35th Computational Complexity Conference

CCC 2020, July 28–31, 2020, Saarbrücken, Germany
(Virtual Conference)

Edited by

Shubhangi Saraf

Editors

Shubhangi Saraf
Department of Mathematics and Department of Computer Science
Rutgers University
Piscatatway, NJ, USA
shubhangi.saraf@rutgers.edu

ACM Classification 2012
Theory of computation

ISBN 978-3-95977-156-6

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-156-6.

Publication date
July, 2020

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

Digital Object Identifier: 10.4230/LIPIcs.CCC.2020.0

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Shubhangi Saraf 0:ix

Awards
..... 0:xi

Conference Organization
..... 0:xiii

External Reviewers
..... 0:xv

Papers

Near-Optimal Erasure List-Decodable Codes
Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma 1:1–1:27

A Quadratic Lower Bound for Algebraic Branching Programs
Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk 2:1–2:21

Super Strong ETH Is True for PPSZ with Small Resolution Width
Dominik Scheder and Navid Talebanfard 3:1–3:12

Approximability of the Eight-Vertex Model
Jin-Yi Cai, Tianyu Liu, Pinyan Lu, and Jing Yu 4:1–4:18

Lower Bounds for Matrix Factorization
Mrinal Kumar and Ben Lee Volk 5:1–5:20

Log-Seed Pseudorandom Generators via Iterated Restrictions
Dean Doron, Pooya Hatami, and William M. Hoza 6:1–6:36

Quantum Lower Bounds for Approximate Counting via Laurent Polynomials
Scott Aaronson, Robin Kothari, William Kretschmer, and Justin Thaler 7:1–7:47

A Generalized Sylvester-Gallai Type Theorem for Quadratic Polynomials
Shir Peleg and Amir Shpilka 8:1–8:33

Simultaneous Max-Cut Is Harder to Approximate Than Max-Cut
Amey Bhangale and Subhash Khot 9:1–9:15

Hitting Sets Give Two-Sided Derandomization of Small Space
Kuan Cheng and William M. Hoza 10:1–10:25

Palette-Alternating Tree Codes
Gil Cohen and Shahar Samocha 11:1–11:29

Search Problems in Algebraic Complexity, GCT, and Hardness of Generators for Invariant Rings
Ankit Garg, Christian Ikenmeyer, Visu Makam, Rafael Oliveira, Michael Walter, and Avi Wigderson 12:1–12:17

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

Statistical Physics Approaches to Unique Games
Matthew Coulson, Ewan Davies, Alexandra Kolla, Viresh Patel, and Guus Regts 13:1–13:27

Schur Polynomials Do Not Have Small Formulas If the Determinant Doesn't
Prasad Chaugule, Mrinal Kumar, Nutan Limaye, Chandra Kanta Mohapatra, Adrian She, and Srikanth Srinivasan 14:1–14:27

Algorithms and Lower Bounds for De Morgan Formulas of Low-Communication Leaf Gates
Valentine Kabanets, Sajin Koroth, Zhenjian Lu, Dimitrios Myrisiotis, and Igor C. Oliveira 15:1–15:41

On the Quantum Complexity of Closest Pair and Related Problems
Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang 16:1–16:43

Limits of Preprocessing
Yuval Filmus, Yuval Ishai, Avi Kaplan, and Guy Kindler 17:1–17:22

Sign Rank vs Discrepancy
Hamed Hatami, Kaave Hosseini, and Shachar Lovett 18:1–18:14

On the Complexity of Modulo-q Arguments and the Chevalley–Warning Theorem
Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis 19:1–19:42

Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions
Shuichi Hirahara 20:1–20:47

Algebraic Branching Programs, Border Complexity, and Tangent Spaces
Markus Bläser, Christian Ikenmeyer, Meena Mahajan, Anurag Pandey, and Nitin Saurabh 21:1–21:24

NP-Hardness of Circuit Minimization for Multi-Output Functions
Rahul Ilango, Bruno Loff, and Igor C. Oliveira 22:1–22:36

A Super-Quadratic Lower Bound for Depth Four Arithmetic Circuits
Nikhil Gupta, Chandan Saha, and Bhargav Thankey 23:1–23:31

Multiparty Karchmer – Wigderson Games and Threshold Circuits
Alexander Kozachinskiy and Vladimir Podolskii 24:1–24:23

Optimal Error Pseudodistributions for Read-Once Branching Programs
Eshan Chattopadhyay and Jyun-Jie Liao 25:1–25:27

Circuit Lower Bounds from NP-Hardness of MCSP Under Turing Reductions
Michael Saks and Rahul Santhanam 26:1–26:13

Finding Small Satisfying Assignments Faster Than Brute Force: A Fine-Grained Perspective into Boolean Constraint Satisfaction
Marvin Künnemann and Dániel Marx 27:1–27:28

Exponential Resolution Lower Bounds for Weak Pigeonhole Principle and Perfect Matching Formulas over Sparse Graphs
Susanna F. de Rezende, Jakob Nordström, Kilian Risse, and Dmitry Sokolov 28:1–28:24

Groups with ALOGTIME-Hard Word Problems and PSPACE-Complete Circuit Value Problems
Laurent Bartholdi, Michael Figelius, Markus Lohrey, and Armin Weiß 29:1–29:29

Optimal Lower Bounds for Matching and Vertex Cover in Dynamic Graph Streams
Jacques Dark and Christian Konrad 30:1–30:14

Connecting Perebor Conjectures: Towards a Search to Decision Reduction for Minimizing Formulas
Rahul Ilango 31:1–31:35

Quantum Query-To-Communication Simulation Needs a Logarithmic Overhead
Sourav Chakraborty, Arkadev Chattopadhyay, Nikhil S. Mande, and Manaswi Paraashar 32:1–32:15

Factorization of Polynomials Given By Arithmetic Branching Programs
Amit Sinhababu and Thomas Thierauf 33:1–33:19

On the Complexity of Branching Proofs
Daniel Dadush and Samarth Tiwari 34:1–34:35

Geometric Rank of Tensors and Subrank of Matrix Multiplication
Swastik Kopparty, Guy Moshkovitz, and Jeroen Zuiddam 35:1–35:21

Hardness of Bounded Distance Decoding on Lattices in ℓ_p Norms
Huck Bennett and Chris Peikert 36:1–36:21

Algebraic Hardness Versus Randomness in Low Characteristic
Robert Andrews 37:1–37:32

Sum of Squares Bounds for the Ordering Principle
Aaron Potechin 38:1–38:37

Preface

The papers in this volume were accepted for presentation at the 35th Computational Complexity Conference (CCC 2020), held between July 28–31, 2020 in a virtual online format. CCC 2020 was originally scheduled to be held in Saarbrücken, Germany, but due to the public health measures related to Covid-19 in place worldwide, the online format was used instead. The conference is organized by the Computational Complexity Foundation (CCF) in cooperation with the ACM Special Interest Group on Algorithms and Computation Theory (SIGACT) and the European Association for Theoretical Computer Science (EATCS).

The call for papers sought original research papers in all areas of computational complexity theory. Of the 101 submissions, the program committee selected 38 for presentation at the conference.

The program committee would like to thank everyone involved in the conference, including all those who submitted papers for consideration as well as the reviewers (listed separately) for their scientific contributions; the board of trustees of the Computational Complexity Foundation and especially its president Venkatesan Guruswami, and secretary Ashwin Nayak for their advice and assistance; Amir Shpilka for sharing his knowledge as prior PC chair for CCC; the Local Arrangements Committee chair Markus Bläser; Shachar Lovett and Thomas Vidick for their invited talks; and Michael Wagner for coordinating the production of these proceedings.

Shubhangi Saraf
Program Committee Chair, on behalf of the Program Committee

Awards

The program committee of the 35th Computational Complexity Conference is very pleased to present the **Best Paper Award** to Daniel Dadush and Samarth Tiwari for their paper

On the Complexity of Branching Proofs;

and the **Best Student Paper Award** to Rahul Ilango for his paper

Connecting Perebor Conjectures: Towards a Search to Decision Reduction for Minimizing Formulas.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

Conference Organization

Program Committee

Per Austrin, KTH Royal Institute of Technology
Zeev Dvir, Princeton University
Prahladh Harsha, Tata Institute of Fundamental Research
Toniann Pitassi, University of Toronto and IAS
Noga Ron-Zewi, Haifa University
Shubhangi Saraf (Chair), Rutgers University
Avishay Tal, University of California at Berkeley
Salil Vadhan, Harvard University
Ryan Williams, Massachusetts Institute of Technology
Ronald de Wolf, CWI and University of Amsterdam
Amir Yehudayoff, Technion – Israel Institute of Technology

Local Arrangements Committee

Markus Bläser (Chair), Universität des Saarlandes

Board of Trustees

Venkatesan Guruswami (President), Carnegie Mellon University
Michal Koucký, Charles University
Shachar Lovett, University of California at San Diego
Ashwin Nayak, University of Waterloo
Ryan O'Donnell, Carnegie Mellon University
Rahul Santhanam, Oxford University
Rocco Servedio, Columbia University
Ronen Shaltiel, University of Haifa
Ryan Williams, Massachusetts Institute of Technology

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

External Reviewers

Amir Abboud
Eric Allender
Shi Bai
Omri Ben-Eliezer
Andrej Bogdanov
Jonah Brown-Cohen
Christopher Cade
Nai-Hui Chia
Gil Cohen
Bill Fefferman
Ran Gelles
Pierre Guillon
Bernhard Haeupler
Rahul Ilango
Valentine Kabanets
Egor V. Kostylev
Guillaume Lagarde
Massimo Lauria
Xin Li
Nikhil Mande
Dylan McKay
Dana Moshkovitz
Jesper Nederlof
Omer Paneth
Chris Peikert
Prasad Raghavendra
Kilian Risse
Dana Ron
Ron Rothblum
Chandan Saha
Luke Schaeffer
Morgan Shirley
Dmitry Sokolov
Suguru Tamaki
Neil Thapen
Thomas Vidick
Osamu Watanabe

Divesh Aggarwal
Sepehr Assadi
David Barrington
Amey Bhangale
Ilario Bonacina
Peter Buergisser
Lijie Chen
Suryajith Chillara
Mina Dalirrooyfard
Noah Fleming
Alexander Golovnev
Zeyu Guo
Shuichi Hirahara
Russel Impagliazzo
Neeraj Kayal
Pravesh K Kothari
Ching-Yi Lai
Chin Ho Lee
Nutan Limaye
Pasin Manurangsi
Pierre McKenzie
Sagnik Mukhopadhyay
Chinmay Nirkhe
Denis Pankratov
Aditya Potukuchi
Anup Rao
Robert Robere
Marc Roth
Aviad Rubinstein
Rahul Santhanam
C. Seshadhri
Adi Shraibman
Aleksa Stankovic
Amnon Tashma
Nithin Varma
Ben Lee Volk
Thomas Watson

Rohit Agrawal
Arturs Backurs
Paul Beame
Jean-Camille Birget
Joshua Brakensiek
Sam Buss
Xi Chen
Kai-Min Chung
Susanna F. de Rezende
Sumegha Garg
Joshua Grochow
Mika Göös
Johan Håstad
Rahul Jain
Alexandr Kazda
Mrinal Kumar
Joseph Landsberg
Euiwoong Lee
Shachar Lovett
Andrew McGregor
Shay Moran
Jack Murtagh
Rafael Oliveira
Ramamohan Paturi
Youming Qiao
Alexander Razborov
Andrei Romashchenko
Guy Rothblum
Sushant Sachdeva
Ramprasad Saptharishi
Ronen Shaltiel
Allan Sly
Joseph Swernofsky
Justin Thaler
Virginia Vassilevska Williams
Nikhil Vyas
Or Zamir

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

Near-Optimal Erasure List-Decodable Codes

Avraham Ben-Aroya
The Blavatnik School of Computer Science, Tel-Aviv University, Israel

Dean Doron
Department of Computer Science, Stanford University, CA, US
ddoron@stanford.edu

Amnon Ta-Shma
The Blavatnik School of Computer Science, Tel-Aviv University, Israel
amnon@tau.ac.il

Abstract

A code $\mathcal{C} \subseteq \{0,1\}^{\bar{n}}$ is (s, L) erasure list-decodable if for every word w, after erasing any s symbols of w, the remaining $\bar{n} - s$ symbols have at most L possible completions into a codeword of $\mathcal{C}$. Non-explicitly, there exist binary $((1-\tau)\bar{n}, L)$ erasure list-decodable codes with rate approaching τ and tiny list-size $L = O(\log \frac{1}{\tau})$. Achieving either of these parameters explicitly is a natural open problem (see, e.g., [26, 24, 25]). While partial progress on the problem has been achieved, no prior nontrivial explicit construction achieved rate better than $\Omega(\tau^2)$ or list-size smaller than $\Omega(1/\tau)$. Furthermore, Guruswami showed no *linear* code can have list-size smaller than $\Omega(1/\tau)$ [24]. We construct an explicit binary $((1-\tau)\bar{n}, L)$ erasure list-decodable code having rate $\tau^{1+\gamma}$ (for any constant $\gamma > 0$ and small τ) and list-size $\text{poly}(\log \frac{1}{\tau})$, answering simultaneously both questions, and exhibiting an explicit non-linear code that provably beats the best possible linear code.

The binary erasure list-decoding problem is equivalent to the construction of explicit, low-error, strong dispersers outputting one bit with minimal entropy-loss and seed-length. For error ε, no prior explicit construction achieved seed-length better than $2\log(\frac{1}{\varepsilon})$ or entropy-loss smaller than $2\log(\frac{1}{\varepsilon})$, which are the best possible parameters for extractors. We explicitly construct an ε-error one-bit strong disperser with near-optimal seed-length $(1+\gamma)\log(\frac{1}{\varepsilon})$ and entropy-loss $O(\log\log\frac{1}{\varepsilon})$.

The main ingredient in our construction is a new (and almost-optimal) *unbalanced* two-source extractor. The extractor extracts one bit with constant error from two independent sources, where one source has length n and tiny min-entropy $O(\log\log n)$ and the other source has length $O(\log n)$ and arbitrarily small constant min-entropy rate. When instantiated as a balanced two-source extractor, it improves upon Raz's extractor [39] in the constant error regime. The construction incorporates recent components and ideas from extractor theory with a delicate and novel analysis needed in order to solve dependency and error issues that prevented previous papers (such as [27, 9, 13]) from achieving the above results.

2012 ACM Subject Classification Theory of computation → Expander graphs and randomness extractors; Theory of computation → Error-correcting codes; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Dispersers, Erasure codes, List decoding, Ramsey graphs, Two-source extractors

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.1

Funding Research supported by the Israel science Foundation grant no. 952/18 and by Len Blavatnik and the Blavatnik Family foundation.

1 Introduction

Extractors and dispersers are important derandomization tools with numerous applications (see, e.g., [40, 43]). Both extractors and dispersers are hash functions $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ that take an input string $x \in \{0,1\}^n$ and an auxiliary seed $y \in \{0,1\}^d$, and output an element $C(x,y)$ in a smaller universe $\{0,1\}^m$ where $m \ll n$. Both extractors and dispersers

are meant to hash any input distribution X that has some crude uniformity to a nearly uniform distribution. Also, the measure of crude uniformity is the same for both objects: We say a distribution X is a k-source if it has k min-entropy, i.e., the probability of each $x \sim X$ is at most 2^{-k}.

Extractors and dispersers differ in the way they measure the proximity of the output distribution to the uniform distribution: Extractors use the total-variation distance, whereas dispersers use support-size distance (that is, they count the number of elements not in the image of the hash function). Extractors are stronger objects, and, roughly speaking, extractors are needed to derandomize two-sided error algorithms whereas dispersers suffice for one-sided error derandomization.

More formally, a function $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a *strong* (k, ε) *extractor* if for any k-source X the output distribution $(U_d, C(X, U_d))$, containing the seed y along with output $C(x, y)$, is ε-close to the uniform distribution over $\{0,1\}^d \times \{0,1\}^m$. We say C is a *strong* (k, ε) *disperser* if for any k-source X, the support of $(U_d, C(X, U_d))$ covers at least $(1-\varepsilon)2^{d+m}$ elements from $\{0,1\}^d \times \{0,1\}^m$.

There are two natural parameters measuring the quality of extractors and dispersers:

1. **Seed length.** Both extractors and dispersers use an auxiliary uniform independent source to extract the entropy from the weak source X. The length d of the auxiliary source is called the *seed-length*. We would like the seed-length to be as small as possible.
2. **Entropy loss.** There are $k + d$ bits of entropy in the system: k bits coming from the k-source X, and d bits from the independent uniform seed. The *entropy-loss* is $k - m$, i.e., the difference between the entropy in the input system (including the seed) and the output system (of length $d + m$).

As noted, strong dispersers are weaker objects than strong extractors. The interest in dispersers stems from the fact that their parameters can outperform those of extractors. For extractors, [37] showed that every strong extractor requires seed-length $d \geq 2\log(\frac{1}{\varepsilon}) + \log(n-k) - O(1)$ and has an unavoidable entropy-loss of $k - m \geq 2\log(\frac{1}{\varepsilon}) - O(1)$. Non-explicitly there exist strong extractors with seed-length $d \leq 2\log(\frac{1}{\varepsilon}) + \log(n-k) + O(1)$ and entropy-loss $k - m \leq 2\log(\frac{1}{\varepsilon}) + O(1)$. For strong dispersers, [37] showed that every strong disperser requires seed-length $d \geq \log(\frac{1}{\varepsilon}) + \log(n-k) - O(1)$ and has an unavoidable entropy-loss $k - m \geq \log\log(\frac{1}{\varepsilon}) - O(1)$. Again, non-explicitly, there exist strong dispersers with seed-length $d \leq \log(\frac{1}{\varepsilon}) + \log(n-k) + O(1)$ and entropy-loss $k - m \leq \log\log(\frac{1}{\varepsilon}) + O(1)$ [37, 33].

For strong dispersers, even the case of outputting just one bit in a way that outperforms extractor constructions has been widely open. Indeed, Gradwohl et al. [22] noticed that such strong dispersers imply good Ramsey graphs, another problem that withstood many attempts for many years, until the recent breakthrough result of Chattopadhyay and Zuckerman [9].

In this paper we go in the reverse direction of that taken in [22]. By using the recent machinery of *non-malleable* extractors and their connection to two-source extractors [9, 7, 15, 28, 29], we construct near-optimal *unbalanced* two-source extractors (which imply near-optimal *unbalanced* bipartite Ramsey graphs). We use these extractors to obtain explicit strong dispersers that output a single bit, with near-optimal seed-length and near-optimal entropy-loss.

▶ **Theorem 1** (see also Theorem 40). *For every constant $0 < \gamma < 1$ and $\varepsilon = n^{-\Omega_\gamma(1)}$ there exists an explicit strong (k, ε) disperser $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ with $d = (1+\gamma)\log(\frac{1}{\varepsilon})$ and $k = O_\gamma(\log\log\frac{1}{\varepsilon})$.*

We remark that the dependence of the seed-length on the error is $(1+\gamma)\log(\frac{1}{\varepsilon}) < 2\log(\frac{1}{\varepsilon})$, and the entropy-loss is $O(\log\log\frac{1}{\varepsilon}) < 2\log(\frac{1}{\varepsilon})$ and both these bounds are optimal for dispersers up to small factors and are *impossible* for extractors. Most previous disperser constructions have not obtained parameters better than the extractors lower bounds, and we are only aware of one exception: Meka et al. [33], extending the techniques in [22], gave a strong disperser with optimal entropy-loss. However, their construction works only for extremely high min-entropy $k = n - \Theta(1)$ and has suboptimal seed-length.[1]

1.1 Erasure List-Decodable Codes

We now turn our attention to binary list-decodable codes in the erasures model. A code $\mathcal{C}$ is a set $\mathcal{C} \subseteq \mathbb{F}_2^n$. We call elements in $\mathbb{F}_2^n$ *words* and elements in C *codewords*. Two interesting parameters of a code are its *redundancy* and its *noise-resiliency*. The redundancy is measured by the *rate* of the code, $\frac{\log|\mathcal{C}|}{n}$. The noise-resiliency is measured according to the model of noise.

In the errors model: A code $\mathcal{C}$ is $(\tau n, L)$ *list-decodable* if for every word $w \in \mathbb{F}_2^n$ there exist at most L codewords in the Hamming ball of radius τn around w.

In the erasures model: A code $\mathcal{C}$ is $(\tau n, L)$ *erasure list-decodable* if for every $z \in \mathbb{F}_2^{(1-\tau)n}$ and every set $T \subseteq [n]$ of size $(1-\tau)n$, the number of codewords that have z in the coordinates indexed by T is at most L.

If $\mathcal{C}$ is $(\tau n, L)$ list-decodable we can recover from τn errors in the following sense: Given a word $w \in \mathbb{F}_2^n$ that was obtained by corrupting at most τn entries of some codeword c, one can (perhaps non-efficiently) produce a small set of size L that necessarily contains c.

Similarly, if $\mathcal{C}$ is $(\tau n, L)$ erasure list-decodable we can recover from τn erasures in the following sense: Given a word $w \in \{0,1,?\}^n$ that was obtained by replacing at most τn entries of some codeword c with the erasure sign '?', one can (perhaps non-efficiently) produce a small set of size L that necessarily contains c.

A strong (k,ε) extractor with one output bit is roughly equivalent to a binary $(\frac{1-\varepsilon}{2}2^d, L = 2^k)$ list-decodable code [42]. In the same spirit, Guruswami [25] observed that strong dispersers with one output bit can be used to construct erasure list-decodable codes. In this paper we complement his argument with the converse statement, showing that erasure list-decodable codes are essentially *equivalent* to strong dispersers with one output bit. Specifically, $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ is a strong (k,ε) disperser if and only if the code $\mathcal{C} : \{0,1\}^n \to \{0,1\}^{2^d}$ defined by $\mathcal{C}(x)_i = \mathsf{Disp}(x,i)$ is $((1-2\varepsilon)2^d, 2^k)$ erasure list-decodable (see Lemma 44).

As we can see, for both extractors and dispersers, the seed-length corresponds to the rate of the code, $\frac{n}{2^d}$, whereas the entropy-loss corresponds to the list-size of the code. Thus, the gap between the seed-lengths of dispersers (which is $\log(\frac{1}{\varepsilon})$) and extractors (which is $2\log(\frac{1}{\varepsilon})$) translates to a difference between rate ε in the erasures model compared with rate ε^2 in the errors model. Similarly, the gap between the entropy-loss of dispersers (which is $\log\log(\frac{1}{\varepsilon})$) and extractors (which is $2\log(\frac{1}{\varepsilon})$) translates to a difference between list-size $\log(\frac{1}{\varepsilon})$ in the erasures model compared with list-size $\mathrm{poly}(\frac{1}{\varepsilon})$ in the errors model. Formally:

- Non-explicitly there exist binary codes having rate $\Omega(\varepsilon^2)$ that are $(\frac{1-\varepsilon}{2}\cdot n, \mathrm{poly}(\frac{1}{\varepsilon}))$ list-decodable and these parameters are tight.
- Non-explicitly there exist binary codes having rate $\Omega(\varepsilon)$ that are $((1-\varepsilon)n, O(\log\frac{1}{\varepsilon}))$ erasure list-decodable, and up to a constant multiplicative factor in the list-size these parameters are tight [24].

[1] Specifically, they support min-entropy $k = n - c$ with seed-length $O(2^c \log(1/\varepsilon))$.

Thus, erasure list-decodable codes can have quadratically better rate and exponentially smaller list-size than list-decodable codes. In fact, Guruswami proved that any *linear* erasure list-decodable codes must have $L = \Omega(1/\varepsilon)$ [24], and so the exponential improvement (or any better than polynomial improvement) is necessarily only possible for non-linear constructions.

The state of affairs for *explicit* binary erasure list-decodable codes is similar to that of *explicit* dispersers. That is, only a few explicit binary erasure list-decodable codes are known to have rate below $\Omega(\varepsilon^2)$ or list-size below $\Omega(1/\varepsilon)$. Guruswami and Indyk [26] gave a *probabilistic* polynomial-time algorithm that outputs with high probability an erasure list-decodable code of rate $\Omega\left(\frac{\varepsilon^2}{\log(1/\varepsilon)}\right)$ and optimal list-size (their construction can be explicitly derandomized when ε is constant). The natural open problem of obtaining erasure list-decodable codes having rate better than ε^2 was explicitly mentioned several times, e.g., in [26, 25]. More concretely, in [23, Open Question 10.2], Guruswami posed the open problem of constructing efficient erasure list-decodable codes of rate ε^{2-a}.

Incorporating the above discussion with Theorem 1, we get the best explicit construction to date:

▶ **Theorem 2** (see also Theorem 46). *For every constant $0 < \gamma < 1$ and $\varepsilon = n^{-\Omega_\gamma(1)}$ there exists an explicit $((1-\varepsilon)\bar{n}, L = \log^{O_\gamma(1)} \frac{1}{\varepsilon})$ erasure list-decodable code $\mathcal{C} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$ of rate $\varepsilon^{1+\gamma}$.*

Thus, Theorem 2 solves Guruswami's problem for the interesting regime of polynomially small ε. We stress that the codes we present are explicit in the sense that they have explicit encoding, but we do not know whether the codes we construct admit *efficient* erasure list-decoding algorithms. We also mention that the list-size $\text{poly}(\log \frac{1}{\varepsilon})$ achieved by our code is exponentially smaller than the best possible list-size by any *linear* code.

1.2 Two-Source Extractors

A function $\mathsf{2Ext} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$ is an $((n_1,k_1),(n_2,k_2),\varepsilon)$ *two-source extractor* if for any two independent sources X and Y, where X is an (n_1,k_1) source and Y is an (n_2,k_2) source, the output distribution $\mathsf{2Ext}(X,Y)$ is ε-close to uniform.

Often, the two-source extractor terminology is more expressive than the extractor notation, as we explain now. Suppose $\mathsf{Ext} : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}$ is a strong (k,ε) extractor. Fix an (n,k) source X, and let ε_i be the distance of the distribution $\mathsf{Ext}(X,i)$ from uniform. By the extractor definition we know that $\mathbb{E}[\varepsilon_i] \leq \varepsilon$. However, the extractor definition does not distinguish between the case where the ε error occurs because all seeds $y \in \mathsf{Supp}(Y)$ have the same error ε, and the case where ε fraction of the seeds have constant error and the rest have none. The situation is different with two-source extractors. Roughly speaking, in an $((n,k),(d,d'),\varepsilon)$ two-source extractor, there are at most $2^{d'-d}$ bad "seeds" y with distance $\varepsilon_y \geq \varepsilon$. Thus, the two-source extractor notation allows separating the fraction of bad seeds from the quality of good seeds.

We would like to explicitly construct a strong (k,ε) disperser $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with parameters better than those of (k,ε) extractors. Thus, on the one hand, for almost every seed y, $\mathsf{Disp}(X,y)$ covers almost all of $\{0,1\}^m$, and, on the other hand, Disp is not a strong extractor, so for almost every seed y, $\mathsf{Disp}(X,y)$ is *far* from uniform. How can this happen?

The situation becomes clearer if we look at strong dispersers with only one additional output bit, i.e., when $m = 1$. As $\mathsf{Disp}(X,y)$ is distributed over one bit, for almost every seed y, $\mathsf{Supp}(\mathsf{Disp}(X,y)) = \{0,1\}$. Yet, it is possible (even necessary, since Disp is not an extractor) that for many seeds y, $\mathsf{Disp}(X,y)$ is ε_0 away from uniform for some constant $\varepsilon_0 \gg \varepsilon > 0$., e.g., when $\mathsf{Disp}(X,y)$ has much more weight on 0 than on 1.

One clean way of capturing this is by using the two-source extractor terminology. We are looking for a two-source extractor $\mathsf{2Ext}$ where almost all seeds (except for ε fraction) are "good" in the sense that y is good if $\mathsf{2Ext}(X, y)$ covers both 0 and 1. Roughly speaking, this amounts to an explicit construction of an $((n,k),(d,d'),\varepsilon_0)$ two-source extractor having $\varepsilon = 2^{d'-d}$ and any non-trivial error $\varepsilon_0 < 1$. Two-source extractors with arbitrary $\varepsilon_0 < 1$ are also called bipartite Ramsey graphs (see Claim 49).

Explicitly constructing two-source extractors (and Ramsey graphs) is a long standing and important challenge. A long line of research (e.g., [10, 39, 8, 5, 6]) culminated in $((n,k),(n,k),\varepsilon_0)$ two-source extractors supporting poly-log min-entropy [14, 9, 32]. This was later improved to $k = O(\log n \frac{\log\log n}{\log\log\log n})$ [7, 15, 28, 29]. However, using the latter two-source extractors gives dispersers with suboptimal entropy-loss and long seed, or, equivalently, erasure list-decodable codes with large list-size and low rate.

Another natural two-source extractor is Raz's two-source extractor [39]. Raz's function is an $((n,k),(d = O(\log \frac{n}{\varepsilon}),d'),\varepsilon_{\mathsf{Raz}})$ two-source extractor that has an unbalanced entropy requirement; the first source is long and very weak (k can be as small as, roughly, $\log\log \frac{n}{\varepsilon_{\mathsf{Raz}}}$), the second source is short and somewhat dense with $d' \geq \delta d$, for any constant $\delta > \frac{1}{2}$. The fact that k can be very small corresponds to a disperser with small entropy-loss, which is good for us. Moreover, d is small, which is again what we want because the length of the corresponding erasure list-decodable code is 2^d. The error $\varepsilon_{\mathsf{Raz}}$ of Raz's extractor is exponentially-small in $\min\{k,d'\}$ which is much better than the mere non-trivial error that we need. However, the second source must be relatively dense, satisfying $\frac{d'}{d} \geq \frac{1}{2}$. This implies that the error ε of the disperser is given by $2^{-d+d'}$ and as a consequence $d \geq 2\log(\frac{1}{\varepsilon})$.

In this paper we show how to explicitly construct the necessary two-source extractor. We show:

▶ **Theorem 3** (see also Theorem 27)**.** *For every two constants $\delta, \varepsilon_0 > 0$ and every $k \geq \Omega_{\delta,\varepsilon_0}(\log\log n)$ there exists an explicit $((n,k),(d,\delta d),\varepsilon_0)$ two-source extractor $\mathsf{2Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ with $d = O_{\delta,\varepsilon_0}(\log n)$.*

Theorem 3 is interesting on its own right. The entropy requirement in both sources is optimal up to constant factors, as both sources have entropy which is logarithmic in the length of the other source. This property is also true for Raz's extractor. On the negative side, Theorem 3 has a large error ε_0, whereas Raz's extractor has a very small error. On the positive side, Raz's extractor works only when $d' = \delta d > 0.5d$ whereas Theorem 3 works with $d' = \delta d$ for any $\delta > 0$, and it is this feature that gives a disperser construction with parameters better than those possible for extractors. Having Theorem 3 immediately gives the strong one output bit disperser and the non-linear near-optimal erasure list-decodable code discussed above.

We also obtain a variant of Theorem 3 that gives a new construction of *balanced* two-source extractors.

▶ **Theorem 4.** *For every two constants $\delta, \varepsilon_0 > 0$ and every $k \geq \Omega_{\delta,\varepsilon_0}(\log n)$ there exists an explicit $((n,k),(n,\delta n),\varepsilon_0)$ two-source extractor $\mathsf{2Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.*

We see that one source has a minimal entropy requirement of $O(\log n)$ while the other has arbitrarily small constant entropy rate. Again, this improves upon [39] in terms of entropy requirement but is worse in terms of error. Theorem 4 is also incomparable to [29] as there, both sources require min-entropy at least $O(\log n \frac{\log\log n}{\log\log\log n})$.

Both Theorem 3 and Theorem 4 follow directly from Theorem 27.

1.3 The Two-Source Extractor Construction

We now give an informal presentation of the two-source extractor construction. We try to keep the discussion intuitive, and for that we omit (or ignore) some technical details. We also assume some familiarity with the field, sometimes using notions that will be formally presented in Section 2.

The input to the $((n,k),(d,\delta d),\varepsilon_0)$ two-source extractor is an (n,k) source X and a $(d,\delta d)$ source Y, for some $0<\delta<\frac{1}{2}$. At a high level, we do the following:

1. Increase the entropy rate of Y from δ to, say, 0.7. For that, we use a constant-error *condenser*. We cannot do it deterministically (because the condenser needs a uniformly random seed) and we still want to keep X fresh. Therefore, we apply the condenser on Y and every possible seed, letting the output of this procedure be a table Y' in which each row corresponds to an application with a different seed. The table Y' has the guarantee that most of the rows of Y' are close to having entropy rate 0.7.
2. Next, we would like to transform the dense rows of Y' to uniformly random strings. For that, we use Raz's extractor with the first source X and the rows of Y' as (independent) seeds. Call the resulting table Y'' and note that it is a function of both X and Y. Also note that although it is now guaranteed that a constant fraction of the rows of Y'' are uniform (Raz's extractor works with entropy rate above half), it is *not* guaranteed (and also not true) that the rows of Y'' are independent of each other.
3. Now we wish to break the dependence between the rows of Y'' so that (ideally) every t of them are uniform and independent (think of t as being poly-logarithmic in the number of rows of Y''). For that, we use a *correlation-breaker* that outputs one bit. The correlation-breaker requires two independent sources, which we do *not* have. Instead, we apply it on Y and Y''. Call the output table Y'''. We shall prove that with high probability, Y''' has many good rows and every t good rows of Y''' are *very* close to being uniform and independent.
4. Finally, we apply a *resilient function* f on the bits of Y'''. The output of our construction is the function's output $f(Y''')$.
   The property that we want from f is that it is nearly balanced and that its output cannot be heavily influenced by any small set of bad bits (the bad rows of Y'''). We need these properties to hold not only when the "good" bits are perfectly uniform and independent, but also under weaker conditions (e.g., that the good players are t-wise independent).

Thus, the coarse structure of our construction is essentially the same as many previous two-source extractor constructions. Namely, we use the two-sources to get a non-oblivious bit-fixing source and then apply a resilient function. There are two known approaches how to implement the first step of getting a non-oblivious bit-fixing source from two independent sources. The first approach was developed by Li [27] and uses alternating extraction (or equivalently, correlation breakers).[2] The second approach, used by Chattopadhyay and Zuckerman [9], uses a non-malleable extractor combined with a sampler. The second approach is more modular, while the first is more flexible.

[2] In [27] Li uses *three* independent sources in his construction. However, Chattopadhyay and Zuckerman [9] remark that their two source extractors could also have been obtained using Li's approach, once a low-depth, highly resilient function is constructed (as is done in [9]). Thus, we view Li's construction as a reduction from *two* independent sources to a non-oblivious bit-fixing source.

We now elaborate more on how Li obtained the above reduction and compare it with our work. The input to Li's protocol are samples from two independent $(n, k = \text{polylog}\, n)$ sources X and Y. The protocol works by applying an extractor E_1 to Y, where we enumerate over every possible seed of E_1 and build a table. Then, each such row is fed as a seed to an extractor E_2, applied on the source X. Li then proceeds to use alternating extraction to get a non-oblivious bit fixing source. (Eventually, he also uses the lightest-bin protocol to obtain a three-source extractor.)

Li's reduction and our construction are very similar, except that:

- In step (1) we replace E_1 with a *constant degree condenser*, and as a result,
- In step (2) the role of E_2 in our construction is played by a *two-source extractor* (of Raz). This is necessary because the output of the condenser is only guaranteed to have high min-entropy. Finally,
- In step (3) a correlation breaker replaces alternating extraction.

While the change seems small it is essential, and the reason why the problem waited its solution for so long. Next, we elaborate on why we use condensers instead of the extractor E_1 and which condensers should be used.

First, we notice that the two-source extractor we are set to construct is different than that of [27] and [9]. [27, 9] construct *balanced* two-source extractors, where each of the two sources is weak (an in particular might have densities well below linear) whereas we are set to construct a highly *non-balanced* extractor where one of the sources (the small one) has *linear* density.

The key observation of the paper is that in such a situation (where the density of one source is linear) the condenser of step (1) has a huge advantage over the extractor E_1 since the condenser might have *constant* seed length (hence a constant number of rows in the table) independent of the row length, and therefore also independent of n, which is totally impossible with extractors. The fact that such explicit condensers exist is a beautiful result of [16]. The analysis (done in Section 4.3) critically uses this fact (that the number of output rows of the condenser is a *constant*) in a delicate way to prove the correctness of the construction.

We also mention that our construction shares steps that are similar to Cohen's construction [12] of three-source extractors. The vital difference is that in [12], a third source is used to achieve complete independence between the rows of a table and then a simple parity can be applied, even if only one row is close to uniform. Here, we only use *two* sources.

To conclude this part, we discuss the dependence problem (to be explained soon), and what aspects of our solution to this problem differ from previous solutions:

- First, there is the issue of lack of independence between the source Y and the seed Y'' in item (3) of the construction. To overcome this, we show a conditioning under which Y'' is still good, Y is independent of Y'' and even after the conditioning the two sources have enough min-entropy. In recent years, such conditioning methods were very successful in constructing an abundance of primitives (e.g., correlation breakers, independence-preserving mergers and non-malleable extractors, etc.).
- Next, there is a delicate issue with the errors. The error $\varepsilon_{\mathsf{cond}}$ of the condenser is high (think of it as a constant). In a naive analysis we would argue that each t good rows are $\varepsilon' > \varepsilon_{\mathsf{cond}}$ close to uniform, and therefore the whole table Y''' is $A^t\varepsilon'$-close to a table where the good rows are perfectly t-wise independent, where A is the number of rows in the table Y'''. However, such an approach is doomed to fail, as necessarily $A\varepsilon_{\mathsf{cond}} > 1$. Our solution for this problem is the heart of the argument. We observe that some of the errors in the construction depend on A, the number of rows in the table, while others depend on the *row length*. In the construction we make sure that A is small (think of it

as a fixed constant) while the row length is unbounded (and, e.g., grows to infinity as n grows to infinity). Thus, we have a natural separation between *large* errors that depend on the number of rows A, and *small* errors that depend on the row length. A similar distinction between large and small errors appears in [27].

The condenser of step (1) and the resilient function of step (4) incur large errors. Raz's extractor (step (2)) and the correlation breaker with advice (step (3)) incur small errors that are exponentially-small in the row length. We show that with some constant probability we succeed in step (1), and that once we have succeeded, the errors δ in steps (2) and (3) are so small that $A^t\delta$ is still small, hence Y''' is close to a table with t-wise independent good players, and so the resilient function in step (4) works (and incurs another constant error). Thus, while the failure probability is high, when we succeed we are exponentially-close to uniform.

Notice that the fact that A is a constant (independent of the row length) is crucial for the argument to work, and this is why we resort to using condensers rather than extractors as in previous solutions.

- Finally, the argument used in the last bullet raises a difficulty regarding the set of good rows. Specifically, in [9], the set of good rows is a function of one of the sources. In our analysis the set of good rows is not just a function of the *sources* X and Y, but also depends on the specific *sample* $y \sim Y$.

Indeed, as we said before, this strategy leads to better unbalanced two-source constructions, and consequently to constructions of near-optimal erasure list-decodable codes (with high rate and small list-size), and one output-bit strong dispersers (with almost optimal seed length and entropy requirement) overcoming barriers that stood open for many years without seeing any progress.

1.4 Non-Strong Dispersers

Strong dispersers are the focal point of this paper. One may wonder why we insist on the strongness property, and whether the problem becomes easier when the strongness property is dropped.

- The answer to the first question is that the strongness property is essential. The equivalence between erasure list-decodable codes and dispersers requires the dispersers to be strong (see Lemma 44, and also notice the correspondence between code coordinates and seeds). Similarly, the connection to Ramsey graphs also requires the disperser to be strong, as already observed by Gradwohl et al. [22]. [22] constructed dispersers that are strong in almost all of the seed, but not strong in some part of the seed, and this drawback is severe enough that none of the applications go through.
- The answer to the second question is that it is easier to construct non-strong dispersers with good parameters. In the paper we prove that it is possible to output more bits from the source at the expense of being strong in only most of the bits (we are non-strong in only $O(1)$ bits of the seed). We prove:

  ▶ **Theorem 5.** *For every constant $0 < \gamma < 1$ and $\varepsilon = n^{-\Omega_\gamma(1)}$ there exists an explicit (k, ε) disperser $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = (1+\gamma)\log(\frac{1}{\varepsilon})$, $k \geq \Omega_\gamma(\log\log\frac{1}{\varepsilon})$ and $m = d + \Omega_\gamma(k)$. The disperser is strong in $d - O_\gamma(1)$ bits of the seed.*

  We sketch a proof of the above theorem in Section 5.2.

1.5 Organization

The rest of the paper is organized as follows. Section 2 covers the preliminaries and notations we use. Section 3 describes the *constant degree* condenser that is used in step (1). Following the above discussion, it is important for us that A, the number of rows in the table, and equivalently the seed-length of the condenser, is a constant independent of the row length. In that section we show one can combine existing constructions of somewhere-random condensers and mergers to achieve that. Next, in Section 4, we describe and analyze the new unbalanced two-source extractor. In Section 5 we use the new two-source extractor to obtain near-optimal strong seeded dispersers, erasure list-decodable codes and unbalanced Ramsey graphs. We conclude with a few open problems in Section 6.

2 Preliminaries

Throughout the paper we use the convention that lowercase variables are the logarithm (in base 2) of their corresponding uppercase variables, e.g., $n = \log N$, $d = \log D$. We denote by $[A]$ the set $\{1, \ldots, A\}$. The density of a set $B \subseteq A$ is $\rho(B) = \frac{|B|}{|A|}$. We say a function $f : A \to B$ is *explicit* if there exists a deterministic polynomial algorithm that runs in time $\text{poly}(\log |A|)$ and computes f.

2.1 Random Variables and Min-Entropy

The *statistical distance* between two distributions X and Y on the same domain Ω is defined as $|X - Y| = \max_{A \subseteq \Omega}(\Pr[X \in A] - \Pr[Y \in A])$. If $|X - Y| \leq \varepsilon$ we say X is ε-close to Y and denote it by $X \approx_\varepsilon Y$. We denote by U_n the random variable distributed uniformly over $\{0,1\}^n$. We say a random variable is *flat* if it is uniform over its support.

For a function $f : \Omega_1 \to \Omega_2$ and a random variable X distributed over Ω_1, $f(X)$ is the random variable distributed over Ω_2 obtained by choosing x according to X and computing $f(x)$. For a set $A \subseteq \Omega_1$, $f(A) = \{f(x) \mid x \in A\}$. For every $f : \Omega_1 \to \Omega_2$ and two random variables X and Y distributed over Ω_1, it holds that $|f(X) - f(Y)| \leq |X - Y|$.

The *min-entropy* of a random variable X is defined by

$$H_\infty(X) \;=\; \min_{x \in \mathsf{Supp}(X)} \log \frac{1}{\Pr[X = x]}.$$

A random variable X is an (n,k) source if X is distributed over $\{0,1\}^n$ and has min-entropy at least k. When n is clear from the context we sometimes omit it and simply say that X is a k-source. Every k-source X can be expressed as a convex combination of *flat* distributions each with min-entropy at least k.

▶ **Definition 6** (average conditional min-entropy)**.** *Let X, Y be two random variables. The* average conditional min-entropy *of X given Y is*

$$\tilde{H}_\infty(X|Y) \;=\; -\log\left(\mathbb{E}_{y \sim Y}\left[2^{-H_\infty(X|Y=y)}\right]\right).$$

We will use the following simple claim about average conditional min-entropy:

▷ Claim 7. For any random variables X, Y, $\tilde{H}_\infty(X|Y) \;\geq\; H_\infty(X) - \log|\mathsf{Supp}(Y)|$.

2.2 Condensers

▶ **Definition 8** (condenser). *A function $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is an $(n,k) \to_{\varepsilon_{\mathsf{cond}}} (m,k')$ condenser, if for every (n,k) source X, $C(X,U_d)$ is $\varepsilon_{\mathsf{cond}}$-close to an (m,k') source. If $k = \delta n$ and $k' = \delta' m$ we say C is a $\delta \to_{\varepsilon_{\mathsf{cond}}} \delta'$ condenser.*

▶ **Lemma 9.** *Suppose $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is an $(n,k) \to_{\varepsilon_{\mathsf{cond}}} (m,k'+d)$ condenser. Let X be an (n,k) source. Let ε_i be the minimal distance of $C(X,i)$ to an (m,k') source. Then, $\mathbb{E}_{i\in\{0,1\}^d}[\varepsilon_i] \le \varepsilon_{\mathsf{cond}}$.*

Proof. Fix an (n,k) source X. For $i \in \{0,1\}^d$, let $H_i \subseteq \{0,1\}^m$ be the set of "heavy" elements of $C(X,i)$,

$$H_i = \left\{ w \in \{0,1\}^m : \Pr_{x\in X}[C(x,i) = w] \ge 2^{-k'} \right\}.$$

The distance of $C(X,i)$ from a k'-source is

$$\varepsilon_i = \Pr_{x\in X}[C(x,i) \in H_i] - 2^{-k'}|H_i|,$$

by redistributing the mass of the heavy elements. Let $H = \bigcup_{i\in\{0,1\}^d} H_i$. Then,

- For every $w \in H$, $\Pr_{x\in X, i\in\{0,1\}^d}[C(x,i) = w] \ge 2^{-d}2^{-k'} = 2^{-(k'+d)}$, and,
- it holds that

$$\begin{aligned}
\varepsilon_{\mathsf{cond}} &= \Pr_{x\in X, i\in\{0,1\}^d}[C(x,i) \in H] - |H|2^{-(k'+d)} \\
&= \sum_{i\in\{0,1\}^d} 2^{-d} \Pr_x[C(x,i) \in H] - |H|2^{-(k'+d)} \\
&\ge \sum_{i\in\{0,1\}^d} 2^{-d} \Pr_x[C(x,i) \in H_i] - 2^{-(k'+d)} \sum_{i\in\{0,1\}^d} |H_i| \\
&= \sum_{i\in\{0,1\}^d} 2^{-d} \left(\Pr_x[C(x,i) \in H_i] - 2^{-k'}|H_i| \right) \\
&= \sum_{i\in\{0,1\}^d} 2^{-d}\varepsilon_i \;=\; \mathbb{E}_{i\in\{0,1\}^d}[\varepsilon_i].
\end{aligned}$$

◀

2.3 Two-Source Extractors

▶ **Definition 10** (two-source extractor). *A function $\mathsf{2Ext} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$ is an $((n_1,k_1),(n_2,k_2),\varepsilon)$ two-source extractor if for every two independent sources X_1 and X_2 where X_1 is an (n_1,k_1) source and X_2 is an (n_2,k_2) source, it holds that $\mathsf{2Ext}(X_1,X_2) \approx_\varepsilon U_m$. We say that* 2Ext *is* strong *if*

$$(\mathsf{2Ext}(X_1,X_2), X_1) \approx_\varepsilon (U_m, X_1)$$

and

$$(\mathsf{2Ext}(X_1,X_2), X_2) \approx_\varepsilon (U_m, X_2).$$

In our construction, we will use the following two-source extractor:

▶ **Theorem 11** ([39]). *For every constant $\delta_{\mathsf{Raz}} > \frac{1}{2}$ there exist constants $c_1 = c_1(\delta_{\mathsf{Raz}}), c_2 = c_2(\delta_{\mathsf{Raz}}) > 1$ such that for every n_1, k_1, n_2, k_2 satisfying*

- $k_1 \geq c_1 \log n_2$,
- $k_2 \geq c_2 \log n_1$,

there exists an explicit strong $((n_1, k_1), (n_2, k_2 = \delta_{\mathsf{Raz}} n_2), \varepsilon_{\mathsf{Raz}})$ two-source extractor

$$\mathsf{Raz} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$$

with $m = \Omega(\min\{k_1, k_2\})$ and $\varepsilon_{\mathsf{Raz}} = 2^{-\Omega(m)}$, where the constants hiding in the asymptotic notation may depend on δ_{Raz}.

▷ Claim 12. Suppose $\mathsf{2Ext} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$ is a strong $((n_1, k_1), (n_2, k_2), \varepsilon)$ two-source extractor. Let X be an (n, k_1) source. Call an element $y \in \{0,1\}^{n_2}$ *bad* if $|\mathsf{2Ext}(X, y) - U_m| > \varepsilon$, and let B_Y denote the set of all bad elements. Then, $|B_Y| < 2^{k_2}$.

Proof. Assume towards contradiction that $|B_Y| \geq 2^{k_2}$ and let Y be the uniform distribution over the set B_Y. Then, $H_\infty(Y) \geq k_2$ and so $(\mathsf{2Ext}(X, Y), Y) \approx_\varepsilon (U_m, Y)$ which implies that

$$\frac{1}{|B_Y|} \sum_{y \in B_Y} |\mathsf{2Ext}(X, y) - U_m| \;\leq\; \varepsilon.$$

However, $|\mathsf{2Ext}(X, y) - U_m| > \varepsilon$ for every $y \in B_Y$, in contradiction. ◀

2.4 Mergers

A *merger* takes as input a list of possibly correlated random variables along with a short uniform seed and outputs one random variable which is close to having high min-entropy, provided at least one of the input variables has high min-entropy. Formally:

▶ **Definition 13** (somewhere-random source). *A source $X = X_1 \circ \ldots \circ X_A$ is an $(n, k, (\alpha, \beta))$* somewhere-random (s.r.) source *if there is a random variable $I \in \{0, \ldots, A\}$ such that for every $i \in [A]$, $(X_i | I = i)$ is α-close to an (n, k) source and $\Pr[I = 0] \leq \beta$. The variable I is called the* indicator *of source. If $\alpha = \beta = 0$ we say X is a (n, k)* s.r. source.

▶ **Definition 14** (merger). *A function $B : (\{0,1\}^n)^D \times \{0,1\}^t \to \{0,1\}^m$ is a (k, k', ε) merger, if for every (n, k) s.r. source $X = X_1 \circ \ldots \circ X_A$, the output $M(X, U_t)$ is ε-close to a k'-source.*

There are explicit constructions of good mergers. Dvir and Wigderson [17] constructed the *curve merger* and proved that it works with $t = O(\log \frac{n}{\varepsilon})$. This was further improved in [16] who proved that $t = O(\log \frac{D}{\varepsilon})$ suffices. Notice that now t only depends on the number of sources D and the requested error ε, but *not* on the source length n, and this remarkable property will be crucial for us. Formally,

▶ **Theorem 15** ([17, 16]). *There exists a constant $c_{\mathsf{DKSS}} \geq 1$ such that the following holds. Fix $\beta, \delta, \varepsilon > 0$. There exists an explicit function $B : (\{0,1\}^n)^D \times \{0,1\}^t \to \{0,1\}^n$ that is a $(k = \delta n, k' = (1 - \beta)\delta n, \varepsilon)$ merger, with $t = c_{\mathsf{DKSS}} \cdot \frac{1}{\beta} \log \frac{D}{\varepsilon}$.*

2.5 Correlation Breakers with Advice

A correlation-breaker with advice is a function $\mathsf{CBA} : \{0,1\}^n \times \{0,1\}^\ell \times [A] \to \{0,1\}^m$ where we think of the first input as a weak source, the second as an independent short seed and the last as an advice string. Roughly speaking, applying CBA on t possibly correlated seeds with t distinct advice strings results in independent random variables. For example, $\mathsf{CBA}(X, Y, \alpha)$ is (nearly) independent of $\mathsf{CBA}(X, Y, \alpha')$ for any $\alpha \neq \alpha'$. Formally,

▶ **Definition 16.** *A function* $\mathsf{CBA} : \{0,1\}^n \times \{0,1\}^\ell \times [A] \to \{0,1\}^m$ *is a* $(t, k, \varepsilon_{\mathsf{CBA}})$ *correlation-breaker with advice if the following holds. If* Y *is a distribution over* $\{0,1\}^n$, $Z = (Z_1, \ldots, Z_t)$ *is a distribution on* $(\{0,1\}^\ell)^t$, $\mathcal{H}$ *is a random variable and* $\delta > 0$ *satisfy:*

- Y *and* Z *are independent, conditioned on* $\mathcal{H}$,
- $\tilde{H}_\infty(Y|\mathcal{H}) \geq k + \log(1/\varepsilon_{\mathsf{CBA}})$,
- $(Z_1, \mathcal{H}) \approx_\delta (U_\ell, \mathcal{H})$, *and,*
- $\alpha_1, \ldots, \alpha_t \in [A]$ *are distinct strings.*

Then,

$$\left(\mathsf{CBA}(Y, Z_1, \alpha_1), (\mathsf{CBA}(Y, Z_i, \alpha_i))_{i=2}^t, \mathcal{H}\right) \approx_{\delta + 2\varepsilon_{\mathsf{CBA}}} \left(U_m, (\mathsf{CBA}(Y, Z_i, \alpha_i))_{i=2}^t, \mathcal{H}\right).$$

We use the following result:

▶ **Theorem 17** ([13, Theorem 4.12])**.** *There exists a constant* $c_{\mathsf{CBA}} \geq 1$ *such that the following holds. Let* n, a *be integers and* $\varepsilon_{\mathsf{CBA}} > 0$. *Then, there exists an explicit* $(t, k_{\mathsf{CBA}}, \varepsilon_{\mathsf{CBA}})$ *correlation-breaker with advice*

$$\mathsf{CBA} : \{0,1\}^n \times \{0,1\}^\ell \times [A] \to \{0,1\}$$

with $\ell = c_{\mathsf{CBA}} \cdot at \cdot \log \frac{n}{\varepsilon_{\mathsf{CBA}}}$ *and* $k_{\mathsf{CBA}} \geq \ell$.

In our setting, the number of rows A is a constant independent of n. For this reason we work with a "basic" correlation-breaker, where there is no attempt to optimize the dependence of ℓ on a. This gives a seed-length which is optimal up to constant multiplicative factors.

We also need the following lemma.

▶ **Lemma 18.** *Let* $X_1, \ldots, X_t$ *be random variables over* $\{0,1\}^m$. *Further suppose that for any* $i \in [t]$,

$$\left(X_i, \{X_j\}_{j \neq i}\right) \approx_\varepsilon \left(U_m, \{X_j\}_{j \neq i}\right).$$

Then, $(X_1, \ldots, X_t) \approx_{t\varepsilon} U_{tm}$.

2.6 Limited Independence and Non-Oblivious Bit-Fixing Sources

▶ **Definition 19.** *A distribution* X *over* $\{0,1\}^A$ *is called* (t, γ)-wise independent *if the restriction of* X *to every* t *coordinates is* γ*-close to* U_t. *A source* X *over* $\{0,1\}^A$ *is called a* (q, t, γ) non-oblivious bit-fixing source *if there exists a subset* $Q \subseteq A$ *of size at most* q *such that the joint distribution of the bits in* $A \setminus Q$ *is* (t, γ)*-wise independent. The bits in* Q *are allowed to arbitrarily depend on the bits in* $A \setminus Q$. *If* $\gamma = 0$ *we often say that* X *is a* (q, t) *non-oblivious bit-fixing source.*

▶ **Lemma 20** ([3])**.** *A* (t, γ)*-wise distribution over* A *bits is* $(A^t\gamma)$*-close to some* t*-wise independent distribution.*

▶ **Definition 21.** *Let* $f : \{0,1\}^A \to \{0,1\}$, $\mathcal{D}$ *a distribution over* $\{0,1\}^A$ *and* $Q \subseteq A$. *Let* $I_{Q,\mathcal{D}}(f)$ *denote the probability that* f *is undetermined when the variables outside* Q *are sampled from* $\mathcal{D}$. *We define* $I_{q,t,\gamma}(f)$ *to be the maximum of* $I_{Q,\mathcal{D}}(f)$ *over all* $Q \subseteq A$ *of size* q *and all* $\mathcal{D}$ *that is a* (t, γ)*-wise independent distribution. We say that* f *is* (t, γ)-independent (q, ε)-resilient *if* $I_{q,t,\gamma}(f) \leq \varepsilon$.

▶ **Theorem 22** ([9, 32]). *For every $0 < \gamma < 1$ there exists a constant $c_\gamma \geq 1$ such that for all $A > 0$ there exists an explicit function $f : \{0,1\}^A \to \{0,1\}$ with the following property:*

For every $t \geq c_\gamma \log^4 A$,

- *f is almost balanced: For any t-wise independent distribution $\mathcal{D}$ on $\{0,1\}^A$,*

$$\Pr_{x\sim\mathcal{D}}[f(x) = 1] = 1/2 \pm A^{-1/c_\gamma}, \text{ and,}$$

- *f is resilient: $I_{q,t,\gamma}(f) \leq c_\gamma \cdot \frac{q}{A^{1-\gamma}}$.*

3 Constant Degree Condensers

In this section we prove:

▶ **Theorem 23.** *For every constant $0 < \delta_1 < \delta_2 = 0.7$, every $s \geq s_0(\delta_1)$ and every integer n_1 and $\varepsilon_{\mathsf{cond}} \geq 2^{-\Omega(n_1)}$ there exists an explicit $\delta_1 \to_{\varepsilon_{\mathsf{cond}}} \delta_2$ condenser $C : \{0,1\}^{n_1} \times \{0,1\}^d \to \{0,1\}^{n_2}$ with $n_2 = (\frac{2}{3})^s n_1$ and $d = 4c_{\mathsf{DKSS}}\left(s + \log\frac{1}{\varepsilon_{\mathsf{cond}}}\right)$, where c_{DKSS} is the constant from Theorem 15. Note that d is independent of n_1.*

Note that, in particular, for every $\delta_1 > 0$ there exists an explicit $\delta_1 \to_{\varepsilon_{\mathsf{cond}}} \delta_2 = 0.7$ condenser $C : \{0,1\}^{n_1} \times \{0,1\}^d \to \{0,1\}^{n_2}$ with $n_2 = \Omega(n_1)$ and $d = O(\log\frac{1}{\varepsilon_{\mathsf{cond}}})$. However, we will need the more precise version that appears in Theorem 23.

The proof goes through *somewhere-random condensers*, so let us first discuss the similarities and differences between condensers and somewhere-random condensers. We begin with the necessary definitions:

▶ **Definition 24** (s.r. condenser). *A function $C : \{0,1\}^n \to (\{0,1\}^m)^A$ is an $(n,k) \to_\varepsilon (m,k')$ s.r. condenser if for every (n,k) source X it holds that $C(X) = C(X,1) \circ \ldots \circ C(X,A)$ is ε-close to a k' s.r. source. If $k = \delta n$ and $k' = \delta m$ we say C is $\delta \to_\varepsilon \delta'$ s.r. condenser.*

We may take a condenser $C : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ and expand it to a table with the outputs of all possible seeds, i.e., define $S : \{0,1\}^n \to (\{0,1\}^m)^D$, with $D = 2^d$, where $S(x)_i = C(x,i)$. The condenser property guarantees that for every k-source X, *most* rows in the table are close to having k' min-entropy. In contrast, a s.r. condenser is a weaker object, because it only guarantees that *one* row has k' entropy (or more precisely that we are in a convex combination of such cases).

The major question we consider now is the dependence of the degree (2^d for condensers and A for s.r. condensers) on n, m, k, k' and ε. We focus on the case where $m = \Omega(n)$, $k = \delta n$, $k' = \delta' m$ and $\delta < \delta'$ are constants. A priori, we could have expected the degree to depend on n and ε, as is indeed the case when m might be arbitrarily small. However, remarkably, things are drastically different when $m = \Omega(n)$. In this case both condensers and s.r. condensers may be of degree that is independent of n and this will be crucial for us. If we consider the dependence on the error, then s.r. condensers may have exponentially-small error and constant D, whereas the degree of a condenser is at least $d \geq \log(\frac{1}{\varepsilon})$. Remarkably, all of that can be explicitly achieved, as we now explain.

The basic building block we use is the following beautiful result of Zuckerman, which is based on additive combinatorics:

▶ **Theorem 25** ([45, Theorem 8.3]). *For every constant $0 < c < 1$ there exists a constant $\alpha = \alpha(c)$ such that for every constant $\delta \leq c$ and integer n there exists an explicit function $C : \{0,1\}^n \to (\{0,1\}^{\frac{2}{3}n})^2$ that is a $\delta \to_\varepsilon (1+\alpha)\delta$ s.r. condenser with $\varepsilon = 2^{-\Omega(\alpha\delta n)}$.*

Somewhere-random condensers can be easily composed. Specifically, Barak et al. [5] showed that if $C_1 : \{0,1\}^{n_1} \to (\{0,1\}^{n_2})^{\ell_1}$ is a $\delta_1 \to_{\varepsilon_1} \delta_2$ s.r. condenser and $C_2 : \{0,1\}^{n_2} \to (\{0,1\}^{n_3})^{\ell_2}$ a $\delta_2 \to_{\varepsilon_2} \delta_3$ s.r. condenser then $C_2 \circ C_1 : \{0,1\}^{n_1} \to (\{0,1\}^{n_3})^{\ell_1 \cdot \ell_2}$ defined by $C_2 \circ C_1(x)_{(i_1,i_2)} = C_2(C_1(x)_{i_1})_{i_2}$ is a $\delta_1 \to_{\varepsilon_1+\varepsilon_2} \delta_3$ s.r. condenser.

Composing the s.r condenser of Theorem 25 with itself s times we get an explicit function $C : \{0,1\}^n \to (\{0,1\}^m)^D$ with $D = 2^s$ and $m = (\frac{2}{3})^s n$ that is a $\delta \to_\varepsilon \delta'$ s.r. condenser with $\varepsilon = \sum_{i=1}^{s} 2^{-\Omega((1+\alpha)^i \delta (\frac{2}{3})^i n)} = 2^{-\Omega(m)}$ and $\delta' \geq (1+\alpha(\delta'))^s \delta$. Therefore:

▶ **Lemma 26.** *For every constants $0 < \delta_1 < \delta_2 < 1$ there exists a constant $s = s(\delta_1, \delta_2)$ and an explicit function $C : \{0,1\}^{n_1} \to (\{0,1\}^{n_2})^D$ that is a $\delta_1 \to_\varepsilon \delta_2$ s.r. condenser with $D = 2^s$, $n_2 = (\frac{2}{3})^s n_1$ and $\varepsilon = 2^{-\Omega(n_2)}$. Note that D is independent of n and ε.*

Right now, if X is a k-source, the table $C(X)$ has D rows, and, roughly speaking, the guarantee is that one of these rows has density δ'. We want to change this to get a condenser, i.e., we are willing to invest a short seed (that is independent of n) and we want to get *one* output which is close to uniform. (Alternatively, we can write the condenser as a table with one row per seed, the number of rows is independent of n and most rows are close to uniform.) This is exactly what a *merger* does and applying the merger of Theorem 15 with $\beta = \frac{1}{4}$ on the s.r. condenser of Lemma 26 (with δ_2 close to 1) gives Theorem 23.

4 The Unbalanced Two-Source Extractor Construction

The main result of this section is the following two-source extractor.

▶ **Theorem 27.** *For every integer n and two constants $\delta_0, \varepsilon_0 > 0$ there exists a constant c such that for $d \geq c \log n$ and $k \geq c \log d$ there exists an explicit $((n,k),(d,\delta_0 d),\varepsilon_0)$ two-source extractor $\mathsf{2Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$.*

The extractor in the above theorem has constant error, and works when:

1. Each source's entropy is in the order of the logarithm of the length of the other source.
2. The shorter source, of length d, has an arbitrarily small constant density δ_0.

We think of n and $d = d(n)$ as growing parameters while ε_0 and δ_0 are constants. We use asymptotic notations (such as $\Omega(\cdot)$) to hide constants that are independent of n and d (but may depend on ε_0 and δ_0).

4.1 The Construction

Recall that ε_0 is the target error of the extractor 2Ext. The input to 2Ext is a pair (x,y) where x is drawn from an (n,k) source X, and y is drawn from an independent $(d,\delta_0 d)$ source Y. Our problem is that the y comes from a $\delta_0 d$-source for some $\delta_0 < \frac{1}{2}$. To overcome this, we do the following:

- We apply the condenser of Theorem 23 on y to get a table y' that is 1-wise 0.7-dense. Notice that the output of this step is a table rather than a single output.
- We apply Raz's extractor (Theorem 11) on the table and the input x from the other source to convert the table y' to another table y'' that is 1-wise uniform.
- We apply the t correlation-breaker with advice of Theorem 17 on y, using the table y'' as the seed, to get a table y''' that is t-wise uniform.
- Finally, we apply the resilient function f of Theorem 22 on the table y''' to collapse the many rows of the table to a single, close to uniform, output.

Formally, these steps work as follows:

Condense the short source: We are given $\delta_0 < \frac{1}{2}$. Set $\delta' = 0.69$ and $\delta_2 = 0.7$.
By Theorem 23 there exists a constant $s_0 = s_0(\delta_0)$ such that for every $s \geq s_0$ there exists an explicit

$$C : \{0,1\}^d \times \{0,1\}^a \to \{0,1\}^{d'}$$

that is a $\delta_0 \to_{\varepsilon_{\mathsf{cond}}} \delta_2 = 0.7$ condenser with $a = 4c_{\mathsf{DKSS}}(s + \log \frac{1}{\varepsilon_{\mathsf{cond}}})$ and $d' = (\frac{2}{3})^s d$.
We set

$$\gamma = \frac{1}{2^5 c_{\mathsf{DKSS}}},$$

and this also fixes c_γ as in Theorem 22. Notice that γ and c_γ are fixed constants independent of all other parameters in our system.
Now, choose $\varepsilon_{\mathsf{cond}}$ so that

$$\left(\frac{1}{\varepsilon_{\mathsf{cond}}}\right)^{\log(3/2)} \geq \frac{4}{\delta_0} 2^{12} c_\gamma c_{\mathsf{DKSS}}^4 \log^4 \frac{1}{\varepsilon_{\mathsf{cond}}}, \tag{1}$$

and also so that $\varepsilon_{\mathsf{cond}} \leq \xi(\varepsilon_0, \delta_0)$, where

$$\xi(\varepsilon_0, \delta_0) = \min\left\{2^{-s_0}, \left(\frac{\varepsilon_0}{8}\right)^2, \left(\frac{\varepsilon_0}{5}\right)^{c_\gamma}, \left(\frac{\varepsilon_0}{5c_\gamma}\right)^{1/\gamma}\right\}. \tag{2}$$

Given $\varepsilon_{\mathsf{cond}}$, we set $s = \log \frac{1}{\varepsilon_{\mathsf{cond}}} \geq s_0$, giving $a = 8c_{\mathsf{DKSS}} \log \frac{1}{\varepsilon_{\mathsf{cond}}}$. Note that the degree of the condenser, $A = 2^a$, satisfies

$$\sqrt{\varepsilon_{\mathsf{cond}}} A = 2^{-\frac{1}{2}\log \frac{1}{\varepsilon_{\mathsf{cond}}} + a} = 2^{-\frac{a}{2^4 c_{\mathsf{DKSS}}} + a} = A^{1-2\gamma}.$$

Observe that $s \geq s_0$ and that $d' = \Omega(d)$. Also, notice that $(\delta_2 - \delta')d' = d'/100 \geq a = \log A$ for large enough d. Thus, C is a $(d, \delta d) \to_{\varepsilon_{\mathsf{cond}}} (d', \log(A) + \delta' d')$ condenser.
Define an $A \times d'$ table Y' where

$$Y'_i = C(Y, i) \in \{0,1\}^{d'}$$

for $i = 1, \ldots, A$.

1-wise uniformity: Let c_1, c_2 be the constants from Theorem 11 for $\delta_{\mathsf{Raz}} = 0.6$.
Notice that $\delta_{\mathsf{Raz}} d' = \Omega(d') = \Omega(d)$. Therefore, for a constant c large enough, $d \geq c \log n$ is large enough so that $\delta_{\mathsf{Raz}} d' \geq c_2 \log n$. We can, in particular, choose c such that in addition $c \geq c_1$. Recalling that $k \geq c \log d$, we have $k \geq c_1 \log d'$. By Theorem 11, there exists an explicit function

$$\mathsf{Raz} : \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{d''}$$

that is a strong $((n,k),(d', \delta_{\mathsf{Raz}} d'), \varepsilon_{\mathsf{Raz}} = 2^{-\Omega(d'')})$ two-source extractor with $d'' = \Omega(\min\{k, \delta_{\mathsf{Raz}} d'\}) = \Omega(k)$.[3] Define an $A \times d''$ table Y'' where

$$Y''_i = \mathsf{Raz}(X, Y'_i)$$

for $i = 1, \ldots, A$.

[3] Although $k \geq c \log d$ we can always assume w.l.o.g. that $k = c \log d$ and so $k \leq \delta_{\mathsf{Raz}} d' = \Omega(d)$.

t-wise uniformity: Let $k_{\mathsf{CBA}} = \frac{\delta_0 d}{8}$ and $\varepsilon_{\mathsf{CBA}} = \frac{1}{d}$. Set

$$t \;=\; \frac{\delta_0}{4}\left(\frac{3}{2}\right)^s.$$

Notice that for a large enough constant c we have $d'' = \Omega(k) = \Omega(c \log d) \geq c_{\mathsf{CBA}} a t \log \frac{d}{\varepsilon_{\mathsf{CBA}}}$, where the latter is the seed-length required by the correlation-breaker from Theorem 17. Also, $k_{\mathsf{CBA}} = \frac{\delta_0 d}{8} \geq d''$ for large enough d, as $d'' = \Omega(k) = \Omega(\log d)$. Hence, by Theorem 17 there exists an explicit function

$$\mathsf{CBA} : \{0,1\}^d \times \{0,1\}^{d''} \to \{0,1\}$$

that is a $(t, k_{\mathsf{CBA}}, \varepsilon_{\mathsf{CBA}})$ correlation-breaker with advice.
Define an $A \times 1$ table Y''' where

$$Y'''_i = \mathsf{CBA}(Y, Y''_i, i)$$

for $i = 1, \ldots, A$.
Keep in mind that the entropy in Y suffices for CBA since $H_\infty(Y) = 8k_{\mathsf{CBA}}$.

Collapse: Take $f : \{0,1\}^A \to \{0,1\}$ to be the $(q = A^{1-2\gamma}, t, \varepsilon_f = c_\gamma A^{-\gamma})$ resilient function of Theorem 22 and output $f(y'''_1, \ldots, y'''_A)$.

4.2 Two Subtleties

As mentioned in the introduction, there are several delicate issues in the analysis:

1. Circular dependence: Y'' depends on both X and Y, and is used as a seed in the application of the correlation-breaker with advice on Y.
2. We need Y''' to be close to a perfect t-wise independent table, while the correlation-breaker with advice only guarantees that every t good rows are close to uniform. To bridge the gap we need the error to be at least polynomially-small in the number of rows, but some of the steps incur a large constant error.

To overcome the first issue we show a conditioning under which Y'' is still good, Y is independent of Y'' and even after the conditioning the two sources have enough min-entropy.

To overcome the second issue we distinguish between large errors that depend on the number of rows A, and small errors that depend on the row length (see Section 1.3 in the introduction). In particular, the errors are of three types:

- The probability p_1 that a value we condition upon is bad. This error is incurred by the condenser and is high (think of it as being a constant).
- We show that when we condition on a good value, every t good rows in Y''' are p_2-close to uniform. We then claim that Y''' as a table is $A^t p_2$-close to a table where the good rows are truly t-wise independent (where A is the number of rows in the table Y'''). The error p_2 is incurred by Raz's extractor and by the correlation-breaker, and can be made very small if we deal with a source X having enough min-entropy. We make p_2 small enough so that $A^t p_2$ is also small.
- A third error p_3 is incurred by the resilient function f. This error is large, say, a constant, and we are fine with that.

Note that we cannot just accumulate all errors as $A^t p_1$ is way larger than 1. Instead, we argue that with a constant probability $1 - p_1$, we get extremely close to perfect behavior, and then we get such a small error p_2 so that $A^t p_2$ is also small.

4.3 The Analysis

Proof of Theorem 27. Fix an (n,k) source X and an independent $(d,\delta d)$ source Y. We decompose the proof into three parts:

- In the first part we prove that very often (except for a small constant probability) the table Y'' contains many rows that are marginally close to uniform.
- Next, we prove that every set of t rows $\{i,j_1,\ldots,j_{t-1}\}$ in Y''' are *product* in the sense that if i is a good row (intuitively meaning that Y''_i is close to uniform) and $j_1,\ldots,j_{t-1}$ are $t-1$ other rows, then in Y''', Y'''_i is close to uniform and *independent* of $Y'''_{j_1},\ldots,Y'''_{j_{t-1}}$. This part involves applying a correlation-breaker with advice on Y and Y''. In order to ensure that Y and Y'' are independent, we condition on the values of Y' in the t rows $\{i,j_1,\ldots,j_{t-1}\}$.
- Together, except for a small constant probability, there are many good rows, and every t rows of Y''' are product, hence the table Y''' is close to a (q,t) non-oblivious bit-fixing source, where every good row is a good bit in the bit-fixing source. Hence, $f(Y''')$ is close to uniform.

Part 1: Often, many rows in Y' are good

Let ε_i be the minimal distance of $C(Y,i)$ from a $\delta' d'$-source. According to Lemma 9,

$$\mathbb{E}_{i\in[A]}[\varepsilon_i] \le \varepsilon_{\mathsf{cond}}.$$

▶ **Definition 28.** *We say $z\in\{0,1\}^{d'}$ is* good *if $\mathsf{Raz}(X,z)$ is $\varepsilon_{\mathsf{Raz}}$-close to uniform. Let GZ be the set of all good z-s, and BZ the rest. We say $i\in[A]$ is* good for $y\in\{0,1\}^d$ *if $C(y,i)\in GZ$ and* bad *otherwise. We define a random variable B_i, where the sample space is Y, and $B_i(y)=1$ if i is bad for y and 0 otherwise.*

By Claim 12, $|BZ|\le 2^{\delta_{\mathsf{Raz}}d'}$. Therefore, in expectation, the number of bad rows for y is small:

▷ Claim 29. $\mathbb{E}_{y\in Y}\left[\sum_{i\in[A]} B_i(y)\right] \le 2\varepsilon_{\mathsf{cond}}A$.

Proof. Fix an $i\in[A]$. We have that $C(Y,i)$ is ε_i-close to some $\delta'd'=0.69d'$-source R. Hence:

$$\mathbb{E}_y[B_i(y)] = \Pr_{y\in Y}[C(y,i)\in BZ] \le \varepsilon_i + \Pr_{r\in R}[r\in BZ] \le \varepsilon_i + \frac{|BZ|}{2^{\delta' d'}} = \varepsilon_i + 2^{-0.09d'}.$$

Thus, for d large enough,

$$\mathbb{E}_y\Big[\sum_{i\in[A]} B_i(y)\Big] = \sum_{i\in[A]}\mathbb{E}_y[B_i(y)] \le \sum_{i\in[A]}\left(\varepsilon_i + 2^{-0.09d'}\right) \le \varepsilon_{\mathsf{cond}}A + 2^{-0.09d'}A \le 2\varepsilon_{\mathsf{cond}}A. \lhd$$

▶ **Definition 30.** *We say $y\in\mathsf{Supp}(Y)$* has many bad rows *if $\sum_{i\in[A]}B_i(y)\ge\sqrt{\varepsilon_{\mathsf{cond}}}A$.*

Denote $p_{1,1}=\frac{\varepsilon_0}{4}$.

▷ Claim 31. $\Pr_{y\in Y}[y \text{ has many bad rows}] \le p_{1,1}$.

Proof. By Markov,

$$\Pr_{y\in Y}\Big[\sum_i B_i(y) \ge \sqrt{\varepsilon_{\mathsf{cond}}}A\Big] \le \frac{\mathbb{E}\left[\sum_i B_i(y)\right]}{\sqrt{\varepsilon_{\mathsf{cond}}}A} \le \frac{2\varepsilon_{\mathsf{cond}}A}{\sqrt{\varepsilon_{\mathsf{cond}}}A} = 2\sqrt{\varepsilon_{\mathsf{cond}}} \le \frac{\varepsilon_0}{4},$$

where the last inequality follows from the fact that $\varepsilon_{\mathsf{cond}}\le(\frac{\varepsilon_0}{8})^2$. $\lhd$

Part 2: The good rows are t-wise independent

We introduce some notations to simplify the expressions in the proof. For $y_0 \in \{0,1\}^d$ and $k \in [A]$, let $Y'''_k(y_0)$ denote $(Y'''_k | Y = y_0)$. Also, for a set $S \subseteq [A]$, define $Y'''_S(y_0) = \left\{Y'''_j(y_0)\right\}_{j\in S}$. Denote $p_2 = \varepsilon_{\mathsf{Raz}} + 2\varepsilon_{\mathsf{CBA}}$.

▶ **Definition 32.** *Let $y_0 \in \{0,1\}^d$ (not necessarily in the support of Y). Let $i \in [A]$ and $S \subseteq [A] \setminus \{i\}$ of cardinality $t-1$. We say y_0* violates the product rule *for (i,S) if $B_i(y_0) = 0$ and*

$$(Y'''_i(y_0), Y'''_S(y_0)) \;\not\approx_{p_2}\; U_1 \times Y'''_S(y_0).$$

▶ **Definition 33.** *Let $y_0 \in \{0,1\}^d$ (not necessarily in the support of Y). Let $i \in [A]$ and $S \subseteq [A] \setminus \{i\}$ of cardinality $t-1$. We say y_0* violates the product rule with distinguisher $\Delta : \{0,1\}^t \to \{0,1\}$ *for (i,S) if $B_i(y_0) = 0$ and*

$$\left| \Pr[\Delta\left(Y'''_i(y_0), Y'''_S(y_0)\right) = 1] - \Pr[\Delta(U_1, Y'''_S(y_0)) = 1] \right| > p_2.$$

Observe that if y_0 violates the product rule then there exists some Δ such that y_0 violates the product rule with distinguisher Δ.

▶ **Lemma 34.** *For every i and S as above, the number of $y \in \{0,1\}^d$ that violate the product rule for (i,S) is at most $2^{\delta_0 d/2 + 2^t}$.*[4]

Proof. Suppose the lemma is false for some (i,S). Then, by the pigeonhole principle there exists some Δ such that the number of elements $y \in \{0,1\}^d$ that violate the product rule for (i,S) with distinguisher Δ is at least $2^{\delta_0 d/2}$. Let BY denote the set of these elements. Identify BY with the uniform distribution over the set BY.

Let $BY'_i = C(BY, i)$, $BY''_i = \mathsf{Raz}(X, BY'_i)$ and $BY'''_i = \mathsf{CBA}(BY, BY''_i, i)$. For a subset $T \subseteq [A]$ Let BY'_T denote the sub-table of BY' corresponding to the rows of T, and similarly BY''_T and BY'''_T. Since for every $y \in BY$, we have that

$$\Delta\left(BY'''_i(y), BY'''_S(y)\right) \;\not\approx_{p_2}\; \Delta(U_1, BY'''_S(y)),$$

this holds also on average, that is

$$\Delta\left(BY'''_i, BY'''_S\right) \;\not\approx_{p_2}\; \Delta(U_1, BY'''_S).$$

Thus, it follows that

$$BY'''_{S\cup\{i\}} \;\not\approx_{p_2}\; U_1 \times BY'''_S. \tag{3}$$

On the other hand, when we condition on the values of $\mathcal{H} = BY'_{S\cup\{i\}}$, the conditions for the correlation-breaker with advice hold:

- BY and $BY''_{S\cup\{i\}}$ are independent given $\mathcal{H} = BY'_{S\cup\{i\}}$, since $\mathcal{H}$ is a function of BY alone, and given that $\mathcal{H} = BY'_{S\cup\{i\}} = h$ for some h, $BY''_{S\cup\{i\}}$ is a function of X alone.

[4] We could have used an alternative argument that avoids the 2^{2^t} factor here by a minor deterioration in the error of the CBA. However, since the t we use is constant the 2^{2^t} factor is negligible.

- It holds that

$$\begin{aligned}\tilde{H}_\infty(BY|\mathcal{H}) &\geq H_\infty(BY) - \log(|\mathsf{Supp}(\mathcal{H})|)\\ &= H_\infty(BY) - td' \geq \frac{\delta_0 d}{2} - td' \geq \frac{\delta_0 d}{4},\end{aligned}$$

because

$$\frac{td'}{d} = t \cdot \left(\frac{2}{3}\right)^s = \frac{\delta_0}{4}.$$

Now, since $k_{\mathsf{CBA}} = \frac{\delta_0 d}{8}$ and $\varepsilon_{\mathsf{CBA}} = \frac{1}{d}$ we also have for d large enough,

$$\tilde{H}_\infty(BY|\mathcal{H}) \;\geq\; \frac{\delta_0 d}{4} \;\geq\; k_{\mathsf{CBA}} + \log\frac{1}{\varepsilon_{\mathsf{CBA}}}.$$

- $B_i(y) = 0$, hence $BY'_i \in GZ$ and $BY''_i = \mathsf{Raz}(X, BY'_i)$ is $\varepsilon_{\mathsf{Raz}}$-close to uniform.

Thus, by the correlation-breaker with advice property,

$$\left(\mathsf{CBA}(BY, BY''_i, i), \left\{\mathsf{CBA}(BY, BY''_j, j)\right\}_{j\in S}\right) \;\approx_{\varepsilon_{\mathsf{Raz}}+2\varepsilon_{\mathsf{CBA}}}\; \left(U_1, \left\{\mathsf{CBA}(BY, BY''_j, j)\right\}_{j\in S}\right),$$

or, equivalently,

$$(BY'''_i, BY'''_S) \;\approx_{p_2}\; U_1 \times BY'''_S,$$

in contradiction to Equation (3). ◀

▶ **Definition 35.** *Say $y \in \{0,1\}^d$* violates the product rule *if it violates it for some $i \in [A]$ and $S \subseteq [A] \setminus \{i\}$ of cardinality $t-1$.*

As $H_\infty(Y) \geq \delta_0 d$, the probability $y \in Y$ violates the product rule for a specific (i, S) is at most $2^{\delta_0 d/2 + 2^t - \delta_0 d} = 2^{2^t - \delta_0 d/2}$. Let $p_{1,2} = \frac{\varepsilon_0}{10}$. Then, by the union bound, for d large enough:

▶ **Corollary 36.** $\Pr_{y\in Y}[y \text{ violates the product rule}] \leq 2^{2^t - \delta_0 d/2} \cdot A^t \leq p_{1,2}$.

Part 3: Completing the proof

▶ **Definition 37.** *We say y is* bad *if it has many bad rows or if it violates the product rule. If y is not bad we say it is* good.

Let $p_1 = p_{1,1} + p_{1,2}$. Clearly, by Claim 31 and Corollary 36, $\Pr_{y\in Y}[y \text{ is bad}] \leq p_1 = \left(\frac{1}{4} + \frac{1}{10}\right)\varepsilon_0$.

▷ Claim 38. Fix any good $y \in Y$. Then, $Y'''(y)$ is a (q, t, tp_2) non-oblivious bit-fixing source, for $q = \sqrt{\varepsilon_{\mathsf{cond}}}A$.

Proof. Let $Q(y) \subseteq [A]$ be the set of bad rows for y. As y does not have many bad rows, $|Q(y)| = \sum_{i\in[A]} B_i(y) \leq \sqrt{\varepsilon_{\mathsf{cond}}}A = q$.

Now, fix any set $S \subseteq [A] \setminus Q(y)$ of cardinality t. Let $i \in S$. As $S \subseteq [A] \setminus Q(y)$ and $i \in S$ we have $i \notin Q(y)$ and therefore $B_i(y) = 0$. Also, y does not violate the product rule, hence,

$$\left(Y'''_i(y), Y'''_{S\setminus\{i\}}(y)\right) \;\approx_{p_2}\; U_1 \times Y'''_{S\setminus\{i\}}(y).$$

As this is true for any $i \in S$, by Lemma 18,

$$Y'''_S(y) \;\approx_{tp_2}\; U_t.$$

Thus, $Y'''(y)$ is a (q, t, tp_2) non-oblivious bit-fixing source. ◁

In particular, by Lemma 20, for every good y, $Y'''(y)$ is $tA^t p_2$-close to a (q,t) non-oblivious bit-fixing source. By the choices we have made above $q = \sqrt{\varepsilon_{\mathsf{cond}}} A \leq A^{1-2\gamma}$. Equation (1) implies that

$$t \;=\; \frac{\delta_0}{4}\left(\frac{1}{\varepsilon_{\mathsf{cond}}}\right)^{\log(3/2)} \;\geq\; c_\gamma \log^4 A.$$

Using the resiliency of f from Theorem 22 (and the fact that it is almost balanced), the output when y is good is p_3-close to uniform for $p_3 = tA^t p_2 + \varepsilon_f + A^{-1/c_\gamma}$, where the first term is due to the distance from a t-wise distribution, the second is due to the resiliency and the third is due to the bias of f (see, e.g., Lemma 2.11 in [9]). To that we also have to add the probability p_1 that y is not good. To finish the proof we notice that:

- It holds that

$$\varepsilon_f \;\leq\; c_\gamma \frac{q}{A^{1-\gamma}} \;\leq\; c_\gamma \frac{A^{1-2\gamma}}{A^{1-\gamma}} \;=\; c_\gamma A^{-\gamma} \;\leq\; c_\gamma 2^{-\gamma \log \frac{1}{\varepsilon_{\mathsf{cond}}}} \;\leq\; \frac{\varepsilon_0}{5},$$

  because $\varepsilon_{\mathsf{cond}} \leq (\frac{\varepsilon_0}{5c_\gamma})^{1/\gamma}$.
- Also,

$$A^{-1/c_\gamma} \;\leq\; 2^{-\frac{1}{c_\gamma}\log\frac{1}{\varepsilon_{\mathsf{cond}}}} \;=\; {\varepsilon_{\mathsf{cond}}}^{1/c_\gamma} \;\leq\; \frac{\varepsilon_0}{5},$$

  because $\varepsilon_{\mathsf{cond}} \leq (\frac{\varepsilon_0}{5})^{c_\gamma}$.
- Finally, $tp_2 = t(\varepsilon_{\mathsf{Raz}} + 2\varepsilon_{\mathsf{CBA}})$, $\varepsilon_{\mathsf{Raz}} = 2^{-\Omega(k)} = d^{-\Omega(1)}$, $\varepsilon_{\mathsf{CBA}} = \frac{1}{d}$. Thus, $tp_2 \leq 4td^{-\Omega(1)}$. A and t are constants, so for d large enough, $tA^t p_2 \leq \frac{\varepsilon_0}{5}$.

Together, the error is at most $p_1 + p_3 \leq \varepsilon_0$ completing the proof of the theorem. ◀

5 Strong Seeded Dispersers and Friends

5.1 Strong Seeded Dispersers

▶ **Definition 39** (strong disperser). *A function* $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a strong* (k,ε) *disperser, if for every* (n,k) *source* X*,*

$$|\mathsf{Supp}((Y, \mathsf{Disp}(X,Y)))| > (1-\varepsilon)DM.$$

We say Disp *is (source) linear if for every* $y \in \{0,1\}^d$ *and every* $x_1, x_2 \in \mathbb{F}_2^n$, $\mathsf{Disp}(x_1+x_2, y) = \mathsf{Disp}(x_1, y) + \mathsf{Disp}(x_2, y)$.

We are interested in the important special case where $m = 1$. In this case, non-explicitly, a random function is (w.h.p.) a strong (k,ε) disperser with $d = \log n + \log(\frac{1}{\varepsilon}) + O(1)$ provided that $k \geq \log\log(\frac{1}{\varepsilon}) + O(1)$ [37, 33]. A matching lower bound, up to additive constant factors, was given by [37].

Using the translation between strong seeded dispersers and erasure list-decodable codes which we discuss in Section 5.3, Guruswami and Indyk's result [26] gives a probabilistic polynomial time algorithm that outputs with high probability a strong seeded disperser with seed-length $d = 2\log(\frac{1}{\varepsilon}) + \log n + \log\log(\frac{1}{\varepsilon})$ and optimal entropy-loss. The construction can be made deterministic, but with running time exponential in $1/\varepsilon$. See Table 1 for a summary of previous results.

Note that as we discuss the one output bit case, the required entropy is essentially the *entropy-loss*. From Theorem 27 we can derive a better explicit construction of a strong disperser with small error.

Table 1 Parameters of strong (k,ε) one-bit dispersers, up to additive $O(1)$ terms. γ is an arbitrarily small positive constant.

	Required entropy k	Seed length d	
Lower-bound, non-explicit	$\log\log(\frac{1}{\varepsilon})$	$\log(\frac{1}{\varepsilon})+\log n$	[37, 33]
[26]	$\log\log(\frac{1}{\varepsilon})$	$(2+\gamma)\log(\frac{1}{\varepsilon})+\log n$	Constant ε, or randomized construction
This work (Theorem 40)	$O(\log\log\frac{1}{\varepsilon})$	$(1+\gamma)\log(\frac{1}{\varepsilon})$	poly$(1/n)$ error

▶ **Theorem 40.** *For every constant $0<\gamma<1$ there exists a constant $c\geq 1$ such that for every integer n and $\varepsilon \leq n^{-\frac{c}{1-\gamma}}$ there exists an explicit strong (k,ε) disperser* $\mathsf{Disp}:\{0,1\}^n\times\{0,1\}^d\to\{0,1\}$ *where $d=(1+\gamma)\log(\frac{1}{\varepsilon})$ and $k=c\log d$.*

Proof. Set $\varepsilon_0=\frac{1}{4}$ and $\delta_0=\frac{\gamma}{1+\gamma}$. Let c be the constant from Theorem 27 for δ_0 and ε_0 and let $\mathsf{2Ext}:[N]\times[D]\to\{0,1\}$ be the $((n,k),(d,k_2=\delta_0 d),\varepsilon_0)$ two-source extractor where $d=(1+\gamma)\log(\frac{1}{\varepsilon})$ and $k=c\log d$. Notice that $d\geq c\log n$ (because $\varepsilon\leq n^{-\frac{c}{1+\gamma}}$) as required. Let $\mathsf{Disp}(x,y)=\mathsf{2Ext}(x,y)$.

Let $X\subseteq[N]$ be a set of size K and call a value $y\in[D]$ *b-bad* if $\mathsf{Disp}(X,y)=\{b\}$. It follows that the sets of 0-bad y-s and 1-bad y-s are each of size less than K_2. Therefore,

$$|\mathsf{Supp}\big((U_d,\mathsf{Disp}(X,U_d))\big)| \;>\; 2K_2+2(D-2K_2) \;=\; 2D-2K_2 \;=\; \left(1-\frac{K_2}{D}\right)2D \;=\; (1-\varepsilon)2D,$$

because $\frac{K_2}{D}=2^{-(1-\delta_0)d}=2^{-\frac{1}{1+\gamma}d}=2^{-\log(\frac{1}{\varepsilon})}=\varepsilon$. ◀

5.2 Non-strong dispersers

We now prove Theorem 5 and output more bits from the source at the expense of being strong in only most of the seed. We construct

$$\mathsf{Disp}:\{0,1\}^n\times\{0,1\}^{d_1}\times\{0,1\}^{d_2}\to\{0,1\}^m,$$

where we think of d_1 and d_2 as two parts of the seed. Disp will be strong in the first d_1 bits of the seed. Using the notations of Section 4 we let

$$\mathsf{Disp}(x,y,i)=\big(y,\mathsf{Raz}(x,C(y,i))\big).$$

We now prove (in sketch) Theorem 5.

Proof. We adopt the notations of Section 4. In those notations, $\mathsf{Disp}(X,Y,I)=(Y,Y''_I)$. First note that the length of $i\in\{0,1\}^{d_2}$ is the logarithm of the number of rows in the table Y'' which is $a=O(1)$. By Claim 31 we know that for nearly every $y\in\{0,1\}^{d_1}$ we have many values $i\in\{0,1\}^{d_2}$ such that $\mathsf{Raz}(X,C(y,i))$ is $\varepsilon_{\mathsf{Raz}}$-close to uniform. In particular, for every y that has many good rows, let i_y be any such row. Then,

$$\begin{aligned}|\mathsf{Supp}(\mathsf{Disp}(X,U_{d_1},U_{d_2}))| &\geq \sum_{y \text{ has many good rows}} |\mathsf{Supp}(\mathsf{Disp}(X,y,i_y))| \\ &\geq \sum_{y \text{ has many good rows}} (1-\varepsilon_{\mathsf{Raz}})2^{d''}.\end{aligned}$$

The theorem now follows since $d''=\Omega(k)$ and $\varepsilon_{\mathsf{Raz}}$ is smaller than $2^{-\Omega(d'')}$, which implies that we can truncate the output of Raz such that when $\mathsf{Raz}(X,C(y,i))$ is $\varepsilon_{\mathsf{Raz}}$-close to uniform it covers its entire support. ◀

Table 2 Parameters of $(\bar{n}, N)_2$ codes, $((1-\varepsilon)\bar{n}, L)$ erasure list-decodable, up to constant multiplicative factors. γ is an arbitrarily small positive constant.

	Rate $R = n/\bar{n}$	List size L	
Lower-bound, non-explicit	ε	$\log(\frac{1}{\varepsilon})$	[24]
[26]	$\frac{\varepsilon^2}{\log(1/\varepsilon)}$	$\log(\frac{1}{\varepsilon})$	Constant ε, or randomized construction
This work (Theorem 46)	$\varepsilon^{1+\gamma}$	$\log^{O(1)}(\frac{1}{\varepsilon})$	poly$(1/n)$ error

5.3 Erasure List-Decodable Codes

An $(\bar{n}, n)$ (binary) code is a mapping $\mathcal{C} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$. The code $\mathcal{C}$ is *linear* if $\mathcal{C}$ is linear, and is denoted by $[\bar{n}, n]$. We identify a code with the image of $\mathcal{C}$. For a linear $\mathcal{C}$ this image is a linear subspace of $\mathbb{F}_2^{\bar{n}}$ of dimension n. A generator matrix for an $[\bar{n}, n]$ code $\mathcal{C}$ is any matrix whose columns form a basis for $\mathcal{C}$. In the *erasures* noise model, an adversarially chosen subset of the codeword's symbols are erased and the positions where erasures have occurred are known.

▶ **Definition 41** (erasure list-decodable code)**.** *A code $\mathcal{C} \subseteq \{0,1\}^{\bar{n}}$ is (s, L) erasure list-decodable if for every $r \in \{0,1\}^{\bar{n}-s}$ and every set $T \subseteq [\bar{n}]$ of size $\bar{n} - s$,*

$$\left|\left\{c \in \mathcal{C} \;\middle|\; c|_T = r\right\}\right| \;<\; L,$$

where $c|_T$ denotes the projection of c to the coordinates in T.

The following folklore lemma (see, e.g., [24, Lemma 1]) gives an alternative characterization of *linear* erasure list-decodable codes.

▶ **Lemma 42.** *An $[\bar{n}, n]_2$ linear code $\mathcal{C}$ is $((1-\varepsilon)\bar{n}, L)$ erasure list-decodable if and only if its $\bar{n} \times n$ generator matrix G has the property that every $\varepsilon\bar{n} \times n$ sub-matrix of G has rank greater than $n - \log L$.*

Non-explicitly, we have:

▶ **Theorem 43** ([24])**.** *For every n and $\varepsilon > 0$, there exists an $(\bar{n}, n)$ binary code that is $((1-\varepsilon)\bar{n}, L)$-erasure list-decodable of rate $\frac{n}{\bar{n}} = \Omega(\varepsilon)$ and $L = O(\log(1/\varepsilon))$.*

See Table 2 for a summary of previous results.

Guruswami [25] observed that strong dispersers can be used to construct erasure list-decodable codes. Here we complement his argument, and note that strong dispersers are *equivalent* to erasure list-decodable codes. Given a function $\mathsf{Disp} : [N] \times [D] \to \{0,1\}$, we consider the (D, n) code $\mathcal{C}_{\mathsf{Disp}} : \{0,1\}^n \to \{0,1\}^D$ defined by $\mathcal{C}_{\mathsf{Disp}}(x)_i = \mathsf{Disp}(x, i)$. Note that the code is linear if and only if Disp is linear.

▶ **Lemma 44** (following [25, Lemma 12])**.** *The function $\mathsf{Disp} : [N] \times [D] \to \{0,1\}$ is a strong (k, ε) disperser if and only if $\mathcal{C}_{\mathsf{Disp}}$ is $((1-2\varepsilon)D, K)$ erasure list-decodable.*

Proof. For one direction, assume Disp is a strong (k, ε) disperser. We wish to prove that $\mathcal{C}_{\mathsf{Disp}}$ is $((1-2\varepsilon)D, K)$ erasure list-decodable. Let $T = \{t_1, \ldots, t_{2\varepsilon D}\} \subseteq [D]$ be an arbitrary set of size $2\varepsilon D$ and $r \in \{0,1\}^{2\varepsilon D}$ an arbitrary string. Let $X_{T,r} \subseteq \{0,1\}^n$ denote the set of all the messages x for which $\mathcal{C}_{\mathsf{Disp}}(x)|_T = r$. Then,

$$\left|\mathsf{Supp}\big((U_d, \mathsf{Disp}(X_{T,r}, U_d))\big)\right| \;\le\; |T| \cdot 1 + (D - |T|) \cdot 2 \;\le\; (1-\varepsilon)2D,$$

where the first inequality follows by considering seeds in T and seeds in $[D] \setminus T$. For a seed $t_i \in T$ we have that $\mathsf{Disp}(X_{T,r}, t_i)$ is fixed, hence each such seed contributes 1 to the support size. For any other seed y, the support size of $\mathsf{Disp}(X_{T,r}, y)$ is at most 2. As Disp is a strong (k,ε) disperser, we conclude that $|X_{T,r}| \leq K$ as desired.

For the other direction assume Disp is a not a strong (k,ε) disperser. Then, there exists a set $X \subseteq \{0,1\}^n$ such that $|X| \geq K$ and $|\mathsf{Supp}\big((U_d, \mathsf{Disp}(X, U_d))\big)| \leq (1-2\varepsilon)2D$. Note that for every $y \in [D]$ we have $|\mathsf{Supp}(\mathsf{Disp}(X,y))| \in \{1,2\}$. Therefore, following the above calculation, there exists a set $T \subseteq D$ of size at least $2\varepsilon D$ such that for each $y \in T$, $|\mathsf{Supp}(\mathsf{Disp}(X,y))| = 1$. But this means that for every $x \in X$, $\mathcal{C}_{\mathsf{Disp}}(x)|_T$ is the same (punctured) codeword. It follows that $\mathcal{C}_{\mathsf{Disp}}$ is not $((1-2\varepsilon)D, K)$ erasure list-decodable. ◀

▶ **Corollary 45.** *If* $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ *is a strong* (k,ε) *disperser with seed-length* $d = a_1 \log n + a_2 \log(\frac{1}{\varepsilon}) + a_3$ *(for some* $a_1 \geq 1, a_2 \geq 1$ *and* a_3*) then* $\mathcal{C}_{\mathsf{Disp}}$ *is a* $((1-2\varepsilon)D, K)$ *erasure list-decodable code of rate* $2^{-a_3} \cdot n^{1-a_1} \cdot \varepsilon^{a_2}$.

When ε is much smaller than $\frac{1}{n}$ the dominant factor is determined by a_2. As we mentioned earlier (and as Guruswami also notes in [25]) previous explicit constructions for binary codes had $a_2 \geq 2$ (usually inherited from extractor constructions). Our construction is the first to get arbitrary close to $a_2 = 1$ and small list-size. Combining Corollary 45 and Theorem 40, we obtain:

▶ **Theorem 46.** *For every constant* $0 < \gamma < 1$ *there exists a constant* $c \geq 1$ *such that for every integer* n *and* $\varepsilon \leq n^{-\frac{c}{1-\gamma}}$ *there exists an explicit code* $\mathcal{C} : \{0,1\}^n \to \{0,1\}^{(\frac{1}{\varepsilon})^{1+\gamma}}$ *that is*

$$\left((1-2\varepsilon)\left(\frac{1}{\varepsilon}\right)^{1+\gamma}\ ,\ \left((1+\gamma)\log\frac{1}{\varepsilon}\right)^{c}\right)$$

erasure list-decodable of rate $n\varepsilon^{1+\gamma}$.

5.4 Ramsey Graphs

Ramsey theory studies inevitable order that appears in large structures. It was initiated by Ramsey [38], who showed that any graph over $N = 2^n$ vertices must contain a clique or an independent set of size $n/2$. A graph over N vertices is called *K-Ramsey* if it contains neither a clique nor an independent set of size K. Inaugurating the probabilistic method, Erdős [18] showed that there are $2n$-Ramsey graphs. He also offered a bounty of \$100 for an *explicit* construction of an $O(n)$-Ramsey graph.

Erdős's challenge initiated a line of beautiful constructions of Ramsey graphs [1, 34, 19, 11, 20]. The study of pseudorandomness gave a new perspective on Ramsey graphs. Specifically, any two-source disperser or extractor gives rise to a bipartite Ramsey graph (and hence, also to a non-bipartite Ramsey graph [41]). This connection led to to new constructions of Ramsey graph [10, 35, 2, 39, 8, 4, 21, 5, 6, 14, 9, 32, 15, 28] culminating in $(N, n^{O(\log\log n/\log\log\log n)})$-Ramsey graphs [7, 29].

In this section we tackle the problem of constructing *unbalanced* Ramsey graphs.

▶ **Definition 47** (Ramsey graph)**.** *A bipartite graph* $\mathsf{Ram} : [N_1] \times [N_2] \to \{0,1\}$ *is a* (K_1, K_2) *bipartite Ramsey graph if every* $K_1 \times K_2$ *induced subgraph of* Ram *is neither a bipartite clique nor a bipartite independent set.*

While it is possible to interpret some pseudorandom objects as unbalanced Ramsey graphs, they were less studied explicitly. See Table 3 for a summary of previous results.

Table 3 Parameters of (K_1, K_2) Ramsey graphs in the unbalanced case, $[N_1 = 2^n] \times [N_2]$. c is any large enough constant and γ is an arbitrarily small positive constant.

	$K_1 : N_1$	$K_2 : N_2$	
Lower-bound	$(c-1)\log n : 2^n$	$n : n^c$	By [37] and Claim 49
Non-explicit	$O(c \log n) : 2^n$	$n : n^c$	Probabilistic method
[39]	$\log^{O(1)} n : 2^n$	$N_2^{0.5+\gamma} : n^{O(1)}$	$O(1)$ terms depend on γ
This work (Theorem 27)	$\log^{O(1)} n : 2^n$	$N_2^{\gamma} : n^{O(1)}$	$O(1)$ terms depend on γ

It is easy to see that a two-source extractor with any nontrivial error is, in fact, a bipartite Ramsey graph, so as a corollary of Theorem 27, we obtain:

▶ **Corollary 48.** *For every integer N_1 and a constant $0 < \delta < 1$ there exists a constant $c = c(\delta) \geq 1$ and an explicit function $\mathsf{Ram} : [N_1] \times [N_2] \to \{0,1\}$ that is a bipartite $(K_1, K_2 = N_2^{\delta})$ Ramsey graph, for $N_2 = \log^c N_1$ and $K_1 = \log^c N_2$.*

We start with the easy claim that bipartite Ramsey graphs are equivalent to strong one-bit dispersers.

▷ Claim 49. If $\mathsf{Ram} : [N_1] \times [N_2] \to \{0,1\}$ is a (K_1, K_2) bipartite Ramsey graph then Ram is a strong $(k_1, \varepsilon \geq \frac{K_2}{N_2})$ disperser with seed-length $n_2 = k_2 + \log(\frac{1}{\varepsilon})$. Also, if Ram is a strong $(k_1, \varepsilon = \frac{K_2}{2N_2})$ disperser then it is a (K_1, K_2) bipartite Ramsey graph.

Proof. The first claim follows from the proof of Theorem 40.

For the other claim, which was already observed in [22], assume Ram is a $(k_1, \varepsilon = \frac{K_2}{2N_2})$ disperser and assume towards contradiction that it is not a $(K_1, K_2 = 2\varepsilon N_2)$ bipartite Ramsey graph. Hence, there exist some $S \subseteq [N_1]$ and $T \subseteq [N_2]$ so that $|S| \geq K_1$ and $|T| \geq K_2$ such that either $\mathsf{Ram}(S,T) = \{0\}$ or $\mathsf{Ram}(S,T) = \{1\}$. Assume w.l.o.g. that $\mathsf{Ram}(S,T) = \{0\}$, so for every $t \in T$, $(t,1) \notin \mathsf{Supp}\big((U_{n_2}, \mathsf{Ram}(S, U_{n_2}))\big)$. But then,

$$|\mathsf{Supp}\big((U_{n_2}, \mathsf{Ram}(S, U_{n_2}))\big)| \;\leq\; 2(N_2 - |T|) + |T| \;\leq\; (1-\varepsilon)2N_2,$$

a contradiction. ◁

As observed in [22], the quality of the Ramsey graph implied by the above theorem crucially depends on the seed-length of the given disperser. Specifically, if the seed-length dependence on the error ε is $2 \cdot \log(\frac{1}{\varepsilon})$ then $K_2 = 2\varepsilon N_2 > \sqrt{N_2}$ and if it is $1 \cdot \log(\frac{1}{\varepsilon})$ then K_2 can be very small.

We mention a more frugal way of obtaining Ramsey graphs from *linear* dispersers. The argument is a straightforward adaptation of an argument of Alon [23, Proposition 10.15].[5] The parameters we obtain are identical to the above claim (and [22]), except that one side of the graph is scaled down (from N to n) as is its entropy (from K to k).

▶ **Theorem 50.** *Suppose $\mathsf{Disp} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}$ is a linear strong (K, ε) disperser. Let G be the $D \times n$ generating matrix of the $[D, n]_2$ linear code $\mathcal{C}_{\mathsf{Disp}}$. Then, G is a $(2\varepsilon D, k+1)$ bipartite-Ramsey-graph.*

[5] Alon's argument is aimed at obtaining *balanced* Ramsey graphs, while we are more concerned with the entropy they can handle.

Proof. Assume Disp is a linear strong (K, ε) disperser. By Lemma 44, $\mathcal{C}_{\mathsf{Disp}}$ is a $((1-2\varepsilon)D, K)$ erasure list-decodable code. Assume towards contradiction that G is not a $(2\varepsilon D, k+1)$ bipartite Ramsey graph. Let M' be a monochromatic $2\varepsilon D \times k+1$ sub-matrix of G. Assume that M' is the all-ones matrix (a similar argument handles the all-zeros matrix). Denote by M the $2\varepsilon D \times n$ sub-matrix of G that is formed by taking the rows of M' and all columns of G. On the one hand, by Lemma 44 and Lemma 42, $\mathrm{rank}(M) > n - \log K = n - k$. On the other hand, as M contains $k+1$ columns of rank 1, $\mathrm{rank}(M) \leq n - k$, a contradiction. ◀

It is natural to ask whether the other direction also holds, namely whether an adjacency matrix of a bipartite Ramsey graph is in fact a generating matrix of a linear, erasure list-decodable code. Stated differently, whether a low-rank matrix must contain large monochromatic rectangles. That question received much attention, as it is tightly related to the famous "log-rank conjecture" in communication complexity [30, 36]. Unfortunately, the acclaimed unconditional upper bound of Lovett [31] still does not give us a meaningful result.

6 Concluding Remarks and Open Problems

- The strong disperser we construct in this paper outputs one bit, and for $k = O(\log\log\frac{1}{\varepsilon})$,
  - has $O(\log\log\frac{1}{\varepsilon})$ entropy-loss, and,
  - $(1+\gamma)\cdot\log(\frac{1}{\varepsilon})$ dependence of the seed-length on the error.

  It is natural to ask to extend the results of the paper to arbitrarily large values of k, matching (up to multiplicative factors) the non-explicit results.
- Our dispersers are inherently non-linear, and therefore we also get non-linear erasure list-decodable codes. How can we obtain near optimal *linear* codes?
- The erasure list-decodable code we construct is explicit in the sense that the code can be efficiently encoded. Does it also admit an efficient erasure list-*decoding* algorithm?
- The seed-length of our strong disperser is $c\log n + \log(\frac{1}{\varepsilon})$. Pushing c closer to 1 is an important open problem. In particular it would imply erasure list-decodable codes of near-optimal rate even for relatively large ε. Such a disperser with many output bits can also be used for simulating one-sided error randomized algorithms using weak random sources with nearly linear overhead [44].

References

1 HL Abbott. Lower bounds for some Ramsey numbers. *Discrete Mathematics*, 2(4):289–293, 1972.

2 Noga Alon. The Shannon capacity of a union. *Combinatorica*, 18(3):301–310, 1998.

3 Noga Alon, Oded Goldreich, and Yishay Mansour. Almost k-wise independence versus k-wise independence. *Information Processing Letters*, 88(3):107–110, 2003.

4 Boaz Barak. A simple explicit construction of an $n^{\tilde{o}(\log n)}$-Ramsey graph. *arXiv preprint*, 2006. `arXiv:math/0601651`.

5 Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. *Journal of the ACM (JACM)*, 57(4):20, 2010.

6 Boaz Barak, Anup Rao, Ronen Shaltiel, and Avi Wigderson. 2-source dispersers for $n^{o(1)}$ entropy, and Ramsey graphs beating the frankl-wilson construction. *Annals of Mathematics*, 176(3):1483–1544, 2012.

7 Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. An efficient reduction from two-source to nonmalleable extractors: achieving near-logarithmic min-entropy. *SIAM Journal on Computing*, pages STOC17–31–STOC17–49, 2019.

8 Jean Bourgain. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory*, 1(01):1–32, 2005.

9 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. *Annals of Mathematics*, 189(3):653–705, 2019.

10 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

11 Fan RK Chung. A note on constructive methods for Ramsey numbers. *Journal of Graph Theory*, 5(1):109–113, 1981.

12 Gil Cohen. Local correlation breakers and applications to three-source extractors and mergers. *SIAM Journal on Computing*, 45(4):1297–1338, 2016.

13 Gil Cohen. Non-malleable extractors-new tools and improved constructions. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

14 Gil Cohen. Two-source dispersers for polylogarithmic entropy and improved Ramsey graphs. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 278–284, 2016.

15 Gil Cohen. Two-source extractors for quasi-logarithmic min-entropy and improved privacy amplification protocols. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, page 114, 2016.

16 Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. *SIAM Journal on Computing*, 42(6):2305–2328, 2013.

17 Zeev Dvir and Avi Wigderson. Kakeya sets, new mergers, and old extractors. *SIAM Journal on Computing*, 40(3):778–792, 2011.

18 Paul Erdös. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53(4):292–294, 1947.

19 Peter Frankl. A constructive lower bound for Ramsey numbers. *Ars Combinatoria*, 3(297-302):28, 1977.

20 Peter Frankl and Richard M. Wilson. Intersection theorems with geometric consequences. *Combinatorica*, 1(4):357–368, 1981.

21 Parikshit Gopalan. Constructing Ramsey graphs from boolean function representations. In *Proceedings of the 21st Annual IEEE Conference on Computational Complexity (CCC 2006)*, pages 14–pp, 2006.

22 Ronen Gradwohl, Guy Kindler, Omer Reingold, and Amnon Ta-Shma. On the error parameter of dispersers. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 294–305. Springer, 2005.

23 Venkatesan Guruswami. *List decoding of error-correcting codes*. PhD thesis, Massachusetts Institute of Technology, 2001.

24 Venkatesan Guruswami. List decoding from erasures: Bounds and code constructions. *IEEE Transactions on Information Theory*, 49(11):2826–2833, 2003.

25 Venkatesan Guruswami. Better extractors for better codes? In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC 2004)*, pages 436–444, 2004.

26 Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 812–821, 2002.

27 Xin Li. Three-source extractors for polylogarithmic min-entropy. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2015)*, pages 863–882. IEEE, 2015.

28 Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC 2017)*, pages 1144–1156, 2017.

29 Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

30 Shachar Lovett. Recent advances on the log-rank conjecture in communication complexity. *Bulletin of EATCS*, 1(112), 2014.

31 Shachar Lovett. Communication is bounded by root of rank. *Journal of the ACM (JACM)*, 63(1):1, 2016.

32 Raghu Meka. Explicit resilient functions matching Ajtai-Linial. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1132–1148, 2017.

33 Raghu Meka, Omer Reingold, and Yuan Zhou. Deterministic coupon collection and better strong dispersers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

34 Zs Nagy. A constructive estimation of the Ramsey numbers. *Mat. Lapok*, 23:301–302, 1975.

35 Moni Naor. Constructing Ramsey graphs from small probability spaces. *IBM Research Report RJ*, 8810, 1992.

36 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995.

37 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

38 FP Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 2(1):264–286, 1930.

39 Ran Raz. Extractors with weak random seeds. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005)*, pages 11–20, 2005.

40 Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the EATCS*, 77(67-95):10, 2002.

41 Ronen Shaltiel. An introduction to randomness extractors. In *International Colloquium on Automata, Languages, and Programming*, pages 21–41. Springer, 2011.

42 Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM (JACM)*, 48(4):860–879, 2001.

43 Avi Wigderson. Randomness extractors – applications and constructions. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

44 David Zuckerman. Simulating BPP using a general weak random source. *Algorithmica*, 16(4):367–391, 1996.

45 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

A Quadratic Lower Bound for Algebraic Branching Programs

Prerona Chatterjee
Tata Institute of Fundamental Research, Mumbai, India
prerona.chatterjee.tifr@gmail.com

Mrinal Kumar
Department of Computer Science & Engineering, IIT Bombay, India
mrinal@cse.iitb.ac.in

Adrian She
Department of Computer Science, University of Toronto, Canada
ashe@cs.toronto.edu

Ben Lee Volk
Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA, USA
benleevolk@gmail.com

Abstract

We show that any Algebraic Branching Program (ABP) computing the polynomial $\sum_{i=1}^{n} x_i^n$ has at least $\Omega(n^2)$ vertices. This improves upon the lower bound of $\Omega(n \log n)$, which follows from the classical result of Baur and Strassen [24, 1], and extends the results of Kumar [13], which showed a quadratic lower bound for *homogeneous* ABPs computing the same polynomial.

Our proof relies on a notion of depth reduction which is reminiscent of similar statements in the context of matrix rigidity, and shows that any small enough ABP computing the polynomial $\sum_{i=1}^{n} x_i^n$ can be depth reduced to essentially a homogeneous ABP of the same size which computes the polynomial $\sum_{i=1}^{n} x_i^n + \varepsilon(\mathbf{x})$, for a structured "error polynomial" $\varepsilon(\mathbf{x})$. To complete the proof, we then observe that the lower bound in [13] is robust enough and continues to hold for all polynomials $\sum_{i=1}^{n} x_i^n + \varepsilon(\mathbf{x})$, where $\varepsilon(\mathbf{x})$ has the appropriate structure.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Branching Programs, Lower Bound

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.2

Related Version Full version available at `https://eccc.weizmann.ac.il/report/2019/170/`.

Funding *Prerona Chatterjee*: Research supported by the Department of Atomic Energy, Government of India, under project no. 12-R&D-TFR-5.01-0500.

Acknowledgements We are thankful to Ramprasad Saptharishi for helpful discussions at various stages of this work. A part the second author's work was done during a postdoctoral stay at University of Toronto.

1 Introduction

Proving that there are explicit polynomials which are hard to compute is the template of many open problems in algebraic complexity theory. Various instances of this problem involve different definitions of explicitness, hardness and computation.

In the most general form, this is the well known VP vs. VNP question, which asks whether every "explicit" polynomial has a polynomial-size algebraic circuit. An algebraic circuit is a very natural (and the most general) algebraic computational model. Informally, it is a computational device which is given a set of indeterminates $\{x_1, \ldots, x_n\}$, and it

can use additions and multiplications (as well as field scalars) to compute a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$. The complexity of the circuit is then measured by the number of operations the circuit performs.

It is trivial to give an explicit n-variate polynomial which requires circuits of size $\Omega(n)$. It is also not hard to show that a degree-d polynomial requires circuits of size $\Omega(\log d)$, since the degree can at most double in each operation. Thus, one trivially obtains a $\max\{n, \log d\} = \Omega(n + \log d)$ lower bound for an n-variate degree-d polynomial.

A major result of Baur and Strassen [24, 1] gives an explicit n-variate degree-d polynomial which requires circuits of size at least $\Omega(n \cdot \log d)$. On the one hand, this is quite impressive since when $d = \text{poly}(n)$, this gives lower bound which is super-linear in n. Such lower bounds for explicit functions in the analogous model of *boolean* circuits are a long-standing and important open problem in boolean circuit complexity. On the other hand, this lower bound is barely super-linear, whereas ideally one would hope to prove super-polynomial or even exponential lower bounds (indeed, it can be proved that "most" polynomials require circuits of size exponential in n).

Despite decades of work, this lower bound has not been improved, even though it has been reproved (using different techniques [23, 2]). Most of the works thus deal with restricted models of algebraic computation. For some, there exist exponential or at least super-polynomial lower bounds. For other, more powerful models, merely improved polynomial lower bound. We refer the reader to [21] for a comprehensive survey of lower bounds in algebraic complexity.

One such restricted model of computation for which we have better lower bounds is *algebraic formulas*. Formulas are simply circuits whose underlying graph is a tree. Kalorkoti [11] has shown how to adapt Nechiporuk's method [16], originally developed for boolean formulas, to prove an almost quadratic lower bound for an n-variate polynomial. [1] This is also the best lower bound obtainable using this technique.

1.1 Algebraic Branching Programs

Algebraic Branching Programs (ABPs, for short), defined below, are an intermediate model between algebraic formulas and algebraic circuits. To within polynomial factors, algebraic formulas can be simulated by ABPs, and ABPs can be simulated by circuits. It is believed that each of the reverse transformations requires a super-polynomial blow-up in the size (for some restricted models of computation, this is a known fact [17, 19, 20, 7, 10]).

Polynomial families which can be efficiently computed by algebraic branching programs form the complexity class VBP, and the determinant is a complete polynomial for this class under an appropriate notion of reductions. Thus, the famous Permanent vs. Determinant problem, unbeknownst to many, is in fact equivalent to showing super-polynomial lower bound for ABPs. In this paper, we focus on the question of proving lower bounds on the size of algebraic branching programs for explicit polynomial families. We start by formally defining an algebraic branching program.

▶ **Definition 1** (Algebraic Branching Programs). *An Algebraic Branching Program (ABP) is a layered graph where each edge is labeled by an affine linear form and the first and the last layer have one vertex each, called the "start" and the "end" vertex respectively.*

[1] In his paper, Kalorkoti proves an $\Omega(n^3)$ lower bound for the $n \times n$ determinant, which has n^2 variables, so the lower bound not quadratic in the number of variables. However, it is possible to get the statement claimed here using a straightforward application of his techniques.

The polynomial computed by an ABP is equal to the sum of the weights of all paths from the start vertex to the end vertex in the ABP, where the weight of a path is equal to the product of the labels of all the edges on it.

The size of an ABP is the number of vertices in it.

While Definition 1 is quite standard, there are some small variants of it in the literature which we now discuss. These distinctions make no difference as far as super-polynomial lower bounds are concerned, since it can be easily seen that each variant can be simulated by the other to within polynomial factors, and thus the issues described here are usually left unaddressed. However, it seems that we are very far from proving super-polynomial lower bounds for general algebraic branching programs, and in this paper we focus on proving polynomial (yet still super-linear) lower bounds. In this setting, those issues do affect the results.

Layered vs. Unlayered

In Definition 1, we have required the graph to be layered. We also consider in this paper ABPs whose underlying graphs are unlayered, which we call *unlayered ABPs*. We are able to prove super-linear (but weaker) lower bounds for this model as well.

One motivation for considering layered graph as the "standard" model is given by the following interpretation. From the definition, it can be observed that any polynomial computable by an ABP with d layers and ℓ_i vertices in the i-th layer can be written as the (only) entry of the 1×1 matrix given by the product $M := \prod_{i=1}^{d-1} M_i$, where M_i is an $\ell_i \times \ell_{i+1}$ matrix with affine forms as entries. One natural complexity measure of such a representation is the total number of non-zero entries in those matrices, which is the number of edges in the ABP. Another natural measure, which can only be smaller, is the sums of dimensions of the matrices involved in the product, which is the same as the number of vertices in the underlying graph.

Branching programs are also prevalent in boolean complexity theory, and in particular in the context of derandomizing the class RL. In this setting again it only makes sense to talk about layered graphs.

Unlayered ABPs can also be thought of as (a slight generalization of) *skew circuits*. These are circuits in which on every multiplication gate, at least one of the operands is a variable (or more generally, a linear function).

Edge labels

In Definition 1 we have allowed each edge to be labeled by an arbitrary affine linear form in the variables. This is again quite standard, perhaps inspired by Nisan's characterization of the ABP complexity of a non-commutative polynomial as the rank of an associated coefficients matrix [17], which requires this freedom. A more restrictive definition would only allow each edge to be labeled by a linear function in 1 variable. On the other hand, an even more general definition, which we sometimes adopt, is to allow every edge to be labeled by an *arbitrary* polynomial of degree at most Δ. In this case we refer to the model as an ABP with edge labels of degree at most Δ. Thus, the common case is $\Delta = 1$, but our results are meaningful even when $\Delta = \omega(1)$. Note that this is quite a powerful model, which is allowed to use polynomials with super-polynomial standard circuit complexity "for free".

We will recall some of these distinctions in Subsection 1.3, where we discuss previous results, some of which apply to several of the variants discussed here.

1.2 Lower Bounds for Algebraic Branching Programs

Our main result is a quadratic lower bound on the size of any algebraic branching program computing some explicit polynomial.

▶ **Theorem 2.** *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program over $\mathbb{F}$ computing the polynomial $\sum_{i=1}^{n} x_i^n$ is of size at least $\Omega(n^2)$.*

When the ABP's edge labels are allowed to be polynomials of degree at most Δ, our lower bound is $\Omega(n^2/\Delta)$.

Note that there also exists an algebraic branching program for $\sum_{i=1}^{n} x_i^n$ of size $O(n^2/\Delta)$. A rough sketch of the construction is as follows. The ABP essentially consists of n parallel paths from the source vertex to the target vertex, with the i^{th} path computing x_i^n. If the labels on the edges are allowed to have degree $\leq \Delta$, then each path consists of n/Δ edges, with all the edges on the i^{th} path being labelled by x_i^{Δ} (except possibly the last edge, which is labelled by $x_i^{n-\Delta((n/\Delta)-1)}$). This shows that the bound we give here is in fact tight.

In a subsequent version of this paper [5], we use the techniques in the proof of Theorem 2 along with some more ideas to prove an $\Omega(n^2)$ lower bound on the size of algebraic formulas computing the elementary symmetric polynomials on n variables. The lower bound essentially settles the question of formula complexity of elementary symmetric polynomials and is the first (non-trivial) $\Omega(n^2)$ lower bound for an n variate polynomial for algebraic formulas; the prior best bound being a lower bound of $\Omega(n^2/\log n)$ due to Kalorkoti [11]. In fact, it was known that the techniques used in [11] cannot directly give a lower bound better than $\Omega(n^2/\log n)$ for an n variate polynomial.

For the unlayered case, we prove a weaker (but still superlinear) lower bound.

▶ **Theorem 3.** *Let $\mathbb{F}$ be a field and $n \in \mathbb{N}$ such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any unlayered algebraic branching program over $\mathbb{F}$ with edge labels of degree at most Δ computing the polynomial $\sum_{i=1}^{n} x_i^n$ has at least $\Omega(n \log n/(\log\log n + \log \Delta))$ edges.*

1.3 Previous Work

The best lower bound known for ABPs prior to this work is a lower bound of $\Omega(n \log n)$ on the number of edges for the same polynomial $\sum_{i=1}^{n} x_i^n$. This follows from the classical lower bound of $\Omega(n \log n)$ by Baur and Strassen [24, 1] on the number of multiplication gates in any algebraic circuit computing the polynomial $\sum_{i=1}^{n} x_i^n$ and the observation that when converting an ABP to an algebraic circuit, the number of product gates in the resulting circuit is at most the number of edges in the ABP. Theorem 2 improves upon this bound quantitatively, and also qualitatively, since the lower bound is on the number of vertices in the ABP.

For the case of homogeneous ABPs,[2] a quadratic lower bound for the polynomial $\sum_{i=1}^{n} x_i^n$ was shown by Kumar [13], and the proofs in this paper build on the ideas in [13]. In a nutshell, the result in [13] is equivalent to a lower bound for ABPs computing the polynomial $\sum_{i=1}^{n} x_i^n$ when the number of layers in the ABP is at most n. In this work, we generalize this to proving essentially the same lower bound as in [13] for ABPs with an unbounded number of layers.

[2] An ABP is *homogeneous* if the polynomial computed between the start vertex and any other vertex is a homogeneous polynomial. This condition is essentially equivalent to assuming that the number of layers in the ABP is upper bounded by the degree of the output polynomial.

In general, an ABP computing an n-variate homogeneous polynomial of degree poly(n) can be homogenized with a polynomial blow-up in size. This is proved in a similar manner to the standard classical result which shows this statement for algebraic circuits [25]. Thus, much like the discussion following Definition 1, homogeneity is not an issue when one considers polynomial vs. super-polynomial sizes, but becomes relevant when proving polynomial lower bounds. In other contexts in algebraic complexity this distinction is even more sharp. For example, exponential lower bounds for homogeneous depth-3 circuits are well known and easy to prove [18], but strong enough exponential lower bounds for non-homogeneous depth-3 circuits would separate VP from VNP [9].

For *unlayered* ABPs, the situation is more complex. If the edge labels are only functions of one variable, it is possible to adapt Nechiporuk's method [16] in order to obtain a lower bound of $\tilde{\Omega}(n^{3/2})$ (for a different polynomial than we consider). This is an argument attributed to Pudlák and sketched by Karchmer and Wigderson [12] for the boolean model of parity branching programs, but can be applied to the algebraic setting. However, this argument does not extend to the case where the edge labels are arbitrary linear or low-degree polynomials in the n variables. The crux of Nechiporuk's argument is to partition the variables into m disjoint sets, to argue (using counting or dimension arguments) that the number of edges labeled by variables from each set must be somewhat large[3], and then to sum the contributions over all m sets. This is hard to implement in models where a single edge can have a "global" access to all variables, since it is not clear how to avoid over-counting in this case.

As mentioned above, the lower bound of Baur-Strassen does hold in the unlayered case, assuming the edge labels are linear functions in the variables. When we allow edge labels of degree at most Δ for some $\Delta \geq 2$, their technique does not seem to carry over. Indeed, even if we equip the circuit with the ability to compute such low-degree polynomials "for free", a key step in the Baur-Strassen proof is the claim that if a polynomial f has a circuit of size τ, then there is a circuit of size $O(\tau)$ which computes all its first order partial derivatives, and this statement does not seem to hold in this new model.

It is possible to get an $\Omega(n \log n / \log \Delta)$ lower bound for this model, for a different polynomial, by suitably extending the techniques of Ben-Or [2, 3]. Our lower bounds are weaker by at most a doubly-logarithmic factor; however, the techniques are completely different. Ben-Or's proofs rely as a black-box on strong modern results in algebraic geometry, whereas our proofs are much more elementary.

Detereminantal Complexity

Another model of computation in algebraic complexity theory, which is related to the discussions in this paper is the notion of determinantal complexity. Given a polynomial $f \in \mathbb{F}[\mathbf{x}]$ of degree d, the determinantal complexity of f is defined to be the smallest k for which there is a $k \times k$ matrix M with affine forms in $\mathbb{F}[\mathbf{x}]$ as entries, such that $\det(M) = f$.

The best lower bound known on the determinantal complexity was hown by Mignnon and Ressayre [15, 4], where they show an $n/2$ lower bound for an n variate polynomial family. Over the field of real numbers, this bound was improved by Yabe [27], to n.

[3] This is usually guaranteed by constructing a function or a polynomial with the property that given a fixed set S in the partition, there are many subfunctions or subpolynomials on the variables of S that can be obtained by different restrictions of the variables outside of S.

Mahajan and Vinay [14] showed that the $n \times n$ determinant polynomial has an ABP of size $O(n^3)$. Thus, note that a *strong enough* lower bound on the ABP complexity of a polynomial can give a lower bound on its determinantal complexity as well. However, since the lower bound here is only quadratic, nothing non-trivial can be said in this case, even though the techniques here could potentially be useful.

1.4 Proof Overview

The first part in the proof of Theorem 2 is an extension of the lower bound proved in [13] for ABPs with at most n layers. This straightforward but conceptually important adaptation shows that a similar lower bound holds for any polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \varepsilon(\mathbf{x}) \,,$$

where the suggestively named $\varepsilon(\mathbf{x})$ should be thought of as an "error term" which is "negligible" as far as the proof of [13] is concerned. The exact structure we require is that $\varepsilon(\mathbf{x})$ is of the form $\sum_{i=1}^{r} P_i Q_i + R$, where P_i, Q_i are polynomials with no constant term and $\deg(R) \leq n-1$. The parameter r measures the "size" of the error, which we want to keep small, and the lower bound holds if, e.g., $r \leq n/10$.

To argue about ABPs with d layers, with $d > n$, we use a notion of depth reduction which is reminiscent of similar statements in the context of matrix rigidity. We show that unless the size τ of the ABP is too large to begin with (in which case there is nothing to prove), it is possible to find a small set of vertices (of size about $\eta = \tau/d$) whose removal adds a small error term $\varepsilon(\mathbf{x})$ as above with at most η summands, but also reduces the depth of the ABP by a constant factor. Repeatedly applying this operation $O(\log n)$ times eventually gives an ABP of depth at most n while ensuring that we have not accumulated too much "error",[4] so that we can apply the lower bound from the previous paragraph.

In the full proof we have to be a bit more careful when arguing about the ABP along the steps of the proof above. The details are presented in Section 3.

The proof of Theorem 3 follows the same strategy, although the main impediment is that general undirected graphs can have much more complex structure then layered graphs. One of the main ingredients in our proof is (a small variant of) a famous lemma of Valiant [26], which shows that for every graph of depth 2^k with m edges, it is possible to find a set of edges, of size at most m/k, whose removal reduces the depth of the graph to 2^{k-1}. This lemma helps us identify a small set of vertices which can reduce the depth of the graph by a constant factor while again accumulating small error terms.

Interestingly, Valiant originally proved this lemma in a different context, where he showed that linear algebraic circuits of depth $O(\log n)$ and size $O(n)$ can be reduced to a special type of depth-2 circuits (and thus strong lower bounds on such circuits imply super-linear lower bounds on circuits of depth $O(\log n)$). This lemma can be also used to show that *boolean* circuits of depth $O(\log n)$ and size $O(n)$ can be converted to depth-3 circuits of size $2^{o(n)}$, and thus again strong lower bounds on depth-3 circuits will imply super-linear lower bounds on circuits of depth $O(\log n)$. Both of these questions continue to be well known

[4] It takes some care in showing that the total number of error terms accumulated is at most $n/10$ as opposed to the obvious upper bound of $O(n \log n)$. In particular, we observe that the number of error terms can be upper bounded by a geometric progression with first term roughly τ/n and common ratio being a constant less than 1.

open problems in algebraic and boolean complexity, and to the best of our knowledge, our proof is the first time Valiant's lemma is successfully used in order to prove circuit lower bounds for explicit functions or polynomials.

2 Notations and Preliminaries

All logarithms in the paper are base 2.

We use some standard graph theory terminology: If G is a directed graph and (u, v) is an edge, v is called the *head* of the edge and u the *tail*. Our directed graphs are always acyclic with designated source vertex s and sink vertex t. The *depth* of a vertex v, denoted $\mathsf{depth}(v)$, is the length (in edges) of a longest path from s to v. The depth of the graph, denoted by $\mathsf{depth}(G)$, is the depth of t.

For any two vertices u and v in an ABP, the polynomial computed between u and v is the sum of weights of all paths between u and v in the ABP. We denote this by $[u, v]$.

The formal degree of a vertex u in an ABP denoted $\mathrm{fdeg}(u)$, is defined inductively as follows: If s is the start vertex of the ABP, $\mathrm{fdeg}(s) = 0$. If u is a vertex with incoming edges from $u_1, \ldots, u_k$, labeled by non-zero polynomials $\ell_1, \ldots, \ell_k$, respectively, then

$$\mathrm{fdeg}(u) = \max_{i \in [k]} \{\deg(\ell_i) + \mathrm{fdeg}(u_i)\} .$$

It follows by induction that for every vertex u, $\deg([s, u]) \leq \mathrm{fdeg}(u)$ (however, cancellations can allow for arbitrary gaps between the two). Also, note that the formal degree of vertices is monotonically non-decreasing along any path from the source vertex to the sink vertex. The formal degree of the ABP is the maximal formal degree of any vertex in it.

We sometimes denote by $\mathbf{x}$ the vector of variables $(x_1, \ldots, x_n)$, where n is understood from the context. Similarly we use $\mathbf{0}$ to denote the n-dimensional vector $(0, 0, \ldots, 0)$.

2.1 A Decomposition Lemma

The following lemma gives a decomposition of a (possibly unlayered) ABP in terms of the intermediate polynomials it computes. Its proof closely resembles that of Lemma 3.5 of [13]. For completeness we prove it here for a slightly more general model.

▶ **Lemma 4** (Kumar [13]). *Let $\mathcal{B}$ be a (possibly unlayered) algebraic branching program which computes a degree d polynomial $P \in \mathbb{F}[x_1, \ldots x_n]$, and has formal degree d. Further, assume that the edges of $\mathcal{B}$ are labelled by arbitrary polynomials of degree at most Δ, where $1 \leq \Delta \leq d/2$. Set $d' = \lfloor d/\Delta \rfloor$.*

For any $i \in \{1, 2, \ldots, d' - 1\}$, let $S_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,m}\}$ be the set of all vertices in $\mathcal{B}$ whose formal degree is in the interval $[i\Delta, (i + 1)\Delta)$.

Then, there exist polynomials $Q_{i,1}, Q_{i,2}, \ldots, Q_{i,m}$ and R_i, each of degree at most $d - 1$ such that

$$P = \sum_{j=1}^{m} [s, u_{i,j}] \cdot Q_{i,j} + R_i .$$

Proof. Fix i as above and set $S_i = \{u_{i,1}, u_{i,2}, \ldots, u_{i,m}\}$ as above (observe that since each edge label is of degree at most Δ, S_i is non empty). Further suppose, without loss of generality, that the elements of S_i are ordered such that there is no directed path from $u_{i,j}$ to $u_{i,j'}$ for $j' > j$.

Consider the unlayered ABP $\mathcal{B}_1$ obtained from $\mathcal{B}$ by erasing all incoming edges to $u_{i,1}$, and multiplying all the labels of the outgoing edges from $u_{i,1}$ by a new variable y_1. The ABP $\mathcal{B}_1$ now computes a polynomial of the form

$$P'(y_1, x_1, \ldots, x_n) = y_1 \cdot Q_{i,1} + R_{i,1}$$

where $P = P'([s, u_{i,1}], x_1, \ldots, x_n)$. $R_{i,1}$ is the polynomial obtained from $\mathcal{B}_1$ by setting y_1 to zero, or equivalently, removing $u_{i,1}$ and all its outgoing edges. We continue in the same manner with $u_{i,2}, \ldots, u_{i,m}$ to obtain

$$P = \sum_{j=1}^{m} [s, u_{i,j}] \cdot Q_{i,j} + R_i.$$

Indeed, observe that since there is no path from $u_{i,j}$ to $u_{i,j'}$ for $j' > j$, removing $u_{i,j}$ does not change $[s, u_{i,j'}]$. The bound on the degrees of $Q_{i,j}$ is immediate from the fact that the formal degree of the ABP is at most d and $\mathrm{fdeg}(u_{i,j}) \geq 1$. It remains to argue the $\deg(R_i) \leq d - 1$.

The polynomial R_i is obtained from $\mathcal{B}$ by erasing all the vertices in S_i and the edges touching them. We will show that every path in the corresponding ABP computes a polynomial of degree at most $d - 1$. Let $s = v_1, v_2, \ldots, v_r = t$ be such a path, which is also a path in $\mathcal{B}$. Let v_k be the minimal vertex in the path whose degree (in $\mathcal{B}$) is at least $(i+1)\Delta$ (if no such v_k exists, the proposition follows). As $v_{k-1} \notin S_i$, the formal degree of v_{k-1} is at most $i\Delta - 1$. The degree of the polynomial computed by this path is thus at most $i\Delta - 1 + \Delta + D = (i+1)\Delta - 1 + D$, where D is the degree of product of the labels on the path $v_k, v_{k+1}, \ldots, t$. To complete the proof, it remains to be shown that $D \leq d - (i+1)\Delta$.

Indeed, if $D \geq d - (i+1)\Delta + 1$ then since the degree of v_k is at least $(i+1)\Delta$, there would be in $\mathcal{B}$ a path of formal degree at least $(i+1)\Delta + D \geq d + 1$, contradicting the assumption on $\mathcal{B}$. ◀

2.2 Variety and its Dimension

One of the important notions we will use in our proofs is that of an affine algebraic variety. Given a set of polynomials, the algebraic variety defined by these polynomials is defined to be the set of their common zeros. That is, if S is a set of polynomials on n variables over a field $\mathbb{F}$, then

$$V = \{\mathbf{a} \in \mathbb{F}^n \ : \ \forall f \in S, f(\mathbf{a}) = 0\}.$$

Given a variety, an important property that is studied is its dimension. Intuitively, it is an appropriate generalisation of the notion of dimension for linear spaces. We will not be defining it formally here and refer the interested reader to the book by Cox, Little and O'Shea [6]. However we state formally the properties, of dimensions of varieties, that we will be using in our proofs.

▶ **Lemma 5** (Section 2.8 in [22]). *Let S be a set of polynomials in n variables over an algebraically closed field $\mathbb{F}$ such that $|S| \leq n$. Let $V = \mathbb{V}(S)$ be the set of common zeros of polynomials in S. If V is non-empty, then the dimension of V is at least $n - |S|$.*

▶ **Lemma 6** (Chapter 4 in [6]). *Let $\mathbb{F}$ be an algebraically closed field, and let $V_1 \subseteq \mathbb{F}^n$ and $V_2 \subseteq F^n$ be two affine varieties such that $V_1 \subseteq V_2$. Then, the dimension of V_1 is at most the dimension of V_2.*

3 A Lower Bound for Algebraic Branching Programs

In this section we prove Theorem 2. We start by restating it.

▶ **Theorem 7.** *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \nmid n$. If $\mathcal{A}$ is an algebraic branching program with edge labels of degree at most Δ that computes the polynomial $\sum_{i=1}^{n} x_i^n$, then the size of $\mathcal{A}$ is at least*

$$\Omega\left(\frac{n^2}{\Delta}\right).$$

For technical reasons, we work with a slightly more general model which we call *multilayered* ABPs, which we now define.

▶ **Definition 8** (Multilayered ABP). *Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be k ABPs with $d_1, \ldots, d_k$ layers and $\tau_1, \ldots, \tau_k$ vertices, respectively. A multilayered ABP $\mathcal{A}$, denoted by $\mathcal{A} = \sum_{i=1}^{k} \mathcal{A}_i$, is the ABP obtained by placing $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ in parallel and identifying their start and end vertices respectively. Thus, the polynomial computed by $\mathcal{A}$ is $\sum_{i=1}^{k} [\mathcal{A}_i]$, where $[\mathcal{A}_i]$ is the polynomial computed by $\mathcal{A}_i$.*

The number of layers of $\mathcal{A}$ is $d := \max\{d_1, \ldots, d_k\}$. The size of $\mathcal{A}$ is the number of vertices in $\mathcal{A}$, and thus equals

$$|\mathcal{A}| := 2 + \sum_i (\tau_i - 2).$$

This model is an intermediate model between (layered) ABPs and unlayered ABPs: given a multilayered ABP of size τ it is straightforward to construct an unlayered ABP of size $O(\tau)$ which computes the same polynomial.

3.1 A Robust Lower Bound for ABPs of Formal Degree at most *n*

In this section, we prove a lower bound for the case where the formal degree of every vertex in the ABP is at most n. In fact, Kumar [13] has already proved a quadratic lower bound for this case.

▶ **Theorem 9** (Kumar [13]). *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \nmid n$. Then any algebraic branching program of formal degree at most n which computes the polynomial $\sum_{i=1}^{n} x_i^n$ has at least $\Omega(n^2)$ vertices.*

However, to prove Theorem 7, we need the following more "robust" version of Theorem 9, which gives a lower bound for a larger class of polynomials. For completeness, we also sketch an argument for the proof which is a minor variation of the proof of Theorem 9.

▶ **Theorem 10.** *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be field such that $\mathrm{char}(\mathbb{F}) \nmid n$. Let $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x})$, $B_1(\mathbf{x}), \ldots, B_r(\mathbf{x})$ and $R(\mathbf{x})$ be polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R is a polynomial of degree at most $n-1$. Then, any algebraic branching program over $\mathbb{F}$, of formal degree at most n and edge labels of degree at most $\Delta \leq n/10$, which computes the polynomial*

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R$$

has at least $\frac{(n/2-r)n}{2\Delta}$ vertices.

The proof of the theorem follows from Lemma 4 and the following lemma which is a slight generalization of Lemma 3.1 in [13]. We include a proof for completeness.

▶ **Lemma 11.** *Let $n \in \mathbb{N}$, and let $\mathbb{F}$ be an algebraically closed field such that* $\mathrm{char}(\mathbb{F}) \nmid n$. *Let $\{P_1, \ldots, P_m, Q_1, \ldots, Q_m, A_1, \ldots, A_r, B_1, \ldots, B_r\}$ be a set of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ such that the set of their common zeros*

$$\mathcal{V} = \mathbb{V}(P_1, \ldots, P_m, Q_1, \ldots, Q_m, A_1, \ldots, A_r, B_1, \ldots, B_r) \subseteq \mathbb{F}^n$$

is non-empty. Finally, suppose R is a polynomial in $\mathbb{F}[\mathbf{x}]$ of degree at most $n-1$, such that

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j = R + \sum_{i=1}^{m} P_i \cdot Q_i.$$

Then, $m \geq \frac{n}{2} - r$.

Proof. Since $\mathcal{V} \neq \emptyset$, by Lemma 5, $\dim(\mathcal{V}) \geq n - 2m - 2r$. Thus, the set of zeros with multiplicity two of

$$\sum_{i=1}^{n} x_i^n - R = \sum_{i=1}^{m} P_i \cdot Q_i - \sum_{j=1}^{r} A_j \cdot B_j,$$

has dimension at least $n - 2m - 2r$. Now if S is the set of common zeros of the set of all first order partial derivatives of $\sum_{i=1}^{n} x_i^n - R$, $\mathcal{V} \subseteq S$. Thus, by Lemma 6, $\dim(S) \geq n - 2m - 2r$. Up to scaling by n (which is non-zero in $\mathbb{F}$, by assumption), the set of all first order partial derivatives of $\sum_{i=1}^{n} x_i^n - R$ is given by

$$\left\{ x_i^{n-1} - \frac{1}{n} \partial_{x_i} R \right\}_{i \in [n]}.$$

Thus, the statement of this lemma immediately follows from the following claim.

▷ Claim 12 (Lemma 3.2 in [13]). Let $\mathbb{F}$ be an algebraically closed field, and D a positive natural number. For every choice of polynomials $g_1, g_2, \ldots g_n \in \mathbb{F}[\mathbf{x}]$ of degree at most $D-1$, the dimension of the variety

$$\mathbb{V}(x_1^D - g_1, x_2^D - g_2, \ldots, x_n^D - g_n)$$

is zero.

Indeed, the above claim shows that $0 = \dim(S) \geq n - 2m - 2r$, and so $m \geq \frac{n}{2} - r$. This completes the proof of Lemma 11. ◀

We now use Lemma 4 and Lemma 11 to complete the proof of Theorem 10.

Proof of Theorem 10. Let $\mathcal{B}$ be an algebraic branching program of formal degree at most n, edge labels of degree at most $\Delta \leq n/10$, and with start vertex s and end vertex t, which computes

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R.$$

We may assume without loss of generality that $\mathbb{F}$ is algebraically closed, by interpreting $\mathcal{B}$ as an ABP over the algebraic closure of $\mathbb{F}$, if necessary.

Let $n' = \lfloor n/\Delta \rfloor$, fix $k \in \{1, 2, \ldots, n'-1\}$, and let $V_k = \{v_{k,1}, v_{k,2}, \ldots, v_{k,m}\}$ be the set of all vertices in $\mathcal{B}$ whose formal degree lies in the interval $[k\Delta, (k+1)\Delta)$. Letting $P'_j = [s, v_{k,j}]$, by Lemma 4, there exist polynomials $Q'_1, Q'_2, \ldots, Q'_m$ and R', each of degree at most $n-1$ such that

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R = \sum_{j=1}^{m} P'_j \cdot Q'_j + R' .$$

Let α_j, β_j be the constant terms in P'_j, Q'_j respectively. Then by defining

$$P_j = P'_j - \alpha_j \qquad \text{and} \qquad Q_j = Q'_j - \beta_j,$$

we have that

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j = R'' + \sum_{j=1}^{m} P_j \cdot Q_j.$$

Here, $R'' = -R + R' + \sum_{j=1}^{m} (\alpha_j \cdot Q'_j + \beta_j \cdot P'_j + \alpha_j \beta_j)$. We now have that for every i, the constant terms of P_i, Q_i are zero and $\deg(R'') \leq n-1$. Let

$$\mathcal{V} = \mathbb{V}(P_1, \ldots, P_m, Q_1, \ldots, Q_m, A_1, \ldots, A_r, B_1, \ldots, B_r).$$

Then $\mathbf{0} \in \mathcal{V}$, and so $\mathcal{V} \neq \emptyset$. Thus by Lemma 11, we know that $m \geq \frac{n}{2} - r$.

Finally, for $k \neq k' \in \{1, 2, \ldots, n'-1\}$, $V_k \cap V_{k'} = \emptyset$. Thus, the number of vertices in $\mathcal{B}$ must be at least

$$\left(\frac{n}{2} - r\right) \cdot (n'-1) \geq \left(\frac{n}{2} - r\right) \cdot \frac{n}{2\Delta} . \qquad \blacktriangleleft$$

3.2 A lower bound for the general case

The following lemma shows how we can obtain, given an ABP with d layers which computes a polynomial F, a multilayered ABP, whose number of layers is significantly smaller, which computes F plus a small "error term".

▶ **Lemma 13.** *Let $\mathcal{A}$ be an ABP over a field $\mathbb{F}$ with d layers, which computes the polynomial F and has m vertices. Let s and t be the start and end vertices of $\mathcal{A}$ respectively, and let $L = \{u_1, u_2, \ldots, u_{|L|}\}$ be the set of vertices in the ℓ-th layer of $\mathcal{A}$. For every $i \in \{1, 2, \ldots, |L|\}$, let α_i and β_i be the constant terms of $[s, u_i]$ and $[u_i, t]$ respectively. Furthermore, let P_i and Q_i be polynomials such that $[s, u_i] = P_i + \alpha_i$ and $[u_i, t] = Q_i + \beta_i$.*

Then, there is a multilayered ABP $\mathcal{A}'$, with at most $\max\{\ell, d - \ell + 1\}$ layers and size at most $|\mathcal{A}|$ that computes the polynomial

$$F - \sum_{i=1}^{|L|} P_i \cdot Q_i + \sum_{i=1}^{|L|} \alpha_i \cdot \beta_i.$$

Proof. Let $u_1, u_2, \ldots, u_{|L|}$ be the vertices in L as described, so that

$$F = [s, t] = \sum_{i=1}^{|L|} [s, u_i] \cdot [u_i, t] .$$

Further, for every $i \in \{1, 2, \ldots, |L|\}$, $[s, u_i] = P_i + \alpha_i$ and $[u_i, t] = Q_i + \beta_i$, where the constant terms of P_i and Q_i are zero (by definition). Having set up this notation, we can thus express the polynomial F computed by $\mathcal{A}$ as

$$F = [s, t] = \sum_{i=1}^{|L|} (P_i + \alpha_i) \cdot (Q_i + \beta_i) \,.$$

On further rearrangement, this gives

$$F - \left(\sum_{i=1}^{|L|} P_i \cdot Q_i \right) + \left(\sum_{i=1}^{|L|} \alpha_i \cdot \beta_i \right) = \left(\sum_{i=1}^{|L|} \alpha_i \cdot (Q_i + \beta_i) \right) + \left(\sum_{i=1}^{|L|} (P_i + \alpha_i) \cdot \beta_i \right) .$$

This is equivalent to the following expression.

$$F - \left(\sum_{i=1}^{|L|} P_i \cdot Q_i \right) + \left(\sum_{i=1}^{|L|} \alpha_i \cdot \beta_i \right) = \left(\sum_{i=1}^{|L|} \alpha_i \cdot [u_i, t] \right) + \left(\sum_{i=1}^{|L|} [s, u_i] \cdot \beta_i \right) .$$

Now, observe that the polynomial $\sum_{i=1}^{|L|} [s, u_i] \cdot \beta_i$ is computable by an ABP $\mathcal{B}$ with $\ell + 1$ layers, obtained by just keeping the vertices and edges within first ℓ layers of $\mathcal{A}$ and the end vertex t, deleting all other vertices and edges, and connecting the vertex u_i in the ℓ-th layer to t by an edge of weight β_i. Similarly, the polynomial $\sum_{i=1}^{|L|} \alpha_i \cdot [u_i, t]$ is computable by an ABP $\mathcal{C}$ with at most $(d - \ell + 1) + 1$ layers, whose set of vertices is s along the vertices in the layers $\ell, \ell + 1, \ell + 2, \ldots, d$ of $\mathcal{A}$. From the definition of $\mathcal{B}$ and $\mathcal{C}$, it follows that the multilayered ABP $\tilde{\mathcal{A}}$ obtained by taking the sum of $\mathcal{B}$ and $\mathcal{C}$ has at most $\max\{\ell + 1, d - \ell + 2\}$ layers.

We are almost done with the proof of the lemma, except for the upper bound on the number of vertices of the resulting multilayered ABP $\tilde{\mathcal{A}}$, and the fact that the upper bound on the depth is slightly weaker than claimed. Both these issues can be solved simultaneously.

The vertices in L appear in both the ABP $\mathcal{B}$ and the ABP $\mathcal{C}$ and are counted twice in the size of $\tilde{\mathcal{A}}$. However, every other vertex is counted exactly once. Hence,

$$|\mathcal{B}| + |\mathcal{C}| = |\mathcal{A}| + |L| \,. \tag{1}$$

In order to fix this issue, we first observe that the edges between the vertices in the ℓ-th layer of $\mathcal{B}$ and the end vertex t are labeled by $\beta_1, \beta_2, \ldots, \beta_{|L|}$, all of which are field constants. In the following claim, we argue that for ABPs with this additional structure, the last layer is redundant and can be removed.

▷ Claim 14. Let $\mathcal{M}$ be an ABP over $\mathbb{F}$ with $k + 1$ layers and edge labels of degree at most Δ such that the labels of all the edges between the k-th layer of $\mathcal{M}$ and its end vertex are scalars in $\mathbb{F}$. Then, there is an ABP $\mathcal{M}'$ with k layers computing the same polynomial as $\mathcal{M}$, with edge labels of degree at most Δ, such that

$$|\mathcal{M}'| \leq |\mathcal{M}| - |V| \,,$$

where V is the set of vertices in the k-th layer of $\mathcal{M}$.

An analogous statement, with an identical proof, is true if we assume that all edge labels between the first and second layer are scalars in $\mathbb{F}$.

We first use Claim 14 to complete the proof of the lemma. As observed above, the edge labels between the last layer L of $\mathcal{B}$ and its end vertex are all constants. Hence, by Claim 14, there is an ABP $\mathcal{B}'$ which computes the same polynomial as $\mathcal{B}$ such that $|\mathcal{B}'| \leq |\mathcal{B}| - |L|$, and $\mathcal{B}'$ has only ℓ layers. Similarly, we can obtain an ABP $\mathcal{C}'$ with at most $d - \ell + 1$ layers.

We consider the multilayered ABP $\mathcal{A}'$ by taking the sum of $\mathcal{B}'$ and $\mathcal{C}'$. Clearly, the number of layers in $\mathcal{A}'$ is at most $\max\{\ell, d-\ell+1\}$ and the size is at most

$$|\mathcal{A}'| \leq |\mathcal{B}'| + |\mathcal{C}'| \leq (|\mathcal{B}| - |L|) + (|\mathcal{C}| - |L|) \leq |\mathcal{A}| \ .$$

Here, the second inequality follows by Claim 14 and the last one follows by Equation 1. To complete the proof of the lemma, we now prove Claim 14. ◀

Proof of Claim 14. For the proof of the claim, we focus on the k-th and $(k-1)$-st layer of $\mathcal{M}$. To this end, we first set up some notation. Let $\{v_1, v_2, \ldots, v_r\}$ be the set of vertices in the k-th layer of $\mathcal{M}$, $\{u_1, u_2, \ldots, u_{r'}\}$ be the set of vertices in $(k-1)$-st layer of $\mathcal{M}$, and a, b denote the start and the end vertices of $\mathcal{M}$ respectively. Then, the polynomial computed by $\mathcal{M}$, can be decomposed as

$$[a, b] = \sum_{i=1}^{r} [a, v_i] \cdot [v_i, b] \, .$$

Note that (v_i, b) is an edge in the ABP. Similarly, the polynomial $[a, v_i]$ can be written as

$$[a, v_i] = \sum_{j=1}^{r'} [a, u_j] \cdot [u_j, v_i] \, .$$

Combining the two expressions together, we get

$$[a, b] = \sum_{i=1}^{r} [v_i, b] \cdot \left(\sum_{j=1}^{r'} [u_j, v_i] \cdot [a, u_j] \right) \ ,$$

which on further rearrangement, gives us

$$[a, b] = \sum_{j=1}^{r'} \left(\sum_{i=1}^{r} [v_i, b][u_j, v_i] \right) \cdot [a, u_j] \, . \tag{2}$$

From the hypothesis of the claim, we know that for every $i \in [r]$, the edge label $[v_i, b]$ is a field constant, and the edge label $[u_j, v_i]$ is a polynomial of degree at most Δ. Thus, for every $j \in [r']$, the expression $(\sum_{i=1}^{r} [u_i, b][u_j, v_i])$ is a polynomial of degree at most Δ.

This gives us the following natural construction for the ABP $\mathcal{M}'$ from $\mathcal{M}$. We delete the vertices $v_1, v_2, \ldots, v_r$ in $\mathcal{M}$ (and hence, all edges incident to them), and for every $j \in \{1, 2, \ldots, r'\}$, we connect the vertex u_j with the end vertex b using an edge with label $(\sum_{i=1}^{r} [v_i, b][u_j, v_i])$. The upper bound on the size and the number of layers of $\mathcal{M}'$ is immediate from the construction, and that it computes the same polynomial as $\mathcal{M}$ follows from Equation 2. ◁

We now state and prove a simple generalization of Lemma 13 for a multilayered ABP.

▶ **Lemma 15.** *Let $\mathcal{A} = \sum_{i=1}^{m} \mathcal{A}_i$ be a multilayered ABP with d layers over a field $\mathbb{F}$ computing the polynomial F, such that each $\mathcal{A}_i$ is an ABP with d_i layers. Also, let $\ell_{i,j}$ be the number of vertices in the j-th layer of $\mathcal{A}_i$ ($\ell_{i,j} = 0$ if $\mathcal{A}_i$ has fewer than j layers), and $\ell = \min_{j \in (d/3, 2d/3)} \{\sum_{i=1}^{m} \ell_{i,j}\}$.*

Then, there is a multilayered ABP with at most $2d/3$ layers and size at most $|\mathcal{A}|$ that computes a polynomial of the form

$$F - \sum_{i=1}^{\ell} P_i \cdot Q_i + \delta \, ,$$

where $\{P_1, \ldots, P_\ell, Q_1, \ldots, Q_\ell\}$ is a set of non-constant polynomials with constant term zero and $\delta \in \mathbb{F}$.

Proof. Let $j_0 \in (d/3, 2d/3)$ be the natural number which minimizes the quantity $\sum_{i=1}^{m} \ell_{i,j}$, and let $S \subseteq [m]$ be the set of all indices i such that $\mathcal{A}_i$ has at least j_0 layers. Let $\mathcal{A}' = \sum_{i \in S} \mathcal{A}_i$ and $\mathcal{A}'' = \sum_{i \notin S} \mathcal{A}_i$. Thus,

$$\mathcal{A} = \mathcal{A}' + \mathcal{A}''.$$

Here, $\mathcal{A}'' = \sum_{i \notin S} \mathcal{A}_i$ is a multilayered ABP with at most $2d/3$ layers. Moreover, $|\mathcal{A}| = |\mathcal{A}'| + |\mathcal{A}''|$.

The idea now is to apply Lemma 13 to every ABP in $\mathcal{A}'$. For every $i \in S$, we know that there exist some polynomials $P_{i,1}, \ldots, P_{i,\ell_{i,j_0}}, Q_{i,1}, \ldots, Q_{i,\ell_{i,j_0}}$ with constant terms zero and a constant δ_i, such that

$$F_i - \sum_{r=0}^{\ell_{i,j_0}} P_{i,r} Q_{i,r} + \delta_i$$

can be computed by a multilayered ABP. Let us denote this multilayered ABP by $\mathcal{B}_i$. From Lemma 13, we know that $\mathcal{B}_i$ has at most $\max\{j_0, d_i - j_0 + 1\} \leq 2d/3$ layers and size at most $|\mathcal{A}_i|$. Taking a sum over all $i \in S$ and re-indexing the summands, we get that there exist polynomials $P_1, \ldots, P_\ell, Q_1, \ldots, Q_\ell$ with constant terms zero and a constant δ such that the polynomial

$$\sum_{i \in S} F_i - \sum_{r=0}^{\ell} P_r Q_r + \delta$$

is computable by a multilayered ABP $\mathcal{B} = \sum_{i \in S} \mathcal{B}_i$ with at most $2d/3$ layers and size at most $\sum_{i \in S} |\mathcal{A}_i| \leq |\mathcal{A}'|$.

Finally, by combining the multilayered ABPs $\mathcal{B}$ and $\mathcal{A}''$, we get that the polynomial

$$F - \sum_{r=0}^{\ell} P_r Q_r + \delta$$

is computable by a multilayered ABP with at most $2d/3$ layers and size at most $|\mathcal{A}|$. ◀

We now use Lemma 15 to prove our main result, which we restate once more.

▶ **Theorem 7.** *Let $n \in \mathbb{N}$ and let $\mathbb{F}$ be a field such that $\mathrm{char}(\mathbb{F}) \nmid n$. If $\mathcal{A}$ is an algebraic branching program with edge labels of degree at most Δ that computes the polynomial $\sum_{i=1}^{n} x_i^n$, then the size of $\mathcal{A}$ is at least*

$$\Omega\left(\frac{n^2}{\Delta}\right).$$

Proof. Let $\mathcal{A}$ be a multilayered ABP with d_0 layers which computes the polynomial $\sum_{i=1}^{n} x_i^n$. As before we may assume without loss of generality that the underlying field $\mathbb{F}$ is algebraically closed. Note that if d_0 is at most n/Δ, then the formal degree of $\mathcal{A}$ is at most $d_0\Delta \leq n$. Thus, by Theorem 10, we know that $|\mathcal{A}|$ is at least $\Omega(n^2/\Delta)$ and we are done. Also, if $d_0 > n^2/\Delta$, then again we have our lower bound since each layer of $\mathcal{A}$ must have at least one vertex. Thus, we can assume that $n/\Delta \leq d_0 \leq n^2/\Delta$.

The proof idea is to iteratively make changes to $\mathcal{A}$ till we get a multilayered ABP $\mathcal{A}'$ of formal degree at most n that computes a polynomial of the type

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R$$

where $r \leq n/10$ and $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x}), R(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R has degree at most $n-1$. Once we have this, we can invoke Theorem 10 and get the required lower bound.

We now explain how to iteratively obtain $\mathcal{A}'$ from $\mathcal{A}$. In one step, we ensure the following.

▷ Claim 16. Let $\mathcal{A}_k$ be a multilayered ABP with edge labels of degree at most Δ, $d_k \geq n/\Delta$ layers and size at most τ that computes a polynomial of the form $\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R$ where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x}), R(\mathbf{x})$ are polynomials such that for every j, $A_j(\mathbf{0}) = B_j(\mathbf{0}) = 0$ and R has degree at most $n-1$.

If $\tau \leq 0.001n^2/\Delta$, then there exists a multilayered ABP $\mathcal{A}_{k+1}$ with at most $2d_k/3$ layers and size at most τ which computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r'} A'_j \cdot B'_j + R' \,,$$

such that $r' \leq r + 0.005\frac{n^2}{\Delta \cdot d_k}$ and $A'_1(\mathbf{x}), \ldots, A'_{r'}(\mathbf{x}), B'_1(\mathbf{x}), \ldots, B'_{r'}(\mathbf{x}), R'(\mathbf{x})$ are polynomials such that for every i, $A'_i(\mathbf{0}) = B'_i(\mathbf{0}) = 0$ and R' has degree at most $n-1$.

Before moving on to the proof of Claim 16, we first use it to complete the proof of Theorem 7. Let us set $\mathcal{A}_0 = \mathcal{A}$. Then, $\mathcal{A}_0$ is a multilayered ABP with d_0 layers and size at most τ that computes the polynomial $\sum_{i=1}^{n} x_i^n$.

If $\tau \geq 0.001n^2/\Delta$, the statement of the theorem follows. Otherwise, we apply Claim 16 iteratively K times, as long as the number of layers is more than n/Δ, to eventually get a multilayered ABP $\mathcal{A}' = \mathcal{A}_K$ with $d' \leq n/\Delta$ layers. Let $d_0, \ldots, d_{K-1}, d_K$ denote the number of layers in each ABP in this sequence, so that $d_{K-1} > n/\Delta$, and $d_k \leq 2d_{k-1}/3$ for $k \in [K]$. $\mathcal{A}'$ is an ABP with at most n/Δ layers and size at most τ, which by induction, computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R \,,$$

where $A_1(\mathbf{x}), \ldots, A_r(\mathbf{x}), B_1(\mathbf{x}), \ldots, B_r(\mathbf{x}), R(\mathbf{x})$ are polynomials such that for every i, $A_i(\mathbf{0}) = B_i(\mathbf{0}) = 0$ and R has degree at most $n-1$. Further, the number of error terms, r, is at most

$$\frac{0.005n^2}{\Delta}\left(\frac{1}{d_{K-1}} + \frac{1}{d_{K-2}} + \cdots + \frac{1}{d_0}\right).$$

Since $d_k \leq \frac{2}{3} \cdot d_{k-1}$, we have that $\frac{1}{d_{k-1}} \leq \frac{2}{3} \cdot \frac{1}{d_k}$ for all $k \in [K]$, so that

$$r \leq \frac{0.005n^2}{\Delta} \cdot \frac{1}{1-2/3} \cdot \frac{1}{d_{K-1}} \leq \frac{n}{10}$$

as $d_{K-1} \geq n/\Delta$.

At this point, since the formal degree is at most n, using Theorem 10 we get

$$\tau \geq |\mathcal{A}'| \geq \frac{(n/2-r)n}{2\Delta} = \Omega\left(\frac{n^2}{\Delta}\right). \qquad \blacktriangleleft$$

To complete the proof of Theorem 7, we now prove Claim 16.

Proof of Claim 16. Let $\mathcal{A}_k = \sum_{i=1}^{m} \mathcal{A}_{k,i}$, and for $j \in [d_k]$, let $\ell_{i,j}$ be the number of vertices in layer j of $\mathcal{A}_{k,i}$. Recall that if the number of layers in $\mathcal{A}_{k,i}$ is strictly less than j, then we set $\ell_{i,j} = 0$. Let ℓ be the total number of vertices in the middle layers of $\mathcal{A}_k$, defined as

$$\ell = \sum_{i=1}^{m} \left(\sum_{j \in (d_k/3, 2d_k/3)} \ell_{i,j} \right) .$$

Since $\ell \leq \tau \leq \frac{0.001n^2}{\Delta}$, by averaging, we know that there is a $j_0 \in (d_k/3, 2d_k/3)$, such that

$$\ell_{j_0} = \sum_{i=1}^{m} \ell_{i,j_0} \leq \frac{\ell}{d_k/3} \leq \frac{0.001n^2}{\Delta} \cdot \frac{1}{d_k/3} \leq 0.005 \frac{n^2}{\Delta \cdot d_k} .$$

This condition, together with Lemma 15, tells us that there is a multilayered ABP $\mathcal{A}'_{k+1}$ with at most $2d_k/3$ layers and size at most τ that computes a polynomial of the form

$$\sum_{i=1}^{n} x_i^n + \sum_{j=1}^{r} A_j \cdot B_j + R - \sum_{i=1}^{\ell_{j_0}} P_i \cdot Q_i + \delta ,$$

where $P_1, \ldots, P_\ell, Q_1, \ldots, Q_\ell$ are a set of non-constant polynomials with constant term zero and $\delta \in \mathbb{F}$. Since $\ell_{j_0} \leq 0.005 \frac{n^2}{\Delta \cdot d_k}$, the claim follows. ◁

4 Unlayered Algebraic Branching Programs

In this section, we prove Theorem 3. We begin with the following definition.

▶ **Definition 17.** *Let $\mathcal{A}$ be an unlayered ABP over $\mathbb{F}$. Let s and t denote the start and end vertices of $\mathcal{A}$, respectively, and let $v \neq s, t$ be a vertex in $\mathcal{A}$. Denote by $\alpha \in \mathbb{F}$ the constant term of $[s, v]$ and by $\beta \in \mathbb{F}$ the constant term of $[v, t]$.*

The cut *of $\mathcal{A}$ with respect to v, denoted $\mathsf{cut}(\mathcal{A}, v)$, is the unlayered ABP obtained from $\mathcal{A}$ using the following sequence of operations:*

1. *Duplicate the vertex v (along with its incoming and outgoing edges). Let v_1, v_2 denote the two copies of v.*
2. *Erase all outgoing edges of v_1, and connect v_1 to t by a new edge labeled β.*
3. *Erase all incoming edges of v_2, and connect s to v_2 by a new edge labeled α.*

We now prove some basic properties of the construction in Definition 17.

▷ Claim 18. Let $\mathcal{A}$ be an unlayered ABP over $\mathbb{F}$ computing a polynomial F, and let v be a vertex in $\mathcal{A}$. Denote $\mathcal{A}' = \mathsf{cut}(\mathcal{A}, v)$. Denote by d the depth of $\mathcal{A}$ and by d_v the depth of v in $\mathcal{A}$. Then the following properties hold:

1. $\mathcal{A}'$ has 1 more vertex and 2 more edges than $\mathcal{A}$.
2. The depth of $\mathcal{A}'$ is at most

$$\max \{ \mathsf{depth}(\mathcal{A} \setminus \{v\}), d_v + 1, d - d_v + 1 \} ,$$

   where $\mathcal{A} \setminus \{v\}$ is the ABP obtained from $\mathcal{A}$ by erasing v and all of its adjacent edges.
3. $\mathcal{A}'$ computes a polynomial of the form $F - P \cdot Q - \delta$ where P and Q have no constant term, and $\delta \in \mathbb{F}$.

Proof. The first property is immediate from the construction. The second property follows from the following reasoning: each path in $\mathcal{A}'$ is of exactly one of the following types: (a) misses both v_1 and v_2, (b) passes through v_1, or (c) passes through v_2. In case (a), the path also appears in the graph of $\mathcal{A} \setminus \{v\}$. In case (b), the only edge going out of v_1 is to t, and all other edges in the path appear in $\mathcal{A}$, hence the length is at most $d_v + 1$. In case (c), the only edge entering v_2 is from s, hence similarly the path is of length at most $d - d_v + 1$.

It remains to show the last property. Let $P' = [s, v]$ and $Q' = [v, t]$ (as computed in $\mathcal{A}$). Denote $P' = P + \alpha$ where P has no constant term and $\alpha \in \mathbb{F}$ and similarly $Q' = Q + \beta$. One may write $F = P' \cdot Q' + R = (P + \alpha)(Q + \beta) + R$ where R is the sum over all paths in $\mathcal{A}$ which do not pass through v. In $\mathcal{A}'$, we have that $[s, v_1] = P'$ and $[v_2, t] = Q'$, and thus $\mathcal{A}'$ computes the polynomial

$$R + \alpha \cdot Q' + P' \cdot \beta = F - P \cdot Q + \alpha\beta. \qquad \lhd$$

Our goal is to perform cuts on a strategically chosen set of vertices. In order to select them, will use the following well known lemma of Valiant [26], simplifying and improving an earlier result of Erdős, Graham and Szemerédi [8]. For completeness, we also sketch a short proof.

▶ **Lemma 19** (Valiant [26]). *Let G be a directed acyclic graph with m edges and depth $d \geq \sqrt{n}$. Then, there exists a set E' of at most $4m/\log n$ edges such that removing E' from G results in a graph of depth at most $d/2$.*

Proof. Let $d' \geq d \geq \sqrt{n}$ be a smallest power of 2 larger than d, so that $d' \leq 2d$. Let $k = \log d'$. A *valid labeling* of a directed graph $G = (V, E)$ is a function $f : V \to \{0, \ldots, N-1\}$ such that whenever (u, v) is an edge, $f(u) < f(v)$. Clearly if G had depth d then there is a valid labeling with image $\{0, \ldots, N-1\} = \{0, \ldots, d-1\}$ by labeling each vertex by its depth. Conversely, if there is a valid labeling with image $\{0, \ldots, N-1\}$ then $\mathsf{depth}(G) \leq N$.

Let f be a valid labeling of G with image $\{0, \ldots, d'-1\}$ and for $i \in [k]$ let E_i be the set of edges such that the most significant bit in which the binary encoding of the labels of their endpoints differ is i. If E_i is removed, we can obtain a valid relabeling of the graph with image $\{0, \ldots, d'/2 - 1\}$ by removing the i-th bit from all labels.

The two smallest sets among the E_i-s have size at most $2m/k \leq 4m/\log n$ (since $k = \log d' \geq \log n/2$), and removing them gives a valid labeling with image $\{0, \ldots, d'/4 - 1\}$, and therefore a graph with depth at most $d'/4 \leq d/2$. ◀

We need a slight variation of this lemma, in which we do not pick edges whose endpoints have too small or too large a depth in the graph.

▶ **Lemma 20.** *Let G be a directed acyclic graph with m edges and depth $d \geq \sqrt{n}$. Then, there exist a set U of vertices, of size at most $4m/\log n$, such for every $v \in U$ we have that $d/9 \leq \mathsf{depth}(v) \leq 8d/9$, and removing U (and the edges touching those vertices) results in a graph of depth at most $3d/4$.*

Proof. Let E denote the set of edges of G and $E' \subseteq E$ be the set of edges guaranteed by Lemma 19. Let $E_1 \subseteq E'$ be the edges in E' whose heads have depth at most $d/9$, and E_2 be the edges in E' whose heads have depth at least $8d/9$. Let $E'' = E' \setminus (E_1 \cup E_2)$. Clearly, $|E''| \leq |E'| \leq 4m/\log n$. Let U be the set of heads of vertices in E''.

Consider now any path in the graph obtained from G by removing U (and hence in particular E''). Given such a path, let e_1 be the last edge from E_1 in the path which appears before all edges from E_2 (if there exists such an edge), and let e_2 the first edge from E_2 (if

any) in the path. We partition the path into three (possibly empty) parts: the first part is all the edges which appear until e_1 (including e_1); the second part is all the edges after e_1 and before e_2; the last part consists of all the edges which appear after e_2 (including e_2). Because the head of e_1 is a vertex of depth at most $d/9$, the first part can contribute at most $d/9$ edges. The second part includes only edges from $E \setminus E'$, and thus its length is at most $d/2$. The last part again has depth at most $d/9 + 1$, as any path leaving a vertex of depth at least $8d/9$ can have at most that many edges (here we add 1 to account for the edge e_2 itself, since the assumption is on the depth of the head of e_2). Thus, the total length of the path is at most

$$d/9 + d/2 + d/9 + 1 \leq 3d/4\,. \qquad \blacktriangleleft$$

The set of vertices given by the lemma above will be the vertices according to which we will cut the ABP. We describe it in the following lemma, and prove some properties of this operation.

▶ **Lemma 21.** *Let $\mathcal{A}$ be an ABP over a field $\mathbb{F}$ of depth $d \geq \sqrt{n}$ computing a polynomial F. Let τ be the number of vertices and m be the number of edges in $\mathcal{A}$. Then, there exist an unlayered ABP $\mathcal{A}'$, with at most $\tau + 4m/\log n$ vertices, at most $m + 8m/\log n$ edges, and depth at most $9d/10$, computing a polynomial of the form $F - \sum_{i=1}^{r} P_i Q_i - \delta$ where $\delta \in \mathbb{F}$ is a field constant, the P_i, Q_i's have no constant term, and $r \leq 4m/\log n$.*

Proof. Let G be the underlying graph of the ABP $\mathcal{A}$. Let $U = \{u_1, \ldots, u_r\}$ be the set of vertices guaranteed by Lemma 20, such that $r \leq 4m/\log n$. We perform the following sequence of cuts on $\mathcal{A}$. Set $\mathcal{A}_0 := \mathcal{A}$ and for $i \in [r]$, $\mathcal{A}_i = \mathsf{cut}(\mathcal{A}_{i-1}, u_i)$. Finally $\mathcal{A}' = \mathcal{A}_r$.

The statements of the lemma now follow from the properties of cuts as proved in Claim 18. The bound on the number of vertices and edges in $\mathcal{A}'$ is immediate. The claim on the polynomial computed by $\mathcal{A}'$ follows by induction on i.

Finally, by induction on i, we have that the depth of $\mathcal{A}'$ is at most

$$\max\{\mathsf{depth}(\mathcal{A}\setminus U), \mathsf{depth}(u_1)+1, \ldots, \mathsf{depth}(u_r)+1, d-\mathsf{depth}(u_1)+1, \ldots, d-\mathsf{depth}(u_r)+1\}\,,$$

where $\mathcal{A} \setminus U$ is the ABP obtained by removing all vertices in U.

By the choice of U as in Lemma 20, for every $i \in [r]$ we have that $d/9 \leq \mathsf{depth}(u_i) \leq 8d/9$, and $\mathsf{depth}(\mathcal{A} \setminus U) \leq 3d/4$, which implies the required upper bound on the depth of $\mathcal{A}'$ (assuming n, and hence d, are large enough). $\blacktriangleleft$

Repeated applications of Lemma 21 give the following statement.

▶ **Corollary 22.** *Let $\mathcal{A}$ be an ABP over a field $\mathbb{F}$, with edge labels of degree at most $\Delta = n^{o(1)}$, computing an n-variate polynomial F. Further suppose $\mathcal{A}$ has depth at least $\sqrt{n}$, and that the number of edges in $\mathcal{A}$ is at most $n \log n/(1000(\log\log n + \log\Delta))$. Let τ denote the number of vertices in $\mathcal{A}$.*

Then, there exists an ABP $\mathcal{A}'$, whose depth is at most n/Δ, which computes a polynomial of the form $F - \sum_{i=1}^{r} P_i Q_i - \delta$, such that P_i, Q_i are all polynomials without a constant term, $\delta \in \mathbb{F}$ is a field constant, and $r \leq n/10$. The number of vertices in $\mathcal{A}'$ is at most $\tau + n/10$.

Proof. Observe that the depth of $\mathcal{A}$ is at most $d := n \log n$. As long as the depth is at least $\sqrt{n}$, apply Lemma 21 repeatedly at most $k := 7(\log\log n + \log\Delta)$ times, to obtain an ABP of depth at most $(0.9)^k \cdot d \leq n/\Delta$.

The upper bound on the number of summands P_iQ_i and the number of vertices after each application is given as a function of the number of edges, which increases in the process. Hence, we first provide a crude estimate on the number of edges at each step. For $i \in [k]$, let $\mathcal{A}_i$ denote the ABP obtained after the i-th application of Lemma 21, and let m_i be the number of edges in that ABP.

We claim that by induction on i, $m_i \leq m_0 \cdot (1 + 8/\log n)^i$. This is true for $i = 0$ by definition. For $i \geq 1$, since we maintain the invariant that the depth is at least $\sqrt{n}$, it follows from Lemma 21 that

$$m_i \leq m_{i-1} + 8m_{i-1}/\log n = m_{i-1}(1 + 8/\log n) \leq m_0(1 + 8/\log n)^{i-1} \cdot (1 + 8/\log n)\,,$$

where the last inequality uses the induction hypothesis. Thus, the final ABP has at most

$$m_k \leq m_0(1 + 8/\log n)^k \leq 2m_0 = n \log n/(500(\log\log n + \log\Delta)) =: M$$

assuming n is large enough (recall that by assumption we have that $\log\Delta = o(\log n)$, so that $\lim_{n\to\infty}(1 + 8/\log n)^{o(\log n)} = 1$). It is convenient to now use M as a uniform upper bound on the number of edges in all stages of this process, so that each step adds at most $4M/\log n$ summands and vertices. It now follows that r is at most

$$\frac{4kM}{\log n} \leq \frac{7(\log\log n + \log\Delta) \cdot 4n}{500(\log\log n + \log\Delta)} \leq n/10,$$

and similarly the total number of vertices added throughout the process is at most $n/10$. ◀

The lower bound given in Theorem 3 now follows by a simple win-win argument. For convenience, we restate the theorem.

▶ **Corollary 23.** *Let $\mathcal{A}$ be an ABP over a field $\mathbb{F}$, with edge labels of degree at most $\Delta = n^{o(1)}$, computing $\sum_{i=1}^n x_i^n$. Then $\mathcal{A}$ has at least $\Omega(n \log n/(\log\log n + \log\Delta))$ edges.*

Proof. Let τ denote the number of vertices in $\mathcal{A}$. If the number of edges is at least $n \log n/(1000(\log\log n + \log\Delta))$, then we already have our lower bound. Else, the number of edges is at most $n \log n/(1000(\log\log n + \log\Delta))$. Now, by Corollary 22, there exists an ABP $\mathcal{A}'$, with $\tau + n/10$ vertices and depth at most n/Δ, computing $\sum_{i=1}^n x_i^n - \sum_{j=1}^r P_iQ_i - \delta$, such that P_j, Q_j have no constant term, $r \leq n/10$, and $\delta \in \mathbb{F}$.

It thus follows that $\mathcal{A}'$ has formal degree at most n. By Theorem 10, it has $\Omega(n^2/\Delta)$ vertices, thus $\tau = \Omega(n^2/\Delta)$, so that the number of edges is also $\Omega(n^2/\Delta)$. ◀

5 Open problems

We conclude with some open problems.

- A natural open question here is to prove an improved lower bound for unlayered algebraic branching programs. In particular, in the absence of an obvious non-trivial upper bound, it seems reasonable to conjecture that any unlayered ABP computing the polynomial $\sum_{i=1}^n x_i^n$ has size at least $\Omega(n^{2-o(1)})$.
- Yet another question which is natural in the context of this work and remains open is to prove stronger lower bounds for ABPs. As a first step towards this, the question of proving super-quadratic lower bound for homogeneous algebraic formulas might be more approachable.

References

1 Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.

2 Michael Ben-Or. Lower bounds for algebraic computation trees (preliminary report). In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 80–86. ACM, 1983. doi:10.1145/800061.808735.

3 Michael Ben-Or. Algebraic computation trees in characteristi p>0 (extended abstract). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 534–539. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365738.

4 Jin-yi Cai, Xi Chen, and Dong Li. Quadratic lower bound for permanent vs. determinant in any characteristic. *Comput. Complex.*, 19(1):37–56, 2010. doi:10.1007/s00037-009-0284-2.

5 Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. A quadratic lower bound for algebraic branching programs and formulas. *CoRR*, abs/1911.11793, 2019. arXiv:1911.11793.

6 David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in mathematics. Springer, 2007. doi:10.1007/978-0-387-35651-8.

7 Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating multilinear branching programs and formulas. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 615–624. ACM, 2012. doi:10.1145/2213977.2214034.

8 Paul Erdős, Ronald L. Graham, and Endre Szemerédi. On sparse graphs with dense long paths. *Computers & Mathematics with Applications*, 1(3):365–369, 1975. doi:10.1016/0898-1221(75)90037-1.

9 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

10 Pavel Hrubes and Amir Yehudayoff. On isoperimetric profiles and computational complexity. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 89:1–89:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.89.

11 Kyriakos Kalorkoti. A Lower Bound for the Formula Size of Rational Functions. *SIAM Journal on Computing*, 14(3):678–687, 1985. doi:10.1137/0214050.

12 Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111. IEEE Computer Society, 1993. doi:10.1109/SCT.1993.336536.

13 Mrinal Kumar. A quadratic lower bound for homogeneous algebraic branching programs. *Computational Complexity*, 28(3):409–435, 2019. doi:10.1007/s00037-019-00186-3.

14 Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci.*, 1997, 1997. URL: http://cjtcs.cs.uchicago.edu/articles/1997/5/contents.html.

15 Thierry Mignon and Nicolas Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, 2004(79):4241–4253, 2004.

16 Eduard Ivanovich Nechiporuk. On a boolean function. *Dokl. Akad. Nauk SSSR*, 169:765–766, 1966. URL: http://mi.mathnet.ru/dan32449.

17 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418. ACM, 1991. doi:10.1145/103418.103462.

18 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. Available on `citeseer:10.1.1.90.2644`. `doi:10.1007/BF01294256`.

19 Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing*, 2(6):121–135, 2006. `doi:10.4086/toc.2006.v002a006`.

20 Ran Raz and Amir Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008. `doi:10.1007/s00037-008-0254-0`.

21 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015. URL: `https://github.com/dasarpmar/lowerbounds-survey/releases/`.

22 Justin R. Smith. *Introduction to Algebraic Geometry*. Textbooks in Mathematics. Taylor & Francis, 2014. URL: `https://books.google.com/books?id=zx7usgEACAAJ`.

23 Roman Smolensky. Easy lower bound for a strange computational model. *Computational Complexity*, 6(3):213–216, 1997. `doi:10.1007/BF01294255`.

24 Volker Strassen. Die berechnungskomplexität von elementarsymmetrischen funktionen und von interpolationskoeffizienten. *Numerische Mathematik*, 20(3):238–251, June 1973. `doi:10.1007/BF01436566`.

25 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. URL: `http://eudml.org/doc/151394`.

26 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In Jozef Gruska, editor, *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977. `doi:10.1007/3-540-08353-7_135`.

27 Akihiro Yabe. Bi-polynomial rank and determinantal complexity. *CoRR*, abs/1504.00151, 2015. `arXiv:1504.00151`.

Super Strong ETH Is True for PPSZ with Small Resolution Width

Dominik Scheder
Shanghai Jiaotong University, China
dominik.scheder@gmail.com

Navid Talebanfard
Institute of Mathematics, The Czech Academy of Sciences, Prague, Czech Republic
talebanfard@math.cas.cz

Abstract

We construct k-CNFs with m variables on which the strong version of PPSZ k-SAT algorithm, which uses resolution of width bounded by $O(\sqrt{\log \log m})$, has success probability at most $2^{-(1-(1+\epsilon)2/k)m}$ for every $\epsilon > 0$. Previously such a bound was known only for the weak PPSZ algorithm which exhaustively searches through small subformulas of the CNF to see if any of them forces the value of a given variable, and for strong PPSZ the best known previous upper bound was $2^{-(1-O(\log(k)/k))m}$ (Pudlák et al., ICALP 2017).

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases k-SAT, PPSZ, Resolution

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.3

Funding *Dominik Scheder*: Supported by the National Natural Science Foundation of China under grant 61502300 and 11671258.
Navid Talebanfard: Supported by GAČR grant 19-27871X.

1 Introduction

The PPSZ algorithm for k-SAT by Paturi, Pudlák, Saks, and Zane [7] is simple to state but famously difficult to analyze. Given a k-CNF formula Φ as input, it first chooses a random ordering π of its variables $x_1, \ldots, x_m$. It goes through them one by one, in the order given by π. For each variable x, it tries to derive the correct value using a certain *proof heuristic* P. P takes as input a k-CNF formula Φ and a variable x and returns a value in $\{0, 1, ?\}$. P must be *sound*, meaning if $P(\Phi, x) = b \in \{0, 1\}$ then $\Phi \models (x = b)$, i.e., every satisfying assignment of Φ sets x to b; however, we allow P to be *incomplete*, i.e., it may answer "?", meaning "I don't know". If $P(\Phi, x) = b \in \{0, 1\}$, then PPSZ sets x to b; otherwise it sets x to some $b \in \{0, 1\}$ chosen uniformly at random. In either case, it simplifies Φ to $\Phi|_{x \mapsto b}$. Once all variables have been processed, the resulting formula either contains the empty clause $\square$, and we declare this run of PPSZ a failure; or it does not, in which case PPSZ has found a satisfying assignment.

If PPSZ has success probability p then we can repeat it $1/p$ times, obtaining a constant success probability. As long as P runs in subexponential time, the overall running time of this Monte Carlo algorithm is dominated by $1/p$ (which will, most likely, be exponential in n). Which proof heuristics P should one consider? There are currently just two on the market. The first one is P_w, which checks whether $(x = b)$ is implied by a set of up to w clauses of Φ. The second one is R_w, which tries to derive the clause $(x = b)$ by resolution, bounded by width w. Obviously they both can be implemented in time $O^*\left(\binom{|\Phi|}{w}\right) \leq O^*\left(\binom{m^k}{w}\right)$, which

is subexponential as long as $w \in o\left(\frac{m}{\log m}\right)$. It is easy to see that $R_{w \cdot k}$ is at least as strong as P_w. We also speak of *weak PPSZ* when it uses P_w and *strong PPSZ* when it uses R_w (ignoring the concrete values of w).

Proving positive results, i.e., lower bounds on the success probability, seems remarkably insensitive to our choice of P. In fact, all lower bounds we currently know work for P_w, for any $w \in \omega(1)$:

▶ **Theorem 1** (Paturi, Pudlák, Saks, and Zane [7] and Hertli [6]). *On k-CNF formulas with m variables, the success probability of PPSZ using the heuristic P_w is at least $2^{-m(1-s_k)+o(m)}$, where $\lim_{k\to\infty} k s_k = \frac{\pi^2}{6}$, provided that $w = w(m) \in \omega(1)$.*

Originally, Paturi, Pudlák, Saks, and Zane stated their algorithm as using R_w, i.e., width-bounded resolution; however, it is easy to see that their analysis works for the weaker heuristic P_w as well, see for example [10] for a formal proof. We do not know any better bound for PPSZ using R_w, for any $w \in o(m)$.

The parameter s_k in the theorem is called the *savings* of the algorithm. Ignoring constant factors, the theorem shows that the savings of PPSZ are at least $\Omega(1/k)$. Other algorithms, arguably much simpler, such as PPZ [8] and Schöning's Random Walk [11] have smaller savings than PPSZ, but also of order $\Omega(1/k)$. In general, let σ_k be the supremum of all σ such that there is a randomized algorithm for k-SAT running in time $O\left(2^{m(1-\sigma)}\right)$. There is a whole hierarchy of conjectures about how large the savings for k-SAT can be. Here is a list, sorted from weak to strong.

1. P $\neq$ NP: k-SAT has no polynomial time algorithm.
2. ETH (exponential time hypothesis): $\sigma_3 < 1$.
3. SETH (strong exponential time hypothesis): $\lim_{k\to\infty} \sigma_k = 0$, i.e., as k grows, the advantage over brute force shrinks to nil.
4. SSETH (super strong exponential time hypothesis): $\sigma_k \in O(1/k)$.

We already know (as shown by PPZ, Schöning's and PPSZ algorithms) that $\sigma_k \in \Omega(1/k)$, so Point 4 actually conjectures that $\sigma_k \in \Theta(1/k)$. Of course proving an unconditional upper bound on σ_k is far out of reach for now. However one could try to prove such upper bounds on the savings of specific algorithms. This would then shed light on the difficulty of improving k-SAT algorithms. In this paper we prove close to tight upper bounds on the savings of the strong PPSZ algorithm showing that its running time is consistent with SSETH, that is the worst case running time of PPSZ is as predicted by SSETH. This is in contrast to a recent result of Vyas and Williams [12] who showed that SSETH is false for random k-SAT.

1.1 Previous Results: Hard Instances

The first hard instances for PPSZ were given by the authors together with Chen and Tang [3]. That work constructed k-CNFs based on a random distribution of linear systems and showed that PPSZ using R_w, that is resolution of bounded width, succeeds with probability at most $2^{-m(1-O(\log^2(k)/k))}$ on these formulas, as long as $w \leq \frac{\ln(k)n}{k}$ (Theorem 1.2 in [3]). Together with Pudlák [9] we then improved this lower bound to $2^{-m(1-O(\log(k)/k))}$, which holds as long as $w \leq n/k$ (Theorem 6 in [9]). This improvement came mainly from clarifying and sharpening a union bound in [3]. However based on a completely different construction, it gave an upper bound of $2^{-m(1-2(1+\epsilon)/k)}$ for the "weak" heuristic P_w, for some $w = n^{\Theta(\epsilon)}$ (Theorem 5 in [9]). This construction is based on Tseitin formulas defined on large girth graphs. For R_w, it was left open whether one can obtain the same bound.

2 Our Results

▶ **Theorem 2** (SSETH Holds for PPSZ). *For every $k \in \mathbb{N}$, there is a polynomial p and a sequence $(F_m)_{m\in\mathbb{N}}$ of satisfiable k-CNF formulas F_m on m variables, such that for every $\epsilon > 0$ and $w \leq \sqrt{\epsilon \cdot \frac{\log\log m}{2\log k}} - 3$, it holds that* $\Pr[\text{ppsz}(F_m, R_w) \text{ succeeds}] \leq p(m)2^{-m(1-2(1+\epsilon)/k)}$.

Thus, the super strong exponential time hypothesis is true for Strong PPSZ, provided that we do not make it too strong, i.e., keep w fairly small. Note that this gives an upper bound on the savings of PPSZ by $2/k$, which is quite close to the currently best lower bound of $(\pi^2/6 + o(1))/k$ [7].

Our result is incomparable to the previous ones. We feel that the "super strong ETH bounds" of $2(1+\epsilon)/k$ in the exponent make this result much stronger than its predecessors. However, the doubly-logarithmic upper bound on w is, of course, much more restrictive than the $w \leq m/k$ bound of Theorem 6 in [9]. Might it be that super strong ETH fails for $w = m/k$? Maybe even for $w = \log(m)$? If we could rule out this possibility, we would have done so in this paper. However, remember that the lower bound on the success probability (Paturi, Pudlák, Saks, and Zane [7]) holds for $P_{\omega(1)}$, which is arguably the weakest possible non-trivial proof heuristic. At the moment, there are no better lower bounds for $R_{o(m)}$, which is much stronger than $P_{\omega(1)}$. Thus, we feel that the parameter w is not as relevant as the savings.

▶ **Conjecture 3.** *Super Strong ETH holds for PPSZ using R_w, as long as $w = o(m)$.*

To be honest, the only supporting evidence we have for this conjecture is the lack of progress in analyzing the success probability of PPSZ. If this conjecture is true, the hard instances proving it might use a very different construction from those in Theorem 2. Thus, we further conjecture:

▶ **Conjecture 4.** *Theorem 2 holds for some $w = \Theta(\log m)$, with the same formulas F_m.*

We have a little bit more evidence supporting the second conjecture: our constructions are based on Tseitin formulas, and our bound on w is related to the girth of a graph H; the graph H has $\Theta(\log m)$ vertices and girth $\Theta(\log\log m)$. However, the resolution width of Tseitin formulas is usually governed by the expansion properties of the underlying graph, not its girth, and thus we hope that some proof also works for $w = \Theta(\log m)$.

Recently, Hansen, Kaplan, Zamir, and Zwick [5] published an improved version of PPSZ, called biased-PPSZ. Roughly stated, the idea of their improvement is that, looking at a formula F with a unique satisfying assignment α, we can identify a set a set $X \subseteq V$ of variables on which α is *biased*, i.e., the number of $x \in X$ set to 1 by α deviates from $|X|/2$ significantly. Thus, setting those variables to 1 with some probability $p \neq 1/2$ gives a higher success probability. We have not checked whether the bounds of Theorem 2 also hold for biased-PPSZ.

2.1 Notation

Given a set of variables X, a *partial assignment* is a function $\alpha : X \to \{0, 1, *\}$, that is an assignment of 0-1 values to some of the variables with $*$ intended to mean unset by α. We denote the set of variables to which α gives a value by $\text{var}(\alpha) := \{x \in X : \alpha(x) \in \{0, 1\}\}$. For two partial assignments α and β we write $\alpha \subseteq \beta$ to mean that for every $x \in \text{var}(\alpha)$, it holds that $\beta(x) = \alpha(x)$. Naturally, $\alpha \subset \beta$ means that $\alpha \subseteq \beta$ and $|\text{var}(\alpha)| < |\text{var}(\beta)|$. Given a

variable x and $b \in \{0,1\}$, $x \mapsto b$ is the partial assignment which sets x to b. The assignment which sets every variable to 0 is denoted by $\mathbf{0}$. For $Y \subseteq X$, we write $Y \mapsto \mathbf{0}$ to denote the partial assignment which sets all variables in Y to 0. If $\mathrm{var}(\alpha) \cap \mathrm{var}(\beta) = \emptyset$, we define $\alpha \cup \beta$ to be the partial assignment which sets all $x \in \mathrm{var}(\alpha)$ to $\alpha(x)$, all $x \in \mathrm{var}(\beta)$ to $\beta(x)$, and all other variables to $*$. Finally the restriction of a formula Φ by α is denoted by $\Phi|_\alpha$.

2.2 The Formula

Let $G = (V, E)$ be a graph. For every $e \in E(G)$ we introduce a variable x_e. Given a *charge* $c : V \to \{0,1\}$, the *Tseitin formula on G with charge c* is the Boolean formula

$$\mathrm{Tseitin}(G, c) := \bigwedge_{u \in V(G)} \left(\sum_{e \in E(G) : u \in e} x_e \equiv c(u) \mod 2 \right) \tag{1}$$

If G has maximum degree k then this can be expressed as a k-CNF formula on $m = |E(G)|$ variables and $|V(G)|2^{k-1}$ clauses. Usually in proof complexity, the charge c is chosen so that $\mathrm{Tseitin}(G, c)$ is unsatisfiable. In this paper, all charges will be 0, and $\mathrm{Tseitin}(G, \mathbf{0})$ is obviously satisfiable: set all variables to 0. We will hence drop c from the notation and simply write $\mathrm{Tseitin}(G)$ to denote this formula. The constraint $\sum_{e \in E(G) : u \in e} x_e \equiv 0 \mod 2$ is called the *Tseitin constraint* of vertex u. Given a set $\mathcal{B}$ of pairs of edges in G consider the following formula

$$\mathrm{Tseitin}(G) \wedge \bigwedge_{\{e,f\} \in \mathcal{B}} (\bar{x}_e \vee \bar{x}_f).$$

The constraint $(\bar{x}_e \vee \bar{x}_f)$ is called a *bridge constraint*. It is easy to see that $\mathbf{0}$ is the unique satisfying assignment of this formula if and only if every cycle in G contains a bridge in $\mathcal{B}$. We will consider a particular instantiation of bridges given by graph homomorphisms. A *graph homomorphism* from a graph G to a graph H is a function $\varphi : V(G) \to V(H)$ such that $\{\varphi(u), \varphi(v)\} \in E(H)$ whenever $\{u, v\} \in E(G)$. Thus, φ also induces a function from $E(G)$ to $E(H)$; $\varphi(\{u, v\}) := \{\varphi(u), \varphi(v)\}$. Given G, H, and a homomorphism φ from G to H, we define a Tseitin formula with bridges on the variable set $\{x_e \mid e \in E(G)\}$:

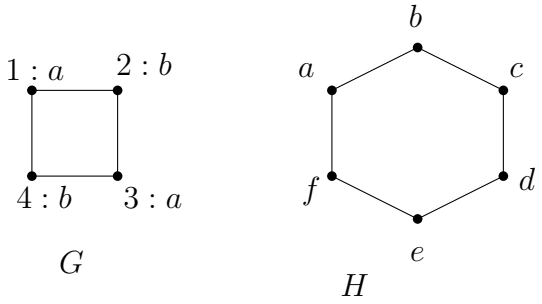
$$\mathrm{TseitinBridge}(G, H, \varphi) := \mathrm{Tseitin}(G) \wedge \bigwedge_{\substack{e,f \in E(G) \\ e \neq f, \varphi(e) = \varphi(f)}} (\bar{x}_e \vee \bar{x}_f)\ . \tag{2}$$

For brevity, we write $V = V(G)$ and $E = E(G)$.

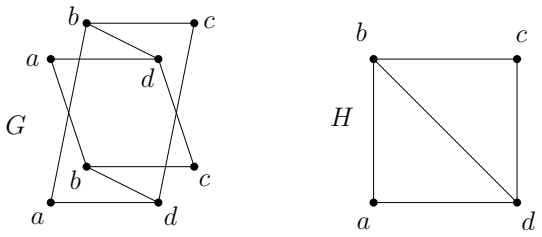
▶ **Observation 5.** *If* $\mathrm{girth}(G) > |E(H)|$ *then* $\mathrm{TseitinBridge}(G, H, \varphi)$ *is uniquely satisfiable by* $\mathbf{0}$.

Proof. Let $\alpha \neq \mathbf{0}$ be a total assignment. Let $F := \{e \in E(G) \mid \alpha(x_e) = 1\}$. If some vertex u has degree 1 in (V, F), then α violates its Tseitin constraint. Otherwise, (V, F) has a cycle, which has length at least $\mathrm{girth}(G)$. By the pigeonhole principle, this cycle contains two edges e, f such that $\varphi(e) = \varphi(f)$, and thus α violates their bridge constraint. ◀

Locally Injective Homomorphisms. A homomorphism φ is called *locally injective* if for every $u \in V(G)$ and any two of its neighbors v_1 and v_2, it holds that $\varphi(v_1) \neq \varphi(v_2)$. Note that $\varphi : G \to H$ being locally injective immediately implies that $\deg_G(u) \leq \deg_H(\varphi(u))$. We call φ *locally bijective* if, additionally, $\deg_G(u) = \deg_H(\varphi(u))$ for all vertices u of G. Note that a locally bijective homomorphism bijectively maps the neighborhood of u to the neighborhood of $\varphi(u)$. The graph G is called a *covering graph* of H or a *lift* of H.



Example of a homomorphism that is not locally injective. The two neighbors of 1 are both mapped to b.



Example of a locally bijective homomorphism. The letters next to the vertices of G are not their names but rather their images under φ.

▶ **Theorem 6.** *Let G be a graph on n vertices and m edges. Suppose there is a locally injective graph homomorphism $\varphi : G \to H$ for some graph H with $|E(H)| < \operatorname{girth}(G)$. Then for all $\epsilon > 0$ and $w := \sqrt{\frac{\epsilon \cdot \operatorname{girth}(H)}{2}} - 3$, the success probability of PPSZ with heuristic R_w on $\Phi := \operatorname{TseitinBridge}(G, H, \varphi)$ is at most*

$$\Pr[\operatorname{ppsz}(\Phi, R_w)] \leq 2^{-m+(1+\epsilon)n} .$$

Proof of Theorem 2 using Theorem 6. We first show how to construct F_m for infinitely many m. Let n_0 be some given, sufficiently large even integer. A well-known fact, first proven by Erdős and Sachs [4], is that there is a k-regular graph G_0 on n_0 vertices having girth at least $g_0 := \frac{\log n_0}{\log(k-1)}$. Set $n_1 := \left\lfloor \frac{2(g_0-1)}{k} \right\rfloor$ or $n_1 := \left\lfloor \frac{2(g_0-1)}{k} \right\rfloor - 1$, whichever is even, and let G_1 be a k-regular graph on n_1 vertices, such that $\operatorname{girth}(G_1) \geq g_1 := \frac{\log n_1}{\log(k-1)}$. This exists, provided that n_0 is sufficiently large. Note that G_1 has at most $g_0 - 1 < \operatorname{girth}(G_0)$ edges.

A result by Angluin and Gardiner [1] states that there is a common lift G of G_0 and G_1. That is, G is a covering graph of G_0 and of G_1. Being a lift of a k-regular graph, G is k-regular as well. A closer inspection of their proof reveals that $n := |V(G)| \leq 4n_0n_1$.

Let $m := \frac{kn}{2}$ be the number of edges in G. We set $\Phi_m := \operatorname{TseitinBridge}(G, G_1, \varphi_1)$, where φ_1 is the locally bijective homomorphism from G to G_1.

It is not difficult to see that lifting cannot decrease the girth, and thus $\operatorname{girth}(G) \geq \operatorname{girth}(G_0) > |E(G_1)|$. Thus, we can apply Theorem 6 to G, G_1, and φ_1, and conclude that the success probability of PPSZ on Φ_m is at most $2^{-m+(1+\epsilon)n}$ when using heuristic R_w. A quick calculation shows that $g_1 \geq \frac{\log\log m}{\log k}$ if n_0 is sufficiently large, and thus $w \geq \sqrt{\epsilon \cdot \frac{\log\log m}{2\log k}} - 3$.

This construction gives us an infinite set $M \subseteq \mathbb{N}$ and, for each $m \in M$, a satisfiable k-CNF formula F_m on m variables for which the claimed hardness result holds. By a simple tweaking of the construction, we can ensure that M is "reasonably dense", meaning that there is some $m^* \in M \cap [m - \log m, m]$ for all sufficiently large m. We then let F_{m^*} be F_m, plus $m - m^*$ dummy variables. The success probability is then at most $2^{-m^*(1-2(1+\epsilon)/k)} \leq \operatorname{poly}(m)2^{-m(1-2(1+\epsilon)/k)}$. We leave the details to the reader. ◀

3 All You Need to Know About PPSZ: Proof of Theorem 6

We will explain the connection between PPSZ and width-bounded resolution lower bounds. After this section, the reader can forget everything about PPSZ and think of this paper as proving a certain resolution width lower bound. If $C = (C' \vee x)$ and $D = (D' \vee \bar{x})$ are clauses, then $(C' \vee D')$ is called the *resolvent* of C and D. It is clear that $C \wedge D$ logically implies $C' \vee D'$. Let Φ be a CNF formula. A *resolution derivation from* Φ is a sequence of clauses $C_1, \ldots, C_t$ such that every C_i is (1) a clause of Φ or (2) the resolvent of two earlier clauses. The *width* of the derivation is $\max_{1 \le i \le t} |C_i|$. For a clause C, we denote by $\text{width}(\Phi \vdash C)$ the minimum width of a resolution derivation from Φ that contains C. Resolution is complete for refutations, that is, Φ is unsatisfiable if and only if there is a derivation of the empty clause, denoted by $\square$, from Φ.

Proof of Theorem 6. Let G, H, φ be as in Theorem 6, and let $\Phi := \text{TseitinBridge}(G, H, \varphi)$. The only satisfying assignment of Φ is $\mathbf{0}$. Consider a variant of PPSZ run on Φ such that whenever it has to pick a random value for a variable, it correctly sets it to 0. Fix a permutation π. Let $Y(\pi)$ be the set of variables for which this variant of PPSZ under π could not derive the value using R_w, and let $Z(\pi) := \text{var}(\Phi) \setminus Y(\pi)$ be the rest, i.e., all variables whose value can be derived using R_w once all variables before them in π are set to 0. It is not difficult to see that the success probability of the actual PPSZ on Φ is exactly $\mathbb{E}_\pi \left[2^{-|Y(\pi)|}\right]$.

Suppose, for the sake of contradiction, that PPSZ using heuristic R_w has success probability greater than $2^{-m+(1+\epsilon)n}$. Then there is some π for which $Z(\pi) \ge (1+\epsilon)n$. Fix this π and set $Z := Z(\pi)$ and $Y := Y(\pi)$. The set of variables Z corresponds to a set F of edges, $F = \{e \in E(G) \ : \ x_e \in Z\}$. Set $G' = (V, F)$. Note that G' has n vertices and at least $(1+\epsilon)n$ edges. Setting a variable in Φ to 0 corresponds to simply deleting the corresponding edge in G, and therefore

$$\Phi|_{Y \mapsto \mathbf{0}} = \text{TseitinBridge}(G', H, \varphi) \ .$$

For a graph $G = (V, E)$ and a set $X \subseteq V$, define the edge boundary $\partial(X) := \{e \in E \ : \ |e \cap X| = 1\}$. Call G an (a, b)-expander if $|\partial(X)| \ge b$ for all sets X of exactly a vertices. The next lemma is basically Lemma 17 from [9], adapted for our purposes. We give a proof for completeness.

▶ **Lemma 7.** *Let $\epsilon > 0$ and let G' be a graph on n vertices with at least $(1+\epsilon)n$ edges. Let $\ell \in \mathbb{N}$ and $h = \ell/\epsilon$. If $h < \text{girth}(G')$ then G' contains a non-empty subgraph G'' that has minimum degree at least 2 and is an $(h, \ell+1)$-expander,*

Proof. Start with $G'' = G'$. If G'' has a vertex of degree 0 or 1, delete it. If G'' contains a set X of h vertices with $|\partial(X)| \le \ell$, delete X from G'', along with all incident edges.

The first type of deletion removes one vertex and at most one edge. The second type removes exactly h vertices. There are at most ℓ edges in the boundary of X; since $|X| < \text{girth}(G')$, the graph $G''[X]$ is a forest, and thus there are at most $h-1$ edges within X. Thus, removing X removes at most $\ell + h - 1 < (1+\epsilon)h$ edges.

We see that a step that removes a vertices removes fewer than $(1+\epsilon)a$ edges. Suppose the process terminates with t vertices deleted. Trivially $t \le n$. Fewer than $(1+\epsilon)n$ edges have been deleted, so G'' is non-empty. ◀

Let G'' be given by Lemma 7 with $\ell := w+1$. We will further restrict Φ so that only edges of G'' remain unset. Let $F'' := E(G) \setminus E(G'')$, $Y'' := \{x_e : e \in F''\}$, and $\Phi'' := \Phi|_{Y'' \mapsto \mathbf{0}}$. Note that $\Phi'' = \text{TseitinBridge}(G'', H, \varphi)$. Recall that all edges of G'' are mentioned in Z and since $Y'' \supseteq Y$ and restricting additional variables cannot increase the resolution width, we conclude that there exists $e \in E(G'')$ such that

$$\text{width}(\Phi'' \vdash \bar{x}_e) \leq w. \tag{3}$$

Towards a contradiction, we claim that in fact this resolution width is *large* for all variables x_e where $e \in E(G'')$. Indeed, we have the following theorem:

▶ **Theorem 8** (Resolution Lower Bound). *Let G be a graph of minimum degree 2 that is an $(h, \ell+1)$-expander. Suppose there is a locally injective homomorphism $\varphi : G \to H$ into some graph H. Then*

$$\text{width}(\text{TseitinBridge}(G, H, \varphi) \vdash \bar{x}_e) > \ell - 1 \ , \tag{4}$$

for all edges e of G, provided that $2h\ell + 5h + \ell < \text{girth}(H)$.

Note that G'' has minimum degree 2 and is a $(h, \ell+1)$-expander for $\ell = w+1$ and $h = \frac{w+1}{\epsilon}$. Also note that $\varphi : V(G'') \to V(H)$ (or rather, the restriction of φ to $V(G'')$) is still a locally injective homomorphism. Recall that $w = \sqrt{\frac{\epsilon \cdot \text{girth}(H)}{2}} - 3$ and hence $2h\ell + 5h + \ell = 2(w+1)^2/\epsilon + 5(w+1)/\epsilon + w + 1 < \text{girth}(H)$, and thus Theorem 8 applies to G''. This contradicts (3) and finishes the proof of Theorem 6. ◀

4 Proof of Theorem 8

Let $\Phi = \text{TseitinBridge}(G, H, \varphi)$ and let e^* be an edge of G. We will show that $\text{width}(\Phi \vdash \bar{x}_{e^*}) > \ell - 1$ for all such edges e^*. In fact, we will prove $\text{width}(\Phi|_{x_{e^*} \mapsto 1} \vdash \square) > \ell - 1$, which is a slightly stronger statement.

We will use the game characterization of resolution width due to Atserias and Dalmau [2]. Given a CNF formula F, the ℓ-bounded Atserias-Dalmau game played by two players, Prover and Delayer is defined as follows. A *position* in this game is a partial assignment α setting up to ℓ variables. The start position is the empty assignment. At position α, Prover can either (1) forget some variables, i.e., replace α by some $\beta \subset \alpha$. Or, (2), if $|\text{var}(\alpha)| \leq \ell - 1$, pick a variable $x \notin \text{var}(\alpha)$ and *query it*; Delayer has to respond with a truth value $b \in \{0, 1\}$, and α is updated to $\alpha \cup (x \mapsto b)$. The game ends if α violates a clause of F, in which case Prover wins. Delayer wins if she has a strategy to play indefinitely.

▶ **Theorem 9** (Atserias and Dalmau [2]). *Let F be an unsatisfiable CNF formula. If Delayer has a winning strategy for the $\ell+1$-bounded game then there is no width-ℓ resolution refutation of F.*

To show that $\text{width}(\Phi|_{x_{e^*} \mapsto 1} \vdash \square) > \ell - 1$ we define a winning strategy for Delayer for the ℓ-bounded game that ensures she never loses. Indeed, we will modify the game a bit: it is now played on Φ instead of $\Phi|_{x_{e^*} \mapsto 1}$; the starting position is the partial assignment $x_{e^*} \mapsto 1$; Prover can never forget x_{e^*} but is now allowed partial assignments up to size $\ell + 1$. That is, he can query a new variable provided $|\text{var}(\alpha)| \leq \ell$. It is easy to see that if Delayer wins this modified game, she wins the original one, too. Since $\Phi = \text{TseitinBridge}(G, H, \varphi)$, we can easily rephrase the rules of the game in terms of sets of edges instead of partial assignments:

The Atserias-Dalmau, Graph View. A position of the game is described by two disjoint set $F_0, F_1 \subseteq E(G)$. F_0 and F_1 correspond to the variables of Φ that the current partial assignment sets to 0 and 1, respectively. The start position is $F_0 = \emptyset$ and $F_1 = \{e^*\}$.

In every step, Prover either (1) removes one edge e from F_0 or F_1 (but never removes e^*). Or (2) he *queries* an edge $e \in E(G) \setminus (F_0 \cup F_1)$, provided $|F_0| + |F_1| \leq \ell$. Delayer can then decide whether to add e to F_0 or F_1.

Prover wins if there is a vertex u in G such that all edges incident to u are in $F_0 \cup F_1$ but $\deg_{F_1}(u)$ is odd (then the partial assignment α violates the Tseitin constraint of u); or if there are two edges $e, f \in F_1$ with $\varphi(e) = \varphi(f)$ (then α violates a bridge constraint).

We will now describe a winning strategy for Delayer. Throughout the game, she maintains a set $\tilde{F}_1$ such that $F_1 \subseteq \tilde{F}_1 \subseteq E \setminus F_0$. Let $V(\tilde{F}_1)$ denote the set of vertices incident to at least one edge of $\tilde{F}_1$. She makes sure $\tilde{F}_1$ satisfies certain invariants:

1. Every connected component of $(V, \tilde{F}_1)$ is a path; a path of positive length (i.e., a path that is not an isolated vertex) is called an $\tilde{F}_1$-path.
2. Every $\tilde{F}_1$-path contains at least one edge of F_1.
3. φ is injective on $V(\tilde{F}_1)$.
4. Each $\tilde{F}_1$-path has length at least $2h + 1$, and the first and last h edges of every $\tilde{F}_1$-path are not in F_1.
5. Each $\tilde{F}_1$-path has length at most $2h\ell + 2h + \ell$.

▶ **Observation 10.** *If $\tilde{F}_1$ satisfies the invariants, then no constraint is violated.*

Proof. In fact we show that invariants 1-4 already give the result. First, consider a Tseitin constraint of a vertex u. Since $\tilde{F}_1$ consists of disjoint paths, so does F_1. Thus, u is incident to 0, 1, or 2 edges of F_1. If it is incident to 0 or 2 edges of F_1, the Tseitin constraint of u is clearly not falsified. If it is incident to exactly one edge of F_1, then it is the endpoint of some path of F_1-edges. By Invariant 4, u is incident to some other edge $f \in \tilde{F}_1 \setminus F_1$. Thus, f is neither in F_0 nor in F_1, and the Tseitin constraint of u is not violated.

Next, consider a bridge constraint $(\bar{x}_e \vee \bar{x}_f)$. By construction we have $\varphi(e) = \varphi(f)$. By Invariant 3, φ is injective on $\tilde{F}_1$, and thus e, f cannot both be in F_1, and the bridge constraint is not violated. ◀

We will use the following property of φ.

▶ **Proposition 11.** *Let G' be a connected subgraph of G of diameter less than* girth(H)*. Then φ is injective on $V(G')$, and thus $\varphi(G')$ is isomorphic to G'.*

Proof. For the sake of contradiction, suppose $u, v \in V(G')$ are two vertices with $\varphi(u) = \varphi(v)$. Let p be a shortest path from u to v in G'. Write p as $u = u_0, u_1, \ldots, u_t = v$. By assumption, $t <$ girth(H). Under φ, the path p is mapped to a reduced walk in H, *reduced* meaning that $\varphi(u_{i-1}) \neq \varphi(u_{i+1})$ for all $1 \leq i \leq t - 1$. Since $\varphi(u) = \varphi(v)$, this is a closed walk and thus contains a cycle. The cycle has length at most $t <$ girth(H), a contradiction. ◀

How to initialize $\tilde{F}_1$. Delayer can easily initialize $\tilde{F}_1$. Write $e^* = \{u^*, v^*\}$. Since G has minimum degree 2, Delayer can start a reduced walk from u^* of length h, and also from v^* and add this to $\tilde{F}_1$. Since $2h + 1 <$ girth(H), this is a path; by Proposition 11, φ is injective on its vertices.

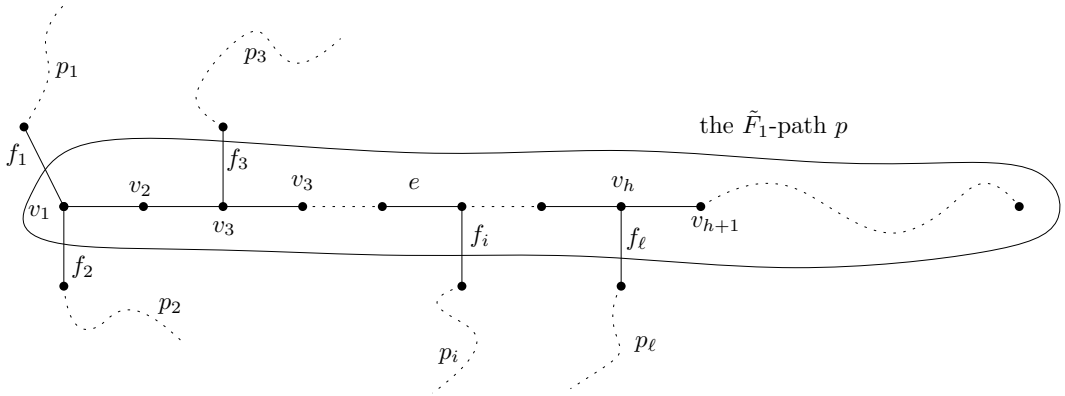
How to handle a Forget Step. Suppose Prover forgets some edge $e \in F_0 \cup F_1$. If $e \in F_0$, Delayer leaves $\tilde{F}_1$ unchanged. If $e \in F_1$, let p be the $\tilde{F}_1$-path containing e. If p contains some other F_1-edge besides e, Delayer does not change $\tilde{F}_1$; otherwise it simply removes all of p from $\tilde{F}_1$. All invariants stay satisfied.

How to handle a Query from Prover. Suppose Prover queries an edge e. Delayer has now to choose whether to include e into F_0 or F_1, and potentially update $\tilde{F}_1$

Case 1: e is not in $\tilde{F}_1$. Then Delayer adds e to F_0 and leaves $\tilde{F}_1$ unchanged. All invariants still hold. This includes the case that e is incident to some vertex on a $\tilde{F}_1$-path, but is not itself inside this path.

Case 2: e is in some $\tilde{F}_1$-path p but not among its first or last h edges. Delayer adds e to F_1 and leaves $\tilde{F}_1$ unchanged. All invariants still hold.

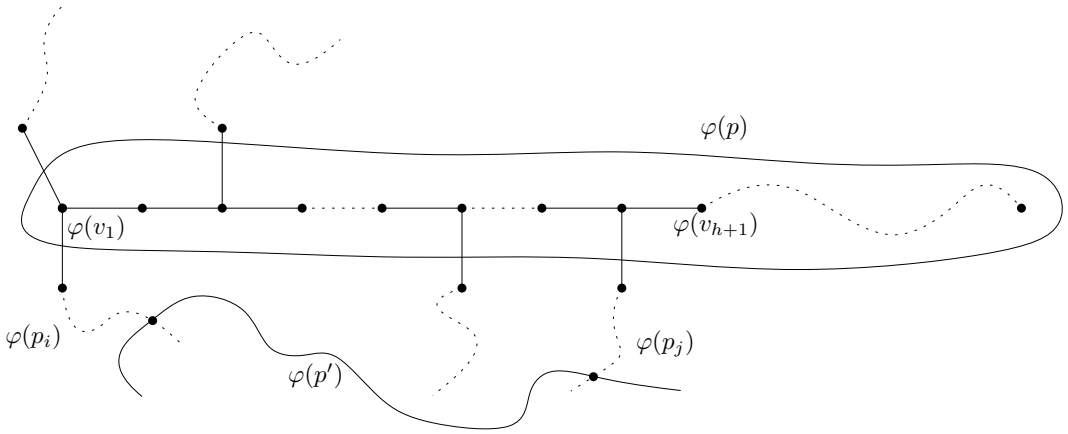
Case 3: e is among the first or last h edges of some $\tilde{F}_1$-path p. Let $v_1, \dots, v_{h+1}$ be the first $h+1$ vertices of p, and let q denote the length-h-path $v_1, \dots, v_{h+1}$. By assumption, e lies on the path q. Since G is an $(h, \ell+1)$-expander, there are edges $f_1, \dots, f_{\ell+1}$, each incident to exactly one vertex in $\{v_1, \dots, v_h\}$. One of those edges could be $\{v_h, v_{h+1}\}$, but without loss of generality, for $1 \le i \le \ell$, edge f_i connects some $a_i \in \{v_1, \dots, v_h\}$ to some b_i outside $\{v_1, \dots, v_{h+1}\}$. Since G has minimum degree 2 and girth larger than h, we can find paths $p_1, \dots, p_\ell$ such that each p_i has length h and starts with a_i as its first and b_i as its second vertex. Since $3h < \text{girth}(H) \le \text{girth}(G)$, the p_i are vertex-disjoint. Since $h + |p| \le h + 2h\ell + 2h + \ell < \text{girth}(H) \le \text{girth}(G)$, the path p_i intersects p only in vertex a_i. Thus, $C := p \cup p_1 \cup \dots \cup p_\ell$ is a tree, and its diameter is at most $h + 2h\ell + 2h + \ell$. This figure shows how C could look like:



Call p_i *blocked by* F_0 if it contains some edge from F_0; at most $|F_0|$ of the ℓ paths are blocked by F_0. Let p' be an $\tilde{F}_1$-path different from p. We say p' *blocks* p_i if the vertex sets of $\varphi(p')$ and $\varphi(p_i)$ intersect.

▶ **Proposition 12.** *Let path p' in $\tilde{F}_1$ be different from p. Then p' blocks at most one of the paths $p_1, \dots, p_\ell$.*

Proof. Let $C = p \cup p_1 \cup \dots \cup p_\ell$. As argued above, this is a tree in G and its diameter is less than girth(H). By Proposition 11, its image $\varphi(C)$ is a tree in H, isomorphic to C. Suppose, for the sake of contradiction, that $\varphi(p')$ intersects $\varphi(p_i)$ and $\varphi(p_j)$. This scenario would look like this:



Since $\varphi(p')$ and $\varphi(p)$ do not share any vertex (by Invariant 3), the subgraph $\varphi(p') \cup \varphi(p_i) \cup \varphi(p_j) \cup \varphi(p)$ contains a cycle. This cycle has size at most $|p'|+|p_i|+|p_j|+|q| \leq 2h\ell+2h+\ell+3h$, a contradiction. ◀

Call p_i *blocked by* $\tilde{F}_1$ if there is some $\tilde{F}_1$-path different from p that blocks p_i. By Proposition 12, at most $|F_1| - 1$ paths p_i are blocked by $\tilde{F}_1$. Thus, a total of at most $|F_0| + |F_1| - 1 \leq \ell - 1$ of the paths p_i are blocked by F_0 or $\tilde{F}_1$. Thus, there exists some path p_i, $1 \leq i \leq \ell$, that is not blocked. We now modify p by removing the edges on the path $v_1, v_2, \ldots, a_i$ and adding p_i. Let $\hat{p}$ denote the new version of p and $\hat{F}_1$ the new version of $\tilde{F}_1$. Note that $F_1 \subseteq \hat{F}_1$ still holds, since we only modify the set $\tilde{F}_1 \setminus F_1$. Obviously, $\hat{F}$ satisfies Invariants 1, 2, and 4. Since p_i is not blocked by F_0, $\hat{F}$ is disjoint from F_0; because p_i is not blocked by $\tilde{F}_1$, Invariant 3 still holds. Invariant 5 might be violated: $\hat{p}$ might be too long. We will deal with this in a minute.

Note that e is now either outside $\hat{F}_1$, and Delayer can include it into F_0; or it is inside $\hat{p}$, but then it is not among the first or last h edges of $\hat{p}$, and Delayer can include it into F_1.

It remains to address the possibility that $\hat{p}$ is too long, violating Invariant 5. If indeed $\hat{p}$ has more than $2h\ell + 2h + \ell$ edges, then it must somewhere contain $2h + 1$ consecutive edges that are not in F_1 (note that $|F_1| \leq \ell$). Let $e_0, \ldots, e_{2h}$ be these edges. Define $\hat{F}_1 := \tilde{F}_1 \setminus \{e_h\}$. That is, we split $\hat{p}$ into two parts, the first ending in $e_0, \ldots, e_{h-1}$, the second starting with $e_{h+1}, \ldots, e_{2h}$. Note that this satisfies Invariant 4. If one of these paths contains no edge from F_1 at all, we delete it from $\hat{F}$. We continue this process until all paths in $\hat{F}$ have size at most $2h\ell + 2h + \ell$. The final $\hat{F}$ satisfies all invariants.

5 Conclusion

We constructed close to tight hard instances for the PPSZ algorithm which uses bounded width resolution to derive values and showed that the savings can be at most $\frac{(1+\epsilon)2}{k}$. Several questions of various levels interest remain open. The first one is to obtain Super Strong ETH hard instance for resolution of larger width, ideally as close to $m/\log(m)$ as possible. Even for the weak heuristic, the hard instances from [9] hold for subformulas of size up to $m^{O(1)}$. The next problem is determining the precise constant in the savings of PPSZ.

References

1 Dana Angluin and A Gardiner. Finite common coverings of pairs of regular graphs. *Journal of Combinatorial Theory, Series B*, 30(2):184–187, 1981. doi:10.1016/0095-8956(81)90062-9.

2 Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.

3 Shiteng Chen, Dominik Scheder, Navid Talebanfard, and Bangsheng Tang. Exponential lower bounds for the PPSZ k-SAT algorithm. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1253–1263. SIAM, 2013. doi:10.1137/1.9781611973105.91.

4 Paul Erdős and Horst Sachs. Reguläre graphen gegebener taillenweite mit minimaler knotenzahl. (regular graphs with given girth and minimal number of knots.). *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg, Math.-Naturwiss.*, 12:251–258, 1963.

5 Thomas Dueholm Hansen, Haim Kaplan, Or Zamir, and Uri Zwick. Faster k-sat algorithms using biased-ppsz. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 578–589. ACM, 2019. doi:10.1145/3313276.3316359.

6 Timon Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014. doi:10.1137/120868177.

7 Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005. doi:10.1145/1066100.1066101.

8 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

9 Pavel Pudlák, Dominik Scheder, and Navid Talebanfard. Tighter hard instances for PPSZ. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 85:1–85:13, 2017. doi:10.4230/LIPIcs.ICALP.2017.85.

10 Dominik Scheder. PPSZ for $k \geq 5$: More is better. *TOCT*, 11(4):25:1–25:22, 2019. doi:10.1145/3349613.

11 Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 410–414. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814612.

12 Nikhil Vyas and R. Ryan Williams. On super strong ETH. In *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, pages 406–423, 2019. doi:10.1007/978-3-030-24258-9_28.

A Existence of Common Lift

▶ **Theorem 13** (Angluin and Gardiner [1]). *Let G and H be k-regular graphs. Then there exists a k-regular graph L that is a common lift of both G and H. Furthermore, $|V(L)| \leq 4|V(G)| \cdot |V(H)|$.*

Proof. Suppose first that both $G = (U, E)$ and $H = (V, F)$ are bipartite. By Hall's Theorem, each has a perfect matching, and in fact, we can partition E and F into k perfect matchings each: $E = E_1 \uplus \cdots \uplus E_k$ and $F = F_1 \uplus \cdots \uplus F_k$. The common lift L has vertex set $U \times V$ and edge set

$$\bigcup_{i=1}^{k} \left\{ \{(u,v),(u',v')\} \in \binom{U \times V}{2} \mid \{u,u'\} \in E_i, \{v,v'\} \in F_i \right\} .$$

It is not difficult to see that the projections $\varphi_G : (u,v) \mapsto u$ and $\varphi_H(u,v) \mapsto v$ are locally bijective homomorphisms from L into G and H, respectively.

If G (or H or both) fails to be bipartite, we first replace it by its 2-*lift* G_2. The vertex set of G_2 is $U \times \{0,1\}$, and we form its edge set by creating, for each $\{u,v\} \in E$, two edges $\{(u,0),(v,1)\}$ and $\{(u,1),(v,0)\}$. The graph G_2 is bipartite, and projection to the first coordinate is a locally bijective homomorphism. Finally, observe that the composition of locally bijective homomorphisms is again a locally bijective homomorphism. Altogether, we can replace G and H by their respective 2-lifts G_2 and H_2; these are bipartite graphs, so we find a common lift L on $4|U| \cdot |V|$ vertices. ◀

Approximability of the Eight-Vertex Model

Jin-Yi Cai
University of Wisconsin-Madison, WI, USA
http://pages.cs.wisc.edu/~jyc/
jyc@cs.wisc.edu

Tianyu Liu
University of Wisconsin-Madison, WI, USA
http://pages.cs.wisc.edu/~tl/
tl@cs.wisc.edu

Pinyan Lu
Shanghai University of Finance and Economics, China
http://itcs.shufe.edu.cn/pinyan/
lu.pinyan@mail.shufe.edu.cn

Jing Yu
Georgia Institute of Technology, Atlanta, GA, USA
jingyu@gatech.edu

Abstract

We initiate a study of the classification of approximation complexity of the eight-vertex model defined over 4-regular graphs. The eight-vertex model, together with its special case the six-vertex model, is one of the most extensively studied models in statistical physics, and can be stated as a problem of counting weighted orientations in graph theory. Our result concerns the approximability of the partition function on all 4-regular graphs, classified according to the parameters of the model. Our complexity results conform to the phase transition phenomenon from physics.

We introduce a *quantum decomposition* of the eight-vertex model and prove a set of *closure properties* in various regions of the parameter space. Furthermore, we show that there are extra closure properties on 4-regular planar graphs. These regions of the parameter space are concordant with the phase transition threshold. Using these closure properties, we derive polynomial time approximation algorithms via *Markov chain Monte Carlo*. We also show that the eight-vertex model is NP-hard to approximate on the other side of the phase transition threshold.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Approximate complexity, the eight-vertex model, Markov chain Monte Carlo

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.4

Related Version A full version of the paper is available at https://arxiv.org/abs/1811.03126.

Funding *Jin-Yi Cai*: Supported by NSF CCF-1714275.
Tianyu Liu: Supported by NSF CCF-1714275.

1 Introduction

The eight-vertex model is one of the most important vertex models in statistical physics [2]. Given a 4-regular graph G, an *even orientation* assigns a direction to every edge such that the number of arrows into (and out of) each vertex is even. In the *unweighted* case, the problem is to count the number of even orientations of G, and this is computable in polynomial time. In general there are *weights* associated with local configurations, and the problem is to compute a weighted sum called the partition function. This becomes a challenging problem, and the complexity picture becomes more intricate [6].

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 4; pp. 4:1–4:18

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

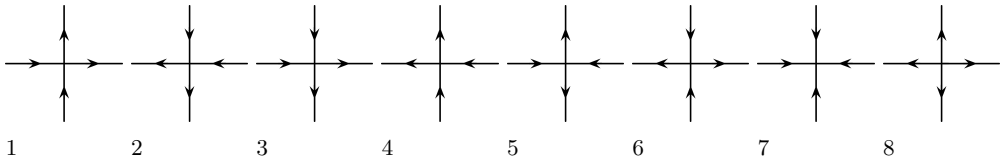


Figure 1 Valid configurations of the eight-vertex model.

In physics, the eight-vertex model is defined on a square lattice region where each vertex of the lattice is connected by an edge to four nearest neighbors, with eight permitted local configurations (see Figure 1). They are associated with eight possible weights $w_1, \ldots, w_8$. In physics, it is typically assumed that the weight is unchanged if all arrows are flipped. In this case we write $w_1 = w_2 = a$, $w_3 = w_4 = b$, $w_5 = w_6 = c$, and $w_7 = w_8 = d$. This is called *arrow reversal symmetry*. In this paper, we make this assumption and further assume that $a, b, c, d \geq 0$, as in *classical* physics. Given a 4-regular graph G, we label four incident edges of each vertex from 1 to 4. The *partition function* of the eight-vertex model with parameters (a, b, c, d) on G is defined as

$$Z(G) = Z(G; a, b, c, d) = \sum_{\tau \in \mathcal{O}_{\mathbf{e}}(G)} a^{n_1+n_2} b^{n_3+n_4} c^{n_5+n_6} d^{n_7+n_8}, \tag{1}$$

where $\mathcal{O}_{\mathbf{e}}(G)$ is the set of all even orientations of G, and n_i is the number of vertices in type i in G ($1 \leq i \leq 8$, locally depicted as in Figure 1) under $\tau \in \mathcal{O}_{\mathbf{e}}(G)$. The famous *six-vertex model* is the special case $d = 0$, i.e., only Figure 1-1 to Figure 1-6 are allowed. In this case, states are *Eulerian orientations*. Further special cases include the ice ($a = b = c$), KDP, and Rys F models. On the square lattice some other important models such as the dimer and zero-field Ising models can be reduced to it [2]. By any metric, these are among the most studied models in statistical physics.

As the problem is to compute the *partition function* $Z(G)$, naturally one should study its computational complexity. For the exact complexity, a dichotomy has been proved [6]. For most parameters, the problem is #P-hard. Regarding approximate complexity, to our best knowledge, there is only one previous result in this regard due to Greenberg and Randall [15]. They showed that on square lattice regions a specific Markov chain is torpidly mixing when d is large. It means that when sinks and sources have large weights, this particular chain cannot be used to approximately sample eight-vertex configurations on the square lattice according to the Gibbs measure.

In this paper we initiate a study toward a classification of the approximate complexity of $Z(G)$ on 4-regular graphs. Our results conform to the *order-disorder phase transitions* of the eight-vertex model in physics. (See the book [2] for more details; the full paper gives a brief description.)

To state our theorems and proofs, we adopt the following notations, for $a, b, c, d \in \mathbb{R}^+$.

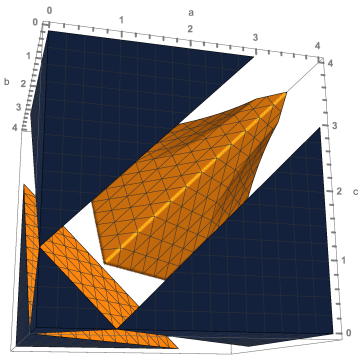
- $\mathcal{F}_{\leq^2} := \{(a, b, c, d) \mid a^2 \leq b^2 + c^2 + d^2, \quad b^2 \leq a^2 + c^2 + d^2, \quad c^2 \leq a^2 + b^2 + d^2, \quad d^2 \leq a^2 + b^2 + c^2\}$;
- $\mathcal{F}_> := \{(a, b, c, d) \mid a > b + c + d \text{ or } b > a + c + d \text{ or } c > a + b + d \text{ or } d > a + b + c$ where at least two of $a, b, c, d > 0\}$;
- $\mathcal{A}_\leq := \{(a, b, c, d) \mid a + d \leq b + c\}$, $\mathcal{B}_\leq := \{(a, b, c, d) \mid b + d \leq a + c\}$, $\mathcal{C}_\leq := \{(a, b, c, d) \mid c + d \leq a + b\}$, $\mathcal{C}_\geq := \{(a, b, c, d) \mid c + d \geq a + b\}$, $\mathcal{C}_= := \{(a, b, c, d) \mid c + d = a + b\}$.

▶ Remark 1.1. We have $\mathcal{F}_{\leq^2} \subset \overline{\mathcal{F}_>}$, and $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq \subset \overline{\mathcal{F}_>}$. Clearly $\mathcal{C}_= = \mathcal{C}_\leq \bigcap \mathcal{C}_\geq$. But $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\geq \not\subseteq \overline{\mathcal{F}_>}$.

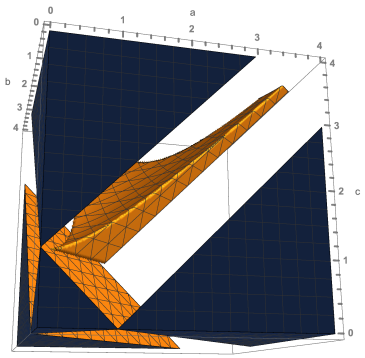
▶ **Theorem 1.2.** *There is an FPRAS* for $Z(a,b,c,d)$ if $(a,b,c,d) \in \mathcal{F}_{\leq^2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$; there is no FPRAS for $Z(a,b,c,d)$ if $(a,b,c,d) \in \mathcal{F}_{>}$ unless RP = NP. In addition, for planar graphs there is an FPRAS for $Z(a,b,c,d)$ if $(a,b,c,d) \in \mathcal{F}_{\leq^2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq}$.*

▶ Remark 1.3. The results in Theorem 1.2 are the first classification results for the approximate complexity of the eight-vertex model on general and planar 4-regular graphs, and they conform to phase transition in physics. After this work was done [10], the first two authors made a connection of the approximate complexity of the eight-vertex model to that of counting perfect matchings on general graphs (#PM) [8], a central open problem in this field. It was proved in [8] that approximating $Z(a,b,c,d)$ on general 4-regular graphs can be reduced to approximating #PM if $(a,b,c,d) \in \mathcal{F}_{\leq^2}$ and is at least as hard as approximating #PM if $(a,b,c,d) \notin \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$. The #PM-hardness result was proved by expressing the eight-vertex model partition function in the Holant framework and utilizing *holographic transformations*. At the end of this section we briefly describe the connection to Holant problems and a major motivation to study the complexity of the eight-vertex model, as part of the classification program of counting problems, quite apart from historical motivations in statistical physics.

▶ Remark 1.4. The relationship of the regions denoted by $\mathcal{F}_{\leq^2}$, $\mathcal{F}_{>}$, $\mathcal{A}_{\leq}$, $\mathcal{B}_{\leq}$, $\mathcal{C}_{\leq}$, $\mathcal{C}_{\geq}$, and $\mathcal{C}_{=}$ may not be easy to visualize, since they reside in 4-dimensional space. See Figure 2 (where we normalize $d = 1$)[†]. The roles of a, b, c, and d are not symmetric. In particular, d is the weight of sinks and sources and has a special role (e.g. see [15]). If $(a,b,c,d) \in \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$ then $d \leq a, b, c$. Surprisingly, by Theorem 1.2 FPRAS can still exist for planar graphs even when sinks/sources have large weights.



(a) The four corner regions constitute $\mathcal{F}_{>}$. The non-corner region depicted is $\mathcal{F}_{\leq^2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$.



(b) An extra region that admits FPRAS on planar graphs.

Figure 2 Regions of known complexity in the eight- vertex model.

To get these FPRAS, our most important contribution is a set of *closure properties*. See Section 2. We then use these closure properties to show that a Markov chain designed for the six-vertex model can be adapted to provide our FPRAS. The Markov chain we adapt is the *directed-loop algorithm* invented by Rahman and Stillinger [21]. The state space of

* Suppose $f : \Sigma^* \to \mathbb{R}$ is a function mapping problem instances to real numbers. A *fully polynomial randomized approximation scheme (FPRAS)* [19] for a problem is a randomized algorithm that takes as input an instance x and $\varepsilon > 0$, running in time polynomial in n (the input length) and ε^{-1}, and outputs a number Y (a random variable) such that $\Pr[(1-\varepsilon)f(x) \leq Y \leq (1+\varepsilon)f(x)] \geq \frac{3}{4}$.

† Some 3D renderings of the parameter space can be found at `https://skfb.ly/6C9LE` and `https://skfb.ly/6C9MS`.

our Markov chain for the eight-vertex model consists of even orientations and near-even orientations, which is an extension of the space of valid configurations; the transitions of this algorithm are composed of creating, shifting, and merging of two "defective" edges.

This leads to a *Markov chain Monte Carlo* approximate counting algorithm by *sampling*. To prove that this is an FPRAS, we show that (1) the above Markov chain is rapidly mixing via a *conductance argument* [17, 12, 22, 16], (2) the valid configurations take a non-negligible proportion in the state space, and (3) there is a (not totally obvious) self-reduction (to reduce the computation of the partition function of a graph to that of a "smaller" graph) [18]. All three parts depend on the closure properties. Specifically, we show that when $(a, b, c, d) \in \mathcal{F}_{\leq^2}$, the conductance of the Markov chain can be polynomially bounded *if* the ratio of near-even orientations over even orientations can be polynomially bounded; when $(a, b, c, d) \in \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$, this ratio is indeed polynomially bounded according to the closure properties. Finally a self-reduction whose success in $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$ requires an additional closure property. Therefore, there is an FPRAS in the intersection of $\mathcal{F}_{\leq^2}$ and $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$ which is in the *disordered phase*.

A 4-ary construction is a 4-regular graph Γ with four "dangling" edges. This defines a *constraint function* of arity 4. In Theorem 2.2 we show that the set of 4-ary constraint functions in $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$ is closed under 4-ary constructions. This is achieved by inventing a *"quantum decomposition"* of even-orientations. To define this, given an even orientation, a *plus pairing* groups the four edges around a vertex into two pairs such that both pairs satisfy "1-in-1-out"; a *minus pairing* groups the four edges around a vertex into two pairs such that both pairs independently satisfy either "2-in" or "2-out". With weights, this leads to a weighted sum of $3^{|V|}$ *"annotated" circuit partitions*. (Details are in Section 2[‡].)

We use these tools to derive our FPRAS. Surprisingly, for planar graphs in the eight-vertex model we can show an additional region where FPRAS exists (also in the *disordered phase*). For planar graphs, in Theorem 2.3 we show that the extra regions $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq} \bigcap \overline{\mathcal{F}_{>}}$ and $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{=}$ also enjoy closure properties. This leads to an FPRAS on planar graphs when the parameter setting is in the intersection of $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq} \bigcap \overline{\mathcal{F}_{>}}$ and $\mathcal{F}_{\leq^2}$. And since $\mathcal{F}_{\leq^2} \subset \overline{\mathcal{F}_{>}}$, combined with the FPRAS on general graphs, we get an FPRAS for $\mathcal{F}_{\leq^2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq}$ for all planar graphs. Considering the fact that the exact complexity for the eight-vertex model on planar graphs is not even understood, this is one of the very few cases where research on approximate complexity has advanced beyond that on exact complexity.

The NP-hardness of approximation in the whole *ordered phases* is shown by reductions from the problem of computing the maximum cut on a 3-regular graph. For the eight-vertex models not included in the six-vertex model ($d \neq 0$), both the reduction source and the "gadgets" we employ to prove the hardness are substantially different from those used in the hardness proof of the six-vertex model [9]. We note that the parameter settings in [15] where torpid mixing is proved are contained in our NP-hardness region.

In addition to the complexity result, we show that there is a fundamental difference in the behavior on the two sides separated by the phase transition threshold, in terms of closure properties. In Theorem 2.2, we show that the set of 4-ary constraint functions lying in the complement of $\mathcal{F}_{>}$ is closed under 4-ary constructions. We prove in this paper that approximation is hard on $\mathcal{F}_{>}$. It is not known if the eight-vertex model in the full region of $\overline{\mathcal{F}_{>}}$ admits FPRAS or not.

[‡] We use the term "quantum" is emphasize that these are linear combinations, or superpositions, of objects called annotated circuit partitions. There are similarities to holographic transformations, where cancellations occur in the analysis. However, currently we do not have any direct link to quantum computing.

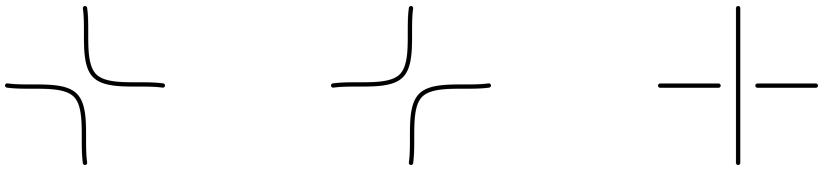
Aside from statistical physics, perhaps a more direct link of the study of partition functions of the eight-vertex model and complexity theory is the classification program of counting problems [4]. The eight-vertex model fits into the wider class of Holant problems. Holant problems are a broad class of counting problems that are more general and expressive than counting constraint satisfaction problems [14, 7], for which complexity dichotomies have been proved [3, 13, 11, 5]. Previous complexity dichotomy theorems have achieved a complete classification for the exact complexity of Holant problems for any set of *symmetric* constraint functions [4]. It turns out that the eight-vertex model is the special case of a Holant problem. In fact, the eight-vertex model can be expressed precisely as a Holant problem in the orientation setting with a single arity-4 *non-symmetric* constraint function that satisfies a parity condition. Under a suitable holographic transformation, it can be expressed as a Holant problem in a standard form. The worst case complexity of the eight-vertex model has been proved in [6]. It is hoped that this set of Holant problems serves as a base case for a future complete worst case complexity classification of Holant problems. The results in this paper are instead on the approximate complexity of the eight-vertex model, and are technically distinct. But a major motivation comes from the classification program on counting problems.

Previous results in approximate counting are mostly about spin systems. The present paper, together with [9], is probably the first fruitful attempt in the Holant literature to make connections to phase transitions. While there is still a gap in the complexity picture for the six-vertex and eight-vertex models, we believe the framework set in this paper gives a starting point for studying the approximate complexity of a broader class of counting problems.

2 Closure Properties

We introduce a "quantum decomposition" for the eight-vertex model, in which every configuration on G is expressed as a "superposition" of $3^{|V|}$ annotated circuit partitions.

Let v be a vertex of G, and e_1, e_2, e_3, e_4 the four labeled edges incident to v. A *pairing* ϱ at v is a partition of $\{e_1, e_2, e_3, e_4\}$ into two pairs. There are exactly three distinct pairings at v (Figure 3) which we denote by three special symbols: ⌐⌐, ⌐⌐, ⊣⊢, respectively. A *circuit partition* of G is a partition of the edges of G into edge-disjoint circuits (in such a circuit vertices may repeat but edges may not). It is in 1-1 correspondence with a family of pairings $\varphi = \{\varrho_v\}_{v \in V}$, where $\varrho_v \in \{⌐⌐, ⌐⌐, ⊣⊢\}$ is a pairing at v—once the pairing at each vertex is fixed, then the two edges paired together at each vertex is also adjacent in the same circuit.



(a) ⌐⌐ **(b)** ⌐⌐ **(c)** ⊣⊢

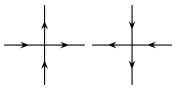
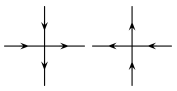
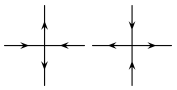
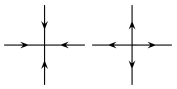
Figure 3 (Unsigned) pairings at a degree 4 vertex.

A *signed pairing* $\boldsymbol{\varrho}_v$ at v is a pairing with a sign, either *plus* (+) or *minus* (−). We denote a signed pairing by ϱ_+ or ϱ_- if the pairing is ϱ and the sign is plus or minus, respectively. An *annotated circuit partition* of G, or *acp* for short, is a circuit partition of G together with

a map $V \to \{+, -\}$ such that along every circuit one encounters an even number of $-$ (a repeat vertex with $-$ counts twice on the circuit). Thus, it is in 1-1 correspondence with a family of signed pairings for all $v \in V$, with the restriction that there is an *even* number of $-$ along each circuit. Each circuit C in an *acp* has exactly two directed *states* – starting at an arbitrary edge in C with one of the two orientations on this edge, one can uniquely orient every edge in C such that for every vertex v on C, two edges incident at v paired up by $+$ have consistent orientations at v (i.e., they form "1-in-1-out" at v), whereas two edges paired up by $-$ have contrary orientations at v (i.e., they form "2-in" or "2-out" at v). These two directed states of C are well-defined because cyclically the direction of edges along C changes an even number of times, precisely at the minus signs. A *directed annotated circuit partition* (*dacp*) is an *acp* with each circuit in a directed state. If an *acp* has k circuits, then it defines 2^k *dacp*'s.

Next we describe an association between even orientations and *acp*'s as well as *dacp*'s. Given an even orientation τ of G, every local configuration of τ at a vertex defines exactly three signed pairings at this vertex according to Table 1. Note that, given τ and a pairing at a vertex v, the two pairs have either *both* consistent or *both* contrary orientations. Thus the same sign, $+$ or $-$, works for both pairs, although this depends on the pairing at v.

Table 1 Map from eight local configurations to signed pairings.

Configurations	**Weight**	**Sign**		
				
	a	-	$+$	$+$
	b	$+$	-	$+$
	c	$+$	$+$	-
	d	-	-	-

In this way, every even orientation τ defines $3^{|V|}$ *acp*'s, denoted by $\Phi(\tau)$. See Table 2 and Table 3 for two examples. Moreover, for any *acp* $\varphi \in \Phi(\tau)$, every circuit in φ is in one of the two well-defined directed states under the orientation τ. Thus each even orientation τ defines $3^{|V|}$ *dacp*'s.

Conversely, for any *dacp*, if we ignore the signs at all vertices we get a valid even orientation (because each sign applies to both pairs). If a *dacp* comes from $\Phi(\tau)$ then we get back the even orientation τ. Therefore, the association from even orientations to *dacp*'s is 1-to-$3^{|V|}$, non-overlapping, and surjective. For every vertex v with the constraint function parameters (a, b, c, d), we define a local weight function w that assigns a local weight to the six signed pairings at v, such that

$$\begin{cases} a = w(\curvearrowleft_-) + w(\curvearrowright_+) + w(\vdash\!\!\dashv_+) \\ b = w(\curvearrowleft_+) + w(\curvearrowright_-) + w(\vdash\!\!\dashv_+) \\ c = w(\curvearrowleft_+) + w(\curvearrowright_+) + w(\vdash\!\!\dashv_-) \\ d = w(\curvearrowleft_-) + w(\curvearrowright_-) + w(\vdash\!\!\dashv_-) \end{cases}. \tag{2}$$

Table 2 An even orientation and its quantum decomposition into *acp*'s.

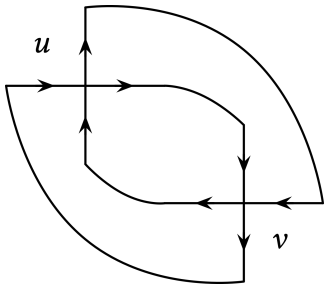
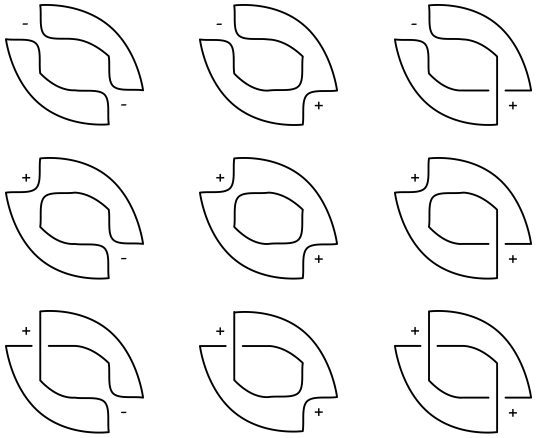
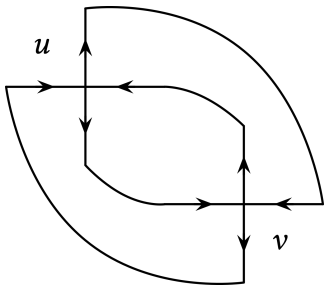
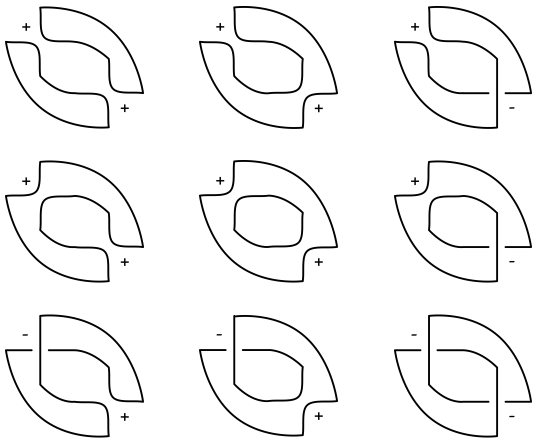
τ	$\Phi(\tau)$
	

Table 3 Another even orientation and its quantum decomposition into *acp*'s.

τ	$\Phi(\tau)$
	

Note that for any a, b, c, d this is a linear system of rank 4 in six variables, and there is a solution space of dimension 2 (Lemma 2.5 discusses this freedom). Define the weight $\tilde{w}(\varphi)$ of an annotated circuit partition φ, either undirected (*acp*) or directed (*dacp*), be the product of weights at each vertex. Then the weight of an eight-vertex model configuration τ is equal to $\sum_{\varphi \in \Phi(\tau)} \tilde{w}(\varphi)$. This is obtained by writing a term in the summation in (1), which is a product of sums by (2), as a sum of products. Note that a single *acp* has the same weight when it becomes directed regardless which directed state the *dacp* is in.

We will illustrate the above in detail by the examples in Table 2 and Table 3. We assume the same constraint (a, b, c, d) is applied at u and v. The orientation at one vertex determines the other in this graph G. There are a total 8 valid configurations, 4 of which are total reversals of the other 4. $Z(G) = 2[a^2 + b^2 + c^2 + d^2]$. When we expand $Z(G)$ using (2) we

get a total of 72 terms. These correspond to 72 *dacp*'s. There are 9 ways to assign a pairing at u and at v. If we consider the configuration in Table 2, these 9 ways are listed under $\Phi(\tau)$, where the local orientation also determines a sign $\pm$ at both u and v. These are 9 *acp*'s (without direction). (Here we take the view of decomposing an orientation into *acp*'s rather than *dacp*'s to illustrate the idea of quantum decomposition which will be exploited later in proofs.) For each *acp* φ, the weight $\tilde{w}(\varphi)$ is defined (without referring to the *dacp*, or the state of orientation on these circuits). Three of the *acp*'s (in the diagonal positions) define two distinct circuits while the other six define one circuit each. For each 2-tuple of pairings (ρ_u, ρ_v) that results in two circuits, the only valid annotations assign $(+,+)$ or $(-,-)$ at (u,v), giving a total of 6 *acp*'s. And since each has two circuits, there are a total of 24 *dacp*'s. For the other six (off-diagonal) 2-tuples of pairings (ρ_u, ρ_v) that results in a single circuit, each has 4 valid annotations, giving a total of 24 *acp*'s. But these have only one circuit and thus give 48 *dacp*'s. To appreciate the "quantum superpostion" of the decomposition, note that the same *acp* that has $(\curvearrowright_+, \curvearrowright_+)$ at (u,v) appears in both decompositions for the distinct configurations in Table 2 and Table 3[§].

▶ Remark 2.1. While a weight function w satisfying (2) is not unique, there are some regions of (a,b,c,d) that can be specified directly in terms of w by any weight function w satisfying (2), and the specification is independent of the choice of the weight function. E.g., the region $\overline{\mathcal{F}_>}$ is specified by $\begin{cases} w(\curvearrowleft_+)+w(\curvearrowright_-)+w(\dashv_-)\geq 0 \\ w(\curvearrowleft_-)+w(\curvearrowright_+)+w(\dashv_-)\geq 0 \\ w(\curvearrowleft_-)+w(\curvearrowright_-)+w(\dashv_+)\geq 0 \\ w(\curvearrowleft_+)+w(\curvearrowright_+)+w(\dashv_+)\geq 0 \end{cases}$. Also $\mathcal{A}_\leq$ is specified by $w(\curvearrowleft_-) \leq w(\curvearrowleft_+)$, $\mathcal{B}_\leq$ by $w(\curvearrowright_-) \leq w(\curvearrowright_+)$, and $\mathcal{C}_\leq$ by $w(\dashv_-) \leq w(\dashv_+)$. In Lemma 2.5 (stated later), we will show that a nonnegative weight function w satisfying (2) exists iff $(a,b,c,d) \in \overline{\mathcal{F}_>}$.

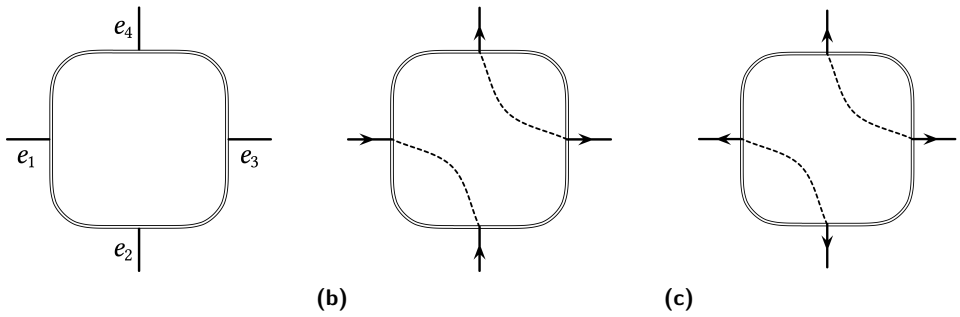


Figure 4 A 4-ary construction in the eight-vertex model.

A *4-ary construction* is a 4-regular graph Γ having four "external" edges (Figure 4a), and a constraint function on each node. It defines a 4-ary constraint function with four input variables as the partial sum in $Z(\Gamma)$ with a given assignment on the dangling edges. If we imagine the graph Γ is shrunken to a single point except the 4 external edges, then a 4-ary construction can be viewed as a virtual vertex with parameters (a',b',c',d') in the eight-vertex model, for some $a',b',c',d' \geq 0$ (satisfying the even orientation rule and arrow reversal symmetry). A *planar 4-ary construction* is a 4-regular plane graph with four dangling edges on the outer face ordered counterclockwise e_1, e_2, e_3, e_4.

We say a set of constraint functions S is closed under 4-ary constructions if the constraint function of any 4-ary construction using functions from S also belongs to S.

[§] There is a superficial similarity between the quantum decomposition and skein relations in knot theory [20]. This remains to be explored further.

▶ **Theorem 2.2.** *Constraint function sets* $\overline{\mathcal{F}_>}$ *and* $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq$ *are closed under 4-ary constructions.*

▶ **Theorem 2.3.** *Constraint function sets* $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\geq \bigcap \overline{\mathcal{F}_>}$ *and* $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_=$ *are closed under 4-ary* plane *constructions.*

A *trail & circuit partition* (*tcp*) for a 4-ary construction Γ is a partition of the edges in Γ into edge-disjoint circuits and exactly two *trails* (walks with no repeated edges) which end in the four external edges. An *annotated trail & circuit partition* (*atcp*) for Γ is a *tcp* with a valid annotation, which assigns an *even* number of $-$ sign along each circuit. Like circuits, each trail in an *atcp* has exactly two directed states. If an *atcp* φ has k circuits (and 2 trails), then φ defines 2^{k+2} *directed annotated trail & circuit partitions* (*datcp*). The weight $\tilde{w}(\varphi)$ of an annotated trail & circuit partition φ, either an *atcp* or *datcp*, can be similarly defined. Again set the weight function as in (2).

For each dangling edge e_i $(1 \leq i \leq 4)$ of Γ, let us describe the state of e_i by 0 (or 1) if it is coming into (respectively going out of) Γ. Denote the constraint function of Γ by f and consider $f(0011)$. Under the eight-vertex model, if a configuration τ of the 4-ary construction with constraint function f has a nonzero contribution to $f(0011)$, it has e_1, e_2 coming in and e_3, e_4 going out. The contribution by τ is a weighted sum over a set $\Phi_{0011}(\tau)$ of *datcp*. Each *datcp* in $\Phi_{0011}(\tau)$ is captured in exactly one of the following three types, according to how e_1, e_2, e_3, e_4 are connected by the two trails:

(1) $\{ \xrightarrow{e_1} \square \xleftarrow{e_2}, \xleftarrow{e_3} \square \xrightarrow{e_4}\}$ and on both trails the numbers of minus pairings are odd; or

(2) $\{ \xrightarrow{e_1} \square \xrightarrow{e_4}, \xrightarrow{e_2} \square \xrightarrow{e_3}\}$ and on both trails the numbers of minus pairings are even (Figure 4b); or

(3) $\{ \xrightarrow{e_1} \square \xrightarrow{e_3}, \xrightarrow{e_2} \square \xrightarrow{e_4}\}$ and on both trails the numbers of minus pairings are even.

Let $\Phi_{0011,\curlywedge_-}$, $\Phi_{0011,\curlyvee_+}$ and $\Phi_{0011,|-|_+}$ be the subsets of *datcp* contributing to $f(0011)$ defined in case (1), (2) and (3) respectively. The value $f(0011)$ is a weighted sum of contributions according to $\tilde{w}$ from these three disjoint sets. Defining the weight of a set Φ of *datcp* by $W(\Phi) = \sum_{\varphi \in \Phi} \tilde{w}(\varphi)$ yields $f(0011) = W(\Phi_{0011,\curlywedge_-}) + W(\Phi_{0011,\curlyvee_+}) + W(\Phi_{0011,|-|_+})$. Similarly we can define $\Phi_{1100,\curlywedge_-}$, $\Phi_{1100,\curlyvee_+}$ and $\Phi_{1100,|-|_+}$, and get $f(1100) = W(\Phi_{1100,\curlywedge_-}) + W(\Phi_{1100,\curlyvee_+}) + W(\Phi_{1100,|-|_+})$. Note that there is a bijective weight-preserving map between $\Phi_{0011,\curlywedge_-}$ and $\Phi_{1100,\curlywedge_-}$ by reversing the direction of every circuit and trail of a *datcp*. Thus, $W(\Phi_{0011,\curlywedge_-}) = W(\Phi_{1100,\curlywedge_-})$, $W(\Phi_{0011,\curlyvee_+}) = W(\Phi_{1100,\curlyvee_+})$, and $W(\Phi_{0011,|-|_+}) = W(\Phi_{1100,|-|_+})$. Consequently $f(0011) = f(1100)$. Similarly we have $f(0110) = f(1001)$, $f(0101) = f(1010)$ and $f(0000) = f(1111)$. For any pairing ϱ, and for every 4-bit pattern $b_1b_2b_3b_4 \in \{0,1\}^4$, we can define $\Phi_{b_1b_2b_3b_4,\varrho_+}$ if (both) paired $b_i \neq b_j$, and $\Phi_{b_1b_2b_3b_4,\varrho_-}$ if (both) paired $b_i = b_j$. We can prove the following and call the common value $W(\curlywedge_-)$:

- $W(\Phi_{0011,\curlywedge_-}) = W(\Phi_{1100,\curlywedge_-}) = W(\Phi_{0000,\curlywedge_-}) = W(\Phi_{1111,\curlywedge_-})$.

We can prove 5 other sets of similar equalities, and call them $W(\curlyvee_-), W(|-|_-), W(\curlywedge_+)$, $W(\curlyvee_+)$, and $W(|-|_+)$. Consequently, f has parameters $\begin{cases} a'=W(\curlywedge_-)+W(\curlyvee_+)+W(|-|_+) \\ b'=W(\curlywedge_+)+W(\curlyvee_-)+W(|-|_+) \\ c'=W(\curlywedge_+)+W(\curlyvee_+)+W(|-|_-) \\ d'=W(\curlywedge_-)+W(\curlyvee_-)+W(|-|_-) \end{cases}$.

Proof Sketch for the closure of $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq$**.** By definition $(a,b,c,d) \in \mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq$ means that $\begin{cases} a+d \leq b+c \\ b+d \leq a+c \\ c+d \leq a+b \end{cases}$. By the weight function w defined in (2) this is equivalent to $\begin{cases} w(\curlywedge_+) \geq w(\curlywedge_-) \\ w(\curlyvee_+) \geq w(\curlyvee_-) \\ w(|-|_+) \geq w(|-|_-) \end{cases}$. Since $\mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq \subset \overline{\mathcal{F}_>}$, by Lemma 2.5 (stated later) we can assume w

is nonnegative. Therefore, we only need to establish $\begin{cases} W(\curlywedge_+) \geq W(\curlywedge_-) \\ W(\curlyvee_+) \geq W(\curlyvee_-) \\ W(\dashv\vdash_+) \geq W(\dashv\vdash_-) \end{cases}$. We prove $W(\curlywedge_+) \geq W(\curlywedge_-)$. Proof for the other two inequalities is symmetric.

An *atcp* is a *tcp* together with a valid annotation. Consider the set Ψ of *tcp* such that the two (unannotated) trails connect e_1 with e_2, and e_3 with e_4. Denote by χ_{12} (respectively χ_{34}) the trail in ψ connecting e_1 and e_2 (respectively e_3 and e_4). Each *tcp* $\psi \in \Psi$ may have many valid annotations.

Since Γ is 4-regular, any vertex inside Γ appears exactly twice counting multiplicity in a *tcp* ψ. It appears either as a self-intersection point of a trail or a circuit, or alternatively in exactly two distinct trails/circuits. So when traversed, in total one encounters an even number of $-$ among all circuits and the two trails in any valid annotation of ψ, and since one encounters an even number of $-$ along each circuit, the numbers of $-$ along χ_{12} and χ_{34} have the same parity. We say a valid annotation of ψ is *positive* if there is an even number of $-$ along χ_{12} (and χ_{34}), and *negative* otherwise.

To prove $W(\curlywedge_+) \geq W(\curlywedge_-)$, it suffices to prove that for each *tcp* $\psi \in \Psi$, the total weight W_+ contributed by the set of positive annotations of ψ is at least the total weight W_- contributed by the set of negative annotations of ψ. We prove this nontrivial statement by induction on the number N of vertices shared by any two distinct circuits in ψ.

Base case: The base case is $N = 0$. In the base case, we can deal with self-intersections on trails and circuits easily, so let us assume that no trail or circuit is self-intersecting. Then every vertex on any circuit C of ψ is shared by C and exactly one trail, χ_{12} or χ_{34}. Also, every vertex on χ_{12} or χ_{34} is shared with some circuit or the other trail.

We will account for the product values of $w(\boldsymbol{\varrho}_v)$ according to how v is shared. We first consider shared vertices of a circuit $C \in \psi$ with the trails. Let $s, t \geq 0$ be the numbers of vertices C shares with χ_{12} and χ_{34}, respectively. Let x_i $(1 \leq i \leq s)$ (if $s > 0$) and y_j $(1 \leq j \leq t)$ (if $t > 0$) be these shared vertices respectively (for $s = 0$ or $t = 0$, the statements below are vacuously true). For any v, if ϱ is the pairing at v according to ψ, then let $w_+(v) = w(\varrho_+)$, and $w_-(v) = w(\varrho_-)$, both at v. In any valid annotation of ψ (either positive or negative), one encounters an even number of $-$ on the vertices along C, each of which is shared with exactly one of χ_{12} and χ_{34}. Hence the number of $-$ in x_i $(1 \leq i \leq s)$ has the same parity as the number of $-$ in y_j $(1 \leq j \leq t)$. Other than having the same parity, the annotation for x_i $(1 \leq i \leq s)$ is independent from the annotation for y_j $(1 \leq j \leq t)$ for a valid annotation, and from the annotations on other circuits. Let $S_+(C)$ (respectively $S_-(C)$) be the sum of products of $w(\boldsymbol{\varrho}_v)$ over $v \in \{x_i \mid 1 \leq i \leq s\}$, summed over valid annotations such that the number of $-$ in x_i $(1 \leq i \leq s)$ is even (respectively odd). Similarly let $T_+(C)$ (respectively $T_-(C)$) be the corresponding sums for y_j $(1 \leq j \leq t)$. We have

$$S_+(C) - S_-(C) = \prod_{i=1}^{s} (w_+(x_i) - w_-(x_i)) \geq 0,$$
$$T_+(C) - T_-(C) = \prod_{j=1}^{t} (w_+(y_j) - w_-(y_j)) \geq 0.$$

Both differences are nonnegative by the hypothesis.

The product $S_+(C)T_+(C)$ is the sum over all valid annotations of vertices on C such that the numbers of $-$ on vertices shared by χ_{12} and C and by χ_{34} and C are both even. Similarly $S_-(C)T_-(C)$ is the sum over all valid annotations of vertices on C such that the numbers of $-$ on vertices shared by χ_{12} and C and by χ_{34} and C are both odd. We have $S_+(C)T_+(C) \geq S_-(C)T_-(C)$.

Next we also account for the vertices shared by χ_{12} and χ_{34} in ψ. Let p be this number and if $p > 0$ let z_k $(1 \le k \le p)$ be these vertices. Let q be the number of circuits in ψ, denoted by C_l $(1 \le l \le q)$ (if $q > 0$). Then we claim that

$$W_+ - W_- = \prod_{k=1}^{p} (w_+(z_k) - w_-(z_k)) \prod_{l=1}^{q} (S_+(C_l)T_+(C_l) - S_-(C_l)T_-(C_l)),$$

and in particular $W_+ - W_- \ge 0$. To prove this claim we only need to expand the product, and separately collect terms that have a $+$ sign and a $-$ sign. In a product term in the fully expanded sum, let p' be the number of $-w_-(z_k)$, and q' be the number of $-S_-(C_l)T_-(C_l)$. Then a product term has a $+$ sign (and thus included in W_+) iff $p' + q' \equiv 0 \pmod 2$.

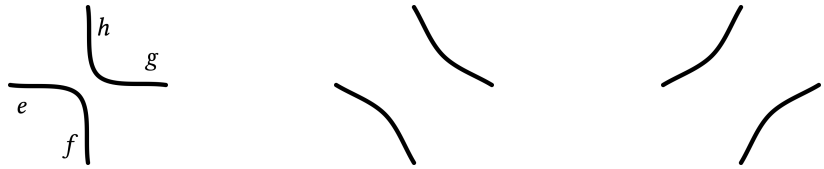


Figure 5 Possible ways of deleting a vertex. The vertex (not explicitly shown) at the center of part (a) is removed in part (b) and (c).

Induction step: Suppose v is a shared vertex between two distinct circuits C_1 and C_2, and let $\{e, f, g, h\}$ be its incident edges in Γ. We may assume the pairing ϱ_v in ψ is $\{e, f\}$ and $\{g, h\}$, and thus e, f are in one circuit, say C_1, while g, h are in another circuit C_2 (Figure 5a). Define Γ' to be the 4-ary construction obtained from Γ by deleting v and merging e with f, and g with h (Figure 5b). Define Γ'' to be the 4-ary construction obtained from Γ by deleting v and merging e with h, and f with g (Figure 5c). Note that in Γ', we have two circuits C_1' and C_2' (each has one fewer vertex v from C_1 and C_2), but in Γ'' the two circuits are merged into one C^*. Define W_+' and W_-' (respectively W_+'' and W_-'') similarly for Γ' (respectively Γ'') with tcp being $\psi' = \psi \setminus \{\varrho_v\}$.

We can decompose $W_+ - W_-$ according to whether the sign on ϱ_v is $+$ or $-$. Recall that for any valid annotation of ψ, one encounters an even number of $-$ along C_1 and C_2. If the sign on ϱ_v is $+$, the number of $-$ along C_1 (and C_2) at all vertices other than v in any valid annotation is always even; if the sign on ϱ_v is $-$, this number (for both C_1 and C_2) is always odd. $W_+ - W_-$ can be decomposed into two parts, corresponding to terms with ϱ_v being $+$ or $-$ respectively. All terms of the first (and second) part have a factor $w_+(v)$ (and $w_-(v)$ respectively). And so we can write

$$W_+ - W_- = w_+(v)[W_+ - W_-]_e + w_-(v)[W_+ - W_-]_o, \tag{3}$$

where $[W_+ - W_-]_e$ and $[W_+ - W_-]_o$ collect terms in $W_+ - W_-$ in the first and second part respectively, but without the factor at v. However by considering valid annotations for Γ' we also have

$$W_+' - W_-' = [W_+ - W_-]_e, \tag{4}$$

because a valid annotation on both C_1' and C_2' is equivalent to a valid annotation on both C_1 and C_2 with v assigned $+$. Similarly, by considering valid annotations for Γ'' we also have

$$W_+'' - W_-'' = [W_+ - W_-]_e + [W_+ - W_-]_o, \tag{5}$$

because depending on whether ϱ_v is assigned $+$ or $-$, a valid annotation on both C_1 and C_2 gives either both an even or both an odd number of $-$ on $C_1 \setminus \{v\}$ and $C_2 \setminus \{v\}$, which is equivalent to an even number of $-$ on the merged circuit C^*. From (3,4,5) we have

$$W_+ - W_- = (w_+(v) - w_-(v))(W'_+ - W'_-) + w_-(v)(W''_+ - W''_-).$$

By induction, $W'_+ \geq W'_-$ and $W''_+ \geq W''_-$. Since $w_+(v) \geq w_-(v)$ is given by hypothesis, we get $W_+ \geq W_-$. ◀

The proof of planar closures is similar but more intricate, and uses the *Jordan Curve Theorem*.

▶ **Lemma 2.4.** *Suppose $x, x', y, y', z, z' \in \mathbb{R}$ satisfy the eight inequalities: $X + Y + Z \geq 0$ where $X \in \{x, x'\}, Y \in \{y, y'\}, Z \in \{z, z'\}$. Then there exist* nonnegative *$\tilde{x}, \tilde{x}', \tilde{y}, \tilde{y}', \tilde{z}, \tilde{z}'$ such that all eight sums $X + Y + Z$ are unchanged when x, x', y, y', z, z' are substituted by the respective values $\tilde{x}, \tilde{x}', \tilde{y}, \tilde{y}', \tilde{z}, \tilde{z}'$.*

▶ **Lemma 2.5.** *The parameter setting (a, b, c, d) belongs to $\overline{\mathcal{F}_>}$ iff there exists a nonnegative weight function w satisfying (2).*

▶ **Notation.** *Fix for each vertex v in a 4-regular graph G a weight function w on signed pairings (satisfying (2) at v). Let $Z_v(\boldsymbol{\varrho})$ be the weighted sum of the set of all* dacp*'s having the signed pairing $\boldsymbol{\varrho}$ at v.*

▶ **Corollary 2.6.** *If at each vertex in a 4-regular graph G we have a nonnegative weight function w such that $w(\curvearrowleft_+) \geq w(\curvearrowleft_-)$, $w(\curvearrowright_+) \geq w(\curvearrowright_-)$, and $w(\dashv\vdash_+) \geq w(\dashv\vdash_-)$, then $Z_v(\curvearrowleft_+) \geq Z_v(\curvearrowleft_-)$, $Z_v(\curvearrowright_+) \geq Z_v(\curvearrowright_-)$, and $Z_v(\dashv\vdash_+) \geq Z_v(\dashv\vdash_-)$ at each vertex v in G.*

▶ **Corollary 2.7.** *If at each vertex in a 4-regular* plane *graph G we have a nonnegative weight function w such that $w(\curvearrowleft_+) \geq w(\curvearrowleft_-)$, $w(\curvearrowright_+) \geq w(\curvearrowright_-)$, and $w(\dashv\vdash_+) \leq w(\dashv\vdash_-)$, then $Z_v(\curvearrowleft_+) \geq Z_v(\curvearrowleft_-)$, $Z_v(\curvearrowright_+) \geq Z_v(\curvearrowright_-)$, and $Z_v(\dashv\vdash_+) \leq Z_v(\dashv\vdash_-)$ at each vertex v in G.*

3 FPRAS

▶ **Theorem 3.1.** *There is an FPRAS for $Z(a, b, c, d)$ if $(a, b, c, d) \in \mathcal{F}_{\leq^2} \bigcap \mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\leq$.*

▶ **Theorem 3.2.** *There is an FPRAS for $Z(a, b, c, d)$ on planar graphs if $(a, b, c, d) \in \mathcal{F}_{\leq^2} \bigcap \mathcal{A}_\leq \bigcap \mathcal{B}_\leq \bigcap \mathcal{C}_\geq$.*

We prove the FPRAS results using the common approach of approximately counting via almost uniformly sampling [18, 17, 12, 22, 16] by showing that the *directed-loop algorithm*, a Markov chain algorithm designed for the six-vertex model[¶], can be adapted for the eight-vertex model. The state space Ω of our Markov chain $\mathcal{MC}$ for the eight-vertex model consists of the set Ω_0 of *even orientations* (e.g. Figure 6a) and the set of *near-even orientations* with exactly two "defective" edges (e.g. Figure 6b and Figure 6c), which is an extension of the space of valid configurations; the transitions of this algorithm are composed of creating (from Figure 6a to Figure 6b), shifting (between Figure 6b and Figure 6c), and merging (from Figure 6b to Figure 6a) of the two defects on edges.

▶ **Notation.** *Let $\mathcal{Z}(S)$ be the weighted sum of states in the set S.*

[¶] The directed-loop algorithm was invented by Rahman and Stillinger [21] and is widely used for the six-vertex model (e.g., [24, 1, 23]).

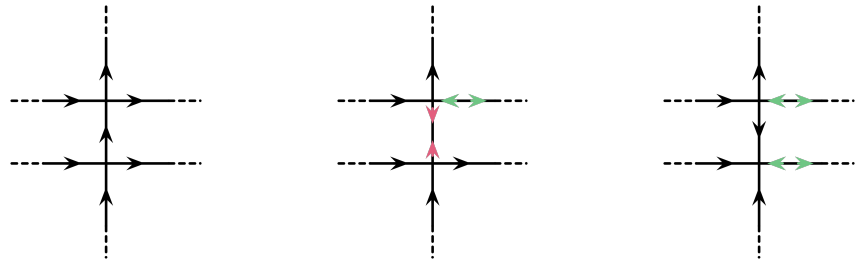


Figure 6 Examples of the states in the directed-loop algorithm.

Proof of Theorem 3.1. It is easy to show that $\mathcal{MC}$ is *irreducible* and *aperiodic*, and it satisfies the *detailed balance condition* under the Gibbs distribution. By the theory of Markov chains, we have an almost uniform sampler of $\Omega_0 \cup \Omega_2$. This sampler is efficient if $\mathcal{MC}$ is rapidly mixing. According to Lemma 3.5, when $(a, b, c, d) \in \mathcal{F}_{\leq^2}$, $\mathcal{MC}$ is rapidly mixing *if* $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)}$ is polynomially bounded via a *conductance argument* [17, 12, 22, 16] in which the *paths* between any two states τ_1 and τ_2 and the amount of *flow* each of them takes are decided by a *quantum decomposition* of the "symmetric difference" $\tau_1 \oplus \tau_2$. According to Corollary 3.3 (a corollary of the *closure property* Theorem 2.2), $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)}$ is polynomially bounded. As a consequence, if all the constraint function comes from $\mathcal{F}_{\leq^2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$, $\mathcal{MC}$ is rapidly mixing and even orientations take a non-negligible proportion in the state space. Therefore, we are able to efficiently sample valid eight-vertex configurations according to the Gibbs measure on Ω_0 (almost uniformly).

In order for self-reduction, we need to extend the type of vertices a graph allows in the eight-vertex model. Previously, a graph can only have degree 4 vertices, on each of which the constraint function satisfies the even orientation rule and has arrow reversal symmetry. Now, a graph can also have degree 2 vertices, on each of which the constraint function satisfies the "1-in-1-out" rule and both valid local configurations have weight 1. Both Lemma 3.5 and Corollary 3.3 still hold with this extension, because such a degree 2 vertex and its two incident edges just work together as a single edge.

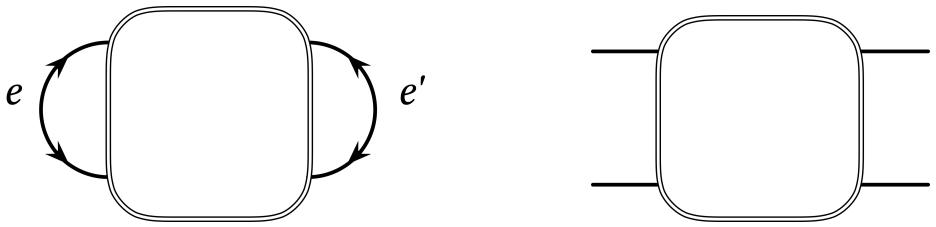
We design the following algorithm to approximately computing $Z(G)$ via sampling with $\mathcal{MC}$. As we have argued in Section 2, the partition function of the eight-vertex models can be viewed as the weighted sum of the set of *dacp*'s (the weight of a *dacp* and its underlying *acp* are the same). Since every constraint function belongs to $\overline{\mathcal{F}_{>}}$, by Lemma 2.5 for each vertex $v \in V$ we can choose a nonnegative weight function w on signed pairings at v. Thus the ratios among different signed pairings $\{\curlywedge, \curlyvee, \dashv\vdash\} \times \{+, -\}$ showing up at v in weighted *dacp*'s can be uniquely determined by the ratios among different local orientations (represented by a, b, c, and d) showing up at v. According to Corollary 2.6 (another corollary of Theorem 2.2), there must be a pairing $\boldsymbol{\varrho} \in \{\curlywedge_+, \curlyvee_+, \dashv\vdash_+\}$ showing up at v with probability at least $\frac{1}{6}$ among all six signed pairings, as long as the partition function is not zero (this can be easily tested in polynomial time). Therefore, running $\mathcal{MC}$ on G, we can approximate, with a sufficient 1/poly(n) precision, the probability of having $\boldsymbol{\varrho} \in \{\curlywedge_+, \curlyvee_+, \dashv\vdash_+\}$ at v, denoted by $\Pr_v(\boldsymbol{\varrho})$. Denote by $G_{v,\boldsymbol{\varrho}}$ the graph with v being split into v_1 and v_2 (both satisfy the "1-in-1-out" rule) and the edges reconnected according to $\boldsymbol{\varrho}$. Write the partition function of $G_{v,\boldsymbol{\varrho}}$ as $Z(G_{v,\boldsymbol{\varrho}})$, we have $\Pr_v(\boldsymbol{\varrho}) = w(\boldsymbol{\varrho})Z(G_{v,\boldsymbol{\varrho}})/Z(G)$ which means $Z(G) = w(\boldsymbol{\varrho})Z(G_{v,\boldsymbol{\varrho}})/\Pr_v(\boldsymbol{\varrho})$. To approximate $Z(G)$ it suffices to approximate $Z(G_{v,\boldsymbol{\varrho}})$, which can be done by running $\mathcal{MC}$ on $G_{v,\boldsymbol{\varrho}}$ and recursing. Repeating this process for $|V|$ steps we decompose the graph G into the base case, a set of disjoint cycles whose partition function is just 2^C where C is the number of cycles. By this self-reduction, the partition function $Z(G)$ can be approximated. ◀

Proof of Theorem 3.2. The proof is similar to that of Theorem 3.1. Given a plane graph G with the constraint function on every vertex from $\mathcal{F}_{\leq 2} \bigcap \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq}$, we can still efficiently sample even orientations according to the Gibbs measure by Lemma 3.5 and Corollary 3.4. However, in order to make our algorithm work, we need to extend the type of vertices in the eight-vertex model again, by allowing degree 2 vertices with the constraint functions satisfiying the "2-in/2-out" rule and both valid local configurations have weight 1. One can check that Lemma 3.5 still hold even with this extension.

The self-reduction still looks at a vertex v at a time. According to Corollary 2.7, there must be a pairing $\boldsymbol{\varrho} \in \{\curvearrowleft_+, \curvearrowright_+, \text{-|-}_-\}$ showing up at v with probability at least $\frac{1}{6}$ among all six signed pairings, as long as the partition function is not zero. If $\boldsymbol{\varrho}$ is $\curvearrowleft_+$ or $\curvearrowright_+$, it can be handled as in the proof of Theorem 3.1. If $\boldsymbol{\varrho}$ is -|-_-, let $G_{v,\text{-|-}_-}$ be the graph still with v being split into v_1 and v_2 and the edges reconnected according to -|-_-, but this time both v_1 and v_2 satisfy the "2-in/2-out" rule. Observe that Corollary 3.4 hold for $G_{v,\text{-|-}_-}$ if and only if it holds for $G'_{v,\text{-|-}_-}$ where we replace v by a virtual vertex v' with parameter setting $(a, b, c, d) = (0, 0, 1, 1)$ (this is equivalent as fixing $w(\text{-|-}_-) = 1$ and w on other five signed pairings being 0, if we require w to be nonnegative). Since $(0, 0, 1, 1) \in \mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq} \bigcap \overline{\mathcal{F}_{>}}$. Theorem 2.3 and consequently Corollary 3.4 still hold for $G'_{v,\text{-|-}_-}$ thus also for $G_{v,\text{-|-}_-}$.

The base case is similar to that in the proof of Theorem 3.1. This time we decompose G into a set of disjoint cycles with an even number of degree 2 vertices that satisfy the "2-in/2-out" rule (by the *Jordan Curve Theorem* argued in the full version). The partition function of this cycle graph is just 2^C where C is the number of cycles. Again, the partition function $Z(G)$ can be approximated. ◀

▶ **Corollary 3.3.** *Given a 4-regular graph $G = (V, E)$, if the constraint function on every vertex is from $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$, then $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)} \leq \binom{|E|}{2}$.*



(a) A near-even orientation with defects at e and e'.

(b) A 4-ary construction by cutting open e and e'.

Figure 7 A 4-ary construction made by cutting open the two edges with defects.

Proof. Let $\Omega_2^{\{e,e'\}} \subseteq \Omega_2$ be the set of near-even orientations in which e, e' are the two defective edges. We have $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)} = \sum_{\{e,e'\} \in \binom{E}{2}} \frac{\mathcal{Z}(\Omega_2^{\{e,e'\}})}{\mathcal{Z}(\Omega_0)}$. For any $\tau \in \Omega_2$, each of e and e' may have both half-edges going out (as in Figure 7a) or coming in, with 4 possibilities. If we "cut open" e and e' as shown in Figure 7b, we get a 4-ary construction Γ using degree 4 vertices with constraint functions in $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\leq}$. Denote the constraint function of Γ by (a', b', c', d'), with the input order being counter-clockwise starting from the upper-left edge. For this 4-ary construction we observe that near-even orientations in $\Omega_2^{\{e,e'\}}$ contribute a total weight $2(a' + d')$ while even orientations in Ω_0 contribute a total weight $2(b' + c')$. By Theorem 2.2 we know that for the 4-ary construction Γ, $a' + d' \leq b' + c'$. Therefore, $\frac{\mathcal{Z}(\Omega_2^{\{e,e'\}})}{\mathcal{Z}(\Omega_0)} \leq 1$. ◀

▶ **Corollary 3.4.** *Given a 4-regular plane graph $G = (V, E)$, if the constraint function on every vertex is from $\mathcal{A}_{\leq} \bigcap \mathcal{B}_{\leq} \bigcap \mathcal{C}_{\geq} \bigcap \overline{\mathcal{F}_{>}}$, then $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)} \leq \binom{|E|}{2}$.*

▶ **Lemma 3.5.** *Assume $\mathcal{Z}(\Omega_0) > 0$ and the constraint function on every vertex belongs to $\mathcal{F}_{\leq 2}$. $\mathcal{MC}$ is rapidly mixing if $\frac{\mathcal{Z}(\Omega_2)}{\mathcal{Z}(\Omega_0)}$ is polynomially bounded.*

4 Hardness

▶ **Theorem 4.1.** *If $(a, b, c, d) \in \mathcal{F}_{>}$, then $Z(a, b, c, d)$ does not have an FPRAS unless RP=NP.*

▶ Remark 4.2. For any $(a, b, c, d) \in \mathcal{F}_{>}$, at least two of a, b, c, and d are nonzero. The case $d = 0$ and $a, b, c > 0$ was proved in [9]. The case $d = 0$ and one of a, b, c is zero can be proved by a reduction from computing the partition function of the anti-ferromagnetic Ising model on 3-regular graphs; we postpone this proof to an expanded version of this paper. In this section, we prove the theorem when $d > 0$ and $a > b + c + d$. Since the proof of NP-hardness for $Z(a, b, c, d)$ is for not necessarily planar graphs, we can permute the parameters a, b, c. Thus the proof for $b > a + c + d$ and $c > a + b + d$ is symmetric. The adaption that we make to prove the case when $d > a + b + c$ can be found in the full version.

▶ Remark 4.3. The construction for the proof when $a > b + c + d$, or $b > a + c + d$, or $c > a + b + d$, is in fact a bipartite graph. This means approximating $Z(a, b, c, d)$ in those cases is NP-hard even for *bipartite graphs*.

Proof. Let 3-MAX CUT denote the NP-hard problem of computing the cardinality of a maximum cut in a 3-regular graph [25]. We reduce 3-MAX CUT to approximating $Z(a, b, c, d)$. Before proving the theorem we briefly state our idea. Denote an instance of 3-MAX CUT by $G = (V, E)$. Given $V_+ \subseteq V$ and $V_- = V \setminus V_+$, an edge $\{u, v\} \in E$ is in the cut between V_+ and V_- if and only if $(u \in V_+, v \in V_-)$ or $(u \in V_-, v \in V_+)$. The maximum cut problem favors the partition of V into V_+ and V_- so that there are as many edges in $V_+ \times V_-$ as possible. We want to encode this local preference on each edge by a local fragment of a graph G' in terms of configurations in the eight-vertex model.

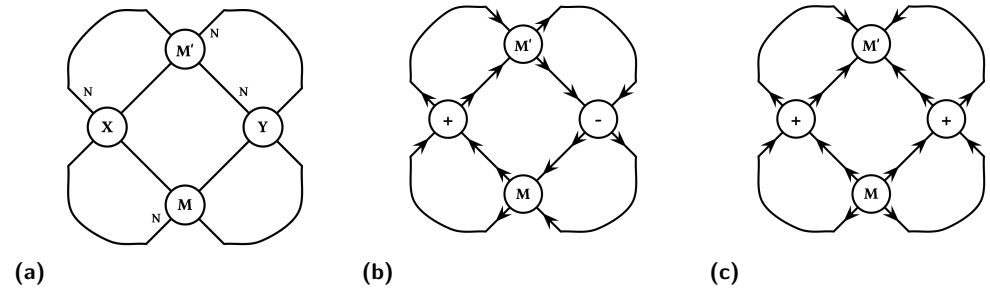


Figure 8 A four-way connection implementing a single edge in 3-MAX CUT.

First we show how to implement a toy example—a single edge $\{u, v\}$—by a construction in the eight-vertex model. Suppose there are four vertices X, Y, M, M' connected as in Figure 8a shows. The order of the 4 edges at each vertex is aligned to Figure 1 by a rotation so that the edge marked by "N" corresponds to the north edge in Figure 1. Let us impose the virtual constraint on X and Y so that the parameter setting on each of them is $\breve{a} > \breve{b} = \breve{c} = \breve{d} = 0$. (We will show how to implement this virtual constraint in the sense of approximation later.)

In other words, the four edges incident on X can only be in two possible configurations, Figure 1-1 or Figure 1-2. The same is true for Y. We say X (and similarly Y) is in state $+$ if its local configuration is in Figure 1-1 (with the "top" two edges going out and the "bottom" two edges coming in); it is in state $-$ if its local configuration is in Figure 1-2 (with the "top" two edges coming in and the "bottom" two edges going out). Hence there are a total of 4 valid configurations given the virtual constraints. When (X, Y) is in state $(+, -)$ (or $(-, +)$), M and M' have local configurations both being Figure 1-1 (or both being Figure 1-2), with weight a (Figure 8b); when (X, Y) is in state $(+, +)$ (or $(-, -)$), M and M' have local configurations both being Figure 1-7 or Figure 1-8, with weight $d < a$ (Figure 8c). This models how two adjacent vertices interact in 3-MAX CUT. We will call the connection pattern described in Figure 8a between the set of 4 dangling edges incident to X and the set of 4 dangling edges incident to Y (each with two on "top" and two on "bottom") a *four-way connection*.

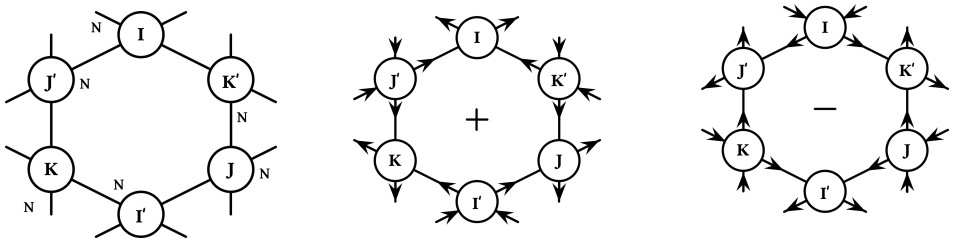


Figure 9 A locking device implementing a vertex of degree 3 in 3-MAX CUT.

To model a vertex of degree 3 in a 3-MAX CUT instance, we use the *locking device* in Figure 9a. Let us assume we have the virtual constraint that each of I, I', J, J', K, K' can only be in two local configurations, Figure 1-1 or Figure 1-2. In fact, each locking device has two states, one shown in Figure 9b with every node in configuration Figure 1-1 (called the $+$ state) and the other shown in Figure 9c with every node in configuration Figure 1-2 (called the $-$ state). If we think of the external edges incident to I, J, K to serve as the "top" edges (with "N" aligned with the "N" at X or Y in Figure 8a), and the edges incident to I', J', K' as the "bottom" edges there, then we simulate the $\pm$ state of a degree 3 vertex as follows: (1) top edges are going out and bottom edges are coming in if the device is in $+$ state, and top edges are coming in and bottom edges are going out if the device is in $-$ state; and (2) the top edges on I, J, K are going out or coming in at the same time.

Next we show how to enforce the virtual constraint in Figure 9a that each vertex has two contrary configurations, in the sense of approximation. The idea is to implement an *amplifier* as a 4-ary construction with parameter $(\hat{a}, \hat{b}, \hat{c}, \hat{d})$ such that $\hat{a} \gg \hat{b} + \hat{c} + \hat{d}$ using polynomially many vertices in the eight-vertex model. We obtain such an amplifier by an iteration of Γ shown in Figure 10. The parameter setting (a', b', c', d') of Γ is $\begin{cases} a'=\Lambda(a,b,c,d) \\ b'=\Lambda(b,c,d,a) \\ c'=\Lambda(c,d,a,b) \\ d'=\Lambda(d,a,b,c) \end{cases}$, where

$$\Lambda(\xi, x, y, z) = \xi^7 + (3x^4 + 3y^4 + 3z^4 + 4x^2y^2 + 4x^2z^2 + 4y^2z^2)\xi^3 + (2x^4y^2 + 2x^4z^2 + 2x^2y^4 + 2y^4z^2 + 2x^2z^4 + 2y^2z^4 + 30x^2y^2z^2)\xi. \quad (6)$$

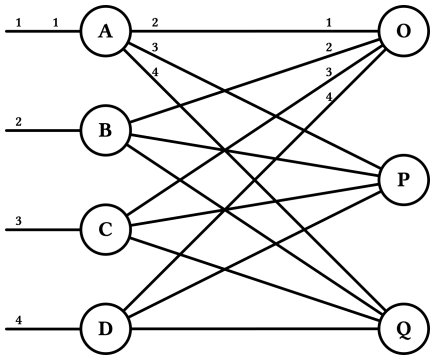


Figure 10 A 4-ary construction that amplify the maximum among a, b, c, d.

This construction uses 7 vertices and is called a 1-*amplifier*. We obtain $(a_1, b_1, c_1, d_1) = (a', b', c', d')$ which amplifies the relative weight of configurations in Figure 1-1 or Figure 1-2. If we plug in the amplifier Γ into each vertex of Γ itself (called a 2-*amplifier*), we can obtain (a_2, b_2, c_2, d_2) using 7^2 vertices. Iteratively, we can construct a series of constraint functions with parameters (a_k, b_k, c_k, d_k) $(k \geq 1)$ such that $\begin{cases} a_{k+1}=\Lambda(a_k,b_k,c_k,d_k) \\ b_{k+1}=\Lambda(b_k,c_k,d_k,a_k) \\ c_{k+1}=\Lambda(c_k,d_k,a_k,b_k) \\ d_{k+1}=\Lambda(d_k,a_k,b_k,c_k) \end{cases}$, using 7^k vertices for each k (called a k-*amplifier*). Lemma 4.4 shows that the asymptotic growth rate is exponential in the number of vertices used.

To reduce the problem 3-MAX CUT to approximating $Z(a, b, c, d)$, let $\kappa > \lambda \geq 1$ be two constants that are sufficiently large. For each 3-MAX CUT instance $G = (V, E)$ with $|V| = n$ and $|E| = m$, we construct a graph G' where a device in Figure 9a is created for each $v \in V$, and a four-way connection is made for every $\{u, v\} \in E(G)$, on the dangling edges corresponding to $\{u, v\}$ as in Figure 8a. For each 4-way connection in Figure 8a, each of the nodes M, M' is replaced by a $(\lambda \log n)$-amplifier to boost the ratio of the configurations in Figure 1-1 or Figure 1-2 over other configurations. For each device in Figure 9a, each of the nodes I, I', J, J', K, K' is replaced by a $(\kappa \log n)$-amplifier to lock in the configurations Figure 9b or Figure 9c. We can prove that the maximum size s of all cuts in G can be recovered from an approximate solution to $Z(G'; f)$. In fact, there is a valid configuration (at the granularity of nodes and edges shown in Figure 9a) of weight $(a_{\kappa \log n})^{6n} (a_{\lambda \log n})^{2s} (d_{\lambda \log n})^{2(m-s)}$, and the weighted sum of all configurations is smaller than $\frac{1}{2} (a_{\kappa \log n})^{6n} (a_{\lambda \log n})^{2(s+1)} (d_{\lambda \log n})^{2(m-(s+1))}$. ◀

▶ **Lemma 4.4.** *Let $(a_k, b_k, c_k, d_k) = \Lambda^{(k)}(a, b, c, d)$ given by (6). Assuming $a_0 > b_0 + c_0 + d_0$, $a_0, d_0 > 0$, and $b_0, c_0 \geq 0$, there exists some constants $\alpha > 0, \beta > 1$ depending only on a_0, b_0, c_0, d_0 such that for all $k \geq 1$, $\frac{a_k}{b_k+c_k+d_k} \geq \alpha \beta^{2^k}$.*

References

1 G. T. Barkema and M. E. J. Newman. Monte carlo simulation of ice models. *Phys. Rev. E*, pages 1155–1166, 1998.

2 R. J. Baxter. *Exactly Solved Models in Statistical Mechanics*. Academic Press Inc., San Diego, CA, USA, 1982.

3 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. *J. ACM*, 60(5), October 2013. doi:10.1145/2528400.

4 Jin-Yi Cai and Xi Chen. *Complexity Dichotomies for Counting Problems*, volume 1. Cambridge University Press, 2017. doi:10.1017/9781107477063.

5 Jin-Yi Cai and Xi Chen. Complexity of counting CSP with complex weights. *J. ACM*, 64(3), June 2017. doi:10.1145/2822891.

6 Jin-Yi Cai and Zhiguo Fu. Complexity classification of the eight-vertex model. *CoRR*, abs/1702.07938, 2017. arXiv:1702.07938.

7 Jin-Yi Cai and Artem Govorov. Perfect matchings, rank of connection tensors and graph homomorphisms. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 476–495, 2019. doi:10.1137/1.9781611975482.30.

8 Jin-Yi Cai and Tianyu Liu. Counting perfect matchings and the eight-vertex model. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming*, ICALP '20, pages 23:1–23:19, 2020. arXiv:1904.10493.

9 Jin-Yi Cai, Tianyu Liu, and Pinyan Lu. Approximability of the six-vertex model. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2248–2261, 2019. doi:10.1137/1.9781611975482.136.

10 Jin-Yi Cai, Tianyu Liu, Pinyan Lu, and Jing Yu. Approximability of the eight-vertex model, 2018. arXiv:1811.03126.

11 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. The complexity of complex weighted Boolean #CSP. *Journal of Computer and System Sciences*, 80(1):217–236, 2014. doi:10.1016/j.jcss.2013.07.003.

12 Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, January 1991. doi:10.1145/102782.102783.

13 Martin Dyer and David Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM Journal on Computing*, 42(3):1245–1274, 2013. doi:10.1137/100811258.

14 Michael Freedman, László Lovász, and Alexander Schrijver. Reflection positivity, rank connectivity, and homomorphism of graphs. *Journal of the American Mathematical Society*, 20:37–51, 2007. doi:10.1090/S0894-0347-06-00529-7.

15 Sam Greenberg and Dana Randall. Slow mixing of Markov chains using fault lines and fat contours. *Algorithmica*, 58(4):911–927, December 2010. doi:10.1007/s00453-008-9246-3.

16 Mark Jerrum. *Counting, Sampling and Integrating: Algorithm and Complexity*. Birkhäuser, Basel, 2003.

17 Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989. doi:10.1137/0218077.

18 Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(Supplement C):169–188, 1986. doi:10.1016/0304-3975(86)90174-X.

19 Richard M. Karp and Michael Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, SFCS '83, pages 56–64, Washington, DC, USA, 1983. IEEE Computer Society. doi:10.1109/SFCS.1983.35.

20 W.B.Raymond Lickorish. *An Introduction to Knot Theory*. Springer New York, 1997. URL: https://books.google.com/books?id=PhHhw_kRvewC.

21 Aneesur Rahman and Frank H. Stillinger. Proton distribution in ice and the kirkwood correlation factor. *The Journal of Chemical Physics*, 57(9):4009–4017, 1972. doi:10.1063/1.1678874.

22 Alistair Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability and Computing*, 1:351–370, 1992.

23 Olav F. Syljuåsen and M. B. Zvonarev. Directed-loop monte carlo simulations of vertex models. *Phys. Rev. E*, 70:016118, July 2004. doi:10.1103/PhysRevE.70.016118.

24 A. Yanagawa and J.F. Nagle. Calculations of correlation functions for two-dimensional square ice. *Chemical Physics*, 43(3):329–339, 1979. doi:10.1016/0301-0104(79)85201-5.

25 Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, STOC '78, pages 253–264, 1978. doi:10.1145/800133.804355.

Lower Bounds for Matrix Factorization

Mrinal Kumar[1]
Department of Computer Science and Engineering, IIT Bombay, India
mrinalkumar08@gmail.com

Ben Lee Volk
Center for the Mathematics of Information, California Institute of Technology, Pasadena, CA, USA
benleevolk@gmail.com

Abstract

We study the problem of constructing explicit families of matrices which cannot be expressed as a product of a few sparse matrices. In addition to being a natural mathematical question on its own, this problem appears in various incarnations in computer science; the most significant being in the context of lower bounds for algebraic circuits which compute linear transformations, matrix rigidity and data structure lower bounds.

We first show, for every constant d, a deterministic construction in time $\exp(n^{1-\Omega(1/d)})$ of a family $\{M_n\}$ of $n \times n$ matrices which cannot be expressed as a product $M_n = A_1 \cdots A_d$ where the total sparsity of $A_1, \ldots, A_d$ is less than $n^{1+1/(2d)}$. In other words, any depth-d linear circuit computing the linear transformation $M_n \cdot \mathbf{x}$ has size at least $n^{1+\Omega(1/d)}$. This improves upon the prior best lower bounds for this problem, which are barely super-linear, and were obtained by a long line of research based on the study of super-concentrators (albeit at the cost of a blow up in the time required to construct these matrices).

We then outline an approach for proving improved lower bounds through a certain derandomization problem, and use this approach to prove asymptotically optimal quadratic lower bounds for natural special cases, which generalize many of the common matrix decompositions.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Circuit complexity

Keywords and phrases Algebraic Complexity, Linear Circuits, Matrix Factorization, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.5

Related Version A full version of the paper is avaiable at `https://arxiv.org/abs/1904.01182`.

Acknowledgements A part of this work was done while the first author was at the semester on Lower Bounds in Computational Complexity at Simons Institute for the Theory of Computing, Berkeley, USA, and at the Department of Computer Science, University of Toronto, Canada. We thank Swastik Kopparty for an insightful discussion on explicit construction of Sidon sets over finite fields. We also thank Rohit Gurjar, Nutan Limaye, Srikanth Srinivasan and Joel Tropp for helpful discussions.

1 Introduction

This work concerns the following (informally stated) very natural problem:

▶ **Open Problem 1.** *Exhibit an explicit matrix $A \in \mathbb{F}^{n \times n}$, such that A cannot be written as $A = BC$, where $B \in \mathbb{F}^{n \times m}$ and $C \in \mathbb{F}^{m \times n}$ are sparse matrices.*

Before bothering ourselves with the precise meaning of the words "explicit" and "sparse" in the above problem, we discuss the various contexts in which this problem presents itself.

[1] A part of this work was done during the semester on Lower Bounds in Computational Complexity at Simons Institute for the Theory of Computing, Berkeley, USA, and at the Department of Computer Science, University of Toronto, Canada.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 5; pp. 5:1–5:20
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

COMPUTATIONAL COMPLEXITY CONFERENCE

1.1 Linear circuits and matrix factorization

Algebraic complexity theory studies the complexity of computing polynomials using arithmetic operations: addition, subtraction, multiplication and division. An algebraic circuit over a field $\mathbb{F}$ is an acyclic directed graph whose vertices of in-degree 0, also called inputs, are labeled by indetermeinates $\{x_1, \ldots, x_n\}$ or field elements from $\mathbb{F}$, and every internal node is labeled with an arithmetic operation. The circuit computes rational functions in the natural way, and the polynomials (or rational functions) computed by the circuit are those computed by its vertices of out-degree 0, called the outputs. This framework is general enough to encompass virtually all the known algorithms for algebraic computational problems. The size of the circuit is defined to be the number of edges in it. For a more detailed background on algebraic circuits, see [50].

Perhaps the simplest non-trivial class of polynomials is the class of linear (or affine) functions. Accordingly, such polynomials can be computed by a very simple class of circuits called *linear circuits*: these are algebraic circuits which are only allowed to use addition and multiplication by a scalar. It is often convenient to consider graphs with labels on the edges as well: every internal node is an addition gate, and for $c \in \mathbb{F}$, an edged labeled c from a vertex v to a vertex u denotes that the output of v is multiplied by c when feeding into u. Thus, every node computes a linear combination of its inputs.

It is not hard to show that any arithmetic circuit for computing a set of linear functions can be converted into a linear circuit with only a constant blow-up in size (see [10], Theorem 13.1; eliminating division gates requires that the field $\mathbb{F}$ in question is large enough. In this paper we will always make this assumption when needed).

Clearly, every set of n linear functions on n variables (represented by a matrix $A \in \mathbb{F}^{n \times n}$) can be computed by a linear circuit of size $O(n^2)$. Using counting arguments (over finite fields) or dimension arguments (over infinite fields), it can be shown that for a random or generic matrix this upper bound is fairly tight. Thus, a central open problem in algebraic complexity theory is to prove any super-linear lower bound for an *explicit* family of matrices $\{A_n\}$ where $A_n \in \mathbb{F}^{n \times n}$. The standard notion of explicitness in complexity theory is that there is a deterministic algorithm that outputs the matrix A_n in $\text{poly}(n)$ time, although more or less stringent definitions can be considered as well.

Despite decades of research and partial results, such lower bounds are not known.[2] In order to gain insight into the general model of computation, research has focused on limited models of linear circuits, such as monotone circuits, circuits with bounded coefficients, or bounded depth circuits. We defer a more thorough discussion on previous work to Subsection 1.5, and proceed to describe bounded depth circuits, which are the focus of this work.

The *depth* of a circuit is the length (in edges) of a longest path from an input to an output. Constant depth circuits appear to be a particularly weak model of computation. However, even this model is surprisingly powerful (see also Subsection 1.2).

The "easiest" non-trivial model is the model of depth-2 linear circuits. A depth 2 linear circuit computing a linear transformation $A \in \mathbb{F}^{n \times n}$ consists of a bottom layer of n input gates, a middle layer of m gates, and a top layer of n output gates. We assume, without loss of generality, that the circuit is *layered*, in the sense that every edge goes either from the bottom to the middle layer, or from the middle to the top layer. Indeed, every edge going directly from the bottom to the top layer can be replaced by a path of length 2; this transformation increases the size of the circuit by at most a factor of 2.

[2] We remark that super-linear lower bounds for general arithmetic circuits are known, but for polynomials of high degree [51, 7].

By letting $C \in \mathbb{F}^{m \times n}$ be the adjacency matrix of the (labeled) subgraph between the bottom and the middle layer, and $B \in \mathbb{F}^{n \times m}$ be the adjacency matrix as the subgraph between the middle and the top layer, it is clear that $A = BC$. Thus, a decomposition of A into the product of two sparse matrices is equivalent to saying that A has a small depth-2 linear circuit. This argument can be generalized, in exactly the same way, to depth-d circuits and decompositions of the form $A = A_1 \cdots A_d$, for constant d.

Weak super-linear lower bounds are known for constant depth linear circuits. They are based on the following observation, due to Valiant [53]: for subsets $S, T \subseteq [n]$ of size k, let $A_{S,T}$ denote the submatrix of A indexed by rows in S and columns in T. If $A_{S,T}$ has rank k, the minimal vertex cut in the subcircuit restricted to input from S and outputs from T is of size at least k: indeed, a smaller cut corresponds to a factorization $A_{S,T} = PQ$ for $P \in \mathbb{F}^{k \times r}$ and $Q \in \mathbb{F}^{r \times k}$ for $r < k$, contradicting the rank assumption. Using Menger's theorem, it is now possible to deduce that if A is a matrix such that for every S, T as above the matrix $A_{S,T}$ is non-singular, then the circuit computing A contains, for every subcircuit which corresponds to such S, T, at least k vertex disjoint paths from S to T. Such graphs were named *superconcentrators* by Valiant, and their minimal size was extensively studied [53, 41, 42, 14, 43, 6, 45].

Superconcentrators of logarithmic depth and linear size do exist, so while this approach cannot show lower bounds for circuits of logarithmic depth, it is possible to show that for constant d, any depth-d superconcentrator has size at least $n \cdot \lambda_d(n)$, where $\lambda_d(n)$ is a function that unfortunately grows very slowly with n. For example, $\lambda_2(n) = \Theta(\log^2 n / \log\log n)$, $\lambda_3(n) = \Theta(\log\log n)$, $\lambda_4(n) = \lambda_5(n) = \log^*(n)$, and so on. Such lower bounds apply for any matrix whose minors of all orders are non-zero, e.g., a Cauchy matrix given by $A_{i,j} = 1/(x_i - y_j)$ for any distinct $x_1, \ldots, x_n, y_1, \ldots, y_n$. Over finite fields it is possible to modify the proof and obtain similar lower bounds for matrices defining good error correcting codes [22].

These lower bounds on the size of superconcentrators are tight: for every $d \in \mathbb{N}$, there exists a super-concentrator of depth d and size $O(n \cdot \lambda_d(n))$. It is thus impossible to improve the lower bounds only using this technique.

1.2 Matrix rigidity

A demonstration of the surprising power of depth-2 circuits can be seen using the notion of *matrix rigidity*, a pseudorandom property of matrices which we now recall. A matrix $A \in \mathbb{F}^{n \times n}$ is (r, s) rigid if A *cannot* be written as a sum $A = R + S$ where R is a matrix of rank r, and S is a matrix with at most s non-zero entries. Valiant [54] famously proved that if A is computed by a linear circuit with bounded fan-in of depth $O(\log n)$ and size $O(n)$, then A is not $(\varepsilon n, n^{1+\delta})$ rigid for every $\varepsilon, \delta > 0$.[3] It follows that an explicit construction of $(\varepsilon n, n^{1+\delta})$ matrix, for some $\varepsilon, \delta > 0$, will imply a super-linear lower bound for linear circuits of depth $O(\log n)$. Pudlák [43] observed that similar rigidity parameters will imply even stronger lower bounds for constant depth circuits. A random matrix (over infinite fields) is $(r, (n-r)^2)$-rigid, but the best explicit constructions have rigidity $(r, n^2/r \cdot \log(n/r))$ [21, 47], which is insufficient for proving lower bounds.

Observe that a decomposition $A = R + S$ where $\mathrm{rank}(R) = \varepsilon n$ and S is $n^{1+\delta}$-sparse corresponds to a depth-2 circuit with a very special structure and with at most $2\varepsilon n^2 + n^{1+\delta}$ edges (this circuit is not layered, but as we explained above, this does not make a significant

[3] In fact, one can obtain slightly better parameters. See, for example, [54] or [16].

difference). In particular, one way of interpreting Valiant's result is as a non-trivial depth reduction from depth $O(\log n)$ to depth 2, so that proving *any* depth-2 $\Omega(n^2)$ lower bound for an explicit matrix, will imply a lower bound for depth $O(\log n)$.[4] This can be seen as the linear circuit analog of similar strong depth reduction theorems for general algebraic circuits [3, 29, 52, 24].

However, we would like to argue that proving lower bounds for depth-2 circuits is in fact *necessary* for proving rigidity lower bounds, by observing that *upper bounds* on the depth-2 complexity of A give upper bounds on its rigidity parameters. Indeed, suppose $A = BC$ can be computed by a depth-2 circuit of size $n^{1+\varepsilon}$. Let m be as before the number of columns of B (which equals the number of rows of C), and note that we may assume $m \leq n^{1+\varepsilon}$, as zero columns of B or zero rows of C can be omitted. For $i \in [m]$, let B_i denote the i-th column of B, and C_i the i-th row of C, so that $A = \sum_{i=1}^{m} B_i C_i$. Fix a constant $\delta > 0$, and say $i \in [m]$ is *dense* if either B_i or C_i has more than n^{ε}/δ non-zero entries; otherwise, i is *sparse*. Since B can have at most δn columns with sparsity of more than n^{ε}/δ, and similarly for the rows of C, the number of dense i-s is at most $2\delta n$. It follows that

$$A = \sum_{i \text{ dense}} B_i C_i + \sum_{i \text{ sparse}} B_i C_i.$$

The first sum is a matrix of rank at most $2\delta n$, and the second is a matrix whose sparsity is at most $m \cdot n^{2\varepsilon}/\delta^2 = n^{1+3\varepsilon}/\delta^2$. Thus, proving rigidity lower bounds of the type required to carry out Valiant's approach necessarily means proving lower bounds of the form "$n^{1+\varepsilon}$" on the depth-2 complexity of A (we remark that the argument above is very similar to the aforementioned result of Pudlák [43]; Pudlák's argument is stated in a slightly different language and in greater generality). Since proving rigidity lower bounds is a long-standing open problem, we view the problem of proving an $\Omega(n^{1+\varepsilon})$ lower bound for depth-2 circuits as an important milestone towards this.

1.3 Data structure lower bounds

The problem of matrix factorization into sparse matrices also appears in the context of proving lower bounds for data structures. A dynamic data structure with n inputs and q queries is a pair of algorithms whose purpose is to update and retrieve certain data under a sequence of operations, while minimizing the memory access. In the group model, it is given by a pair of algorithms. The update algorithm is represented by a matrix $U \in \mathbb{F}^{s \times n}$. Given $x \in \mathbb{F}^n$, thought of as assignment of weights to the n inputs, Ux computes a linear combination of those weights and stores them in memory. The query algorithm is given by a matrix $Q \in \mathbb{F}^{q \times s}$. Given a query, it computes a linear function of the s memory cells, and returns the answer. Hence, an "update" operation followed by a "retrieve" operation computes the linear transformation given by $A = QU$.

The worst case update time of the database is the maximal number of non-zero elements in a column of U, and the worst case query time is the maximal number of non-zero elements in a row of Q. The value s denotes the space required by the data structure. It now directly follows that a matrix $A \in \mathbb{F}^{q \times n}$ which cannot be factored as $A = QU$ for a row-sparse Q and column-sparse U gives a data structure problem with a lower bound on its worst case query

[4] We note that this statement makes sense only over large fields, as over fixed finite fields, it is always possible to prove an *upper bound* of $O(n^2/\log n)$ on the depth-2 complexity of any matrix [27]. This does not contradict the fact that rigid matrices exist over finite fields – a decomposition to $R + S$ is a very special type of depth-2 circuit.

or update time. It is also possible to define an analogous average case notion. Lower bounds for this model were proved by [19, 20, 40, 39, 30, 31, 32], but none of these results beats the lower bounds for depth-2 circuits obtained using superconcentrators.

A related model is that of a static data structures, which is again given by a factorization $A = QP$, where now we are interested in trade-offs between the space s of the data structure and its worst case query time, while not being charged for the total sparsity of P. A recent work of Dvir, Golovnev and Weinstein [16] showed that proving lower bounds for this model is related to the problem of matrix rigidity from Subsection 1.2.

Despite the overall similarity, there are several key technical differences between the linear circuit complexity and the data structure problems. The first and obvious issue is that worst-case lower bounds on the update or query time do not necessarily imply that Q or U are dense matrices: the total sparsity of Q and U is related to the average-case update and query time. The second, more severe issue, is that in many applications the number of queries q is polynomially larger than n, while the lower bounds on running time are still measured as functions of the number of inputs n. This makes sense in the data structure settings, but from a circuit complexity point of view, a set of say n^3 linear functions trivially requires a circuit of size n^3, and thus a lower bound of say $n\,\mathrm{polylog}(n)$ is meaningless in that setting.

This issue also comes up when studying the so-called *succinct space* setting, where we require $s = n(1 + o(1))$. The lower bounds we are aware of for this setting are worst case lower bounds, and require the number of outputs q to be at least Cn for some $C > 1$ [23, 16], so that in the corresponding circuit the number of vertices in the middle layer is required to be much smaller than the number of outputs, which may be considered quite unnatural. In particular, we are unaware of any improved lower bounds on the sparsity of matrix factorization for $A \in \mathbb{F}^{n\times n}$ when $s = n(1 + o(1))$ or even $s = n$ which come from the data structure lower bounds literature.

1.4 Machine learning

We briefly remark that the problem of factorizing a matrix into a product of two or more sparse matrices is also ubiquitous in machine learning and related areas. Naturally, research in those areas did not focus on lower bounds but rather on algorithms for finding such a representation, assuming it exists, sometimes heuristically, and it is usually enough to approximate the target matrix A. In particular, algorithms have been proposed for the very related problems of non-negative matrix factorization [33][5] or sparse dictionary learning [36], and there are also connections to the analysis of deep neural networks [38].

1.5 Previous work

As mentioned in Subsection 1.1, there are no non-trivial known lower bounds for general linear circuits, and for bounded depth circuits, the best lower bounds follow from the lower bounds on bounded depth super-concentrators, which are barely super-linear.

Shoup and Smolensky [49] give a lower bound of $\Omega(dn^{1+1/d})$ for depth-d circuits computing a certain linear transformation given by a matrix $A \in \mathbb{R}^{n\times n}$. Unfortunately, the matrices for which their lower bound holds are not explicit from the complexity theoretic point of

[5] It is interesting to observe that for the problem of factorizing matrices into non-negative matrices it is quite easy to prove almost-optimal lower bounds even for unbounded depth linear circuits, as mentioned in Subsection 1.5

view, despite having a very succinct mathematical description (for example, one can take $A_{i,j} = \sqrt{p_{i,j}}$ for n^2 distinct prime numbers $p_{i,j}$). For the same matrix, they in fact prove super-linear lower bounds for circuits of depth up to polylog(n).

Quite informally, the intuition behind their lower bounds is that all small bounded depth linear circuits can be described as lying in the image of a low-degree polynomial map in a small number of variables, and thus, if the elements of A are sufficiently "algebraically rich", for a certain specific measure, A cannot be computed by such a circuit. This same philosophy lies behind Raz's elusive function approach for proving lower bounds for algebraic circuits [46]. In particular, among other results, Raz uses an argument which can be seen as a modification of the technique of Shoup and Smolensky (as worked out in [50]) to prove lower bounds for bounded depth algebraic circuits computing bounded degree polynomials.

One class of linear circuits which has attracted significant attention is the class of circuits with bounded coefficients. Here, the circuit is only allowed to multiply by scalars with absolute value of at most some constant. For definiteness, we may assume this constant is 1 (this does not affect the complexity by more than a constant factor). The earliest result for this model is Morgenstern's ingenious proof [37] of an $\Omega(n \log n)$ lower bound on bounded coefficient circuits computing the discrete Fourier transform matrix (this lower bound is matched by the upper bound given by the Cooley-Tukey FFT algorithm, which is a bounded coefficient linear circuit). For depth-d circuits, Pudlák [44] has proved lower bounds of the form $\Omega(dn^{1+1/d})$ for the same matrix.

Another natural subclass which was considered in earlier works is the class of monotone linear circuits. These are circuits which are defined over $\mathbb{R}$, and can only use non-negative scalars. Chazelle [12] observed that it is possible to prove lower bounds in this model, even against unbounded-depth circuits, for any boolean matrix with no large monochromatic rectangle. Instantiated with the recent explicit constructions of bipartite Ramsey graphs [11, 8, 13, 34], this gives an almost optimal $n^{2-o(1)}$ lower bound against such circuits. The main observation in the proof is that if A does not have monochromatic $t \times t$ rectangle, then since the model is monotone and no cancellations are allowed, every internal node which computes a linear function supported on at least t variables cannot be connected to more than t output gates.

For a more detailed survey on these results and some other related results, see the survey by Lokam [35].

1.6 Our results

In this paper, we prove several results regarding bounded depth linear circuits which we now discuss.

Lower bounds for depth-d linear circuits

We start by considering general depth-d circuits. We give the first deterministic construction in time $2^{o(n)}$ of matrices which require depth-d circuits of size $n^{1+\Omega(1/d)}$.

▶ **Theorem 2.** *Let $\mathbb{F}$ be a field. There exists a family of matrices $\{A_n\}_{n\in\mathbb{N}}$, which can be constructed in time $\exp(n^{1-\Omega(1/d)})$, such that every depth-d linear circuit computing A_n, even over the algebraic closure of $\mathbb{F}$, has size at least $n^{1+\Omega(1/d)}$.*

If $\mathbb{F} = \mathbb{Q}$, the entries of A are integers of bit complexity $\exp(n^{1-\Omega(1/d)})$. If $\mathbb{F} = \mathbb{F}_q$ is a finite field, the entries of A are elements of an extension $\mathbb{E}$ of $\mathbb{F}$ of degree $\exp(n^{1-\Omega(1/d)})$.

This theorem is proved in Section 2. We remark again that the previous best lower bounds against general depth-d linear circuits (for matrices that can be constructed in polynomial time) are barely super-linear and much weaker than $n^{1+\varepsilon}$. In the recent work of Dvir, Golovnev and Weinstein [16] it was pointed out that currently there are not even known constructions of rigid matrices (with parameters that would imply lower bounds) in classes such as $\mathbf{E}^{\mathbf{NP}}$. By arguing directly about circuit size, and not about rigidity, Theorem 2 gives constructions of matrices in a much smaller complexity class, which enjoy the same bounded-depth complexity lower bounds as would follow from optimal constructions of rigid matrices using the results of Pudlák [43].

In a related and independent work, Alman and Chen [4] constructed in $\mathbf{P}^{\mathbf{NP}}$ (i.e., in polynomial time and using an **NP** oracle), for infinitely many n's, an $n \times n$ matrix with rigidity parameters which suffice for proving a lower bound of $\Omega(n \cdot 2^{\log(n)^{1/4-\varepsilon}})$ on its depth-2 complexity. Compared to their work, our construction lies in an incomparable complexity class (we do not use an NP oracle at the expense of a longer running time), extends for all depths $d \geq 2$, works for all large enough n, and provides stronger lower bounds. Furthermore, Alman and Chen use complexity theoretic techniques which are very different from our algebraic techniques. We refer to [4] for some further discussion on the differences and similarities.

While the statement in Theorem 2 holds for any $d \geq 2$, for $d = 2$ there is a much simpler construction of a hard family of matrices in quasi-polynomial time.

▶ **Theorem 3.** *Let $\mathbb{F}$ be any field and c be any positive constant. Then, there is a family $\{A_n\}_{n\in\mathbb{N}}$ of $n \times n$ matrices which can be constructed in time $\exp(O(\log^{2c+1} n))$ such that any depth-2 linear circuit computing A_n even over the algebraic closure of $\mathbb{F}$ has size at least $\Omega(n \log^c n)$.*

For every constant $c \geq 2$, this theorem already improves upon the current best lower bound of $\Omega(n \log^2 n / \log\log n)$ known for this problem (see [45]). This construction is based on an exponential time construction of a small hard matrix, and then amplifying its hardness using a direct sum construction (note, however, that over infinite fields even the fact that a hard matrix can be constructed in exponential time, while not very hard to prove, is not *completely* obvious). For completeness, we describe this simple construction in Subsection 2.7.

Lower bounds for restricted depth-2 linear circuits

Given the importance of the model of depth-2 linear circuits, as explained above, and its resistance to strong lower bounds, we then move on to consider several natural subclasses of depth-2 circuits. These classes in particular correspond to almost all common matrix decompositions. We are able to prove asymptotically optimal $\Omega(n^2)$ lower bounds for these restricted models. As mentioned above, such lower bounds for general depth-2 circuits will imply super-linear lower bounds for logarithmic depth linear circuits, thus resolving a major open problem.

Symmetric circuits

A symmetric depth-2 circuit (over $\mathbb{R}$) is a circuit of the form $B^T B$ for some $B \in \mathbb{R}^{m\times n}$ (considered as a graph, the subgraph between the middle and the top layer is the "mirror image" of the subgraph between the bottom and middle layer). Over $\mathbb{C}$, one should take the conjugate transpose B^* instead of B^T.

Symmetric circuits are a natural computational model for computing positive semi-definite (PSD) matrix. Clearly, every symmetric circuit computes a PSD matrix, and every PSD matrix has a (non-unique) symmetric circuit. In particular, a Cholesky decomposition of PSD matrices corresponds to a computation by a symmetric circuit (of a very special form).

We prove asymptotically optimal lower bounds for this model.

▶ **Theorem 4.** *There exists an explicit family of real $n \times n$ PSD matrices $\{A_n\}_{n\in\mathbb{N}}$ such that every symmetric circuit computing A_n (over $\mathbb{R}$ or $\mathbb{C}$) has size $\Omega(n^2)$.*

We do not know whether every depth-2 linear circuit for a PSD matrix can be converted to a symmetric circuit with a small blow-up in size. One way to phrase this question is given below.

▶ **Question 5.** *Is there a constant $c < 2$, such that every PSD matrix $A \in \mathbb{R}^{n\times n}$ which can be computed by a linear circuit of size s, can be computed by a symmetric circuit of size $O(s^c)$?*

A positive answer for Question 5 will imply, using Theorem 4, an $\Omega(n^{1+\varepsilon})$ lower bound for depth-2 linear circuits.

Invertible circuits

Invertible circuits are circuits of the form BC, where either B or C are invertible (but not necessarily both). We stress that invertible circuits can (and do) compute non-invertible matrices. In particular, if $B \in \mathbb{F}^{n\times m}$ and $C \in \mathbb{F}^{m\times n}$, here we require $m = n$.

Invertible circuits generalize many of the common matrix decompositions, such as QR decomposition, eigendecomposition, singular value decomposition[6] and LUP decomposition (in the case where the matrix L is required to be unit lower triangular).[7]

We prove optimal lower bounds for invertible circuits.

▶ **Theorem 6.** *Let $\mathbb{F}$ be a large enough field. There exists an explicit family of $n \times n$ matrices $\{A_n\}_{n\in\mathbb{N}}$ over $\mathbb{F}$ such that every invertible circuit computing A_n has size $\Omega(n^2)$.*

If A is an invertible matrix, then clearly every depth-2 circuit with $m = n$ must be an invertible circuit. However, our technique for proving Theorem 6 crucially requires the hard matrix A to be non-invertible.

1.7 Proof Overview

Our proofs rely on a few different ideas coming from algebraic complexity theory, coding theory, arithmetic combinatorics and the theory of derandomization. We now discuss some of the key aspects.

---

[6] A diagonal matrix can be multiplied with the matrix to its left or to its right, without increasing the sparsity, to obtain an invertible depth-2 circuit.

[7] The sparsity of UP equals the sparsity of U, as P simply permutes the columns of U, so every LUP decomposition corresponds to the invertible depth-2 circuit given by $L(UP)$.

Shoup-Smolensky dimension

For the proof of Theorem 2, we rely on the notion of *Shoup-Smolensky* dimension as a measure of complexity of matrices. Shoup-Smolensky dimensions are a family of measures, parametrized by $t \in \mathbb{N}$, of "algebraic richness" of the entries of a matrix (see Definition 8 for details), which is supposed to capture the intuition that matrices with small circuits should depend on a few "parameters" and thus should not posses much richness.

Shoup and Smolensky [49] showed that for an appropriate choice of parameters, this measure is non-trivially small for linear transformations with small linear circuits of depth at most $\text{poly}(\log n)$. Informally, as the order t gets larger, this measure becomes useful against stronger models of computation; however, it also becomes harder to construct matrices which have a large complexity with respect to this measure (and hence cannot be computed by a small linear circuit). Shoup and Smolensky do this by constructing hard matrices which do not have small bit complexity (and hence this construction is not complexity theoretically explicit) but do have short and succinct mathematical description.

For our proof, we first observe that for bounded depth circuits it suffices to use much smaller order t than what Shoup and Smolensky used. This observation was also made by Raz [46] in a similar context, but using the language of elusive functions.

We then use this observation to "derandomize", in a certain sense, an exponential time construction of a hard matrix, by giving deterministic constructions of matrices with large Shoup-Smolensky dimension.

A key ingredient of our proof is a connection between the notion of Sidon Sets in arithmetic combinatorics and Shoup-Smolensky dimension (see Subsection 2.4 for details). Our construction is in two steps. In the first step we construct matrices with entries in $\mathbb{F}[y]$ which have a large Shoup-Smolensky dimension over $\mathbb{F}$, and degree of every entry is not too large. In the next step, we go from these univariate matrices to a matrix with entries in an appropriate low degree extension of $\mathbb{F}$ while still maintaining the Shoup-Smolensky dimension over $\mathbb{F}$. Our construction of hard matrices over the field of complex numbers is based on similar ideas but differs in some minor details.

Lower bounds via Polynomial Identity Testing

Our proofs for Theorem 4 and Theorem 6 are based on a derandomization argument. Connections between derandomization and lower bounds are prevalent in algebraic and Boolean complexity, but in our current setting they have not been widely studied before.

We say that a set $\mathcal{H}$ of $n \times n$ matrices is a *hitting set* for a class $\mathcal{C}$ of matrices if for every non-zero $A \in \mathcal{C}$ there is $H \in \mathcal{H}$ such that $\langle A, H \rangle := \sum_{i,j} A_{i,j} H_{i,j} \neq 0$.

Every class $\mathcal{C}$ has a hitting set of size n^2, namely the indicator matrices of each of the entries. A hitting set is non-trivial if its size is at most $n^2 - 1$. Observe that a non-trivial hitting set for $\mathcal{C}$ gives an efficient algorithm for finding a matrix $M \notin \mathcal{C}$, by finding a non-zero A such that $\langle A, H \rangle = 0$ for every $H \in \mathcal{H}$. Such an A exists and can be found in polynomial time because the set $\mathcal{H}$ imposes at most $n^2 - 1$ homogeneous linear constraints on the n^2 entries of A. This argument is a special case of a more general theorem showing how efficient algorithms for black box polynomial identity testing give lower bounds for algebraic circuits [1, 26].

In practice, it is often convenient (although by no means necessary) to consider hitting sets that contain only rank 1 matrices $\mathbf{x}\mathbf{y}^T$, since $\langle A, \mathbf{x}\mathbf{y}^T \rangle = \mathbf{x}^T A \mathbf{y}$, and thus we find ourselves in the more familiar territory of polynomial identity testing, trying to construct a hitting set for the class of polynomials of the form $\mathbf{x}^T A \mathbf{y}$ for $A \in \mathcal{C}$. This approach was also

taken by Forbes and Shpilka [18], who considered this exact problem where $\mathcal{C}$ is the class of low-rank matrices, and remarked that hitting sets for the class of low-rank matrices plus sparse matrices will give an explicit construction of a rigid matrix.

We carry out this idea for two different classes in the proofs of Theorem 4 and Theorem 6. However, the following problem remains open.

▶ **Open Problem 7.** *For some $0 < \varepsilon \leq 1$, construct an explicit hitting set of size at most $n^2 - 1$ for the class of $n \times n$ matrices A which can be written as $A = BC$ where B, C have at most $n^{1+\varepsilon}$ non-zero entries.*

A solution to Open Problem 7 will imply lower bounds of the form $n^{1+\varepsilon}$ for an explicit matrix. If $\varepsilon = 1$, this will imply lower bounds for logarithmic depth linear circuits.

A useful ingredient in our constructions is the use of maximum distance separable (MDS) codes (for example, Reed-Solomon codes), as their dual subspace is a small dimensional subspace which does not contain sparse non-zero vectors. Over the reals, it is also easy to give such construction based on the well known Descartes' rule of signs which says that a sparse univariate real polynomial cannot have too many real roots. We refer the reader to Subsection 3.1 for details.

2 Lower bounds for constant depth linear circuits

In this section, we prove Theorem 2. We start by describing the notion of Shoup-Smolensky dimension, but first we set up some notation.

2.1 Notation

We work with matrices whose entries lie in an appropriate extension of a base finite field $\mathbb{F}_p$. We follow the natural convention that the elements of this extension will be represented as univariate polynomials of appropriate degree over the base field, and the arithmetic is done modulo an explicitly given irreducible polynomial.

We use boldface letters $(\mathbf{x}, \mathbf{y})$ to denote vectors. The length of the vectors is understood from the context.

For a matrix M, $\|M\|_0$ denotes the number of non-zero entries in M.

2.2 Shoup-Smolensky Dimension

A useful concept will be the notion of Shoup-Smolensky dimension of sequences of elements of an extension $\mathbb{E}$ of a field $\mathbb{F}$.

▶ **Definition 8** (Shoup-Smolensky dimension)**.** *Let $\mathbb{F}$ be a field, and $\mathbb{E}$ be an extension field of $\mathbb{F}$. Let $S = (a_1, \ldots, a_m)$ a sequence of elements of $\mathbb{E}$. For $t \in \mathbb{N}$, denote by $\Pi_t(S)$ the set of t-wise products of distinct entries of S that is,*

$$\Pi_t(S) = \left\{ \prod_{j=1}^{t} a_{i_j} : 1 \leq i_1 < i_2 < \cdots < i_t \leq m \right\}.$$

The Shoup-Smolensky dimension *of S of order t, denoted by $\Gamma_{t,\mathbb{F}}(S)$ is defined to be the dimension, over $\mathbb{F}$, of the vector space spanned by $\Pi_t(S)$.*

We also denote by $\Sigma_t(S)$ the number of distinct elements of $\mathbb{E}$ that can be obtained by summing *distinct elements of $\Pi_t(S)$.*

When $M \in \mathbb{E}^{n \times n}$ is a matrix we also regard it as a sequence of $m = n^2$ elements of $\mathbb{E}$ (under some order on the entries) and refer to the Shoup-Smolensky dimension of M.

2.3 Upper bounding the Shoup-Smolensky dimension for Sparse Products

The following lemma shows that any matrix computable by a depth-d linear circuit of size at most s has a somewhat small Shoup-Smolensky dimension.

▶ **Lemma 9.** *Let $\mathbb{F}$ be a field, $\mathbb{E}$ an extension of $\mathbb{F}$ and $A \in \mathbb{E}^{n\times n}$ be a matrix such that $A = \prod_{i=1}^{d} P_i$ for $P_i \in \mathbb{E}^{n_i \times m_i}$, where $\sum_{i=1}^{d} \|P_i\|_0 \leq s$. Then, for every $t \leq n^2/4$ such that $s \geq dt$ it holds that*

$$\Gamma_{t,\mathbb{F}}(A) \leq \left(e^d(2s/dt)^d\right)^t.$$

Proof. Since

$$A_{i,j} = \left(\prod_{\ell=1}^{d} P_\ell\right)_{i,j} = \sum_{k_1,\ldots,k_{d-1}} (P_1)_{i,k_1} \cdot \left(\prod_{\ell=2}^{d-1} (P_\ell)_{k_{\ell-1},k_\ell}\right) \cdot (P_d)_{k_{d-1},j},$$

every element in $\Pi_t(A)$ is a sum of monomials of degree dt in the entries of $P_1, P_2, \ldots, P_d$, that is,

$$\Gamma_{t,\mathbb{F}}\left(\prod_{i=1}^{d} P_i\right) \leq \binom{s+dt}{dt},$$

with the right hand side being the number of monomials of degree dt in s variables. Using the inequality $\binom{n}{k} \leq (en/k)^k$,

$$\Gamma_{t,\mathbb{F}}(A) \leq (e(1+s/dt))^{dt} \leq \left(e^d(2s/dt)^d\right)^t. \quad \blacktriangleleft$$

Over $\mathbb{Q}$, we do not wish to use field extensions (which would give rise to elements with infinite bit complexity). Thus, we use a similar argument that replaces the measure $\Gamma_{t,\mathbb{F}}$ with Σ_t (recall Definition 8) for a small tolerable penalty.

▶ **Lemma 10.** *Let d be a positive integer. Let $A \in \mathbb{Q}^{n\times n}$ be a matrix such that $A = \prod_{i=1}^{d} P_i$ for $P_i \in \mathbb{Q}^{n_i \times m_i}$, where $\sum_{i=1}^{d} \|P_i\|_0 \leq s$. Assume that for each i, $n_i \leq n^2$ and $m_i \leq n^2$. Then, for every $t \leq n^2/4$ such that $s \geq dt$ it holds that*

$$\Sigma_t(A) \leq 2^{2n^3 \cdot \left(e^d(2s/dt)^d\right)^t}.$$

Proof. We follow the same steps as in the proof of Lemma 9, replacing the measure $\Gamma_{t,\mathbb{F}}(A)$ by $\Sigma_t(A)$. As before,

$$A_{i,j} = \left(\prod_{\ell=1}^{d} P_\ell\right)_{i,j} = \sum_{k_1,\ldots,k_{d-1}} (P_1)_{i,k_1} \cdot \left(\prod_{\ell=2}^{d-1} (P_\ell)_{k_{\ell-1},k_\ell}\right) \cdot (P_d)_{k_{d-1},j}.$$

Every element in $\Pi_t(A)$ can be written as

$$\sum_{\alpha \in \mathcal{M}} c_\alpha \cdot \alpha \tag{1}$$

where $\mathcal{M}$ is the set of monomials of degree dt in the entries of $P_1, P_2, \ldots, P_d$, and each c_α is a non-negative integer of absolute value at most $s^{dt} \leq 2^{n^3}$ (since $s \leq n^2 d$ and d is $O(1)$). It now follows that each element in $\Sigma_t(A)$ has the same form as in (1), with $c_\alpha \leq |\Pi_t(A)| \cdot 2^{n^3} \leq 2^{2n^3}$. We conclude that

$$\Sigma_t(A) \leq (2^{2n^3})^{\binom{s+dt}{dt}},$$

which implies the statement of the lemma using the same bounds on binomial coefficients as in Lemma 9. ◀

We now move on to describe constructions of matrices which have large Shoup-Smolensky dimension, and then deduce lower bounds for them.

2.4 Sidon sets and hard univariate matrices

In this section, we describe a construction of a matrix $G \in \mathbb{F}[y]^{n\times n}$ which has a large value of $\Gamma_{t,\mathbb{F}}$. Let us denote $G_{i,j} = y^{e_{i,j}}$ for some non-negative integer $e_{i,j}$. For G to have a large Shoup-Smolensky dimension of order t, the set $S = \{e_{1,1}, e_{1,2}, \ldots, e_{n,n}\} \subseteq \mathbb{N}$ should have the property that $S^{(t)} := \{a_1 + a_2 + \ldots + a_t : a_i \in S \text{ distinct}\}$ has size comparable to $\binom{|S|}{t}$. A set S such that every subset of size t of S has a distinct sum is called a *t-wise Sidon set*. These are very well studied objects in arithmetic combinatorics, and explicit constructions are known for them in poly(n) time (e.g., Lemma 60 in [9]). However, another important parameter in the construction is the degree of y, and such a set will inevitably contain integers of size roughly $n^{\Omega(t)}$. Thus, the construction of G would take time which is not polynomially bounded in n. Below we give an elementary construction of such a set in time $n^{O(t)}$ (cf. [2]).

▶ **Lemma 11.** *Let t be a positive integer. There is a set $S \subseteq \mathbb{N}$ of size m such that:*
1. *$S^{(t)} := \{a_1 + a_2 + \ldots + a_t : a_i \in S$ distinct$\}$ has size $\binom{m}{t}$.*
2. *The maximal element in S is at most $m^{O(t)}$.*
3. *S can be constructed in time $m^{O(t)}$.*

Proof. Let $S' = \left\{1, 2, 2^2, \ldots, 2^{m-1}\right\}$. Clearly, every subset of S' has a distinct sum. For a prime p we denote $S_p = S' \bmod p = \{a \bmod p : a \in S'\}$, and we claim that there exists a prime $p \leq m^{O(t)}$ such that $|(S_p)^{(t)}| = \binom{m}{t}$. Since this condition can be checked in time $m^{O(t)}$, this would immediately imply the statement of the lemma, by checking this condition for every $p \leq m^{O(t)}$ and letting $S = S_p$ for a p which satisfies this condition.

For every subset $T \subseteq S'$ of size t, let σ_T denote the sum of its elements, and observe that $\sigma_T \leq 2^m$. Clearly, $\sigma_T \bmod p = \sigma_{T'} \bmod p$ if and only if $p \mid \sigma_T - \sigma_{T'}$, so it is enough to show that there exists $p \leq m^{O(t)}$ which does not divide

$$N := \prod_{\substack{T \neq T' \subseteq S' \\ |T| = |T'| = t}} (\sigma_T - \sigma_{T'}),$$

and therefore does not divide any of the terms on the right hand size. It further holds that $0 \neq N \leq (2^m)^{m^{O(t)}} = 2^{m^{O(t)}}$, so the existence of p now follows from the fact that N can have at most $\log N = m^{O(t)}$ distinct prime divisors, and from the prime number theorem. ◀

Given the above construction of t-wise Sidon sets, we now describe the construction of matrices with univariate polynomial entries which has large Shoup-Smolensky dimension.

▶ **Construction 12.** *Let $S = \{e_{i,j} : i, j \in [n]\}$ be a t-wise Sidon set of positive integers of size n^2 as in Lemma 11. Then, the matrix $G_{t,n} \in \mathbb{F}[y]^{n\times n}$ is defined as follows as $(G_t)_{i,j} = y^{e_{i,j}}$.*

The useful properties of Construction 12 are given by the following lemma.

▶ **Lemma 13.** *Let $t \leq n$ be a parameter, $S \subseteq N$ be a t-wise Sidon set of size n^2 and let $G_{t,n}$ be the matrix defined in Construction 12. Then, the following are true.*
1. *Every entry of $G_{t,n}$ is a monomial of degree at most $n^{O(t)}$.*
2. *$\Gamma_{t,\mathbb{F}}((G_{t,n})) \geq \binom{n^2}{t} \geq \left(\frac{n^2}{t}\right)^t$.*

Proof. The first item follows from the definition of $G_{t,n}$ and the properties of the set S in Lemma 11. The second item also follows from the properties of S and the definition of Shoup-Smolensky dimension, since every t-wise product of elements of $G_{t,n}$ gives a distinct monomial in y, and thus they are all linearly independent over the base field $\mathbb{F}$. ◀

2.5 Hard matrices over finite fields

From the univariate matrix in Construction 12, we now construct, for every p and parameter t, a matrix M over an extension of $\mathbb{F}_p$ which has large Shoup-Smolensky dimension over $\overline{\mathbb{F}}_p$ with the same parameters as $G_{t,n}$.

▶ **Lemma 14.** *Let p be a prime, and t be any positive integer. There is a matrix $M_{t,n} \in \mathbb{E}^{n\times n}$ over an extension $\mathbb{E}$ of $\mathbb{F}_p$ of degree $\exp(O(t\log n))$, which can be deterministically constructed in time $n^{O(t)}$, and satisfies*

$$\Gamma_{t,\mathbb{F}_p}(M_{t,n}) \geq \left(\frac{n^2}{t}\right)^t$$

Proof. Let $G_{t,n}$ be as in Construction 12, and let Δ be the maximum degree of any entry of $G_{t,n}$. Set $D = 10\cdot t\cdot\Delta = \exp(O(t\log n))$. We use Shoup's algorithm (see Theorem 3.2 in [48]) to construct an irreducible polynomial $g(z)$ of degree $D+1$ over $\mathbb{F}_p$ in deterministic $\mathrm{poly}(D, |\mathbb{F}_p|)$ time. Let α be a root of $g(z)$ in an extension $\mathbb{E}$ of $\mathbb{F}_p$, where $\mathbb{E} \equiv \mathbb{F}_p[z]/\langle g(z)\rangle$.[8] Then, it follows that $1, \alpha, \alpha^2, \ldots, \alpha^D$ are linearly independent over $\mathbb{F}$.

The matrix $M_{t,n}$ is obtained from G_t by just replacing every occurrence of the variable y by α. We now need to argue that $M_{t,n}$ continues to satisfy $\Gamma_{t,\mathbb{F}_p}(M_{t,n}) \geq \left(\frac{n^2}{t}\right)^t$. By the choice of α, it immediately follows that $\Gamma_{t,\mathbb{F}_p}(M_{t,n}) = \Gamma_{t,\mathbb{F}_p}(G_{t,n})$, since every monomial in the set $\Pi_t(M_{t,n})$ is mapped to a distinct power of α in $\{0,1,\ldots,D\}$, which are all linearly independent over $\mathbb{F}_p$.

The upper bound on the running time needed to construct $M_{t,n}$ now follows from the upper bound on the degree of the extension $\mathbb{E}$, and from Lemma 11. ◀

The following theorem now directly follows.

▶ **Theorem 15.** *Let p be any prime and $d \geq 2$ be a positive integer. Then, there exists a family of matrices $\{A_n\}_{n\in\mathbb{N}}$ which can be constructed in time $n^{O(n^{1-1/2d})}$ such that every depth-d linear circuit $\overline{\mathbb{F}}_p$ computing A_n has size at least $\Omega(n^{1+1/2d})$. Moreover, the entries of A_n lie in an extension of $\mathbb{F}_p$ of degree at most $\exp(O(n^{1-1/2d}\log n))$.*

Proof. We invoke Lemma 14 with parameter t set to $n^{1-1/2d}$ to get matrices $\{A_n\}$ in time $n^{O(t)}$ with the following lower bound on their Shoup-Smolensky dimension.

$$\Gamma_{t,\mathbb{F}_p}(M_n) \geq \left(\frac{n^2}{t}\right)^t .$$

If there is a depth d linear circuit of size s computing the linear transformation $A_n\cdot\mathbf{x}$, the following inequality must hold (from Lemma 9),

$$\left(e^d(2s/dt)^d\right)^t \geq \left(\frac{n^2}{t}\right)^t . \tag{2}$$

If $s \leq n^{1+1/2d}/2$, we have,

$$\left(e^d(2s/dt)^d\right)^t \leq (O(e/d))^{dt}\cdot n^t .$$

[8] We identify the elements of $\mathbb{E}$ with coefficient vectors of polynomials of degree at most D in $\mathbb{F}_p[z]$, and in this representation α is identified with the polynomial z.

We also have,

$$\left(\frac{n^2}{t}\right)^t \geq \left(n^{1+1/2d}\right)^t .$$

For any constant d, these estimates contradict Equation 2, thereby implying a lower bound of $\Omega(n^{1+1/2d})$ on s. ◀

2.6 Hard matrices over $\mathbb{C}$

An analog for Lemma 14 can be proved over $\mathbb{C}$ by constructing a matrix whose Σ_t-measure (rather than $\Gamma_{t,\mathbb{F}}$ as before) is large. The full statement and its proof appear in the full version of the paper. The analog of Theorem 15 for $\mathbb{C}$ is given below, with the proof again deferred to the full version.

▶ **Theorem 16.** *There exists a family of matrices $\{A_n\}_{n\in\mathbb{N}}$ over $\mathbb{Q}$ which can be constructed in time $n^{O(n^{1-1/2d})}$ such that every depth-d linear circuit $\mathbb{C}$ computing A_n has size at least $\Omega(n^{1+1/2d})$. Moreover, the entries of A_n are positive integers of bit complexity at most $\exp(O(n^{1-1/2d}\log n))$.*

2.7 Lower bounds for depth-2 linear circuits

The lower bounds of Theorem 16 and Theorem 15 apply to any constant depth. However, here we briefly remark that in the special case of $d = 2$ there is in fact a much simpler construction. As discussed in the introduction, for depth-2 linear circuits, the best lower bounds currently known is a lower bound of $\Omega\left(n\frac{\log^2 n}{\log\log n}\right)$ based on the study of super-concentrator graphs in the work of Radhakrishnan and Ta-Shma [45].

In the full version of the paper, we give two simple constructions of matrices in quasi-polynomial time which improve upon this bound.

3 Lower bounds via Hitting Sets

In this section, we prove lower bounds for several classes of depth 2 circuits using hitting sets for matrices. We first recall the definition.

▶ **Definition 17** (Hitting set for matrices, [18]). *Let $\mathcal{C} \subseteq \mathbb{F}^{n\times n}$ be a set of matrices. A set $\mathcal{H} \subseteq \mathbb{F}^n \times \mathbb{F}^n$ is said to be a* hitting set *for $\mathcal{C}$, if for every non-zero $M \in \mathcal{C}$, there is a pair $(\mathbf{a}, \mathbf{b}) \in \mathcal{H}$ such that*

$$\langle \mathbf{a}, M \cdot \mathbf{b}\rangle = \sum_{i\in[n],j\in[m]} M_{i,j}a_i b_j \neq 0.$$

3.1 Matrices with no sparse vectors in their kernel

In this section, we recall some simple, deterministic and efficient constructions of matrices which do not have any sparse non-zero vector in their kernel. Such a construction forms the basic building block for building hard instances of matrices for various cases of the matrix factorization problem that we discuss in the rest of this paper. We start by describing such a construction over the field of real numbers.

3.1.1 Construction over $\mathbb{R}$

The following is a weak form of a classical lemma of Descartes.

▶ **Lemma 18** (Descartes' rule of signs). *Let $d_1 < d_2 < \cdots < d_k$ be non-negative integers, and let $a_1, a_2, \ldots, a_k$ be arbitrary real numbers. Then, the number of distinct positive roots of the polynomial $\sum_{i=1}^{k} a_i x^{d_i}$ is at most $k-1$.*

Lemma 18 immediately gives the following construction of a small set of vectors, such that not all of them can lie in the kernel of any matrix with at least one sparse row.

▶ **Lemma 19.** *For $i \in [n]$, let $\mathbf{v}_i := \left(1, i, i^2, \ldots, i^{n-1}\right) \in \mathbb{R}^n$. Then, for every $1 \leq s \leq n$ and for every $m \times n$ matrix B over real numbers that has a non-zero row with at most s non-zero entries, there is an $i \in [s]$ such that $B \cdot \mathbf{v}_i \neq \mathbf{0}$.*

Proof. Let $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{R}^n$ be any non-zero vector with at most s non zero entries. So, the polynomial $P(x) = \sum_{i=0}^{n-1} a_i x^i$ has sparsity at most s. From Lemma 18, it follows that P has at most $s-1$ positive real roots. Therefore, there exists an $i \in [s]$ such that i is *not* a root of $P(x)$, i.e., $P(i) \neq 0$. The lemma now follows immediately by taking $(a_0, a_1, \ldots, a_{n-1})$ to be any non-zero s-sparse row of B. ◀

We remark that Lemma 19 also holds for matrices over $\mathbb{C}$ which have a sparse non-zero row for the choice of the vectors v_i as above. This follows from the application of Lemma 18 separately for the real and complex parts of a sparse complex polynomial, both of which are individually sparse, with real coefficients and at least one of them is not identically zero. This observation extends our results over $\mathbb{R}$ in Subsection 3.2 to the field of complex numbers.

3.1.2 Construction over finite fields

We now recall some basic properties of Reed-Solomon codes, and observe they can be used as well in lieu of the construction in Lemma 19.

The proofs for these properties can be found in any standard reference on coding theory, e.g., Chapter 5 in [25].

▶ **Definition 20** (Reed Solomon codes). *Let $\mathbb{F}_q = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ be the finite field with q elements and let $k \in \{0, 1, \ldots, q-1\}$. The Reed-Solomon code of block length q and dimension k are defined as follows.*

$$RS_q[q,k] = \{(P(\alpha_0), P(\alpha_1), \ldots, P(\alpha_{q-1})) : P(z) \in \mathbb{F}_q[z], \deg(P) \leq k-1\}.$$

▶ **Lemma 21.** *Let $\mathbb{F}_q$ be the finite field with q elements and let $k \in \{0, 1, \ldots, q-1\}$. The linear space $RS_q[q,k]$ as in Definition 20 satisfies the following properties.*

- *Every non-zero vector in $RS_q[q,k]$ has at least $q-k+1$ non-zero coordinates.*
- *The dual of $RS_q[q,k]$ is the space of Reed Solomon codes of block length q and dimension $q-k$.*

▶ **Lemma 22.** *Let $\mathbb{F}_q = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ be the finite field with q elements. For any $k \leq q-1$, let G_k be the $q \times k$ matrix over $\mathbb{F}_q$ whose i-th row is $(1, \alpha_{i-1}, \alpha_{i-1}^2, \ldots, \alpha_{i-1}^{k-1})$. Then, every non-zero vector in $\mathbb{F}_q^q$ in the kernel of $(G_k)^T$ has at least $k+1$ non-zero coordinates.*

Proof. Observe that G_k is the precisely the generator matrix of Reed Solomon codes of block length q and dimension k over $\mathbb{F}_q$. In particular, the linear space $RS_q[q,k]$ as in Lemma 21 is spanned by the columns of G_k. Thus any vector $\mathbf{w}$ in the kernel of $(G_k)^T$ is in fact a

codeword of the dual of these codes, which as we know from Item 2 of Lemma 21, is itself a Reed Solomon code of block length q and dimension $q-k$. From the first item of Lemma 21, it now follows that $\mathbf{w}$ has at least $k+1$ non-zero coordinates. ◀

The following lemma is an analog of Lemma 19.

▶ **Lemma 23.** *Let $\mathbb{F}_q = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ be the finite field with q elements, $s \in [q]$ be a parameter and let $\mathbf{v}_i$ be the i-th column of the matrix G_k as in Lemma 22 for $k = s$.*

Then, for every $m \times n$ matrix B over $\mathbb{F}_q$ that has a non-zero row with at most s non zero entries, there is an $i \in [s]$ such that $B \cdot \mathbf{v}_i \neq 0$.

Proof. The proof follows from the observation that any non-zero vector orthogonal to all the vectors $v_1, v_2, \ldots, v_s$ must be in the kernel of the matrix G_s^T and hence by Lemma 22 must have at least $s+1$ non-zero entries. ◀

3.2 Lower bounds for symmetric circuits

We now prove our lower bounds for symmetric circuits. Recall that a symmetric circuit is a linear depth-2 circuit of the form $B^T B$.

▶ **Theorem 24.** *There is an explicit family of positive semidefinite matrices $\{M_n\}$ such that every symmetric circuit computing M_n has size at least $n^2/4$.*

For the proof of this theorem, we give an efficient deterministic construction of a hitting set $\mathcal{H}$ for the set of matrices which factor as $B^T \cdot B$ for B of sparsity less than $n^2/4$, and as outlined in Subsection 1.7, we construct a hard matrix $M = \tilde{M}^T \cdot \tilde{M}$ which is not hit by such a hitting set and has a high rank.

We start by describing the construction of M.

▶ **Lemma 25.** *Let $\{\mathbf{v}_i : i \in [n]\}$ be the set of vectors defined in Lemma 19. There exists an explicit PSD matrix M of rank $n/2$ such that $\mathbf{v}_i^T M \mathbf{v}_i = 0$ for $i \in [n/2]$.*

Proof. We wish to find a matrix $\tilde{M}$ of high rank such that $\tilde{M}\mathbf{v}_i = 0$ for $i = 1, \ldots, n/2$. This can be done by completing $\{\mathbf{v}_i : i \in \{1, 2, \ldots, n/2\}\}$ to a basis (in an arbitrary way) and requiring that the other $n/2$ basis elements are mapped to linearly independent vectors under $\tilde{M}$. Conveniently, the set $\{\mathbf{v}_i : i \in [n]\}$ is itself a basis for $\mathbb{R}^n$: the matrix V whose rows are the $\mathbf{v}_i$'s is a Vandermonde matrix.

We now describe this in some more detail. For $i \in [n]$, let $\mathbf{e}_i$ by the i-th elementary basis vector. For a set of n^2 variables $Y = (y_{i,j})_{n \times n}$ consider the system of (non-homogeneous) linear equations on the variables Y given by the n constraints.

$$Y \cdot \mathbf{v}_i = 0 \quad \text{for } i \in \{1, 2, \ldots, n/2\}$$
$$Y \cdot \mathbf{v}_i = \mathbf{e}_i \quad \text{for } i \in \{n/2+1, \ldots, n\}\,.$$

Since the vectors $\{\mathbf{v}_i : i \in [n]\}$ are linearly independent, this system has a solution, which can be found in polynomial time using basic linear algebra. More explicitly the j-th row of Y, $\mathbf{y}_j$, is given by the solution to the linear system $V \cdot (\mathbf{y}_j)^T = 0$ for $1 \leq j \leq n/2$ and $V \cdot (\mathbf{y}_j)^T = \mathbf{e}_j$ for $n/2+1 \leq j \leq n$ where V is the Vandermonde matrix whose rows are the $\mathbf{v}_i$'s. Let $\tilde{M}$ be the matrix whose rows are the solution to the system above. Also, note that the rank of $\tilde{M}$ is at least $n/2$, as linearly independent vectors $\mathbf{e}_{n/2+1}, \mathbf{e}_{n/2+2}, \ldots, \mathbf{e}_n$ are in the image of the linear transformation given by $\tilde{M}$.

Now let $M = (\tilde{M}^T) \cdot \tilde{M}$, so that indeed M is a positive semi-definite matrix, and rank $M = n/2$ as well. It immediately follows that

$$\mathbf{v}_i^T M \mathbf{v}_i = (\mathbf{v}_i^T \tilde{M}^T)(\tilde{M}\mathbf{v}_i) = 0. \quad \blacktriangleleft$$

We are now ready to prove Theorem 24.

Proof of Theorem 24. Let M be the matrix from Lemma 25. Let $B \in \mathbb{R}^{m \times n}$ be real matrix such that $\|B\|_0 < n^2/4$, and suppose towards contradiction that $M = B^T B$.

It follows that the rank of B must be at least $n/2$. Thus, B must have at least $n/2$ non-zero rows. Now, since the total sparsity of B is at most $n^2/4 - 1$, there must be a non-zero row of B with sparsity at most $(n^2/4 - 1)/(n/2) \leq n/2$. From Lemma 19, it follows that there is an $i \in [n/2]$ such that $B \cdot \mathbf{v}_i$ is non-zero. Thus, for this index i, we have that

$$\mathbf{v}_i^T (B^T B)\mathbf{v}_i = \|B\mathbf{v}_i\|_2^2 \neq 0,$$

contradicting Lemma 25. $\blacktriangleleft$

We remark that the proof of Theorem 24 goes through almost verbatim for symmetric circuits over $\mathbb{C}$ (recall that over $\mathbb{C}$ these are circuits of form B^*B, where B^* is the conjugate transpose of B).

3.3 Lower bounds for invertible circuits

Recall that an invertible circuit is a circuit of them form BC where either B or C is invertible. In this section, we prove Theorem 6, which shows a quadratic lower bound for such circuits. For convenience, we restate the theorem.

▶ **Theorem 26.** *There exists an explicit family of $n \times n$ matrices $\{A_n\}$, over any field $\mathbb{F}$ such that $\mathbb{F} \geq \text{poly}(n)$, such that every invertible circuit computing A_n has size $n^2/4$.*

The proof of this theorem appears in the full version of the paper.

4 Open Problems

An important problem that continues to remain open is to prove a lower bound of the form $\Omega(n^{1+\varepsilon})$ for some constant $\varepsilon > 0$ for the depth-2 complexity of an explicit matrix. Such a lower bound would follow from an explicit hitting set of size at most $n^2 - 1$ for the class of polynomials of the form $\mathbf{x}^T BC\mathbf{y}$ such that $\|B\|_0 + \|C\|_0 \leq n^{1+\varepsilon}$.

Another natural question here is to understand if this PIT based approach can be used for explicit constructions of rigid matrices, which improve the state of art. One concrete question in this direction would be to construct explicit hitting sets for the set of matrices which are *not* (r, s) rigid for $rs > \omega(n^2 \log(n/r))$. Using the techniques in this paper, it is possible to construct hitting sets of size $O(rs)$ for matrices which are not (r, s) rigid. But, this is non-trivial only when $rs \leq cn^2$ for some constant $c < 1$, which is a regime of parameters for which explicit construction of rigid matrices is already known. A sequence of recent results [5, 15, 17] showed that many natural candidates for rigid matrices that posses certain symmetries are in fact not as rigid as suspected. This approach might circumvent these obstacles by giving an explicit construction which is not ruled out by these results.

A lower bound of s on the size of depth d linear circuits computing the linear transformation $A\mathbf{x}$ implies a lower bound of $\Omega(s)$ for depth $\Omega(d)$ algebraic circuits computing the degree-2 polynomial $\mathbf{y}^T A\mathbf{x}$ [7, 28] (so, we can convert lower bounds for circuits with n outputs to

lower bounds for circuits with 1 output). A notable open problem in algebraic complexity, which is very related to this work, is to prove any super-linear lower bound for algebraic circuits of depth $O(\log n)$ computing a polynomial with constant total degree. We refer to [46] for a discussion on the importance of this problem.

References

1 Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, pages 92–105, 2005. doi:10.1007/11590156_6.

2 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. doi:10.1137/140975103.

3 Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. doi:10.1109/FOCS.2008.32.

4 Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an np oracle. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*, 2019. URL: http://joshalman.com/AlmanChenFOCS19.pdf.

5 Josh Alman and R. Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC 2017)*, pages 641–652. ACM, 2017. doi:10.1145/3055399.3055484.

6 Noga Alon and Pavel Pudlák. Superconcentrators of depths 2 and 3; odd levels help (rarely). *J. Comput. Syst. Sci.*, 48(1):194–202, 1994. doi:10.1016/S0022-0000(05)80027-3.

7 Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.

8 Avraham Ben-Aroya, Dean Doron, and Amnon Ta-Shma. An efficient reduction from two-source to non-malleable extractors: achieving near-logarithmic min-entropy. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC 2017)*, pages 1185–1194. ACM, 2017. doi:10.1145/3055399.3055423.

9 Nader H. Bshouty. Testers and their applications. In *Innovations in Theoretical Computer Science, ITCS'14, 2014*, pages 327–352, 2014. doi:10.1145/2554797.2554828.

10 Peter Bürgisser, Michael Clausen, and Mohammad A. Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1997. doi:10.1007/978-3-662-03338-8.

11 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC 2016)*, pages 670–683. ACM, 2016. doi:10.1145/2897518.2897528.

12 Bernard Chazelle. *The discrepancy method - randomness and complexity*. Cambridge University Press, 2001. URL: https://www.cs.princeton.edu/~chazelle/pubs/book.pdf.

13 Gil Cohen. Towards optimal two-source extractors and ramsey graphs. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC 2017)*, pages 1157–1170. ACM, 2017. doi:10.1145/3055399.3055429.

14 Danny Dolev, Cynthia Dwork, Nicholas Pippenger, and Avi Wigderson. Superconcentrators, generalizers and generalized connectors with limited depth (preliminary version). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC 1983)*, pages 42–51. ACM, 1983. doi:10.1145/800061.808731.

15 Zeev Dvir and Benjamin Edelman. Matrix rigidity and the croot-lev-pach lemma. *CoRR*, abs/1708.01646, 2017. arXiv:1708.01646.

16 Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static data structure lower bounds imply rigidity. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:188, 2018. URL: https://eccc.weizmann.ac.il/report/2018/188.

17 Zeev Dvir and Allen Liu. Fourier and circulant matrices are not rigid. *CoRR*, abs/1902.07334, 2019. arXiv:1902.07334.

18 Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 163–172. ACM, 2012. doi:10.1145/2213977.2213995.

19 Michael L. Fredman. The complexity of maintaining an array and computing its partial sums. *J. ACM*, 29(1):250–260, 1982. doi:10.1145/322290.322305.

20 Michael L. Fredman and Michael E. Saks. The cell probe complexity of dynamic data structures. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC 1989)*, pages 345–354. ACM, 1989. doi:10.1145/73007.73040.

21 Joel Friedman. A note on matrix rigidity. *Combinatorica*, 13(2):235–239, June 1993. doi:10.1007/BF01303207.

22 Anna Gál, Kristoffer Arnsfelt Hansen, Michal Koucký, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. *IEEE Trans. Information Theory*, 59(10):6611–6627, 2013. doi:10.1109/TIT.2013.2270275.

23 Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. *Theor. Comput. Sci.*, 379(3):405–417, 2007. doi:10.1016/j.tcs.2007.02.047.

24 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

25 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory, 2018. URL: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

26 Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 262–272, 1980. doi:10.1145/800141.804674.

27 Stasys Jukna and Igor Sergeev. Complexity of linear boolean operators. *Foundations and Trends in Theoretical Computer Science*, 9(1):1–123, 2013. doi:10.1561/0400000063.

28 Erich Kaltofen and Michael F. Singer. Size efficient parallel algebraic circuits for partial derivatives. In *IV International Conference on Computer Algebra in Physical Research*, pages 133–145, 1991. URL: https://users.cs.duke.edu/~elk27/bibliography/91/KaSi91.pdf.

29 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theoretical Computer Science*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

30 Kasper Green Larsen. The cell probe complexity of dynamic range counting. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 85–94. ACM, 2012. doi:10.1145/2213977.2213987.

31 Kasper Green Larsen. On range searching in the group model and combinatorial discrepancy. *SIAM J. Comput.*, 43(2):673–686, 2014. doi:10.1137/120865240.

32 Kasper Green Larsen, Omri Weinstein, and Huacheng Yu. Crossing the logarithmic barrier for dynamic boolean data structure lower bounds. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 978–989. ACM, 2018. doi:10.1145/3188745.3188790.

33 Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000*, pages 556–562. MIT Press, 2000. URL: http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.

34 Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. *CoRR*, abs/1804.04005, 2018. arXiv:1804.04005.

35 Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009. doi:10.1561/0400000011.

36 Julien Mairal, Francis R. Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 689–696. ACM, 2009. doi:10.1145/1553374.1553463.

37 Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast fourier transform. *J. ACM*, 20(2):305–306, 1973. doi:10.1145/321752.321761.

38 Behnam Neyshabur and Rina Panigrahy. Sparse matrix factorization. *CoRR*, abs/1311.3315, 2013. arXiv:1311.3315.

39 Mihai Pătraşcu. Lower bounds for 2-dimensional range counting. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 40–46. ACM, 2007. doi:10.1145/1250790.1250797.

40 Mihai Pătraşcu and Erik D. Demaine. Logarithmic lower bounds in the cell-probe model. *SIAM J. Comput.*, 35(4):932–963, 2006. doi:10.1137/S0097539705447256.

41 Nicholas Pippenger. Superconcentrators. *SIAM J. Comput.*, 6(2):298–304, 1977. doi:10.1137/0206022.

42 Nicholas Pippenger. Superconcentrators of depth 2. *J. Comput. Syst. Sci.*, 24(1):82–90, 1982. doi:10.1016/0022-0000(82)90056-3.

43 Pavel Pudlák. Communication in bounded depth circuits. *Combinatorica*, 14(2):203–216, 1994. doi:10.1007/BF01215351.

44 Pavel Pudlák. A note on the use of determinant for proving lower bounds on the size of linear circuits. *Inf. Process. Lett.*, 74(5-6):197–201, 2000. doi:10.1016/S0020-0190(00)00058-2.

45 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24, 2000. doi:10.1137/S0895480197329508.

46 Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6(1):135–177, 2010. doi:10.4086/toc.2010.v006a007.

47 Mohammad Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. A remark on matrix rigidity. *Inf. Process. Lett.*, 64(6):283–285, 1997. doi:10.1016/S0020-0190(97)00190-7.

48 Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54:435–447, 1990. URL: https://www.ams.org/journals/mcom/1990-54-189/S0025-5718-1990-0993933-0/S0025-5718-1990-0993933-0.pdf.

49 Victor Shoup and Roman Smolensky. Lower bounds for polynomial evaluation and interpolation problems. *Computational Complexity*, 6(4):301–311, December 1996. doi:10.1007/BF01270384.

50 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010. doi:10.1561/0400000039.

51 Volker Strassen. Die berechnungskomplexität von elementarsymmetrischen funktionen und von interpolationskoeffizienten. *Numerische Mathematik*, 20(3):238–251, June 1973. doi:10.1007/BF01436566.

52 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*. doi:10.1016/j.ic.2014.09.004.

53 Leslie G. Valiant. On non-linear lower bounds in computational complexity. In *Proceedings of the 7th Annual ACM Symposium on Theory of Computing (STOC 1975)*, pages 45–53. ACM, 1975. doi:10.1145/800116.803752.

54 Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Proceedings of the 2nd Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1977)*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977. doi:10.1007/3-540-08353-7_135.

Log-Seed Pseudorandom Generators via Iterated Restrictions

Dean Doron
Department of Computer Science, Stanford University, CA, USA
https://cs.stanford.edu/~ddoron/
ddoron@stanford.edu

Pooya Hatami
Department of Computer Science & Engineering, Ohio State University, Columbus, OH, USA
https://pooyahatami.org/
pooyahat@gmail.com

William M. Hoza
Department of Computer Science, University of Texas at Austin, TX, USA
https://williamhoza.com/
whoza@utexas.edu

Abstract

There are only a few known general approaches for constructing explicit pseudorandom generators (PRGs). The "iterated restrictions" approach, pioneered by Ajtai and Wigderson [2], has provided PRGs with seed length polylog n or even $\widetilde{O}(\log n)$ for several restricted models of computation. Can this approach ever achieve the optimal seed length of $O(\log n)$?

In this work, we answer this question in the affirmative. Using the iterated restrictions approach, we construct an explicit PRG for *read-once depth-2* $\mathbf{AC}^0[\oplus]$ *formulas* with seed length

$$O(\log n) + \widetilde{O}(\log(1/\varepsilon)).$$

In particular, we achieve optimal seed length $O(\log n)$ with near-optimal error $\varepsilon = \exp(-\widetilde{\Omega}(\log n))$. Even for constant error, the best prior PRG for this model (which includes read-once CNFs and read-once $\mathbb{F}_2$-polynomials) has seed length $\Theta(\log n \cdot (\log\log n)^2)$ [22].

A key step in the analysis of our PRG is a tail bound for *subset-wise symmetric polynomials*, a generalization of elementary symmetric polynomials. Like elementary symmetric polynomials, subset-wise symmetric polynomials provide a way to organize the expansion of $\prod_{i=1}^{m}(1+y_i)$. Elementary symmetric polynomials simply organize the terms by *degree*, i.e., they keep track of the number of variables participating in each monomial. Subset-wise symmetric polynomials keep track of more data: for a fixed partition of $[m]$, they keep track of the number of variables *from each subset* participating in each monomial. Our tail bound extends prior work by Gopalan and Yehudayoff [17] on elementary symmetric polynomials.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Pseudorandom generators, Pseudorandom restrictions, Read-once depth-2 formulas, Parity gates

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.6

Funding *Dean Doron*: Supported by NSF grant CCF-1763311. Part of this work was done while at UT Austin and supported by NSF grant CCF-1705028.
Pooya Hatami: Supported by NSF grant CCF-1947546. Part of this work was done while at UT Austin and supported by a Simons Investigator Award (#409864, David Zuckerman).
William M. Hoza: Supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin.

Acknowledgements We thank David Zuckerman for very helpful discussions.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 6; pp. 6:1–6:36

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

1 Introduction

The famous "**L** vs. **BPL**" problem asks whether randomness is ever truly necessary for space-efficient computation. To prove $\mathbf{L} = \mathbf{BPL}$, it suffices to design a suitable *pseudorandom generator* (PRG), i.e., an efficient algorithm that stretches a short truly random seed to a long bitstring that "looks random". To be more specific, the action of a small-space algorithm on its random bits can be modeled by a *read-once branching program* (ROBP). Therefore, to prove $\mathbf{L} = \mathbf{BPL}$, it suffices to design an efficient PRG with seed length $O(\log n)$ that fools polynomial-width ROBPs.

A large and growing body of work has made significant progress toward this ambitious goal. Most work on **L** vs. **BPL** can be broadly divided into two main approaches.

1.1 The "Seed Recycling" Approach

The "classical" approach to **L** vs. **BPL** is based on the observation that there is limited communication between the first half of an ROBP and its second half. Therefore, after using a few truly random bits to generate the first half of a pseudorandom string, the truly random bits can be efficiently *recycled* to generate the second half of the pseudorandom string. This insight is essentially due to Nisan [26].

Of the line of work that uses this approach, some highlights include PRGs for polynomial-width ROBPs with seed length $O(\log^2 n)$ [26, 20, 15]; PRGs for constant-width "regular" ROBPs with seed length $\widetilde{O}(\log n)$ [7, 11, 21, 32, 6]; and derandomization techniques that go beyond the construction of PRGs [27, 31]. More recently, this "seed recycling" approach has been used to obtain improved generators for polynomial-width ROBPs when the error parameter ε is very small [5, 19].

1.2 The "Iterated Restrictions" Approach

The more "modern" approach to **L** vs. **BPL** is to design a pseudorandom generator by *iterated pseudorandom restrictions*. That is, we pseudorandomly assign values to a pseudorandomly chosen subset of the variables, and then repeat the process to assign values to all variables. Intuitively, designing a pseudorandom restriction for some function f is easier than fooling f outright, because designing a pseudorandom restriction amounts to fooling a "smoothed out" version of f [16], or equivalently, designing a PRG that *would* fool f if some *noise* were added [18]. This "iterated restrictions" approach goes back to early work by Ajtai and Wigderson [2], but its modern incarnation is largely due to Gopalan et al. [16].

Of the line of work that takes this approach, some highlights include PRGs for *arbitrarily-ordered* ROBPs with seed length $\operatorname{polylog} n$ [33, 9, 14]; PRGs for width-3 ROBPs with seed length $\widetilde{O}(\log n)$ [16, 33, 24]; PRGs for bounded-depth read-once formulas with seed length $\widetilde{O}(\log n)$ [16, 10, 13]; and near-optimal PRGs for arbitrary-order product tests [18, 22].

1.3 Log-Seed PRGs and Our Main Result

At two extremes, one can either try to derandomize *all* of **BPL** as *efficiently as possible* (e.g. [26, 31]), or else one can try to *optimally* derandomize *as much* of **BPL** as possible (e.g. [28, 29]). Let us adopt the second goal.

In some cases, the "seed recycling" approach has indeed yielded PRGs with truly optimal seed length, at least for moderate error. For example, PRGs are known with seed length $O(\log n)$ that fool all $O(\log n)$-space algorithms that use only $\operatorname{polylog}(n)$ random bits in the first place [1, 28, 19]. For another example, PRGs for constant-width "permutation" ROBPs are known with seed length $O(\log n)$ [11, 21, 32].

The present work considers the question of whether the "iterated restrictions" approach can also yield a PRG with seed length $O(\log n)$ for some interesting class of tests. At first glance, this might seem doubtful, since after all we must pay for many pseudorandom restrictions. Nevertheless, we answer in the affirmative, proving the following theorem.

▶ **Theorem 1.** *For all $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for read-once depth-2 $\mathbf{AC}^0[\oplus]$ formulas on n input bits with seed length*

$$O(\log n) + \widetilde{O}(\log(1/\varepsilon)).$$

Specifically, the seed length of our PRG is $O\left(\log n + \log(1/\varepsilon) \cdot (\log\log(1/\varepsilon))^5\right)$. One can prove a lower bound of $\Omega(\log n + \log(1/\varepsilon))$ on the seed length of any PRG for this model.[1]

1.4 Read-Once Depth-2 $\mathbf{AC}^0[\oplus]$ Formulas

The class of functions that is fooled by our PRG (read-once depth-2 formulas over the basis $\{\wedge, \vee, \oplus\}$, with negations allowed at the inputs for free) is certainly of interest. It includes *read-once CNFs* and *read-once $\mathbb{F}_2$-polynomials* as special cases. The problems of fooling these classes have both received a lot of attention [12, 16, 4, 23, 24, 22]. Previously, even for read-once CNFs, PRGs with seed length $O(\log n)$ were only known for *constant* error [8, 12], whereas our PRG maintains seed length $O(\log n)$ with *near-optimal* error $\varepsilon = \exp(-\widetilde{\Omega}(\log n))$. Meanwhile, for read-once $\mathbb{F}_2$-polynomials, no PRGs with seed length $O(\log n)$ were known at all prior to our work.

Gopalan et al. did give a PRG with *near*-optimal seed length $\widetilde{O}(\log(n/\varepsilon))$ for read-once CNFs, and more generally for read-once depth-2 $\mathbf{AC}^0[\oplus]$ formulas with the property that the output gate is not $\oplus$ [16]. They used their PRG to construct a near-optimal hitting set for width-3 ROBPs [16]. A subsequent line of work provided near-optimal PRGs for *all* read-once depth-2 $\mathbf{AC}^0[\oplus]$ formulas [23, 24, 22].[2]

Conversely, a read-once depth-2 $\mathbf{AC}^0[\oplus]$ formula can be simulated by a width-4 ROBP (after suitably permuting the variables). The problems of designing improved PRGs for width-4 ROBPs and for read-once $\mathbf{AC}^0[\oplus]$ formulas of any constant depth are two major frontiers in unconditional pseudorandomness [24, 13]. The model we study in this paper is an interesting special case.

1.5 Overview of Our Approach

Let us focus on the problem of designing a PRG with seed length $O(\log n)$, with ε as small as possible. For simplicity, assume the test function is a read-once $\mathbb{F}_2$-polynomial $f = f_1 \oplus \cdots \oplus f_m$.

1.5.1 One Restriction

Ultimately, we wish to design a full PRG via *iterated* pseudorandom restrictions. To begin, we will explain how to construct just *one* pseudorandom restriction that assigns values to a constant fraction of the inputs. We use almost $O(\log n)$-wise independence to select the subset of inputs to keep "alive" for each coordinate, where the probability of staying alive is a constant $p \approx 1$. We use a small-bias distribution to assign values to the remaining inputs. Sampling this pseudorandom restriction only costs $O(\log n)$ truly random bits.

[1] This lower bound holds already for fooling parity functions.

[2] The PRGs we are referring to were designed to fool read-once $\mathbb{F}_2$-polynomials, but in fact they fool all of read-once depth-2 $\mathbf{AC}^0[\oplus]$.

We must show that our pseudorandom restriction X is correct. That is, we need to show that

$$\left| \mathop{\mathbb{E}}_{X,U} [f|_X(U)] - \mathbb{E}[f] \right| \leq \varepsilon,$$

where U is a uniform random variable over $\{0,1\}^n$.

We will outline three different arguments for proving correctness, each of which works under certain assumptions about f. We defer to the full proof to explain how to stitch these three arguments together to get a general proof of correctness for any f.

1.5.1.1 Argument 1: Keeping Many Terms Alive

Assume f is a *homogeneous* $\mathbb{F}_2$-polynomial of degree $w \gg \log\log n$, and assume there are many terms, $m \geq 3^w$. (For simplicity, in this informal discussion, we are making stronger assumptions than necessary.) Since f is the *parity* of all these terms, one can show from these assumptions that f is approximately *balanced*, i.e., $\mathbb{E}[f] \approx \frac{1}{2}$. Under a *truly* random restriction, for each term, the probability that all variables in the term remain alive would be p^w, so with high probability, the number of nonconstant terms after the restriction would be at least $m \cdot p^w \geq (3p)^w$. Standard techniques suffice to derandomize this calculation, so after our pseudorandom restriction, with high probability, there are still many terms alive – enough that the restricted function is still approximately balanced.

1.5.1.2 Argument 2: The Forbes-Kelley Approach [14]

Building on prior work [30, 18, 9], Forbes and Kelley showed that a restriction based on δ-biased distributions preserves the expectation of any arbitrary-order constant-width ROBP to within error $1/n$, where $\log(1/\delta) = O(\log n \log\log n)$ [14]. Our test function f can be simulated by a width-4 ROBP under some variable order. Unfortunately, given our budget of $O(\log n)$ truly random bits, we can only afford to sample from a $(1/\operatorname{poly}(n))$-biased distribution.

To move forward, let us turn things around a little: the analysis of Forbes and Kelley shows that a restriction based on δ-biased distributions preserves the expectation to within error ε, where $\varepsilon = \exp(-\Omega(\log(1/\delta)/\log\log(1/\delta)))$. The point is that this latter statement holds even for a relatively large δ, *assuming* the ROBP reads at most $1/\varepsilon$ variables. Therefore, if we assume that our test function f only reads a few variables (say, $\operatorname{polylog} n$ many), then the Forbes-Kelley approach shows that our pseudorandom restriction preserves the expectation of f to within error $\varepsilon = \exp(-\Omega(\log n/\log\log n))$.

1.5.1.3 Argument 3: Subset-Wise Symmetric Polynomials

Assume this time that the degree of every term of f is in the interval $[C\log\log n, C\log n]$ for some appropriate constant C. Assume also that for every w, there are at most 3^w terms of degree w. For this case, we return to an older approach based on symmetric polynomials [16, 17, 24], introduced by Gopalan et al. [16]. The idea is as follows. Let $Z \in \{0,1\}^n$ indicate which variables will remain alive. For convenience, for any $\{0,1\}$-valued function f, let $\overline{f} = (-1)^f$. Having already sampled Z, our remaining task is to argue that the small-bias distribution Y fools the "bias function" defined by

$$\widetilde{f}(x) = \mathop{\mathbb{E}}_{U}[\overline{f}(x + Z \wedge U)].$$

Translating $\{0,1\}$ to $\{\pm 1\}$, the $\oplus$ operation becomes multiplication, i.e., $\overline{f} = \prod_i \overline{f_i}$. For independent random variables, product and expectation can be interchanged, so the bias function of f is the product of the bias functions of the f_i-s. Define $\breve{f}_i$ so that the bias function of f_i is $\mathbb{E}[\overline{f_i}] \cdot (1 + \breve{f}_i)$. That way,

$$\widetilde{f} = \mathbb{E}[\overline{f}] \cdot \prod_{i=1}^{m} (1 + \breve{f}_i). \tag{1}$$

The approach used in prior work [16, 17, 24] is to expand Equation (1) in terms of *elementary symmetric polynomials*. Recall that for $y \in \mathbb{R}^m$, the k-th elementary symmetric polynomial $S_k(y)$ is defined by

$$S_k(y) = \sum_{\substack{I \subseteq [m] \\ |I| = k}} \prod_{i \in I} y_i.$$

We can expand Equation (1) as

$$\widetilde{f} = \mathbb{E}[\overline{f}] \cdot \sum_{k=0}^{m} S_k(\breve{f}_1, \ldots, \widecheck{f_m}). \tag{2}$$

Therefore, the error of our pseudorandom restriction is captured by $\sum_{k=1}^{m} S_k(\breve{f}_1, \ldots, \widecheck{f_m})$. Now we can reason as follows. Pick a cutoff point k_0.

- For $k \leq k_0$, we do a Fourier L_1 calculation to show that $S_k(\breve{f}_1, \ldots, \widecheck{f_m})$ has near-zero expectation even under the small-bias distribution Y.
- For $k \approx k_0$, we do a variance calculation to show that $S_k(\breve{f}_1, \ldots, \widecheck{f_m})$ is small *with high probability* under the uniform distribution, hence also under Y by the previous L_1 calculation.
- Finally we invoke a *tail bound* [17], which says that if S_{k_0} and S_{k_0+1} are both small, then the sum of all subsequent values is also small.

How should we choose the cutoff point k_0? If f is a *homogeneous* $\mathbb{F}_2$-polynomial of degree w, then we should pick $k_0 = \Theta(\frac{\log n}{w})$. That way, k_0 is small enough for the L_1 calculation to work out, because the number of monomials in $S_{k_0}(y_1, \ldots, y_m)$ is

$$\binom{m}{k_0} \leq m^{k_0} \leq 3^{wk_0} \leq \operatorname{poly}(n).$$

But at the same time, k_0 is large enough to sufficiently dampen $S_k(\breve{f}_1, \ldots, \widecheck{f_m})$ for $k \approx k_0$. In fact, one can show that

$$\mathbb{E}[S_k^2(\breve{f}_1(Y), \ldots, \widecheck{f_m}(Y))] \leq \frac{\exp(-\Omega(wk))}{k!},$$

which for $k \approx k_0$ is $\frac{1}{\operatorname{poly}(n) \cdot k!}$. This is small enough for the tail bound to give an overall error of $1/\operatorname{poly}(n)$.

The difficulty, of course, is that f is not necessarily homogeneous, i.e., the terms of f do not necessarily all have the same degree. To address this difficulty, following prior work, let us partition the terms of f into $Q = O(\log \log n)$ *buckets* based on degree, say $f = F_1 \oplus F_2 \oplus \cdots \oplus F_Q$. For each bucket $q \in [Q]$, there is a suitable cutoff point k_0, so our restriction preserves the expectation of F_q.

At this point, the approach taken by prior work has been to invoke a generic *XOR lemma* (see Lemma 6) to argue that our restriction must also preserve the expectation of the parity of the F_q's, i.e., our test function f. This XOR lemma is a suitable generalization of the fact that the Fourier L_1 norm is submultiplicative. Unfortunately, invoking the XOR lemma would require us to start with a smaller-bias distribution Y. Effectively, to invoke the XOR lemma, we would have to pay a factor of Q in the seed length, which we cannot afford.

Therefore, we take a different approach. Our observation is that ideally, the cutoff point k_0 should guarantee that every product $\prod_{i\in I} \widecheck{f}_i$ appearing in $S_{k_0}(\widecheck{f}_1, \ldots, \widecheck{f_m})$ involves $\Theta(\log n)$ of the *input* variables $x_1, \ldots, x_n$. Intuitively, that's why the right choice is $k_0 = \Theta(\frac{\log n}{w})$ for degree w. When the terms of f do not all have the same degree, the products $\prod_{i\in I} \widecheck{f}_i$ appearing in $S_k(\widecheck{f}_1, \ldots, \widecheck{f_m})$ do not all involve the same number of input variables $x_1, \ldots, x_n$, hence there isn't a well-defined correct choice of k_0. This suggests that Equation (2) is simply *not the best expansion* of Equation (1).

These observations motivate the definition of *subset-wise symmetric polynomials*. We defer to Section 2 for the precise definition, but the point is that they allow us to give a more refined expansion of Equation (1), where instead of just keeping track of k (the number of f_i-s participating in each monomial of S_k) we keep track of a whole vector $\vec{k}$ giving the numbers of f_i-s *from each bucket* participating in each monomial of $S_{\vec{k}}$. This allows us to define a norm $\|\vec{k}\|$ that measures the number of *input* variables $x_1, \ldots, x_n$ that participate in each monomial of $S_{\vec{k}}(\widecheck{f}_1, \ldots, \widecheck{f_m})$.

We expand Equation (1) in terms of subset-wise symmetric polynomials by summing over all vectors $\vec{k}$:

$$\widetilde{f} = \mathbb{E}[f] \cdot \sum_{\vec{k}\in\mathbb{N}^Q} S_{\vec{k}}(\widecheck{f}_1, \ldots, \widecheck{f_m}).$$

Now we can cut off this sum at $\|\vec{k}\| = \Theta(\log n)$. To complete the argument, we extend known tail bounds for elementary symmetric polynomials [17] to the case of subset-wise symmetric polynomials.

1.5.2 Iterating the Restriction to Get a Full PRG

So far, we have outlined the proof that our pseudorandom restriction preserves the expectation of the test function f. Our pseudorandom restriction costs $O(\log n)$ truly random bits. But our goal is to design a full PRG with seed length $O(\log n)$. It seems that one restriction already uses up our entire budget of truly random bits, so how can we afford to iterate the process?

A key insight is that if f only reads n' variables ($n' \le n$), then a pseudorandom restriction for f ought to only cost $O(\log n')$ truly random bits rather than $O(\log n)$. This intuition can be justified using standard constructions of n'-wise small-bias distributions [25, 3], provided $n' \ge \log n$. (A similar insight was used previously by Lee and Viola [23].) Let C be a constant such that one pseudorandom restriction costs $C \log n'$ truly random bits.

To simplify the discussion, assume f is homogeneous of degree $w = \Theta(\log n)$. Each restriction keeps approximately a p-fraction of variables alive. For simplicity, assume that in each term, *exactly* a p-fraction of variables remain alive, i.e., assume that after i pseudorandom restrictions, the restricted $\mathbb{F}_2$-polynomial is homogeneous of degree $p^i w$.

We divide into two cases. For the first case, suppose that the number of terms is always at most exponential in the degree. Specifically, suppose the number of terms is at most $16^{w'}$, where w' is the degree at that stage. In this case, our pseudorandom restrictions get cheaper and cheaper as we go. Quantitatively, after i restrictions, the restricted polynomial reads only n' variables, where $n' = p^i w \cdot 16^{p^i w}$. Therefore, the cost of restriction $i+1$ is only

$$C \log\left(p^i w \cdot 16^{p^i w}\right) \leq 5C \cdot p^i w.$$

Therefore, if we do a total of t pseudorandom restrictions, the total cost is bounded by

$$\sum_{i=0}^{t-1} 5Cp^i w.$$

This geometric sum is bounded by $O(w) = O(\log n)$, *regardless* of t. To optimize the error of our PRG, we choose $t = O(\log \log \log n)$; after this many restrictions, the number of living variables is small enough that we can stop the iteration and apply a prior *near*-optimal PRG by Lee [22] to finish the job.

For the second case, suppose that at some stage the number of terms is enormous compared to the degree: the degree is w' and the number of terms is more than $16^{w'}$. This setting was studied previously by Meka, Reingold, and Tal [24], who gave an *optimal* PRG for any function that can be written as a parity of an enormous number of functions on small disjoint variable sets. Therefore, in this case, we can stop doing pseudorandom restrictions, and instead fool the function outright using the PRG by Meka et al. [24].

Of course we do not know in advance which case we are in, but this difficulty can be resolved by straightforward XORing.

2 Subset-Wise Symmetric Polynomials

In this section, we will formally define subset-wise symmetric polynomials and prove suitable tail bounds for them. This section can be read on its own, independent of the application to PRGs. We start by recalling known tail bounds for elementary symmetric polynomials.

2.1 Gopalan and Yehudayoff's Bounds for Symmetric Polynomials

As a reminder, the k-th elementary symmetric polynomial is defined by

$$S_k(y) = \sum_{\substack{I \subseteq [m], \\ |I|=k}} \prod_{i \in I} y_i.$$

We rely on the following tail bound by Gopalan and Yehudayoff [17]. As discussed in Section 1.5.1, the bound says that if two S_k-s in a row are small, then all subsequent S_k-s are small.

▶ **Theorem 2** ([17]). *Let $y \in \mathbb{R}^m$, $\theta > 0$, and $\ell \in \mathbb{N}$ satisfy $S_\ell^2(y) \leq \frac{\theta^\ell}{\ell!}$ and $S_{\ell+1}^2(y) \leq \frac{\theta^{\ell+1}}{(\ell+1)!}$. Then, for every $k \geq \ell$,*

$$|S_k(y)| \leq \left(\frac{64e^2\theta\ell}{k}\right)^{k/2}.$$

The exact statement of Theorem 2 does not appear in Gopalan and Yehudayoff's work [17], but it follows readily from their analysis, and it was used previously by Meka et al. [24, Theorem 5.2].[3]

2.2 Our Tail Bounds for Subset-Wise Symmetric Polynomials

Let $\mathcal{B} = (B_1, \ldots, B_Q)$ be a partition of $[m]$, namely $[m] = B_1 \sqcup \cdots \sqcup B_Q$. (The sets $B_1, \ldots, B_Q$ correspond to the "buckets" discussed in Section 1.5.1.) Throughout this paper, let $\mathbb{N}$ denote the set of *nonnegative* integers, $\mathbb{N} = \{0, 1, 2, \ldots\}$. For a vector $\vec{k} = (\vec{k}[1], \ldots \vec{k}[Q]) \in \mathbb{N}^Q$ and $y \in \mathbb{R}^m$, we define the following polynomial:

$$S_{\vec{k},\mathcal{B}}(y) = \sum_{\substack{I \subseteq [m], \\ \forall q, |B_q \cap I| = \vec{k}[q]}} \prod_{i \in I} y_i.$$

We name these polynomials as *subset-wise symmetric polynomials*, since for every $q \in [Q]$, $S_{\vec{k}}(y)$ when restricted to the B_q variables is a degree $\vec{k}[q]$ symmetric polynomial.

Throughout this section we fix $\mathcal{B} = (B_1, ..., B_Q)$ to be a partition of $[m]$. When the partition $\mathcal{B}$ is clear from the context, we will simply write $S_{\vec{k}}$ instead of $S_{\vec{k},\mathcal{B}}$. To formulate our tails bounds for the subset-wise symmetric polynomials, we will need the following auxiliary polynomials:

$$R_{\vec{k}}(y) \stackrel{\text{def}}{=} S^2_{\vec{k}}(y) \cdot \prod_{q=1}^{Q} \vec{k}[q]!.$$

Given $c > 1$, we will assign each vector $\vec{k} \in \mathbb{N}^Q$ a weight, defined as

$$\|\vec{k}\|_{(c)} = \sum_{q=1}^{Q} c^q \vec{k}[q].$$

(In our PRG application, B_q will be the set of terms with approximately c^q input variables, so $\|\vec{k}\|_{(c)}$ will be approximately the number of input variables participating in each monomial of $S_{\vec{k}}$, as outlined in Section 1.5.1.) It is easy to verify that the above weight function is indeed a norm; however, we will not be using this observation.

The main result of this section is a tail-bound for subset-wise symmetric polynomials. In Lemma 3, the parameter A is analogous to the "cutoff point" k_0 discussed in Section 1.5.1.

▶ **Lemma 3.** *Suppose $c > 1$ and $Q, A \in \mathbb{N}$ satisfy $A > \max\left\{\left(\frac{10^6 c}{c-1}\right) \cdot c^Q, 2^{60} Q^2\right\}$. Let Y be a random variable taking values in $\mathbb{R}^m$. Moreover, suppose for every $\vec{k} \in \mathbb{N}^Q$ with $\|\vec{k}\|_{(c)} \le A$,*

$$\mathop{\mathbb{E}}_{Y}\left[R_{\vec{k}}(Y)\right] \le 2^{-\frac{1}{8}\|\vec{k}\|_{(c)}}.$$

Then, except with probability $2^{-A/2^{23}}$ over $y \sim Y$,

$$\sum_{\substack{\vec{k} \in \mathbb{N}^Q, \\ \|\vec{k}\|_{(c)} > A}} |S_{\vec{k}}(y)| \le 2^{-\frac{A}{1024}}.$$

[3] The careful reader will notice a slight discrepancy between the exact constants of Theorem 2 on the one hand and the statements by Gopalan and Yehudayoff [17] and Meka et al. [24] on the other. This discrepancy reflects a minor mistake in the original paper by Gopalan and Yehudayoff [17] that we have here corrected.

Lemma 3 is similar in spirit to Theorem 2: it says that if the "early" subset-wise symmetric polynomials are small (with high probability), then the "late" subset-wise symmetric polynomials are all small (with high probability).

2.3 Non-probabilistic Tail Bound

Before moving to the proof of Lemma 3 in the next subsection, here we first give a tail-bound in the case when the input y satisfies some useful properties. We will later prove Lemma 3, by showing that a random Y satisfies these properties with high probability. Given a vector $\vec{k} \in \mathbb{N}^Q$, we define the restriction of $\vec{k}$ to a set $\mathcal{Q} \subseteq [Q]$ by

$$\vec{k}|_{\mathcal{Q}}[q] = \begin{cases} \vec{k}[q] & \text{if } q \in \mathcal{Q}, \\ 0 & \text{if } q \notin \mathcal{Q}. \end{cases}$$

Our non-probabilistic tail bound goes as follows.

▶ **Lemma 4.** *Suppose $c > 1$ and $Q, A \in \mathbb{N}$ satisfy $A > \max\left\{\left(\frac{10^6 c}{c-1}\right) \cdot c^Q, 2^{60}Q^2\right\}$. Let $y \in \mathbb{R}^m$ be a fixed vector. Suppose that for every $\vec{k} \in \mathbb{N}^Q$, with $A/10^5 \leq \|\vec{k}\|_{(c)} \leq A$, and for every pair of disjoint sets $\mathcal{Q}_1, \mathcal{Q}_2 \subseteq [Q]$ satisfying $\{q : \vec{k}[q] > 1\} \subseteq \mathcal{Q}_1 \cup \mathcal{Q}_2$, we have*

$$R_{(\vec{k}|_{\mathcal{Q}_1})}(y) \cdot R_{(\vec{k}|_{\mathcal{Q}_2})}(y)^4 \leq 2^{-\frac{1}{32} \cdot \|\vec{k}\|_{(c)}}.$$

Then,

$$\sum_{\substack{\vec{k} \in \mathbb{N}^Q, \\ \|\vec{k}\|_{(c)} > A}} \left|S_{\vec{k}}(y)\right| \leq 2^{-\frac{A}{1024}}.$$

Proof. For a fixed $\ell \in \mathbb{N}$ and $q \in [Q]$, define

$$S_{\ell,q} = \sum_{I \subseteq B_q, |I| = \ell} \prod_{i \in I} y_i,$$

which is the ℓ-th elementary symmetric polynomial applied to $(y_i)_{i \in B_q}$. Similarly, define

$$R_{\ell,q} = S_{\ell,q}^2(y) \cdot \ell!.$$

Fix $\vec{k}$ with $\|\vec{k}\|_{(c)} > A$, let $\lambda = 10^5 \cdot \|\vec{k}\|_{(c)}/A$ and let $\vec{k}' \in \mathbb{N}^Q$ be such that $\vec{k}'[q] = \lceil \vec{k}[q]/\lambda \rceil$. Thus, $A/10^5 \leq \|\vec{k}'\|_{(c)} \leq A/2$. Let $\mathcal{Q} := \{q \in Q : \vec{k}[q] \geq 1\}$, and for each $q \in \mathcal{Q}$, let $\theta_q > 0$ be the smallest[4] number satisfying

$$R_{\vec{k}'[q],q} \leq \theta_q^{\vec{k}'[q]} \quad \text{and} \quad R_{\vec{k}'[q]+1,q} \leq \theta_q^{\vec{k}'[q]+1}. \tag{3}$$

By Theorem 2,

$$\left|S_{\vec{k}[q],q}\right| \leq \left(\frac{64e^2\theta_q \vec{k}'[q]}{\vec{k}[q]}\right)^{\vec{k}[q]/2}.$$

[4] It is possible that $\theta_q = 0$ satisfies Equation (3). In this degenerate case, we must have $S_{\vec{k}[q],q} = 0$. This implies $S_{\vec{k}}(y) = 0$, hence Equation (4) trivially holds.

Subset-wise symmetric polynomials by design can be expressed as a product of elementary symmetric polynomials, hence

$$|S_{\vec{k}}(y)| = \prod_{q=1}^{Q} \left|S_{\vec{k}[q],q}(y)\right| \le \prod_{q\in\mathcal{Q}} \left(\frac{64e^2\theta_q \vec{k'}[q]}{\vec{k}[q]}\right)^{\vec{k}[q]/2}$$
$$= \left(\prod_{q\in\mathcal{Q}} \theta_q^{\vec{k}[q]/\lambda}\right)^{\lambda/2} \cdot \prod_{q\in\mathcal{Q}} \left(8e\sqrt{\vec{k'}[q]/\vec{k}[q]}\right)^{\vec{k}[q]}.$$

By our choice of θ_q,

$$\theta_q^{\vec{k'}[q]} = \max\left\{R_{\vec{k'}[q],q}(y), R_{\vec{k'}[q]+1,q}(y)^{\vec{k'}[q]/(\vec{k'}[q]+1)}\right\}$$
$$\le \max\left\{R_{\vec{k'}[q],q}(y), R_{\vec{k'}[q]+1,q}(y), \sqrt{R_{\vec{k'}[q]+1,q}(y)}\right\}.$$

Observe that $\vec{k}[q]/\lambda \in \left[\vec{k'}[q]-1, \vec{k'}[q]\right]$, and thus $\theta_q^{\vec{k}[q]/\lambda}$ is between $\theta_q^{\vec{k'}[q]-1}$ and $\theta_q^{\vec{k'}[q]}$. If $\vec{k'}[q]=1$, then $\theta_q^{k'_q-1}=1$, and otherwise $\theta_q^{\vec{k'}[q]-1}$ is between $\theta_q^{\vec{k'}[q]}$ and $\sqrt{\theta_q^{\vec{k'}[q]}}$. Therefore,

$$\theta_q^{\vec{k}[q]/\lambda} \le \max\left\{\theta_q^{\vec{k'}[q]}, \sqrt{\theta_q^{\vec{k'}[q]}}, \mathbb{1}_{\vec{k'}[q]=1}\right\}$$
$$\le \max\left\{R_{\vec{k'}[q],q}(y), R_{\vec{k'}[q],q}(y)^{1/4}, R_{\vec{k'}[q]+1,q}(y), R_{\vec{k'}[q]+1,q}(y)^{1/4}, \mathbb{1}_{\vec{k'}[q]=1}\right\}.$$

For every q, choose $\vec{k''}[q] \in \{\vec{k'}[q], \vec{k'}[q]+1\}$ such that

$$\theta_q^{\vec{k}[q]/\lambda} \le \max\left\{R_{\vec{k''}[q],q}(y), R_{\vec{k''}[q],q}(y)^{1/4}, \mathbb{1}_{\vec{k''}[q]=1}\right\}.$$

Note that $\|\vec{k''}\|_{(c)} \ge \|\vec{k'}\|_{(c)}$ and

$$\|\vec{k''}\|_{(c)} \le \|\vec{k'}\|_{(c)} + \sum_{q=1}^{Q} c^q \le \|\vec{k'}\|_{(c)} + \frac{A}{10^6} < A.$$

Therefore, there exist disjoint sets $\mathcal{Q}_1, \mathcal{Q}_2 \subseteq [Q]$ such that $\{q : \vec{k''}[q] > 1\} \subseteq \mathcal{Q}_1 \cup \mathcal{Q}_2$, and that for every $q \in \mathcal{Q}$,

$$\theta_q^{\vec{k}[q]/\lambda} \le \begin{cases} R_{\vec{k''}[q],q}(y) & \text{if } q \in \mathcal{Q}_1, \\ R_{\vec{k''}[q],q}(y)^{1/4} & \text{if } q \in \mathcal{Q}_2, \\ 1 & \text{otherwise.} \end{cases}$$

Multiplying over $q \in \mathcal{Q}$, we get

$$\prod_{q\in\mathcal{Q}} \theta_q^{\vec{k}[q]/\lambda} \le \prod_{q\in\mathcal{Q}_1} R_{\vec{k''}[q],q}(y) \cdot \prod_{q\in\mathcal{Q}_2} \mathbb{R}_{\vec{k''}[q],q}(y)^{1/4}$$
$$= \left(R_{(\vec{k''}|_{\mathcal{Q}_1})}(y)^4 \cdot R_{(\vec{k''}|_{\mathcal{Q}_2})}(y)\right)^{1/4} \le 2^{-\frac{1}{128}\cdot\|\vec{k''}\|_{(c)}} \le 2^{-\frac{1}{128}\cdot\|\vec{k'}\|_{(c)}}.$$

As a result,

$$\begin{aligned}|S_{\vec{k}}(y)| &\le 2^{-\frac{\|\vec{k'}\|_{(c)}}{128}\cdot\frac{\lambda}{2}}\cdot\prod_{q\in\mathcal{Q}}\left(8e\sqrt{\vec{k'}[q]/\vec{k}[q]}\right)^{\vec{k}[q]}\\ &\le 2^{-\frac{\|\vec{k'}\|_{(c)}}{256}\cdot\lambda}\cdot\prod_{q\in\mathcal{Q}}\left(8e\sqrt{2/10^5}\right)^{\vec{k}[q]}\cdot\left(\sqrt{10^5/2}\right)^{\lambda}\\ &\le 2^{-\frac{\|\vec{k'}\|_{(c)}}{256}\cdot\lambda}\cdot 2^{8Q\cdot\lambda}\cdot 4^{-\|\vec{k}\|_1}\le 2^{-\frac{\|\vec{k'}\|_{(c)}}{512}\cdot\lambda}\cdot 4^{-\|\vec{k}\|_1}\le 2^{-\frac{\|\vec{k}\|_{(c)}}{512}}\cdot 4^{-\|\vec{k}\|_1}. \qquad (4)\end{aligned}$$

To see the second inequality, observe that when $\vec{k}[q]>\lambda$, then $\left(8e\sqrt{\vec{k'}[q]/\vec{k}[q]}\right)^{\vec{k}[q]}\le(8e\sqrt{2/10^5})^{\vec{k}[q]}$, and otherwise $\left(8e\sqrt{\vec{k'}[q]/\vec{k}[q]}\right)^{\vec{k}[q]}\le(8e)^{\lambda}$. Summing up over all choices of $\vec{k}$ we get,

$$\begin{aligned}\sum_{\vec{k}\in\mathbb{N}^Q,\|\vec{k}\|_{(c)}>A}|S_{\vec{k}}(y)| &\le \sum_{L=1}^{m}\sum_{\substack{\vec{k}\in\mathbb{N}^Q,\\ \|\vec{k}\|_{(c)}>A,\|\vec{k}\|_1=L}}2^{-\frac{\|\vec{k}\|_{(c)}}{512}}\cdot 4^{-L}\\ &\le 2^{-\frac{A}{512}}\cdot\sum_{L=1}^{m}4^{-L}\cdot\left|\left\{\vec{k}\in\mathbb{N}^q:\|\vec{k}\|_1=L\right\}\right|\\ &= 2^{-\frac{A}{512}}\cdot\sum_{L=1}^{m}4^{-L}\cdot\binom{Q-1+L}{Q-1}\\ &\le 2^{-\frac{A}{512}}\cdot\sum_{L=1}^{m}4^{-L}\cdot 2^{Q-1+L}\le 2^{-\frac{A}{512}}\cdot 2^{Q-1}\cdot\sum_{L=1}^{m}2^{-L}\le 2^{-\frac{A}{1024}}.\end{aligned}$$

◀

2.4 Probabilistic Tail Bound: Proof of Lemma 3

Proof. Let $\vec{k}$, $\mathcal{Q}_1$, and $\mathcal{Q}_2$ be as in the statement of Lemma 4. Using the Cauchy-Schwarz inequality and the concavity of $(\cdot)^{1/4}$, we get

$$\begin{aligned}\mathbb{E}\left[\left(R_{\vec{k}|_{\mathcal{Q}_1}}(Y)\right)^{1/8}\cdot\left(R_{\vec{k}|_{\mathcal{Q}_2}}(Y)\right)^{1/2}\right] &\le \left(\mathbb{E}\left[\left(R_{\vec{k}|_{\mathcal{Q}_1}}(Y)\right)^{1/4}\right]\cdot\mathbb{E}\left[R_{\vec{k}|_{\mathcal{Q}_2}}(Y)\right]\right)^{1/2}\\ &\le \left(\mathbb{E}\left[R_{\vec{k}|_{\mathcal{Q}_1}}(Y)\right]^{1/4}\cdot\mathbb{E}\left[R_{\vec{k}|_{\mathcal{Q}_2}}(Y)\right]\right)^{1/2}\\ &\le \left(2^{-\frac{1}{32}\cdot\|\vec{k}|_{\mathcal{Q}_1}\|_{(c)}}\cdot 2^{-\frac{1}{8}\cdot\|\vec{k}|_{\mathcal{Q}_2}\|_{(c)}}\right)^{1/2}\\ &\le 2^{-\frac{1}{64}\cdot\|\vec{k}|_{\mathcal{Q}_1\cup\mathcal{Q}_2}\|_{(c)}}\\ &\le 2^{-\frac{1}{64}\cdot\left(\|\vec{k}\|_{(c)}-\left(\frac{c}{c-1}\right)\cdot c^Q\right)}\\ &\le 2^{-\frac{1}{64}\cdot\left(\|\vec{k}\|_{(c)}-\frac{A}{20000}\right)}\\ &\le 2^{-\frac{1}{128}\cdot\|\vec{k}\|_{(c)}}.\end{aligned}$$

Therefore, by Markov's inequality, except with probability at most $2^{-\|\vec{k}\|_{(c)}/256}\le 2^{-A/2560000}$, we have

$$\left(R_{\vec{k}|_{\mathcal{Q}_1}}(Y)\right)\cdot\left(R_{\vec{k}|_{\mathcal{Q}_2}}(Y)\right)^4\le 2^{-\frac{\|\vec{k}\|_{(c)}}{32}}.$$

The above analysis was done for a fixed choice of $\vec{k}$, $\mathcal{Q}_1$, and $\mathcal{Q}_2$. The number of choices for such $\vec{k}$ is A^Q (which is subexponential in A), and the number of such $\mathcal{Q}_1$, $\mathcal{Q}_2$ is at most 3^Q (which is a polynomial in A), thus Lemma 3 follows by a union bound. More precisely, one can check that since $A \geq 2^{60}Q^2$, then $(3A)^Q \cdot 2^{-A/2560000} \leq 2^{-A/2^{23}}$. ◀

3 Pseudorandomness Preliminaries

Having completed our analysis of subset-wise symmetric polynomials, we now move on to setting the groundwork for our PRG construction and analysis.

3.1 Probability Basics

Let U_n denote the uniform distribution over $\{0,1\}^n$. We will simply write U if n is clear from context. For $f\colon \{0,1\}^n \to \mathbb{R}$, as a shorthand, we write $\mathbb{E}[f]$ to denote $\mathbb{E}[f(U)]$ and $\mathrm{Var}[f]$ to denote $\mathrm{Var}[f(U)]$. If X is a distribution over $\{0,1\}^n$, we say that X *ε-fools* f, or X *fools* f *with error* ε, if

$$|\mathbb{E}[f(X)] - \mathbb{E}[f]| \leq \varepsilon.$$

We say that X ε-fools a family $\mathcal{F}$ of functions, if it ε-fools every $f \in \mathcal{F}$.

3.2 Small Bias

A *parity function* is a function of the form $f(x) = \bigoplus_{i \in I} x_i$ for some set $I \subseteq [n]$. We say that a random variable $Y \in \{0,1\}^n$ is *δ-biased* if it δ-fools all parity functions. We say that Y is *n'-wise δ-biased* if it δ-fools all parity functions on at most n' bits, i.e., all parity functions with $|I| \leq n'$. There are explicit constructions of n'-wise δ-biased distributions that can be sampled with $O(\log(n'/\delta) + \log\log n)$ truly random bits [25, 3].

Recall that for a function $f\colon \{0,1\}^n \to \mathbb{R}$ with Fourier expansion $f = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot \chi_S$, the L_1 norm of f is defined by

$$L_1(f) = \sum_{S \subseteq [n]} |\widehat{f}(S)|.$$

This norm is subadditive ($L_1(f+g) \leq L_1(f) + L_1(g)$) and submultiplicative ($L_1(f \cdot g) \leq L_1(f) \cdot L_1(g)$). Functions with bounded L_1 norm are fooled by small-bias distributions:

▷ Claim 5. If $f\colon \{0,1\}^n \to \mathbb{R}$ and Y is δ-biased, then Y fools f with error $2\delta \cdot L_1(f)$.

We will also rely on the following "XOR lemma" for small-bias distributions.

▶ **Lemma 6** ([16, 24]). *Let $0 < \delta < \varepsilon \leq 1$. Let $f_1, \ldots, f_k\colon \{0,1\}^n \to [-1,1]$ depend on disjoint variable sets, and define*

$$f(x) = \prod_{i=1}^{k} f_i(x).$$

If every δ-biased distribution ε-fools every f_i, then every δ^k-biased distribution fools f with error $16^k \cdot 2\varepsilon$.

3.3 Limited Independence

For $p \in [0,1]$, let Bernoulli$(p)^{\otimes n}$ denote the distribution over $\{0,1\}^n$ where the bits are i.i.d. and each bit has expectation p. For example, $U_n = \text{Bernoulli}(1/2)^{\otimes n}$. For a set $I = \{i_1 < i_2 < \cdots < i_\ell\} \subseteq [n]$ and a string $z \in \{0,1\}^n$, we let $z|_I = z_{i_1} z_{i_2} \dots z_{i_\ell} \in \{0,1\}^\ell$. We say that $Z \in \{0,1\}^n$ is *γ-almost k-wise independent with marginals p* if for every set $I \subseteq [n]$ with $|I| \le k$, the total variation distance between $Z|_I$ and Bernoulli$(p)^{\otimes |I|}$ is at most γ.

▷ Claim 7. For every $n, k, C \in \mathbb{N}$ and $\gamma > 0$, there is an explicit γ-almost k-wise independent distribution with marginals $p = 1 - 2^{-C}$ that can be sampled with $O(Ck + \log(1/\gamma) + \log\log n)$ truly random bits.

Proof. Sample $Y \in \{0,1\}^{Cn}$ from a (Ck)-wise $(2^{-Ck/2-1}\gamma)$-biased distribution. Note that as discussed above Y can be sampled using

$$O\left(\log(2^{Ck}/\gamma) + \log\log n\right) = O\left(Ck + \log(1/\gamma) + \log\log n\right)$$

truly random bits. Divide Y into n blocks $Y^{(1)}, \dots, Y^{(n)} \in \{0,1\}^C$, and set

$$Z_i = 0 \iff Y^{(i)} = 1^C.$$

The desired distribution is $Z \in \{0,1\}^n$.

To prove correctness, let $f\colon \{0,1\}^n \to \{0,1\}$ be any test function depending on only k variables. There is a function $g\colon \{0,1\}^{Cn} \to \{0,1\}$ depending on only Ck variables such that $f(Z) = g(Y)$. By Claim 5,

$$\begin{aligned} |\,\mathbb{E}[f(Z)] - \mathbb{E}[f(\text{Bernoulli}(p)^{\otimes n})]| &= |\,\mathbb{E}[g(Y)] - \mathbb{E}[g]| \\ &\le 2^{-Ck/2-1} \cdot 2\gamma \cdot L_1(g) \le \gamma. \end{aligned}$$ ◁

The expectation parameter p can be "amplified" by drawing independent samples and combining with a coordinate-wise conjunction:

▷ Claim 8. Let Z be γ-almost k-wise independent with marginals p. Draw t independent samples $z^{(1)}, \dots, z^{(t)} \sim Z$, and let $Z' = z^{(1)} \wedge \cdots \wedge z^{(t)}$. Then Z' is $(t\gamma)$-almost k-wise independent with marginals p^t.

Proof sketch. The proof is a simple hybrid argument. Draw t independent samples $r^{(1)}, \dots, r^{(t)} \sim \text{Bernoulli}(p)^{\otimes n}$, and let

$$Z^{(i)} = z^{(1)} \wedge \cdots \wedge z^{(i)} \wedge r^{(i+1)} \wedge \cdots \wedge r^{(t)}.$$

One can show by induction on i that $Z^{(i)}$ is $(i\gamma)$-almost k-wise independent with marginals p^t. ◁

3.4 PARITY ∘ AND Formulas

Recall that our main result (Theorem 1) is a PRG for read-once depth-2 $\mathbf{AC}^0[\oplus]$. For most of the paper, we will focus on the special case that the root gate is $\oplus$ and its immediate children are $\wedge$ gates. That is, define a PARITY ∘ AND *formula* to be a function of the form

$$f(x) = \bigoplus_{i=1}^{m} f_i(x),$$

where each f_i is a conjunction of literals, i.e., variables or their negations. We refer to $f_1, \ldots, f_m$ as the *terms* of f. We say that the formula is *read-once* if each variable x_i appears in at most one term. Most of our effort will be spent fooling read-once PARITY $\circ$ AND formulas. Note that this is a slight generalization of read-once $\mathbb{F}_2$-polynomials due to the availability of $\neg$ gates. We will explain in Section 5.6 why it is sufficient to focus on this special case.

The *width* of a term is the number of variables in the term; the width of f is the maximum width of its terms. The *length* of f is m, the number of its terms.

For convenience, if f is a function taking values in $\{0,1\}$, we let $\overline{f} = (-1)^f$. That way, if f is a PARITY $\circ$ AND formula,

$$\overline{f} = \prod_{i=1}^{m} \overline{f_i}.$$

3.5 Restrictions

A *restriction* is a string $x \in \{0,1,\star\}^n$; intuitively, $x_i = \star$ means that x_i has still not been assigned a value. We define an associative *composition* operation on restrictions by the formula

$$(x \circ x')_i = \begin{cases} x_i & \text{if } x_i \neq \star, \\ x'_i & \text{otherwise.} \end{cases}$$

For a function f on $\{0,1\}^n$, the *restricted* function $f|_x$ on $\{0,1\}^n$ is defined by

$$f|_x(x') = f(x \circ x').$$

A restriction x can be specified by two strings $y, z \in \{0,1\}^n$ using the following notation[5]. Define $\mathrm{Res}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1,\star\}^n$ by

$$(\mathrm{Res}(y,z))_i = \begin{cases} \star & \text{if } z_i = 1, \\ y_i & \text{if } z_i = 0. \end{cases}$$

In words, z indicates the $\star$ positions, and y provides the bits in the non-$\star$ positions.

3.6 Pseudorandom Restrictions

Let Y, Z be distributions over $\{0,1\}^n$, and let $X = \mathrm{Res}(Y,Z)$. For a function $f\colon \{0,1\}^n \to \mathbb{R}$, we say that the distribution X *preserves the expectation of* f *with error* ε if

$$|\mathbb{E}[f|_X(U)] - \mathbb{E}[f]| \leq \varepsilon.$$

An equivalent condition is that $|\mathbb{E}[f(Y + Z \wedge U)] - \mathbb{E}[f]| \leq \varepsilon$, where $+$ denotes addition over $\mathbb{F}_2^n$ and $\wedge$ denotes coordinate-wise conjunction. This second condition is the "pseudorandomness plus noise" perspective [18] (the string $Z \wedge U$ can be thought of as a noise vector.)

[5] With apologies, we here flip the order of the arguments to Res compared to the notation used in the authors' prior work [13].

If f takes on values in $\{0,1\}$, for each particular value z that Z might take on, we define the *bias function* [16] $\widetilde{f_z}\colon \{0,1\}^n \to [-1,1]$ by

$$\widetilde{f_z}(x) = \mathbb{E}\left[\overline{f}(x + z \wedge U)\right].$$

(We use $\overline{f}$ rather than f simply for convenience.) The statement that X preserves the expectation of f with error ε is also equivalent to the condition

$$\left|\mathop{\mathbb{E}}_{Z}\left[\mathop{\mathbb{E}}_{Y}[\widetilde{f_Z}(Y)] - \mathbb{E}\left[\overline{f}\right]\right]\right| \leq 2\varepsilon.$$

When z is clear from context, we will just write $\widetilde{f}$ instead of $\widetilde{f_z}$.

If X is a distribution over $\{0,1,\star\}^n$ and $t \in \mathbb{N}$, let $X^{\circ t}$ denote the distribution over $x \in \{0,1,\star\}^n$ obtained by drawing independent samples $x^{(1)}, \ldots, x^{(t)} \sim X$ and composing them, $x = x^{(1)} \circ \cdots \circ x^{(t)}$.

Suppose $\mathcal{F}$ is a class of Boolean functions that is closed under restriction. If X preserves the expectation of every $f \in \mathcal{F}$ with error ε, then $X^{\circ t}$ preserves the expectation of every $f \in \mathcal{F}$ with error $t\varepsilon$. Furthermore, informally, if X "has $\star$-probability p", then $X^{\circ t}$ "has $\star$-probability p^t". To be precise, we can consider the case $X = \mathrm{Res}(Y, Z)$ where Z is γ-almost k-wise independent with marginals p. Then the distribution of $\star$ positions in $X^{\circ t}$ is described by Claim 8.

4 Applying a Single Restriction

In this section, we prove that the expectation of a PARITY $\circ$ AND formula is preserved under a suitable pseudorandom restriction. The cost of the restriction is only $O(\log n)$ truly random bits, the error is $\exp(-\widetilde{\Omega}(\log n))$ (near-optimal), and the restriction assigns values to a constant fraction of the inputs.

4.1 Restriction Construction

Set $C = 500$, $\overline{C} = 2000C$, $c = 1.1$, and $\beta = 0.95$, and consider the following two distributions.

- Let Y be a δ^3-biased distribution over $\{0,1\}^n$ for $\delta = \min\left\{n^{-12\overline{C}}, \frac{1}{2}n^{-\frac{5c}{c-1}-1}\right\} = n^{-12{,}000{,}000}$.[6]
- Let Z be a γ-almost k-wise independent distribution over $\{0,1\}^n$ with marginals $p = 1 - 2^{-C}$, for $k = 6\log n$ and $\gamma = n^{-9}$.

Our restriction is $\mathrm{Res}(Y, Z)$, i.e., Z indicates where to put $\star$ and Y fills in the non-$\star$ bits.

▶ **Lemma 9.** *Let f be a read-once* PARITY $\circ$ AND *formula over n variables of width at most $C\log n$. Then, $\mathrm{Res}(Y, Z)$ preserves the expectation of f to within error $2^{-C\frac{\log n}{\log\log n}}$, i.e.,*

$$|\mathbb{E}[f(Y + Z \wedge U)] - \mathbb{E}[f]| \leq 2^{-C\frac{\log n}{\log\log n}}.$$

4.2 Buckets

Toward proving Lemma 9, we first set some preliminary notations. Recall that f is of the form

$$f = \bigoplus_{i=1}^{m} f_i = \bigoplus_{i=1}^{m} \bigwedge_{j=1}^{w_i} \ell_{ij},$$

where every literal ℓ_{ij} is either some variable in $\{x_1, \ldots, x_n\}$ or its negation.

[6] No attempt was made to optimize the constants.

Set $Q = \lceil \log_c(C \log n) \rceil = O(\log \log n)$. We partition the terms of f into Q buckets according to their *width*. Namely, for each $q \in [Q]$ we define the interval $I_q = [c^{q-1}, c^q)$ and define $B_q \subseteq [m]$ to be the set of indices i such that $w_i \in I_q$. Also, for $q \in [Q]$ we define

$$F_q = \bigoplus_{i \in B_q} f_i,$$

so $f = \bigoplus_{q=1}^{Q} F_q$. For every $q \in [Q]$ we further denote $m_q = |B_q|$.

We divide into two cases (Section 4.3 and Section 4.4) depending on whether there exists a bucket with substantially many terms. Lemma 9 will follow immediately from Lemma 10 and Lemma 15, which cover these two cases respectively.

4.3 Case I – There Exists a Heavy Bucket

Say that bucket $q \in [Q]$ is *heavy* if both $m_q > 3^{c^q}$ and $m_q > \log^C n$. The first case is that there exists a heavy bucket (i.e., there are many terms of roughly the same width, even relative to q). In this case, we will argue that f itself is balanced and also that it stays balanced, w.h.p., after a pseudorandom restriction.

▶ **Lemma 10.** *Let f be a read-once* PARITY $\circ$ AND *formula over n variables of width at most $C \log n$. Suppose there exists a heavy bucket as defined above. Then, with probability at least $1 - \frac{1}{n}$ over $(y, z) \sim Y \times Z$,*

$$\left| \mathbb{E}[\overline{f}|_{\mathrm{Res}(y,z)}] - \mathbb{E}[\overline{f}] \right| \leq \frac{1}{n}.$$

Toward proving Lemma 10, let us define a few more auxiliary notations. Write

$$f = f_{\mathsf{rest}} \oplus F_q,$$

where q is a heavy bucket.

▷ Claim 11. It holds that $\left| \mathbb{E}[\overline{f}] \right| \leq \frac{1}{4n}$.

Proof. By the read-once property and the fact that $\overline{f_{\mathsf{rest}}}$ is bounded,

$$\left| \mathbb{E}[\overline{f}] \right| = \left| \mathbb{E}[\overline{f_{\mathsf{rest}}}] \, \mathbb{E}[\overline{F_q}] \right| \leq \left| \mathbb{E}[\overline{F_q}] \right| = \prod_{i \in B_q} \left| \mathbb{E}[\overline{f_i}] \right|.$$

Each term in F_q has width at least c^{q-1}, so

$$\left| \mathbb{E}[\overline{f}] \right| \leq \left(1 - 2 \cdot 2^{-c^{q-1}}\right)^{m_q} \leq e^{-2 \cdot 2^{-c^{q-1}} \cdot m_q}.$$

Recalling that $m_q \geq 3^{c^q}$, we have $2^{-c^{q-1}} \geq m_q^{\gamma}$ for $\gamma = \log_3 2^{c^{-1}} < \frac{3}{4}$. Thus, using that fact that $m_q \geq \log^C n$,

$$\left| \mathbb{E}[\overline{f}] \right| \leq e^{-2 m_q^{1-\gamma}} \leq e^{-2 \log^{(1-\gamma)C} n} \leq 2^{-\log^{100} n}. \qquad \triangleleft$$

Next, we must analyze the bias of f after the pseudorandom restriction. Let n_q be the number of variables read by F_q. Let $b = \lceil \log_3 n_q \rceil$. We will group the terms of F_q into *blocks*, each of which reads roughly b variables. To define this grouping, first observe that

$b \geq \log_3 m_q$, as each term reads at least one variable. Recalling that $c^q < \log_3 m_q$, we know that $b > c^q$. Therefore, since each term in F_q has width at most c^q, we can write

$$F_q = \bigoplus_{i=1}^{B} g_i,$$

where each block g_i reads b_i variables for $b_i \in \left[b - \frac{1}{2}c^q, b + \frac{1}{2}c^q\right]$.

Let us now estimate B, the number of blocks. Since $b > c^q$, $b_i \in \left[\frac{b}{2}, \frac{3b}{2}\right]$. Also, $m_q > \log^C n$ so $b > \frac{C}{2}\log\log n$. Thus, on the one hand,

$$B \geq \frac{2n_q}{3b} \geq \frac{2 \cdot 3^b}{9b},$$

and on the other hand, $B \leq n_q \leq 3^b$.

Toward arguing that f is balanced after pseudorandom restrictions, we wish to show that with high probability, $z \sim Z$ keeps many variables in many terms alive.

▶ **Definition 12.** *For $z \in \{0,1\}^n$ and a formula f, we say f is* good under z *if z assigns* 1 *to at least a $(1-\beta)$-fraction of the variables f reads.*

▷ Claim 13. For a fixed $z \in \{0,1\}^n$, let $\mathbf{X}_z \subseteq [B]$ be the set of blocks g_i that are not good under z. Then, with probability at least $1 - \frac{1}{2n}$ over $z \sim Z$,

$$|\mathbf{X}_z| \leq \left\lceil \frac{4\log n}{b} \right\rceil.$$

Proof. Set $k_0 = \left\lceil \frac{4\log n}{b} \right\rceil$. Let $S \subseteq [B]$ be some subset of cardinality k_0. We first bound the probability p that every block g_i for $i \in S$ is bad under $z \sim Z$. For a *truly random* $z \sim \text{Bernoulli}(1-2^{-C})^{\otimes n}$, the above probability is bounded by

$$\prod_{i \in S} \binom{b_i}{\beta b_i} 2^{-C\beta b_i} \leq \prod_{i \in S} 2^{b_i} 2^{-5b_i} \leq \left(2^{-4 \cdot \frac{b}{2}}\right)^{|S|} \leq n^{-8}.$$

Now, for every $i \in [B]$, $k \geq k_0 b_i$ so for $z \sim Z$, we get that $p \leq n^{-8} + \gamma \leq 2n^{-8}$. Thus, by the union bound, with probability at most

$$\binom{B}{k_0} p \leq 2B^{k_0} n^{-8} \leq 2\left(3^b\right)^{\frac{4\log n}{b}} n^{-8} \leq 2n^{-(2-\log 3)4} \leq n^{-\frac{4}{3}} < \frac{1}{2n}$$

there will be some S whose all blocks are bad. Taking the contrapositive, we infer that with probability at least $1 - \frac{1}{2n}$ over $z \sim Z$, at most k_0 of the g_i-s are bad under z. ◁

▶ **Lemma 14.** *With probability at least $1 - \frac{1}{n}$ over $(y,z) \sim Y \times Z$, it holds that*

$$\left|\mathbb{E}\left[\overline{f|_{\text{Res}(y,z)}}\right]\right| \leq \frac{1}{2n}.$$

Proof. Fix a *good* z, for which at most $\frac{4\log n}{b}$ of the g_i-s are not good under it. By Claim 13, z is good with probability at least $1 - \frac{1}{2n}$. Let $B^{\mathsf{alive}} = [B] \setminus \mathbf{X}_z$, so

$$f = f_{\mathsf{rest}} \oplus \left(\bigoplus_{i \in B^{\mathsf{alive}}} g_i\right) \oplus \left(\bigoplus_{i \in [B] \setminus B^{\mathsf{alive}}} g_i\right).$$

For every $i \in B^{\mathsf{alive}}$, set the following notations.

- For every $y \in \{0,1\}^n$, let g_i^y denote the function $g_i|_{\mathrm{Res}(y,z)}$.
- Let $I_i^{\mathsf{dead}} \subseteq [n]$ be the literals read by g_i for which $z = 0$. As $i \in B^{\mathsf{alive}}$, $\left|I_i^{\mathsf{dead}}\right| \leq \beta b_i \leq \frac{3\beta}{2} b$. Note that each literal $j \in I_i^{\mathsf{dead}}$ is set by $y \sim Y$.
- Let $I_i^{\mathsf{alive}} \subseteq [n]$ be the literals read by g_i for which $z = 1$. As $i \in B^{\mathsf{alive}}$, $\left|I_i^{\mathsf{alive}}\right| \geq (1-\beta) b_i \geq \frac{1-\beta}{2} b$.

Define the function h_i so that $h_i(y) = 1$ if g_i^y is a nonconstant function, and 0 otherwise. Namely,

$$h_i(y) = \bigwedge_{j \in I_i^{\mathsf{dead}}} y'_j,$$

where y'_j is either y_j or $\neg y_j$ depending on whether y_j appears positively or negatively in g_i. Also, define

$$S(y) = \sum_{i \in B^{\mathsf{alive}}} h_i(y),$$

where the sum is over the reals. Denote

$$\mu = \mathbb{E}[S(U)] = \sum_{i \in B^{\mathsf{alive}}} 2^{-\left|I_i^{\mathsf{dead}}\right|},$$

and note that $\mu \geq |B^{\mathsf{alive}}| \cdot 2^{-\frac{3\beta}{2} b}$. Set $\Delta S = S - \mu$. The spectral norm of the AND function is 1, and so by the sub-additivity we get that $L_1(\Delta S) \leq 2|B^{\mathsf{alive}}|$. Set $\ell = 2\left\lceil \frac{C \log n}{2 \log(2|B^{\mathsf{alive}}|)} \right\rceil$. By the sub-multiplicativity of the spectral norm we have that

$$L_1\left(\Delta S^\ell\right) \leq \left(2\left|B^{\mathsf{alive}}\right|\right)^\ell \leq n^C.$$

For $\varepsilon = \frac{1}{2}$, note that $\delta \leq \frac{\varepsilon}{2} \cdot L_1\left(\Delta S^\ell\right)^{-1}$. By Claim 5, Y ε-fools the function ΔS^ℓ, so

$$\left|\mathbb{E}\left[(S(Y) - \mu)^\ell\right] - \mathbb{E}\left[(S(U) - \mu)^\ell\right]\right| \leq \varepsilon. \tag{5}$$

Next, observe that $\Delta S(U)$ is the sum of zero-mean *independent* random variables, as the h_i-s are supported over disjoint set of variables. Set $A = |B^{\mathsf{alive}}| \cdot 2^{-4\beta b}$. By the Chernoff bound,

$$\begin{aligned}\mathbb{E}\left[\Delta S(U)^\ell\right] &\leq \mathbb{E}\left[\Delta S(U)^\ell \mid \Delta S(U)^\ell \leq A^\ell\right] \\ &\qquad + \mathbb{E}\left[\Delta S(U)^\ell \mid \Delta S(U)^\ell \geq A^\ell\right] \cdot \Pr\left[\Delta S(U)^\ell \geq A^\ell\right] \\ &\leq A^\ell + \left|B^{\mathsf{alive}}\right|^\ell \cdot \Pr\left[\Delta S(U) \geq A\right] \leq \left|B^{\mathsf{alive}}\right|^\ell \cdot \left(2^{-4\beta b\ell} + e^{-\frac{2A^2}{|B^{\mathsf{alive}}|}}\right).\end{aligned}$$

Recall that $b > \frac{C}{2} \log\log n$, so $3^b \geq 36 \log n$ for a large enough n, and since $B \geq \frac{2}{9b} 3^b$ we get that $B \geq \frac{8 \log n}{b}$ and $|B^{\mathsf{alive}}| \geq B - \frac{4 \log n}{b} \geq \frac{B}{2}$. Next, we observe that

$$\frac{2A^2}{|B^{\mathsf{alive}}|} = 2\left|B^{\mathsf{alive}}\right| 2^{-8\beta b} \geq B \cdot 2^{-8\beta b} \geq \frac{2}{9b} 2^{(\log 3 - 8\beta) b} \geq 2^b.$$

As $b\ell \leq \frac{Cb \log n}{\log B} \leq C \log n$, we can conclude that $2^b \geq 4\beta b\ell$ and so $e^{-\frac{2A^2}{|B^{\mathsf{alive}}|}} \leq 2^{-4\beta b\ell}$, which implies that $\mathbb{E}\left[\Delta S(U)^\ell\right] \leq 2|B^{\mathsf{alive}}|^\ell \cdot 2^{-4\beta b\ell}$.

Using Equation (5) and the above bound yields a bound on $\mathbb{E}\left[\Delta S(Y)^\ell\right]$. By Markov's inequality,

$$\Pr\left[S(Y) < \frac{\mu}{2}\right] \leq \frac{\mathbb{E}\left[(S(Y)-\mu)^\ell\right]}{(\mu/2)^\ell} \leq \frac{\varepsilon + 2|B^{\mathsf{alive}}|^\ell \cdot 2^{-4\beta b\ell}}{(\mu/2)^\ell} \leq \left(\frac{8\left|B^{\mathsf{alive}}\right| 2^{-4\beta b}}{\mu}\right)^\ell. \quad (6)$$

Recalling that $\mu \geq |B^{\mathsf{alive}}| \cdot 2^{-\frac{3\beta}{2}b}$, Equation (6) becomes

$$\Pr\left[S(Y) < \frac{\mu}{2}\right] \leq \left(8 \cdot 2^{\left(-4\beta+\frac{3\beta}{2}\right)b}\right)^\ell < 2^{-2\beta b\ell} \leq 2^{-\frac{1}{2}\beta C \log n} \leq \frac{1}{2n},$$

where we have used the fact that $b\ell \geq \frac{C\log n}{4}$.

Overall, with probability at least $1 - \frac{1}{2n}$ over $y \sim Y$, g_i^y is nonconstant for at least $\frac{\mu}{2}$ of the i-s, and recall that each such g_i^y is over at least $(1-\beta)b_i$ variables. Fix such a good y, and let $\mathbf{G} \subseteq [B^{\mathsf{alive}}]$ be the set of nonconstant g_i^y-s. Again, we can write

$$\bigoplus_{i \in B^{\mathsf{alive}}} g_i^y = \left(\bigoplus_{i \in \mathbf{G}} g_i^y\right) \oplus \left(\bigoplus_{i \in B^{\mathsf{alive}} \setminus \mathbf{G}} g_i^y\right) \triangleq t_1 \oplus t_2.$$

Similarly to Claim 11, in order to bound the bias of $\overline{f|_{\mathrm{Res}(Y,Z)}}$ it is sufficient to bound the bias of $\overline{t_1}$, and so

$$\mathbb{E}[\overline{t_1}] \leq \left(1 - 2^{-\frac{3b}{2}}\right)^{\frac{\mu}{2}}.$$

Using the fact that $\mu \geq \frac{1}{2}B \cdot 2^{-\frac{3\beta}{2}b} \geq \frac{1}{9b}2^{\left(\log 3 - \frac{3\beta}{2}\right)b} > 2^{\frac{301}{200}b}$, we get

$$\mathbb{E}[\overline{t_1}] \leq e^{-2^{-\frac{3b}{2}} 2^{\frac{301b}{200}}} \leq e^{-\log^{\frac{C}{400}} n} \leq \frac{1}{2n}. \quad \blacktriangleleft$$

Proof of Lemma 10. Finally, the fact that with probability at least $1 - \frac{1}{n}$ over $(y,z) \sim Y \times Z$, $\left|\widetilde{f}_z(y) - \mathbb{E}[\overline{f}]\right| \leq \frac{1}{n}$, follows immediately from Claim 11 and Lemma 14. $\blacktriangleleft$

4.4 Case II – There Are No Heavy Buckets

In this subsection, we prove that a single pseudorandom restriction preserves the expectation in the case where there is no such a heavy B_q. Namely, for every $q \in [Q]$, either $m_q \leq 3^{c^q}$ or $m_q \leq \log^C n$ (or both).

▶ **Lemma 15.** *Let f be a read-once* PARITY $\circ$ AND *formula over n variables in which the width of every term is at most $C \log n$, and in which there are no heavy buckets as described above. Then, with probability at least $1 - \frac{1}{2} \cdot 2^{-C\frac{\log n}{\log\log n}}$ over $z \sim Z$ it holds that*

$$\left|\mathbb{E}\left[\widetilde{f}_z(Y)\right] - \mathbb{E}[\overline{f}]\right| \leq \frac{1}{2} \cdot 2^{-C\frac{\log n}{\log\log n}}.$$

Toward proving Lemma 15, we partition the Q buckets into two sets and treat terms that fall into each set of buckets separately. Namely, define the two sets as follows.

- $\mathcal{A} = \left\{q \in [Q] : m_q \leq \log^{2C} n\right\}$. We refer to these buckets as the *sparse buckets*.
- $\mathcal{B} = [Q] \setminus \mathcal{A}$. We refer to these buckets as the *well-behaved buckets*.

For each set $\mathcal{T} \in \{\mathcal{A}, \mathcal{B}\}$ we denote

$$f_{\mathcal{T}} = \bigoplus_{i \in \mathcal{T}} F_i,$$

and so $f = f_{\mathcal{A}} \oplus f_{\mathcal{B}}$. The next two subsections will be devoted to proving that the expectation of each $f_{\mathcal{T}}$ is preserved after a single pseudorandom restriction. In Section 4.4.3 we will combine the two results using the XOR lemma for small-bias distributions (Lemma 6) to prove Lemma 15.

4.4.1 Handling Sparse Buckets

For the sparse buckets, we will follow the Forbes-Kelley approach [14] to prove the following.

▶ **Lemma 16.** *With probability at least* $1 - \frac{1}{4} \cdot 2^{-C\frac{\log n}{\log\log n}}$ *over* $z \sim Z$*, it holds that*

$$\left| \mathbb{E}\left[\left(\widetilde{f_{\mathcal{A}}}\right)_z (Y) \right] - \mathbb{E}[\widetilde{f_{\mathcal{A}}}] \right| \leq \frac{1}{4} \cdot 2^{-C\frac{\log n}{\log\log n}}.$$

As outlined in Section 1.5.1, Lemma 16 follows readily from the work by Forbes and Kelley [14]. We require our restriction to work *with high probability* over $z \sim Z$, not merely in expectation, so we must redo some of Forbes and Kelley's analysis. (No substantial modification is needed.) The details follow.

Proof of Lemma 16. First, recall that each term in $f_{\mathcal{A}}$ is of width at most $C \log n$. There are at most $\log^{2C} n$ terms in each bucket, and at most $Q = O(\log\log n)$ such buckets, so overall $f_{\mathcal{A}}$ reads at most $n' = \log^{2C+2} n$ variables.

Note that $f_{\mathcal{A}}$ can be computed by a width-4 ROBP of length n'. We follow [14] and let $G\colon \{0,1\}^{n'} \to \mathbb{R}^{4\times 4}$ encode the transition of the branching program. Namely, perhaps after renumbering the variables, we have $G(x) = G_1(x_1) \cdot \ldots \cdot G_{n'}(x_{n'})$ where $G_i(x_i) = A_{i,x_i}$ for $A_{i,b}$ being the transition matrix that corresponds to taking the bit b while at layer i. Set $k_0 = \frac{8\log n}{\log\log n}$, and note that $k_0 \leq k$. By [14, Lemma 4.1], G can be written as

$$G = \mathbb{E}[G] + L + \sum_{i=1}^{n'} H_i \cdot G^{>i},$$

where L has degree[7] less than k_0, H_i is of degree exactly k_0, $G^{>i}$ is a width-4 ROBP, and H_i and $G^{>i}$ are on disjoint set of variables. More specifically,

$$L = \sum_{\alpha \in \mathbb{F}_2^{n'}, 0<|\alpha|<k_0} \widehat{G}_\alpha \chi_\alpha$$

is the truncated Fourier expansion of G, $G^{>i}(x_{i+1}, \ldots, x_n) = G_{i+1}(x_{i+1}) \cdot \ldots \cdot G_{n'}(x_{n'})$, and

$$H_i = \sum_{\alpha \in \mathbb{F}_2^{n'}, |\alpha|=k_0, \alpha_i=1} \widehat{G^{\leq i}}_\alpha \chi_\alpha,$$

where $G^{\leq i}(x_1, \ldots, x_i) = G_1(x_1) \cdot \ldots \cdot G_i(x_i)$. Let $\|\cdot\|$ be the Frobenius norm. By sub-additivity, we have

[7] We say a function $H\colon \{0,1\}^n \to \mathbb{R}^{w\times w}$ having Fourier expansion $\sum_{\alpha\in\mathbb{F}_2^n} \widehat{H}_\alpha \chi_\alpha$ has degree d if $\widehat{H}_\alpha$ is the zero matrix for every α with Hamming weight larger than d.

$$\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}[G(Y+Z\wedge U)]-\mathbb{E}[G]\right\|\right]\leq\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}[L(Y+Z\wedge U)]\right\|\right]+\sum_{i=1}^{n'}\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}\left[(H_i\cdot G^{>i})(Y+Z\wedge U)\right]\right\|\right]. \quad (7)$$

Just as in [14], the low-degree term L is dealt with a δ-biased distribution. From the work of Chattopadhyay, Hatami, Reingold, and Tal [9] we know that

$$L_1(L)=\sum_{k'=1}^{k_0}(c_{\mathsf{CHRT}}\log n')^{4k'}\leq 2(c_{\mathsf{CHRT}}\log n')^{4k_0}$$

for some universal constant $c_{\mathsf{CHRT}}\geq 1$. Thus, by Claim 5, we get that the first term of Equation (7) is bounded by

$$2\delta\cdot 2(c_{\mathsf{CHRT}}\log n')^{4k_0}\leq 2^{-C\log n}\cdot 2^{8k_0\log\log\log n}\leq n^{-\frac{C}{2}},$$

taking into account the fact that $\mathbb{E}[L(U)]=0$.

For each i of the second term of Equation (7), we use sub-multiplicativity and the fact that H_i and $G^{>i}$ are on disjoint set of variables to get

$$\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}\left[(H_i\cdot G^{>i})(Y+Z\wedge U)\right]\right\|\right]\leq\mathop{\mathbb{E}}_{Y,Z}\left[\left\|\mathop{\mathbb{E}}_{U}[H_i(Y+Z\wedge U)]\right\|\cdot\left\|\mathop{\mathbb{E}}_{U}[G^{>i}(Y+Z\wedge U)]\right\|\right].$$

As $G^{>i}$ is a width-4 ROBP, $\left\|\mathbb{E}_U[G^{>i}(y+z\wedge U)]\right\|\leq 2$ for all $y\sim Y$ and $z\sim Z$. Continuing the above bound, by Cauchy-Schwarz we get

$$\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}(H_i\cdot G^{>i})[Y+Z\wedge U]\right\|\right]\leq 2\sqrt{\mathop{\mathbb{E}}_{Y,Z}\left[\left\|\mathop{\mathbb{E}}_{U}[H_i(Y+Z\wedge U)]\right\|^2\right]}.$$

Following [14, Lemma 7.1][8], using the bound by Chattopadhyay et al. [9] and Parseval's identity [14, Proposition 3.1], we get

$$\begin{aligned}\mathop{\mathbb{E}}_{Y,Z}\left[\left\|\mathop{\mathbb{E}}_{U}[H_i(Y+Z\wedge U)]\right\|^2\right]&\leq\left(2^{-Ck_0}+\gamma\right)\cdot\left(\delta\left(\sum_{\alpha\in\mathbb{F}_2^{n'}}\left\|(\widehat{H_i})_\alpha\right\|\right)^2+\sum_{\alpha\in\mathbb{F}_2^{n'}}\left\|(\widehat{H_i})_\alpha\right\|^2\right)\\&\leq\left(2^{-Ck_0}+\gamma\right)\cdot\left(\delta\cdot L_1^2\left(G^{\leq i}\right)+\mathbb{E}\left[\left\|G^{\leq i}(U)\right\|^2\right]\right)\\&\leq 8\cdot 2^{-Ck_0}.\end{aligned}$$

Overall, we get that

$$\mathop{\mathbb{E}}_{Z}\left[\left\|\mathop{\mathbb{E}}_{Y,U}[G(Y+Z\wedge U)]-\mathbb{E}[G]\right\|\right]\leq n^{-\frac{C}{2}}+2n'\sqrt{8\cdot 2^{-Ck_0}}\leq\frac{1}{16}\cdot 2^{-\frac{C}{4}k_0}=\frac{1}{16}\cdot 2^{-\frac{2C\log}{\log\log n}},$$

and we can choose the encoding G so that $f_{\mathcal{A}}(x)=G(x)_{1,1}$. Markov's inequality completes the proof. ◀

[8] Forbes and Kelley [14] take the bits of Z to have marginals $p=\frac{1}{2}$, but one can extend the lemma easily for the case of a general p.

4.4.2 Handling Well-Behaved Buckets

We will use our tail bounds for subset-wise symmetric polynomials to prove the following lemma.

▶ **Lemma 17.** *With probability at least $1-\frac{1}{2n}$ over $z \sim Z$, $f_{\mathcal{B}}$ can be written as $f_{\mathcal{B}} = f'_{\mathcal{B}} \oplus f''_{\mathcal{B}}$, where $f'_{\mathcal{B}}$ and $f''_{\mathcal{B}}$ are over disjoint set of variables, and for every $g \in \{f'_{\mathcal{B}}, f''_{\mathcal{B}}\}$ it holds that*

$$|\mathbb{E}\left[\widetilde{g}_z(Y)\right] - \mathbb{E}\left[\overline{g}\right]| \leq \frac{1}{n}.$$

The proof of Lemma 17 will follow immediately from Claim 20 and Lemma 21. Toward proving the above lemma, let us set some preliminaries.

▷ Claim 18. If $q \in \mathcal{B}$ then $c^q \in [C \log\log n, C \log n]$ and $m_q \leq 3^{c^q}$.

Proof. The upper bound on c^q follows immediately from the assumption in Lemma 15 that every term has width at most $C \log n$. Also, $m_q > \log^{2C} n$ since $q \notin \mathcal{A}$. Since we are at Case II, $m_q > \log^{2C} n$ implies that $m_q \leq 3^{c^q}$. From the fact that $\log^{2C} n < 3^{c^q}$ we get $c^q > \log_3(\log^{2C} n) > C \log\log n$. ◁

Recall that a term f_i is *good* under z if the variables read by f_i intersects with z in at least $1-\beta$ fraction.

▷ Claim 19. For a fixed $z \in \{0,1\}^n$, let $\mathbf{X}_z \subseteq [m]$ be the set of terms in $f_{\mathcal{B}}$ that are not good under z. Then, with probability at least $1-\frac{1}{2n}$ over $z \sim Z$,

$$|\mathbf{X}_z| \leq \frac{3c}{c-1} \log n.$$

Proof. The proof is very similar to Claim 13. Fix a bucket $q \in \mathcal{B}$, set $k_q = \frac{3 \log n}{c^q}$ and observe that $k \geq k_q$. Let $S \subseteq B_q$ be some subset of cardinally k_q. We first bound the probability p that every term f_i for $i \in S$ is bad under $z \sim Z$.

For a *truly random* $z \sim \text{Bernoulli}(1-2^{-C})^{\otimes n}$, the above probability is bounded by

$$\prod_{i \in S} \binom{w_i}{\beta w_i} 2^{-\beta C w_i} \leq \prod_{i \in S} 2^{w_i} 2^{-5 w_i} \leq \left(2^{-4 \cdot c^{q-1}}\right)^{k_q} \leq 2^{-3 k_q c^q} \leq n^{-9}.$$

For $z \sim Z$, we get that $p \leq n^{-9} + \gamma \leq 2n^{-9}$. Thus, with probability at most $\binom{m_q}{k_q} p$ over $z \sim Z$ there exists a set of k_q terms in B_q whose all terms are bad under z. By using Claim 18, we get

$$\binom{m_q}{k_q} p \leq m_q^{k_q} \cdot 2^{-9 \log n + 1} \leq 3^{k_q c^q + \log_3 2 \cdot (-9 \log n + 1)} \leq 3^{-\frac{9}{4} \log n} \leq n^{-3}.$$

Moreover, with probability at most $|\mathcal{B}| n^{-3} \leq n^{-2}$ over $z \sim Z$ there exists a $q \in \mathcal{B}$ and a set of k_q terms in B_q whose all terms are bad under z. Taking the contrapositive, we infer that with probability at least $1 - n^{-2} \geq 1 - \frac{1}{2n}$ over $z \sim Z$, we have at most

$$\sum_{q \in \mathcal{B}} k_q \leq \sum_{q=1}^{Q} \frac{3 \log n}{c^q} \leq \frac{3c}{c-1} \log n.$$

terms that are bad for z. ◁

From here onwards, we fix a z satisfying $|\mathbf{X}_z| \leq \frac{3c}{c-1}\log n$. Write

$$f_{\mathcal{B}} = \bigoplus_{i \in \mathbf{C} \setminus \mathbf{X}_z} f_i \oplus \bigoplus_{i \in \mathbf{X}_z} f_i \triangleq f'_{\mathcal{B}} \oplus f''_{\mathcal{B}},$$

where $\mathbf{C} = \bigcup_{q \in \mathcal{B}} B_q \subseteq [m]$ is the set of all terms that belong to $\mathcal{B}$'s buckets. Simply put, we divide $f_{\mathcal{B}}$ to the parity of exceptional terms $f''_{\mathcal{B}}$ and non-exceptional terms $f'_{\mathcal{B}}$ for whom we will refer to as *good terms*. We stress that both $f'_{\mathcal{B}}$ and $f''_{\mathcal{B}}$ depend on z.

▷ Claim 20 (Exceptional terms).

$$\left| \mathbb{E}\left[\left(\widetilde{f''_{\mathcal{B}}}\right)_z (Y) \right] - \mathbb{E}[\overline{f''_{\mathcal{B}}}] \right| \leq \frac{1}{n}.$$

Proof. For brevity, let $g = f''_{\mathcal{B}}$. For a fixed $w \in \{0,1\}^n$, let $g^w(x) = g(x+w)$. The proof will follow from bounding the spectral norm of $\overline{g^w}$. Indeed, $\overline{g^w}$ is a multiplication of at most $\frac{3c}{c-1}\log n$ terms, each of which has spectral norm at most 3. By sub-multiplicativity,

$$L_1\left(\overline{g^w}\right) \leq 3^{\frac{3c}{c-1}\log n} \leq n^{\frac{5c}{c-1}}.$$

Now, $\delta \leq \frac{1}{2} n^{-\frac{5c}{c-1}-1}$, so by Claim 5 we get that $|\mathbb{E}[\overline{g^w}(Y)] - \mathbb{E}[\overline{g^w}]| \leq \frac{1}{n}$ for every $w \in \{0,1\}^n$. Fooling $\overline{g^w}$ is sufficient to fool $\widetilde{g}_z$. To see this, note that

$$\begin{aligned} |\mathbb{E}\left[\widetilde{g}_z(Y)\right] - \mathbb{E}[\overline{g}]| &= |\mathbb{E}[\overline{g}(Y + z \wedge U)] - \mathbb{E}[\overline{g}(U + z \wedge U')]| \\ &= \left| \mathop{\mathbb{E}}_{w \sim U} \left[\mathbb{E}\left[\overline{g^w}(Y)\right] - \mathbb{E}\left[\overline{g^w}\right] \right] \right| \leq \frac{1}{n}, \end{aligned}$$

where U' is an independent copy of U. ◁

Next, we prove:

▶ **Lemma 21** (Good terms)**.**

$$\left| \mathbb{E}\left[\left(\widetilde{f'_{\mathcal{B}}}\right)_z (Y) \right] - \mathbb{E}[\overline{f'_{\mathcal{B}}}] \right| \leq \frac{1}{n}.$$

Proof. For brevity, let $g = f'_{\mathcal{B}}$ and recall that its set of terms is given by $\mathbf{C} \setminus \mathbf{X}_z$. Shifting the bias function $\widetilde{g} = \widetilde{g}_z$ to mean zero, recall that we define

$$\check{g}(x) = \frac{\widetilde{g}(x)}{\mathbb{E}[\widetilde{g}]} - 1.$$

Thus, we can write

$$\widetilde{g} = \mathbb{E}[g] \prod_{i \in \mathbf{C} \setminus \mathbf{X}_z} (1 + \check{g}_i) = \mathbb{E}[g] \sum_{I \subseteq \mathbf{C} \setminus \mathbf{X}_z} \prod_{i \in I} \check{g}_i = \mathbb{E}[g] \sum_{\vec{k} \in \mathbb{N}^Q} \sum_{I \subseteq \mathbf{C} \setminus \mathbf{X}_z, K(I) = \vec{k}} \prod_{i \in I} \check{g}_i,$$

where by $K(I) = \vec{k}$ we mean that for every $q \in [Q]$, there are $\vec{k}[q]$ terms in I that belong to the q-th bucket, i.e., $|I \cap B_q| = \vec{k}[q]$. For simplicity, we reorder the terms of g and write $g = \bigoplus_{i \in [m']} g_i$ for $m' = |\mathbf{C} \setminus \mathbf{X}_z|$, and for $q \in [Q]$, $B_q \subseteq [m']$ is the set of terms in g that belong to the q-th bucket. We abbreviate $\vec{g} = (\check{g_1}, \ldots, \widetilde{g_{m'}})$, and write

$$S_{\vec{k}}(\vec{g}) = \sum_{I \subseteq [m'], K(I) = \vec{k}} \prod_{i \in I} \check{g}_i.$$

Under these notations, $\widetilde{g} = \mathbb{E}[g] \sum_{\vec{k} \in \mathbb{N}^Q} S_{\vec{k}}(\vec{g})$.

Let $I_g(x)$ be the Boolean-valued function which is 1 if and only if

$$\sum_{\vec{k}\in\mathbb{N}^Q, \|\vec{k}\|_{(c)}>A} \left|S_{\vec{k}}\left(\vec{g}(x)\right)\right| \leq 2^{-\frac{A}{1024}},$$

where $A = \overline{C}\log n$ and $\|\vec{k}\|_{(c)} = \sum_{q=1}^{Q} c^q \cdot \vec{k}[q]$. Section 4.4.4 will be devoted to showing that $\mathbb{E}[I_g(Y)]$ is very close to 1. Namely,

▶ **Lemma 22.** *The following two inequalities hold.*

1. $\mathbb{E}[I_g(Y)] \geq 1 - e^{-c_I A}$ *for* $c_I = \frac{\ln 2}{2^{23}}$.
2. $\mathbb{E}\left[S^2_{\vec{k}}(\vec{g}(Y))\right] \leq 2^{-\frac{1}{8}\|\vec{k}\|_{(c)}}$.

For now, let us take Lemma 22 as given and continue with the proof of Lemma 21. We proceed by writing

$$|\mathbb{E}[\widetilde{g}(Y)] - \mathbb{E}[\widetilde{g}]| \leq |\mathbb{E}\left[\widetilde{g}(Y) \mid I_g(Y)=1\right] - \mathbb{E}[\widetilde{g}]| + 2\Pr\left[I_g(Y)=0\right]. \tag{8}$$

By Lemma 22, we have that $\Pr[I_g(Y)=0] \leq e^{-c_I A}$. Next, observe that

$$|\mathbb{E}\left[\widetilde{g}(Y) \mid I_g(Y)=1\right] - \mathbb{E}[\widetilde{g}]| = \left|\mathbb{E}[g] \sum_{\vec{k}\in\mathbb{N}^Q, \|\vec{k}\|_{(c)}>0} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \mid I_g(Y)=1\right]\right|,$$

and set

$$\boldsymbol{\Delta} = \left|\sum_{\vec{k}\in\mathbb{N}^Q, \|\vec{k}\|_{(c)}>0} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \mid I_g(Y)=1\right]\right|,$$

so Equation (8) gives us

$$|\mathbb{E}[\widetilde{g}(Y)] - \mathbb{E}[\widetilde{g}]| \leq \boldsymbol{\Delta} + 2e^{-c_I A}. \tag{9}$$

We bound $\boldsymbol{\Delta}$ as follows.

$$\boldsymbol{\Delta} \leq \left|\sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \mid I_g(Y)=1\right]\right| + \max_{y\in\{0,1\}^n, I_g(y)=1} \sum_{\vec{k}\in\mathbb{N}^Q, \|\vec{k}\|_{(c)}>A} \left|S_{\vec{k}}(\vec{g}(y))\right|.$$

By definition, the second term is at most $2^{-\frac{A}{1024}}$. The first term, call it $\boldsymbol{\Delta}_1$, can be split into two terms as follows.

$$\begin{aligned}
\boldsymbol{\Delta}_1 &= \frac{1}{\Pr[I_g(Y)=1]} \left|\sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \cdot I_g(Y)\right]\right| \\
&\leq 2\left|\sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \cdot I_g(Y)\right]\right| \qquad (10) \\
&\leq 2\left|\sum_{\substack{\vec{k}\in\mathbb{N}^Q, \\ 0<\|\vec{k}\|_{(c)}\leq A}} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y))\right]\right| + 2\left|\sum_{\substack{\vec{k}\in\mathbb{N}^Q, \\ 0<\|\vec{k}\|_{(c)}\leq A}} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y)) \cdot (1-I_g(Y))\right]\right|
\end{aligned}$$

$$\leq 2\left|\sum_{\substack{\vec{k}\in\mathbb{N}^Q,\\ 0<\|\vec{k}\|_{(c)}\leq A}} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y))\right]\right| + 2\sqrt{\mathbb{E}[1-I_g(Y)]}\cdot \sum_{\substack{\vec{k}\in\mathbb{N}^Q,\\ 0<\|\vec{k}\|_{(c)}\leq A}} \sqrt{\mathbb{E}\left[S^2_{\vec{k}}(\vec{g}(Y))\right]}, \tag{11}$$

where the last inequality follows from the triangle inequality followed by Cauchy-Schwarz. By Lemma 22, the second term of Equation (11), $\mathbf{\Delta}_{1,2}$, is at most

$$\begin{aligned}
\mathbf{\Delta}_{1,2} &\leq 2\cdot e^{-c_I A}\cdot \sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \sqrt{2^{-\frac{1}{8}\|\vec{k}\|_{(c)}}} \\
&\leq 2\cdot e^{-c_I A}\cdot \sum_{w=1}^{A-1} \left|\left\{\vec{k}\in\mathbb{N}^Q : w<\|\vec{k}\|_{(c)}\leq w+1\right\}\right| 2^{-\frac{1}{\sqrt{8}}w} \\
&\leq 2\cdot e^{-c_I A}(A+1)^Q \sum_{w=1}^{A} 2^{-\frac{1}{\sqrt{8}}w} \leq 8(A+1)^Q e^{-c_I A} \leq 2^{\frac{2}{\log c}(\log\log n)^2} e^{-c_I A} \leq \frac{1}{8n}.
\end{aligned}$$

To finish bounding $\mathbf{\Delta}_1$, it is left to bound the first term of Equation (11), denoted by $\mathbf{\Delta}_{1,1}$.

▷ Claim 23. $\mathbf{\Delta}_{1,1} = 2\left|\sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y))\right]\right| \leq \frac{1}{8n}$.

Proof. The proof goes by bounding the spectral norm of the function $S_{\vec{k}}(\vec{g}(x))$. As for every $\vec{k}\in\mathbb{N}^Q$ with $\|\vec{k}\|_{(c)}\neq 0$, $\mathbb{E}[S_{\vec{k}}(\vec{g}(U))]=0$, the claim will follow by using Claim 5, together with sub-additivity and sub-multiplicativity. First, note that:

▷ Claim 24. For every $i\in[m]$, $L_1(\breve{g}_i)\leq 4$.

Proof. Consider the function $h_i = 1-g_i$, so $L_1(\widetilde{g}_i)\leq L_1(\widetilde{h}_i)+1$ and $\mathbb{E}[\widetilde{h}_i]=\mathbb{E}[h_i]=1-\mathbb{E}[g_i]\geq\frac{1}{2}$. Now, $L_1(\breve{h}_i)\leq\frac{1}{\mathbb{E}[h_i]}L_1(\widetilde{h}_i)+1\leq 2L_1(\widetilde{h}_i)+1$. Recalling that $\widetilde{h}_i(x)=\mathbb{E}[h_i(x+z\wedge U)]$, we get $L_1(\widetilde{h}_i)\leq 1$ as every shift of h_i is a negated conjunction of literals. Thus, $L_1(\breve{h}_i)\leq 3$ and $L_1(\breve{g}_i)\leq 4$. ◁

Then, for every such $\vec{k}\in\mathbb{N}^Q$,

$$\begin{aligned}
L_1\left(S_{\vec{k}}(\vec{g})\right) &\leq \sum_{I\subseteq\mathbf{C}\setminus\mathbf{X}_T, K(I)=\vec{k}} \prod_{i\in I} L_1\left(\breve{g}_i\right) \leq \sum_{I\subseteq\mathbf{C}\setminus\mathbf{X}_T, K(I)=\vec{k}} 4^{|I|} \\
&= \sum_{I\subseteq\mathbf{C}\setminus\mathbf{X}_T, K(I)=\vec{k}} \prod_{q\in[Q]} 4^{\vec{k}[q]} = \prod_{q\in[Q]} \binom{\vec{k}[q]}{m_q} 4^{\vec{k}[q]}.
\end{aligned}$$

Recall that Claim 18 tells us that $m_q\leq 3^{c^q}$, so

$$L_1\left(S_{\vec{k}}(\vec{g})\right) \leq \prod_{q\in[Q]} 3^{c^q(1+\log_3 4)\vec{k}[q]} \leq 12^{\|\vec{k}\|_{(c)}}. \tag{12}$$

Finally,

$$L_1\left(\sum_{\vec{k}\in\mathbb{N}^Q, 0<\|\vec{k}\|_{(c)}\leq A} \mathbb{E}\left[S_{\vec{k}}(\vec{g}(Y))\right]\right) \leq (A+1)^Q\cdot 12^A \leq 2^{6A} \leq n^{6\overline{C}},$$

and the claim follows by observing that $\delta\leq\frac{1}{32n}n^{-6\overline{C}}$. ◁

Incorporating the above claim, we get that $\mathbf{\Delta}_1 = \mathbf{\Delta}_{1,1} + \mathbf{\Delta}_{1,2} \leq \frac{1}{8n} + \frac{1}{8n} \leq \frac{1}{4n}$, which readily gives $\mathbf{\Delta} \leq \frac{1}{4n} + 2^{-\frac{A}{1024}} \leq \frac{1}{2n}$. Plugging-it in Equation (9), we finally get

$$|\mathbb{E}[\widetilde{g}(Y)] - \mathbb{E}[\widetilde{g}]| \leq \frac{1}{2n} + 2e^{-c_I A} \leq \frac{1}{n}$$

and the desired result. ◀

4.4.3 Putting It Together

Here we finally incorporate Lemma 16 and Lemma 17.

Proof of Lemma 15. By Lemma 16 and Lemma 17, with probability at least $1 - \frac{1}{4} \cdot 2^{-C\frac{\log n}{\log\log n}} - \frac{1}{n} \geq 1 - \frac{1}{2} \cdot 2^{-C\frac{\log n}{\log\log n}}$ over $z \sim Z$, we can write

$$f = f_{\mathcal{A}} \oplus f'_{\mathcal{B}} \oplus f''_{\mathcal{B}},$$

where the three functions are over disjoint set of variables, and it holds that for each $\mathcal{T} \in \{\mathcal{A}, \mathcal{B}, \mathcal{B}'\}$,

$$\left|\left(\widetilde{f_{\mathcal{T}}}\right)_z (Y') - \mathbb{E}\left[\overline{f_{\mathcal{T}}}\right]\right| \leq \frac{1}{4} \cdot 2^{-C\frac{\log n}{\log\log n}}$$

for any δ-biased distribution Y'. Using the XOR lemma for small-biased spaces (see Lemma 6), taking into account that our distribution Y is in fact δ^3-biased, we conclude that

$$\left|\mathbb{E}[\widetilde{f}_z(Y)] - \mathbb{E}[\overline{f}]\right| \leq 16^3 \cdot 2 \cdot \frac{1}{4} \cdot 2^{-C\frac{\log n}{\log\log n}} \leq \frac{1}{2} \cdot 2^{-C\frac{\log n}{\log\log n}},$$

and the lemma follows. ◀

4.4.4 I_g Almost Always Happens

We keep using the notations of Section 4.4.2. Specifically, recall that $g = f'_{\mathcal{B}} = \bigoplus_{i\in[m']} g_i$ for $m' = |\mathbf{C} \setminus \mathbf{X}_z|$, and for $q \in [Q]$, $B_q \subseteq [m']$ is the set of terms in g that belong to the q-th bucket. Also, for $\vec{g} = (\widetilde{g_1}, \ldots, \widetilde{g_{m'}})$,

$$S_{\vec{k}}(\vec{g}) = \sum_{I\subseteq[m'], K(I)=\vec{k}} \prod_{i\in I} \widetilde{g}_i.$$

Recall that $I_g(x) \in \{0,1\}$ is 1 if and only if

$$\sum_{\vec{k}\in\mathbb{N}^Q, \|\vec{k}\|_{(c)}>A} \left|S_{\vec{k}}\left(\vec{g}(x)\right)\right| \leq 2^{-\frac{A}{1024}},$$

where $A = \overline{C}\log n$ and $\|\vec{k}\|_{(c)} = \sum_{q\in[Q]} c^q \cdot \vec{k}[q]$.

Proof of Lemma 22. As in Section 2, we define

$$R_{\vec{k}}(\vec{g}) = S^2_{\vec{k}}(\vec{g}) \cdot \prod_{q\in[Q]} \vec{k}[q]!.$$

By Lemma 3, to prove the bound on $\Pr[I_g(Y) = 0]$ it is sufficient to prove that for every $\vec{k} \in \mathbb{N}^Q$ with $\|\vec{k}\|_{(c)} \leq A$ we have that

$$\mathbb{E}\left[R_{\vec{k}}(\vec{g}(Y))\right] \leq 2^{-\frac{1}{8}\|\vec{k}\|_{(c)}}.$$

By now a standard course of action, we aim at bounding the spectral norm of the function $R_{\vec{k}}(\vec{g})$, together with its expectation under the uniform distribution. To this end, let us define, for $q \in [Q]$ and an integer ℓ,

$$\check{S}_{\ell,q} = \sum_{I \subseteq B_q, |I|=\ell} \prod_{i \in I} \breve{g}_i,$$

so $R_{\vec{k}}(\vec{g}) = \prod_{q \in [Q]} \check{S}^2_{\vec{k}[q],q} \vec{k}[q]!$. First, we record that:

▷ Claim 25. For every $i \in [m']$, $\mathbb{E}\left[\breve{g}_i^{\,2}\right] \leq 2^{-(2-2\beta)w_i}$.

Proof. Let $V_i \subseteq [n]$ be the set of variables read by g_i, of cardinality w_i, and let $\ell_i = |V_i \cap \{j \in [n] : z_j = 1\}|$ be the number of *live* variables read by g_i. Note that

$$\widetilde{g}_i(x) = \mathbb{E}[g_i(x + z \wedge U)] = \begin{cases} 0 & \text{if there exists } j \in V_i \text{ such that } x_j = z_j = 0, \\ 2^{-\ell_i} & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned} \mathbb{E}\left[\widetilde{g}_i^{\,2}\right] &= 2^{-2\ell_i} \Pr_{x \sim U}[\text{for every} j \in V_i \text{ s.t. } z_j = 0 \text{ it holds that } x_j = 1] \\ &= 2^{-2\ell_i} 2^{-(w_i - \ell_i)} = 2^{-w_i - \ell_i}. \end{aligned}$$

Recalling that $\ell_i \geq (1-\beta)w_i$ (g_i is good under z), we have $\mathbb{E}[\widetilde{g}_i^{\,2}] \leq 2^{-(2-\beta)w_i}$. Let $h_i = 1 - g_i$, and note that

$$\mathbb{E}\left[\breve{h}_i^{\,2}\right] = \mathrm{Var}\left[\breve{h}_i\right] = \frac{\mathrm{Var}[\widetilde{g}_i]}{\mathbb{E}^2[h_i]} \leq 4 \cdot \mathbb{E}\left[\widetilde{g}_i^{\,2}\right] \leq 4 \cdot 2^{-(2-\beta)w_i} \leq 2^{-(2-2\beta)w_i}.$$

The fact that $\mathrm{Var}[\breve{h}_i] = \mathrm{Var}[\breve{g}_i] = \mathbb{E}[\breve{g}_i^{\,2}]$ finishes the proof. ◁

Now,

$$\begin{aligned} \mathbb{E}\left[\check{S}^2_{\ell,q}\right] &= \sum_{I \subseteq B_q, |I|=\ell} \prod_{i \in I} \mathbb{E}\left[\breve{g}_i^{\,2}\right] \leq \sum_{I \subseteq B_q, |I|=\ell} \prod_{i \in I} 2^{-(2-2\beta)w_i} \\ &\leq \sum_{I \subseteq B_q, |I|=\ell} 2^{-(2-2\beta)c^q \ell} \leq \binom{m_q}{\ell} 2^{-(2-2\beta)c^q \ell} \leq \frac{3^{c^q \ell} e^\ell 2^{-(2-2\beta)c^q \ell}}{\ell!} \leq \frac{1}{\ell!} 2^{-\frac{c^q \ell}{4}}. \end{aligned}$$

Plugging it in our expression for $R_{\vec{k}}$, we get

$$\mathbb{E}\left[R_{\vec{k}}(\vec{g})\right] = \prod_{q \in [Q]} \mathbb{E}\left[\check{S}_{\vec{k}[q],q} \vec{k}[q]!\right] \leq \prod_{q \in [Q]} 2^{-\frac{c^q \vec{k}[q]}{4}} = 2^{-\frac{1}{4}\|\vec{k}\|_{(c)}}. \tag{13}$$

Finally, let us bound $L_1(R_{\vec{k}}(\vec{g}))$. In Equation (12) we established the fact that $L_1(S_{\vec{k}}(\vec{g})) \leq 12^{\|\vec{k}\|_{(c)}} \leq 12^A$. Thus,

$$L_1\left(R_{\vec{k}}(\vec{g})\right) \leq 12^{2A} \prod_{q \in [Q]} \vec{k}[q]! \leq 12^{2A} e^{\sum_{q \in [Q]} \vec{k}[q] \ln \vec{k}[q]} \leq 12^{2A} e^{(\ln A) \sum_{q \in [Q]} \vec{k}[q]}.$$

As $\|\vec{k}\|_{(c)} = \sum_{q \in [Q]} c^q \vec{k}[q] \leq A$ and $c^q \geq C \log\log n$ (see Claim 18), $\sum_{q \in [Q]} \vec{k}[q] \leq \frac{A}{C \log\log n}$ and we get

$$L_1\left(R_{\vec{k}}(\vec{g})\right) \leq 12^{2A} e^{\ln A \frac{A}{C \log\log n}} \leq 12^{2A} 2^{\frac{\overline{C}}{C} \log n} \leq n^{10\overline{C}}.$$

Note that $\delta \leq \frac{1}{32} n^{-10\overline{C}} 2^{-\frac{A}{4}}$. Thus, by Claim 5,

$$\mathbb{E}\left[R_{\vec{k}}(\vec{g}(Y))\right] \leq 2^{-\frac{1}{4}\|\vec{k}\|_{(c)}} + \delta \cdot n^{10\overline{C}} \leq 2^{-\frac{1}{8}\|\vec{k}\|_{(c)}},$$

and we are done with bounding $\Pr[I_g(Y) = 0]$. For the bound on $\mathbb{E}\left[S^2_{\vec{k}}(\vec{g}(Y))\right]$, simply observe that $\mathbb{E}\left[S^2_{\vec{k}}(\vec{g}(Y))\right] < \mathbb{E}\left[R_{\vec{k}}(\vec{g}(Y))\right]$. ◀

5 Full PRG via Iterated Restrictions

So far, we have shown how to pseudorandomly assign values to a *constant fraction* of the inputs of any read-once PARITY ∘ AND formula using $O(\log n)$ truly random bits, preserving the expectation of the formula to within near-optimal error. In this section, to complete the proof of Theorem 1, we show how to pseudorandomly assign values to *all* the inputs, i.e., we give a genuine PRG.

For convenience, we make the following definitions.

▶ **Definition 26.** *Let $w > 0$. A* w-proper formula *is a read-once* PARITY ∘ AND *formula of width at most w and length most 2^{8w}. We say that such a formula is* short *if its length is at most 2^{4w}; otherwise, we say that the formula is* long.

Our main goal is to fool $(C \log n)$-proper formulas, but along the way, we will obtain a PRG for w-proper formulas with seed length $O(w)$ and error $\exp(-\widetilde{\Omega}(w))$, even for w substantially smaller than $\log n$.

5.1 Restrictions for Proper Formulas

Recall that Lemma 9 provides a pseudorandom restriction that uses only $O(\log n)$ truly random bits. We now generalize this fact in two respects. First, in the case of w-proper formulas ($\log \log n \leq w \leq C \log n$), we improve the seed length to $O(w)$. Second, in the case of *short* w-proper formulas, we argue that the restriction *simplifies* the formula, in the sense that it transforms it into a $(w/2)$-proper formula.

▶ **Lemma 27.** *For every $w, n \in \mathbb{N}$ with $w \leq C \log n$, there is a distribution X over $\{0,1,\star\}^n$ with the following properties.*

1. *(Seed length) There is an explicit algorithm to sample from X using just $O(w + \log \log n)$ truly random bits.*
2. *(Expectation preservation) If f is a w-proper formula, then X preserves the expectation of f with error $\exp(-\Omega(w/\log w))$.*
3. *(Simplification) If f is a* short *w-proper formula, then*

$$\Pr[f|_X \text{ is a } (w/2)\text{-proper formula}] \geq 1 - 2^{-w}.$$

Proof. Let $n' = 2^{8w} \cdot w$. Let Y be an n'-wise δ^3-biased distribution where $\delta = (n')^{-12\overline{C}}$, and let Z be γ-almost k-wise independent with marginals $1 - 2^{-C}$, where $k = 6 \log n'$ and $\gamma = (n')^{-9}$. Our restriction is

$$X = \mathrm{Res}(Y, Z)^{\circ 2^{C+4}}.$$

By standard constructions [25, 3] and Claim 7, X can be explicitly sampled using $O(w + \log \log n)$ truly random bits.

Now, to prove expectation preservation, let f be a w-proper formula. By w-properness, there is some set of indices $I \subseteq [n]$, $|I| \leq n'$, such that $f(x)$ only depends on $x|_I$. Let $g\colon \{0,1\}^{|I|} \to \{0,1\}$ be the w-proper formula such that $f(x) = g(x|_I)$. Since $Y|_I$ is δ^3-biased and $Z|_I$ is γ-almost k-wise independent with marginals $1 - 2^{-C}$, Lemma 9 implies that $\mathrm{Res}(Y|_I, Z|_I)$ preserves the expectation of g with error $\exp(-\Omega(\frac{\log n'}{\log\log n'}))$, which is $\exp(-\Omega(w/\log w))$. It follows that $\mathrm{Res}(Y, Z)$ preserves the expectation of f with the same error. The error of X is only larger by a constant factor 2^{C+4}, because any restriction of a w-proper formula is trivially another w-proper formula.

Finally, to prove simplification, let f be a *short* w-proper formula, and let f_i be a term. Since $k > w/2$, by Claim 8, the probability that more than $w/2$ variables from f_i are assigned $\star$ by X is bounded by

$$\binom{w}{w/2} \cdot \left((1-2^{-C})^{2^{C+4}\cdot w/2} + 2^{C+4}\gamma\right) \leq 2^w \cdot \left(e^{-2^{-C}\cdot 2^{C+3}w} + 2^{C+4}\cdot (n')^{-9}\right) < 2^{-5w}.$$

The number of terms in f is at most 2^{4w}, so by the union bound, except with probability 2^{-w}, $f|_X$ has maximum width at most $w/2$. Furthermore, restricting cannot *increase* the number of terms, so the number of terms is still bounded by $2^{4w} = 2^{8(w/2)}$. Therefore, in this case, $f|_X$ is $(w/2)$-proper. ◀

5.2 Full PRGs for Long Proper Formulas [24]

The simplification clause of Lemma 27 only applies if f is *short*. If f is *long*, we will therefore need a different approach. We will take a similar approach as Meka, Reingold, and Tal [24]. A *full PRG* for long w-proper formulas follows readily from their work.

▶ **Lemma 28** ([24]). *For every $w, n \in \mathbb{N}$, there is an explicit (2^{-w})-PRG for* long *w-proper formulas with seed length*

$$O(w + \log\log n).$$

Proof sketch. In short, the PRG is one of the PRGs by Meka et al. [24, full version, Lemma 6.2], except we replace every δ-biased distribution with a $(\cdot)$-wise δ-biased distribution to optimize the seed length.

In more detail, let $n' = 2^{8w}\cdot w$. Sample $v \in \{0,1\}^{wn}$ from an $(n'w)$-wise (c_{MRT}^{-w})-biased distribution, where c_{MRT} is a suitable constant. Think of v as n blocks of w bits. Define a set $I \subseteq [n]$ as follows: include i in I if and only if the i-th block of v is 1^w.

Sample $x^{(0)}, x^{(1)}, \ldots, x^{(16)} \in \{0,1\}^n$ independently from an (n')-wise (c_{MRT}^{-w})-biased distribution. The PRG outputs the string x defined by

$$x_i = \begin{cases} x_i^{(0)} & \text{if } i \notin I \\ \bigoplus_{j=1}^{16} x_i^{(j)} & \text{if } i \in I. \end{cases}$$

By standard constructions [25, 3], the seed length of this PRG is

$$O(\log n' + w + \log\log n) = O(w + \log\log n).$$

As for correctness, let f be a long w-proper formula. Let $J \subseteq [n]$ be the set of indices of variables that f reads, so there is some long w-proper formula g on $|J|$ input bits such that $f(x) = g(x|_J)$. Let X be the distribution output by the PRG. Since $|J| \leq n'$, the

distribution $X|_J$ is exactly the pseudorandom distribution designed by Meka et al. [24, full version, Lemma 6.2]. Furthermore, since f is long, $|J| > 2^{4w}$. It follows that g is in the class of functions fooled by Meka et al.'s pseudorandom distribution: g is an XOR of m non-constant Boolean functions on disjoint variables, where each function is on at most w variables, with $16^w < m \le 16^{2w}$ and $\log\log(|J|/2^w) \ll w \le \log|J|$. Therefore, $X|_J$ fools g with error 2^{-w}, and hence X fools f with error 2^{-w}. ◀

5.3 Full PRGs for Width-$O(\log n)$ Formulas

For *short* proper w-formulas, to get a full PRG, we will iterate the restriction of Lemma 27 several times, assigning values to more and more variables. Eventually, we'll stop this recursive process and use a different PRG. Specifically, for the "base case," we'll use a PRG by Lee [22] with minor modifications:

▶ **Lemma 29** ([22]). *For every $w, n \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit ε-PRG for w-proper formulas with seed length*

$$O((w + \log(1/\varepsilon)) \cdot (\log w + \log\log(1/\varepsilon))^2) + \mathrm{poly}(\log\log(n/\varepsilon)).$$

Proof sketch. In short, the PRG is one of the PRGs by Lee [22, Theorem 6], except we replace every δ-biased distribution with a $(\cdot)$-wise δ-biased distribution to optimize the seed length, just like the proofs of Lemma 27 and Lemma 28.

To give a little more detail, let $n' = 2^{8w} \cdot w$; a w-proper formula only reads n' variables. Lee's PRG [22, Theorem 6] is designed to fool *arbitrary-order combinatorial checkerboards*, i.e., parities of functions on disjoint variable sets of size at most w. This class includes w-proper formulas as a special case. Lee's original PRG has seed length

$$O((w + \log(n/\varepsilon)) \cdot (\log w + \log\log(n/\varepsilon))^2).$$

After making suitable replacements, one can show that the seed length is reduced to

$$O((w + \log(n'/\varepsilon)) \cdot (\log w + \log\log(n'/\varepsilon))^2) + \mathrm{poly}(\log\log(n/\varepsilon)).$$

(We omit the full proof, since it repeats much of Lee's analysis [22].) Plugging in the value of n', we get the claimed seed length. ◀

We now give our full PRG for general formulas of width at most $C \log n$. The PRG follows a similar approach to one of the PRGs by Meka et al. [24, full version, Algorithm 3]: iteratively apply the restriction of Lemma 27, but at each step, XOR with the PRG of Lemma 28 in case the formula is long.

▶ **Lemma 30.** *For every $n \in \mathbb{N}$, there is an explicit PRG for read-once* PARITY $\circ$ AND *formulas of width at most $C \log n$ with seed length $O(\log n)$ and error*

$$2^{-\Omega\left(\frac{\log n}{(\log\log n)^3}\right)}.$$

Proof. Define

$$w_0 = \frac{\log n}{(\log\log n)^2}.$$

We recursively define a PRG G_w for w-proper formulas, $w_0 \le w \le C \log n$, as follows.

- (Base case) If $w \leq 2w_0$, then G_w is the (2^{-w_0})-PRG of Lemma 29 based on Lee's work [22].
- (Recursive case) If $w > 2w_0$, sample $X \in \{0,1,\star\}^n$ from the distribution guaranteed by Lemma 27 based on the work in Section 4. Sample $Y \in \{0,1\}^n$ using the PRG of Lemma 28 based on Meka et al.'s work [24]. Recursively sample $G_{\lceil w/2 \rceil}$, and set

$$G_w = Y \oplus (X \circ G_{\lceil w/2 \rceil}).$$

For the analysis, observe first that in the base case $w \leq 2w_0$, G_w fools w-proper formulas with error 2^{-w_0}. Now, for the inductive step, consider some $w > 2w_0$. Assume $G_{\lceil w/2 \rceil}$ fools $\lceil w/2 \rceil$-proper formulas with error $\varepsilon_{\lceil w/2 \rceil}$; we will show that G_w fools w-proper formulas with error ε_w, where

$$\varepsilon_w = \varepsilon_{\lceil w/2 \rceil} + 2^{-\Omega(w/\log w)}.$$

Let f be a w-proper formula, and for brevity, let $G = G_{\lceil w/2 \rceil}$. For the first case, suppose f is long. Any shift of f is also a long w-proper formula, so

$$\begin{aligned}
|\mathbb{E}[f(G_w)] - \mathbb{E}[f]| &= \left| \mathop{\mathbb{E}}_{X,G} \left[\mathop{\mathbb{E}}_{Y} [f(Y \oplus (X \circ G))] \right] - \mathbb{E}[f] \right| \\
&\leq \mathop{\mathbb{E}}_{X,G} \left[\left| \mathop{\mathbb{E}}_{Y} [f(Y \oplus (X \circ G))] - \mathbb{E}[f] \right| \right] \\
&= \mathop{\mathbb{E}}_{X,G} \left[\left| \mathop{\mathbb{E}}_{Y} [f(Y \oplus (X \circ G))] - \mathop{\mathbb{E}}_{U} [f(U \oplus (X \circ G))] \right| \right] \\
&\leq 2^{-w}.
\end{aligned}$$

For the second case, suppose f is short. For each $y \in \{0,1\}^n$, define $f_y(x) = f(y \oplus x)$, another short w-proper formula. Fix $y \sim Y$, and let E be the event that $f_y|_X$ is $(w/2)$-proper, so whether E occurs depends only on X. Then

$$\begin{aligned}
&|\mathbb{E}[(f_y|_X)(G)] - \mathbb{E}[f]| \\
&\quad\leq \left| \mathop{\mathbb{E}}_{X} \left[\mathop{\mathbb{E}}_{G} [(f_y|_X)(G)] \;\middle|\; E \right] - \mathbb{E}[f] \right| + \Pr[\neg E] \\
&\quad\leq \left| \mathop{\mathbb{E}}_{X} \left[\mathop{\mathbb{E}}_{U} [(f_y|_X)(U)] \;\middle|\; E \right] - \mathbb{E}[f] \right| + \varepsilon_{\lceil w/2 \rceil} + \Pr[\neg E] && \text{(Induction)} \\
&\quad\leq \left| \mathop{\mathbb{E}}_{X} \left[\mathop{\mathbb{E}}_{U} [(f_y|_X)(U)] \right] - \mathbb{E}[f] \right| + \varepsilon_{\lceil w/2 \rceil} + 2\Pr[\neg E] \\
&\quad\leq 2^{-\Omega(w/\log w)} + \varepsilon_{\lceil w/2 \rceil} + 2\Pr[\neg E] && \text{(Item 2 of Lemma 27)} \\
&\quad\leq 2^{-\Omega(w/\log w)} + \varepsilon_{\lceil w/2 \rceil} + 2 \cdot 2^{-w} && \text{(Item 3 of Lemma 27).}
\end{aligned}$$

Let ε_w be the final right-hand side, so indeed $\varepsilon_w = \varepsilon_{\lceil w/2 \rceil} + \exp(-\Omega(w/\log w))$. Then

$$\begin{aligned}
|\mathbb{E}[f(G_w)] - \mathbb{E}[f]| &\leq \mathop{\mathbb{E}}_{Y} \left[\left| \mathop{\mathbb{E}}_{X,G} [(f_Y|_X)(G)] - \mathbb{E}[f] \right| \right] \\
&\leq \varepsilon_w.
\end{aligned}$$

Now, let us add up all these errors. Since $w \geq w_0$ always holds, we have $\varepsilon_w \leq \varepsilon_{\lceil w/2 \rceil} + \exp(-\Omega(w_0/\log w_0))$. Starting at $w = C \log n$, we only need to halve w a total of $O(\log\log\log n)$ times to reach the base case $w \leq 2w_0$. Therefore, the total error of $G_{C \log n}$ is bounded by

$$2^{-w_0} + 2^{-\Omega(w_0/\log w_0)} \cdot O(\log\log\log n) = 2^{-\Omega\left(\frac{\log n}{(\log\log n)^3}\right)}.$$

Finally, let us bound the seed length of G_w. In the base case $w \le 2w_0$, by our choice of w_0, the seed length s_w of G_w is bounded by some value $s_{\mathsf{base}} \le O(\log n)$. In the recursive case $w > 2w_0$, the seed length s_w of G_w is bounded by

$$s_w = s_{\lceil w/2 \rceil} + O(w + \log\log n) = s_{\lceil w/2 \rceil} + O(w).$$

The point is that this is essentially a geometric series. More precisely, let c_{seed} be a constant such that $s_w \le s_{\lceil w/2 \rceil} + c_{\mathsf{seed}} \cdot w$ for all $w > 2w_0$. Then by induction, for all $w \ge w_0$, we have

$$s_w \le s_{\mathsf{base}} + 3c_{\mathsf{seed}} w,$$

because

$$\begin{aligned} s_w &\le s_{\lceil w/2 \rceil} + c_{\mathsf{seed}} w \\ &\le s_{\mathsf{base}} + 3c_{\mathsf{seed}} \lceil w/2 \rceil + c_{\mathsf{seed}} w && \text{(Induction)} \\ &< s_{\mathsf{base}} + 3c_{\mathsf{seed}} w. \end{aligned}$$

Therefore, we can take the desired PRG to be $G_{C \log n}$, because $s_{C \log n} \le O(\log n)$, and any read-once PARITY $\circ$ AND formula of width at most $C \log n$ is $(C \log n)$-proper. ◀

5.4 Arbitrary-Error PRGs for Width-$O(\log(n/\varepsilon))$ Formulas

At this point, the main work of proving Theorem 1 is complete. We just need to address three minor issues: small ε, large width, and formulas not of the form PARITY $\circ$ AND. We begin by addressing the case of small ε. Recall that we wish to achieve seed length $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$ for an *arbitrary* error ε. This follows readily by combining the PRG of Lemma 30 with Lee's PRG (Lemma 29).

▶ **Lemma 31.** *For any $n \in \mathbb{N}, \varepsilon > 0$, there is an explict ε-PRG for read-once* PARITY $\circ$ AND *formulas of width at most $\frac{C}{2}\log(n/\varepsilon)$ with seed length*

$$O(\log n + \log(1/\varepsilon) \cdot (\log\log(1/\varepsilon))^5).$$

Proof. Let ε_0 be the error parameter in Lemma 30, so $\varepsilon_0 = \exp(-\Omega(\frac{\log n}{(\log\log n)^3}))$. If $\varepsilon \ge \varepsilon_0$, the PRG of Lemma 30 works, because $\frac{C}{2}\log(n/\varepsilon) < C \log n$. If $\varepsilon < \varepsilon_0$, use Lee's PRG [22], i.e., the ε-PRG of Lemma 29 for $(\frac{C}{2}\log(n/\varepsilon))$-proper formulas, which has seed length

$$O(\log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^2) \le O(\log(1/\varepsilon) \cdot (\log\log(1/\varepsilon))^5).$$ ◀

(In the proof of Lemma 31, we could just as well have used Lee's original PRG [22, Theorem 6] instead of the slightly modified version given by Lemma 29.)

5.5 PRGs for Any Width

In this section, we eliminate the assumption that the maximum width is bounded.

▶ **Lemma 32.** *For all $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for read-once* PARITY $\circ$ AND *formulas on n input bits with seed length*

$$O\left(\log n + \log(1/\varepsilon) \cdot (\log\log(1/\varepsilon))^5\right).$$

Proof. Sample G from the $(\varepsilon/3)$-PRG for formulas of width $\frac{C}{2}\log(3n/\varepsilon)$ guaranteed by Lemma 31. Sample Y from an $(\frac{\varepsilon}{6n})$-biased distribution. Our final PRG outputs

$$H \stackrel{\text{def}}{=} G \oplus Y.$$

To prove that this works, let f be a read-once PARITY $\circ$ AND formula. Write $f = f' \oplus f''$, where every term in f' has width at most $\frac{C}{2}\log(3n/\varepsilon)$ and every term in f'' has width greater than $\frac{C}{2}\log(3n/\varepsilon)$.

Since any shift of a width-w read-once PARITY $\circ$ AND formula is another width-w read-once PARITY $\circ$ AND formula, H fools f' with error $\varepsilon/3$. Meanwhile, since each term f''_i of f'' is a conjunction of more than $\frac{C}{2}\log(3n/\varepsilon)$ literals,

$$\mathbb{E}[f''_i] \leq \left(\frac{\varepsilon}{3n}\right)^{C/2} < \frac{\varepsilon}{6n}.$$

Furthermore, the L_1 norm of any conjunction of literals is 1, and H is $(\frac{\varepsilon}{6n})$-biased, so by Claim 5, $\mathbb{E}[f''_i(H)] < \frac{\varepsilon}{3n}$. Therefore, by the union bound, for either distribution $X \in \{H, U\}$,

$$\mathbb{E}[f''(X)] < \varepsilon/3.$$

This allows us to bound the error of the final PRG as follows:

$$\begin{aligned}
|\mathbb{E}[f(H)] - \mathbb{E}[f]| &\leq |\mathbb{E}[f(H)] - \mathbb{E}[f'(H)]| + |\mathbb{E}[f'(H)] - \mathbb{E}[f']| + |\mathbb{E}[f'] - \mathbb{E}[f]| \\
&\leq \mathbb{E}[|f(H) - f'(H)|] + |\mathbb{E}[f'(H)] - \mathbb{E}[f']| + \mathbb{E}[|f' - f|] \\
&= \mathbb{E}[f''(H)] + |\mathbb{E}[f'(H)] - \mathbb{E}[f']| + \mathbb{E}[f''] \\
&< \varepsilon/3 + \varepsilon/3 + \varepsilon/3 = \varepsilon.
\end{aligned}$$

◀

5.6 Proof of Theorem 1

In this section, we finally complete the proof of Theorem 1 by showing that fooling read-once PARITY $\circ$ AND formulas is sufficient for fooling read-once depth-2 $\mathbf{AC}^0[\oplus]$:

▶ **Lemma 33.** *Let X be a distribution over $\{0,1\}^n$, and let $\varepsilon > 0$. If X fools all read-once* PARITY $\circ$ AND *formulas with error ε, then X fools all read-once depth-2 $\mathbf{AC}^0[\oplus]$ formulas with error 2ε.*

Proof. Let f be a read-once depth-2 $\mathbf{AC}^0[\oplus]$ formula.

For the first case, suppose the output gate of f is $\oplus$. By merging the output gate with any $\oplus$ children and introducing trivial $\wedge$ gates with fan-in 1 as necessary, we see that without loss of generality, every child of the output gate is either $\wedge$ or $\vee$. By de Morgan's laws, it follows that either f or $\neg f$ can be computed by a read-once PARITY $\circ$ AND formula. Either way, this implies that X ε-fools f.

For the second case, suppose the output gate of f is $\wedge$, say $f = \bigwedge_{i=1}^m f_i$. Using the Fourier expansion of the m-input AND function, we get

$$\begin{aligned}
f &= \sum_{I \subseteq [m]} \frac{(-1)^{|I|}}{2^m} \cdot \prod_{i \in I} (-1)^{f_i} \\
&= \sum_{I \subseteq [m]} \frac{(-1)^{|I|}}{2^m} \cdot \left(1 - 2 \cdot \bigoplus_{i \in I} f_i\right).
\end{aligned}$$

By our analysis for the first case, X fools $\bigoplus_{i \in I} f_i$ with error ε. Therefore, by the triangle inequality,

$$\begin{aligned} |\mathbb{E}[f(X)] - \mathbb{E}[f]| &\leq \sum_{I \subseteq [m]} \left| \frac{(-1)^{|I|} \cdot (-2)}{2^m} \right| \cdot \left| \mathbb{E}\left[\left(\bigoplus_{i \in I} f_i \right)(X) \right] - \mathbb{E}\left[\bigoplus_{i \in I} f_i \right] \right| \\ &\leq \sum_{I \subseteq [m]} \frac{2}{2^m} \cdot \varepsilon = 2\varepsilon. \end{aligned}$$

For the final case, suppose the output gate of f is $\vee$. By de Morgan's laws, $\neg f$ can be computed by a read-once depth-2 $\mathbf{AC}^0[\oplus]$ formula with output gate $\wedge$. By our analysis for the second case, X fools $\neg f$ with error 2ε, hence X fools f with the same error. ◀

6 Directions for Further Work

Is there any setting where the iterated restrictions approach (with $\omega(1)$ iterations) can give a pseudorandom generator (or even a hitting set generator) with truly *optimal* seed length $O(\log(n/\varepsilon))$?

Suppose X, X', X'' are three independent small-bias distributions. Does $X + X' \wedge X''$ fool read-once CNFs with optimal seed length $O(\log(n/\varepsilon))$?

References

1. M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in LOGSPACE. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 132–140, New York, NY, USA, 1987. ACM. doi:10.1145/28395.28410.
2. Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5(199-222):1, 1989.
3. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
4. Louay Bazzi and Nagi Nahas. Small-bias is not enough to hit read-once CNF. *Theory of Computing Systems*, 60(2):324–345, February 2017. doi:10.1007/s00224-016-9680-6.
5. Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 0(0):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.
6. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014.
7. J. Brody and E. Verbin. The coin problem and pseudorandomness for branching programs. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 30–39, October 2010. doi:10.1109/FOCS.2010.10.
8. Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Improved algorithms via approximations of probability distributions. *Journal of Computer and System Sciences*, 61(1):81–107, 2000. doi:10.1006/jcss.1999.1695.
9. Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 363–375, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188800.
10. Sitan Chen, Thomas Steinke, and Salil Vadhan. Pseudorandomness for read-once, constant-depth circuits. *arXiv preprint*, 2015. arXiv:1504.04675.
11. Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Annual IEEE 26th Annual Conference on Computational Complexity (CCC)*, pages 221–231. IEEE, 2011.

12 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Approximation, randomization, and combinatorial optimization*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517. Springer, Berlin, 2010. doi:10.1007/978-3-642-15369-3_38.

13 Dean Doron, Pooya Hatami, and William M Hoza. Near-optimal pseudorandom generators for constant-depth read-once formulas. In *34th Computational Complexity Conference (CCC)*, 2019.

14 Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018.

15 Anat Ganor and Ran Raz. Space pseudorandom generators by communication complexity lower bounds. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 28. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

16 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–129. IEEE, 2012.

17 Parikshit Gopalan and Amir Yehudayoff. Inequalities and tail bounds for elementary symmetric polynomial with applications. *arXiv preprint*, 2014. arXiv:1402.3543.

18 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM Journal on Computing*, 47(2):493–523, 2018. doi:10.1137/17M1129088.

19 William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018.

20 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 356–364. ACM, 1994.

21 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 263–272. ACM, New York, 2011. doi:10.1145/1993636.1993672.

22 Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:25, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2019.7.

23 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 167, 2017.

24 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 626–637. ACM, New York, 2019.

25 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

26 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

27 Noam Nisan. RL $\subseteq$ SC. *computational complexity*, 4(1):1–11, March 1994. doi:10.1007/BF01205052.

28 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.

29 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4):Art. 17, 24, 2008. doi:10.1145/1391289.1391291.

30 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670. Springer, 2013.

31 Michael E. Saks and Shiyu Zhou. $BP_HSPACE(S) \subseteq DSPACE(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

32 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2012.

33 Thomas Steinke, Salil Vadhan, and Andrew Wan. Pseudorandomness and Fourier-growth bounds for width-3 branching programs. *Theory of Computing*, 13(12):1–50, 2017. doi:10.4086/toc.2017.v013a012.

Quantum Lower Bounds for Approximate Counting via Laurent Polynomials

Scott Aaronson
University of Texas at Austin, TX, USA
https://www.scottaaronson.com/
aaronson@cs.utexas.edu

Robin Kothari
Microsoft Quantum, Redmond, WA, USA
Microsoft Research, Redmond, WA, USA
http://www.robinkothari.com/
robin.kothari@microsoft.com

William Kretschmer
University of Texas at Austin, TX, USA
https://www.cs.utexas.edu/~kretsch/
kretsch@cs.utexas.edu

Justin Thaler
Georgetown University, Washington, D.C., USA
http://people.cs.georgetown.edu/jthaler/
justin.thaler@georgetown.edu

Abstract

We study quantum algorithms that are given access to trusted and untrusted quantum witnesses. We establish strong limitations of such algorithms, via new techniques based on *Laurent polynomials* (i.e., polynomials with positive and negative integer exponents). Specifically, we resolve the complexity of *approximate counting*, the problem of multiplicatively estimating the size of a nonempty set $S \subseteq [N]$, in two natural generalizations of quantum query complexity.

Our first result holds in the standard Quantum Merlin–Arthur (QMA) setting, in which a quantum algorithm receives an untrusted quantum witness. We show that, if the algorithm makes T quantum queries to S, and also receives an (untrusted) m-qubit quantum witness, then either $m = \Omega(|S|)$ or $T = \Omega\left(\sqrt{N/|S|}\right)$. This is optimal, matching the straightforward protocols where the witness is either empty, or specifies all the elements of S. As a corollary, this resolves the open problem of giving an oracle separation between SBP, the complexity class that captures approximate counting, and QMA.

In our second result, we ask what if, in addition to a membership oracle for S, a quantum algorithm is also given "QSamples"– i.e., copies of the state $|S\rangle = \frac{1}{\sqrt{|S|}} \sum_{i \in S} |i\rangle$ – or even access to a unitary transformation that enables QSampling? We show that, even then, the algorithm needs either $\Theta\left(\sqrt{N/|S|}\right)$ queries or else $\Theta\left(\min\left\{|S|^{1/3}, \sqrt{N/|S|}\right\}\right)$ QSamples or accesses to the unitary.

Our lower bounds in both settings make essential use of Laurent polynomials, but in different ways.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory; Theory of computation → Oracles and decision trees

Keywords and phrases Approximate counting, Laurent polynomials, QSampling, query complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.7

Related Version This paper subsumes preprints https://arxiv.org/abs/1808.02420 and https://arxiv.org/abs/1902.02398 by the first and third authors, respectively.

Funding *Scott Aaronson*: Supported by a Vannevar Bush Fellowship from the US Department of Defense, a Simons Investigator Award, and the Simons "It from Qubit" collaboration.
William Kretschmer: Supported by a Vannevar Bush Fellowship from the US Department of Defense and a Simons Investigator Award.
Justin Thaler: Supported by NSF CAREER award CCF-1845125.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 7; pp. 7:1–7:47

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

Acknowledgements We are grateful to many people: Paul Burchard, for suggesting the problem of approximate counting with queries and QSamples; MathOverflow user "fedja" for letting us include Lemma 22 and Lemma 23; Ashwin Nayak, for extremely helpful discussions, and for suggesting the transformation of linear programs used in our extension of the method of dual polynomials to the Laurent polynomial setting; Thomas Watson, for suggesting the intersection approach to proving an SBP vs. QMA oracle separation; and Patrick Rall, for helpful feedback on writing. JT would particularly like to thank Ashwin Nayak for his warm hospitality and deeply informative discussions during a visit to Waterloo.

1 Introduction

Understanding the power of quantum algorithms has been a central research goal over the last few decades. One success story in this regard has been the discovery of powerful methods that establish limitations on quantum algorithms in the standard setting of *query complexity*. This setting roughly asks, for a specified function f, how many bits of the input must be examined by any quantum algorithm that computes f (see [16] for a survey of query complexity).

A fundamental topic of study in complexity theory is algorithms that are "augmented" with additional information, such as an untrusted witness provided by a powerful prover. For example, the classical complexity class NP is defined this way. In the quantum setting, if we go beyond standard query algorithms, and allow algorithms to receive a quantum state, the model becomes much richer, and we have very few techniques to establish lower bounds for these algorithms. In this paper, we develop such techniques. Our methods crucially use *Laurent polynomials*, which are polynomials with positive and negative integer exponents.

We demonstrate the power of these lower bound techniques by proving optimal lower bounds for the *approximate counting* problem, which captures the following task. Given a nonempty finite set $S \subseteq [N] := \{1, \ldots, N\}$, estimate its cardinality, $|S|$, to within some constant (say, 2) multiplicative accuracy. Approximate counting is a fundamental task with a rich history in computer science. This includes the works of Stockmeyer [54], which showed that approximate counting is in the polynomial hierarchy, and Sinclair and Jerrum [52], which showed the equivalence between approximate counting and approximate sampling that enabled the development of a whole new class of algorithms based on Markov chains. Additionally, approximate counting precisely highlights the limitations of current lower bound techniques for the complexity class QMA (as we explain in Section 1.1).

Formally, we study the following decision version of the problem in this paper:

▶ **Problem 1** (Approximate Counting). *In the* $\mathsf{ApxCount}_{N,w}$ *problem, our goal is to decide whether a nonempty set* $S \subseteq [N]$ *satisfies* $|S| \geq 2w$ *(YES) or* $|S| \leq w$ *(NO), promised that one of these is the case.*

In the query model, the algorithm is given a membership oracle for S: one that, for any $i \in [N]$, returns whether $i \in S$. How many queries must we make, as a function of both N and $|S|$, to solve approximate counting with high probability?

For classical randomized algorithms, it is easy to see that $\Theta(N/|S|)$ membership queries are necessary and sufficient. For quantum algorithms, which can query the membership oracle on superpositions of inputs, Brassard et al. [14, 13] gave an algorithm that makes only $O\left(\sqrt{N/|S|}\right)$ queries. It follows from the optimality of Grover's algorithm (i.e., the BBBV Theorem [10]) that this cannot be improved. Hence, the classical and quantum complexity of approximate counting with membership queries alone is completely understood. In this paper, we study the complexity of approximate counting in models with untrusted and trusted quantum states.

1.1 First result: QMA complexity of approximate counting

Our first result, presented in Section 3, considers the standard Quantum Merlin–Arthur (QMA) setting, in which the quantum algorithm receives an untrusted quantum state (called the witness). This model is the quantum analogue of the classical complexity class NP, and is of great interest in quantum complexity theory. It captures natural problems about ground states of physical systems, properties of quantum circuits and channels, noncommutative constraint satisfaction problems, consistency of representations of quantum systems, and more [12].

In a QMA protocol, a skeptical verifier (Arthur) receives a quantum witness state $|\psi\rangle$ from an all-powerful but untrustworthy prover (Merlin), in support of the claim that $f(x) = 1$. Arthur then needs to verify $|\psi\rangle$, via some algorithm that satisfies the twin properties of *completeness* and *soundness*. That is, if $f(x) = 1$, then there must exist some $|\psi\rangle$ that causes Arthur to accept with high probability, while if $f(x) = 0$, then every $|\psi\rangle$ must cause Arthur to reject with high probability. We call such a protocol a QMA (Quantum Merlin–Arthur) protocol for computing f.

In the query complexity setting, there are two resources to consider: the length of the quantum witness, m, and the number of queries, T, that Arthur makes to the membership oracle. A QMA protocol for f is efficient if both m and T are polylog(N).

The known lower bound technique for QMA

Prior to our work, all known QMA lower bounds used the same proof technique.[1] The technique establishes (and exploits) the complexity class containment QMA $\subseteq$ SBQP, where SBQP is a complexity class that models quantum algorithms with tiny acceptance and rejection probabilities. Specifically, we say that a function f has SBQP query complexity at most k if there exists a k-query quantum algorithm that

- outputs 1 with probability $\geq \alpha$ when $f(x) = 1$, and
- outputs 1 with probability $\leq \alpha/2$ when $f(x) = 0$,

for some α that does not depend on the input (but may depend on the input size). Note that when $\alpha = 2/3$, we recover standard quantum query complexity. But α could be also be exponentially small, which makes SBQP algorithms very powerful.

Nevertheless, one can establish significant limitations on SBQP algorithms, by using a variation of the polynomial method of Beals et al. [8]. If a function f can be evaluated by an SBQP algorithm with k queries, then there exists a real polynomial p of degree $2k$ such that $p(x) \in [0, 1]$ whenever $f(x) = 0$ and $p(x) \geq 2$ whenever $f(x) = 1$. The minimum degree of such a polynomial is also called *one-sided approximate degree* [19].

The relationship between SBQP and QMA protocols is very simple: if f has a QMA protocol that receives an m-qubit witness and makes T queries, then it also has an SBQP algorithm that makes $O(mT)$ queries. This was essentially observed by Marriott and Watrous [36, Remark 3.9] and used by Aaronson [4] to show an oracle relative to which SZK $\not\subset$ QMA.

[1] There is one special case in which it is trivial to lower-bound QMA complexity. Consider the AND_N function on N bits that outputs 1 if and only if all N bits equal 1. For this function, since Merlin wants to convince Arthur that $f(x) = 1$, intuitively there is nothing interesting that Merlin can say to Arthur other than "x is all ones" since that is the only input with $f(x) = 1$. Formally, Arthur can simply create the witness state that an honest Merlin would have sent on the all ones input, and hence Arthur does not need Merlin [45]. For such functions, QMA complexity is the same as standard quantum query complexity.

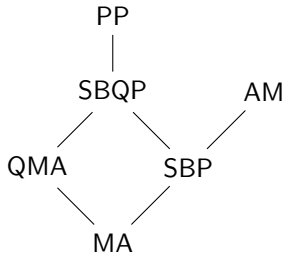


Figure 1 Relationships between complexity classes. An upward line indicates that a complexity class is contained in the one above it relative to all oracles.

Beyond the known lower bound technique for QMA

Our goal is to find a new method of lower bounding QMA, that does not go through SBQP complexity. The natural way to formalize this quest is to find a problem that has an efficient SBQP algorithm, and show that it does not have an efficient QMA protocol. A natural candidate for this is the $\mathsf{ApxCount}_{N,w}$ problem. We know that $\mathsf{ApxCount}_{N,w}$ *does* have a very simple SBQP algorithm of cost 1: the algorithm picks an $i \in [N]$ uniformly at random, and accepts if and only if $i \in S$. Clearly the algorithm accepts with probability greater than $2w/N$ on yes inputs and with probability at most w/N on no inputs.

Our first result establishes that $\mathsf{ApxCount}_{N,w}$ does *not* have an efficient QMA protocol.

▶ **Theorem 2.** *Consider a* QMA *protocol that solves* $\mathsf{ApxCount}_{N,w}$*. If the protocol receives a quantum witness of length* m*, and makes* T *queries to the membership oracle for* S*, then either* $m = \Omega(w)$ *or* $T = \Omega\big(\sqrt{N/w}\big)$*.*

This lower bound proved in Section 3.2 resolves the QMA complexity of $\mathsf{ApxCount}_{N,w}$, as (up to a $\log N$ factor) it matches the cost of two trivial QMA protocols. In the first, Merlin sends $2w$ items claimed to be in S, and Arthur picks a constant number of the items at random and confirms they are all in S with one membership query each. This protocol has witness length $m = O(w \log N)$ (the number of bits needed to specify $2w$ elements out of N) and $T = O(1)$. In the second protocol, Merlin does nothing, and Arthur solves the problem with $T = O\big(\sqrt{N/w}\big)$ quantum queries.

Oracle separation

Our result also yields new oracle separations. The approximate counting problem is complete for the complexity class SBP [11], which is sandwiched between MA (Merlin–Arthur) and AM (Arthur–Merlin). The class SBQP (discussed above), first defined by Kuperberg [33], is a quantum analogue of SBP that contains both SBP and QMA.

By the usual connection between oracle separations and query complexity lower bounds, Theorem 2 implies an oracle separation between SBP and QMA – i.e., there exists an oracle A such that $\mathsf{SBP}^A \not\subset \mathsf{QMA}^A$ (see Corollary 20). Prior to our work, it was known that there exist oracles A, B such that $\mathsf{SBP}^A \not\subset \mathsf{MA}^A$ [11] and $\mathsf{AM}^B \not\subset \mathsf{QMA}^B$, which follows from $\mathsf{AM}^B \not\subset \mathsf{PP}^B$ [56], but the relation between SBP and QMA remained elusive.[2] Figure 1 shows the known inclusion relations among these classes (all of which hold relative to all oracles).

[2] It is interesting to note that in the non-relativized world, under plausible derandomization assumptions [38], we have NP = MA = SBP = AM. In this scenario, all these classes are equal, and all are contained in QMA.

Previous techniques were inherently unable to establish this oracle separation for the reason stated above: all existing QMA lower bounds intrinsically apply to SBQP as well. Since SBP is contained in SBQP, prior techniques cannot establish $\mathsf{SBP}^A \not\subset \mathsf{QMA}^A$, or even $\mathsf{SBQP}^A \not\subset \mathsf{QMA}^A$, for any oracle A. Our analysis also yields the first oracle with respect to which SBQP is not closed under intersection.

Proof overview

To get around the issue of $\mathsf{ApxCount}_{N,w}$ being in SBQP, we use a clever strategy that was previously used by Göös et al. [26], and that was suggested to us by Thomas Watson (personal communication). Our strategy exploits a structural property of QMA: the fact that QMA is closed under intersection, but (at least relative to oracles, and as we'll show) SBQP is not.

Given a function f, let $\mathsf{AND}_2 \circ f$ be the AND of two copies of f on separate inputs.[3] Then if f has small QMA query complexity, it's not hard to see that $\mathsf{AND}_2 \circ f$ does as well: Merlin simply sends witnesses corresponding to both inputs; then Arthur checks both of them independently. While it's not completely obvious, one can verify that a dishonest Merlin would gain nothing by entangling the two witness states. Hence if $\mathsf{ApxCount}_{N,w}$ had an efficient QMA protocol, then so would $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$, with the witness size and query complexity increasing by only a constant factor.

By contrast, even though $\mathsf{ApxCount}_{N,w}$ does have an efficient SBQP algorithm, we will show that $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ does not. This is the technical core of our proof and proved in Section 3.1.

▶ **Theorem 3.** *Consider an* SBQP *algorithm for* $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ *that makes* T *queries to membership oracles for the two instances of* $\mathsf{ApxCount}_{N,w}$. *Then* $T = \Omega\left(\min\{w, \sqrt{N/w}\}\right)$.

Theorem 3 is quantitatively optimal, as we'll exhibit a matching SBQP upper bound. Combined with the connection between QMA and SBQP, Theorem 3 immediately implies a QMA lower bound for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$, and by extension $\mathsf{ApxCount}_{N,w}$ itself. However, this QMA lower bound is not quantitatively optimal. To obtain the optimal bound of Theorem 2, we exploit additional analytic properties of the SBQP protocols that are derived from QMA protocols.

At a high level, the proof of Theorem 3 assumes that there's an efficient SBQP algorithm for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$. This assumption yields a low-degree one-sided approximating polynomial for the problem in $2N$ Boolean variables, where N variables come from each $\mathsf{ApxCount}_{N,w}$ instance. We then symmetrize the polynomial (using the standard Minsky–Papert symmetrization argument [39]) to obtain a bivariate polynomial in two variables x and y that represent the Hamming weight of the original instances.[4] This yields a polynomial $p(x, y)$ that for *integer pairs* x, y (also called lattice points) satisfies $p(x, y) \in [0, 1]$ when either $x \in \{0, \ldots, w\}$ and $y \in \{0, \ldots, w\} \cup \{2w, \ldots, N\}$, or (symmetrically) $y \in \{0, \ldots, w\}$ and $x \in \{0, \ldots, w\} \cup \{2w, \ldots, N\}$. If both $x \in \{2w, \ldots, N\}$ and $y \in \{2w, \ldots, N\}$, then $p(x, y) \geq 2$. This polynomial p is depicted in Figure 2.

[3] Because we focus on lower bounds, for a promise problem f (such as $\mathsf{ApxCount}_{N,w}$), we take the promise for $\mathsf{AND}_2 \circ f$ to be that both instances of f must satisfy f's promise. Then, any lower bound also applies to more relaxed definitions, such as only requiring one of the two instances to be in the promise.

[4] The term "symmetrization" originally referred to the process of averaging a multivariate polynomial over permutations of its inputs to obtain a symmetric polynomial. More recently, authors have used "symmetrization" more generally to refer to any method for turning a multivariate polynomial into a

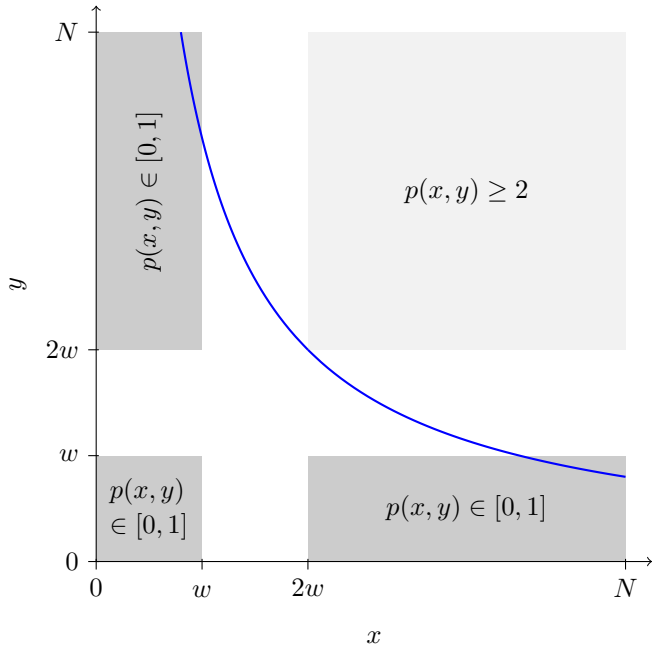


Figure 2 The behavior of the (Minsky–Papert symmetrized) bivariate polynomial $p(x, y)$ at integer points (x, y) in the proof of Theorem 3. The polynomial q obtained by erase-all-subscripts symmetrization is not depicted. We later restrict q to a hyperbola similar to the one drawn in blue.

One difficulty is that we have a guarantee on the behavior of p at lattice points only, whereas the rest of our proof requires precise control over the polynomial even at non-integer points. We ignore this issue for now and assume that $p(x, y) \geq 2$ for all real values $x, y \in [2w, N]$, and $p(x, y) \in [0, 1]$ whenever $x \in [0, w]$ and $y \in [2w, N]$ or vice versa. We outline how we address integrality issues one paragraph hence.

The key remaining difficulty is that we want to lower-bound the degree of a bivariate polynomial, but almost all known lower bound techniques apply only to univariate polynomials. To address this, we introduce a new technique to reduce the number of variables (from 2 to 1) in a degree-preserving way: we pass a *hyperbola* through the xy plane (see Figure 2) and consider the polynomial p restricted to the hyperbola. Doing so gives us a new univariate *Laurent* polynomial $\ell(t) = p(2wt, 2w/t)$, whose positive and negative degree is at most $\deg(p)$. This Laurent polynomial has an additional symmetry, which stems from the fact that $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ is the AND of two identical problems (namely, $\mathsf{ApxCount}_{N,w}$). We leverage this symmetry to view $\ell(t)$, a Laurent polynomial in t, as an ordinary univariate polynomial r in $t + 1/t$ of degree $\deg(p)$. We know that $r(2) = \ell(1) = p(2w, 2w) \geq 2$, while for all $k \in [2.5, N/w + w/N]$, we know that $r(k) \in [0, 1]$. It then follows from classical results in approximation theory that this univariate polynomial must have degree $\Omega\big(\sqrt{N/w}\big)$.

Returning to integrality issues, to obtain a polynomial whose behavior we can control at non-integer points, we use a different symmetrization argument (dating back at least to work of Shi [51]) that we call "erase-all-subscripts" symmetrization (see Lemma 12). This symmetrization yields a bivariate polynomial q of the same degree as p that is bounded

univariate one in a degree non-increasing manner (see, e.g., [48, 49]). In this paper, we use the term "symmetrization" in this more general sense.

in $[0,1]$ at all *real-valued* inputs in $[0,N] \times [0,N]$. However, while we have more control over q's values at non-integer inputs relative to p, we have *less* control over q's values at integer inputs relative to p, and this introduces additional challenges. (These challenges are not merely annoyances; they are why the SBQP complexity of $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ is $T = \Theta\big(\min\{w, \sqrt{N/w}\}\big)$, and not $\Theta\big(\sqrt{N/w}\big)$). Ultimately, both types of symmetrization play an important role in our analysis, as we use p to bound q when the polynomials have degree $o(w)$, using tools from approximation theory and Chernoff bounds.

1.2 Second result: Approximate counting with quantum samples

Our second result resolves the complexity of $\mathsf{ApxCount}_{N,w}$ in a different generalization of the quantum query model, in which the algorithm is given access to certain (trusted) quantum states.

Quantum samples

In practice, when trying to estimate the size of a set $S \subseteq [N]$, often we can do more than make membership queries to S. At the least, often we can efficiently generate nearly uniform *samples* from S, for instance by using Markov Chain Monte Carlo techniques. To give two examples, if S is the set of perfect matchings in a bipartite graph, or the set of grid points in a high-dimensional convex body, then we can efficiently sample S using the seminal algorithms of Jerrum, Sinclair, and Vigoda [29] or of Dyer, Frieze, and Kannan [21], respectively.

The natural quantum generalization of uniform sampling from a set S is *QSampling* S – a term coined in 2003 by Aharonov and Ta-Shma [7], and which means that we can approximately prepare the uniform superposition

$$|S\rangle := \frac{1}{\sqrt{|S|}} \sum_{i \in S} |i\rangle \tag{1}$$

via a polynomial-time quantum algorithm (where "polynomial" here means $\mathrm{polylog}(N)$). Because we need to uncompute garbage, the ability to prepare $|S\rangle$ as a coherent superposition is a more stringent requirement than the ability to classically sample from S. Indeed, Aharonov and Ta-Shma [7] showed that the ability to QSample lends considerable power: all problems in the complexity class SZK (which contains problems that are widely believed be hard on average [24, 25, 37, 23, 44]) can be efficiently reduced to the task of *QSampling* some set that can be *classically* sampled in polynomial time. To be clear, QSampling supposes that the algorithm is given trusted copies of $|S\rangle$; unlike in the QMA setting, the state need not be "verified" by the algorithm.

On the other hand, Aharonov and Ta-Shma [7], and Grover and Rudolph [27], observed that many interesting sets S can be efficiently QSampled as well.[5]

[5] In particular, this holds for all sets S such that we can approximately count not only S itself, but also the restrictions of S obtained by fixing bits of its elements. So in particular, the set of perfect matchings in a bipartite graph, and the set of grid points in a convex body, can both be efficiently QSampled. There are other sets that can be QSampled but not because of this reduction. A simple example would be a set S such that $|S| \geq \frac{N}{\mathrm{polylog}N}$: in that case we can efficiently prepare $|S\rangle$ using postselection, but approximately counting S's restrictions might be hard.

QSampling via unitaries

In many applications (such as when S is the set of perfect matchings in a bipartite graph or grid points in a convex body), the reason an algorithm can QSample S is because it is possible to efficiently construct a quantum circuit implementing a unitary operator U that prepares the state $|S\rangle$. Access to this unitary U potentially conveys substantially more power than QSampling alone. For example, access to U conveys (in a black box manner) the ability not only to QSample, but also to perform reflections about $|S\rangle$: that is, to apply the unitary transformation

$$\mathcal{R}_S := \mathbb{1} - 2|S\rangle\langle S|, \tag{2}$$

which has eigenvalue -1 for $|S\rangle$ and eigenvalue $+1$ for all states orthogonal to $|S\rangle$. More concretely, let U be the unitary that performs the map $U|0\rangle = |S\rangle$, for some canonical starting state $|0\rangle$. Since we know the circuit U, we can also implement $U^\dagger$, by reversing the order of all the gates and replacing all the gates with their adjoints. Then $\mathcal{R}_S$ is simply

$$\mathcal{R}_S = \mathbb{1} - 2|S\rangle\langle S| = U\left(\mathbb{1} - 2|0\rangle\langle 0|\right)U^\dagger. \tag{3}$$

Note that *a priori*, QSamples and reflections about $|S\rangle$ could be incomparable resources; it is not obvious how to simulate either one using the other. On the other hand, it is known how to apply a quantum channel that is ε-close to $\mathcal{R}_S$ (in the diamond norm) using $\Theta(1/\varepsilon)$ copies of $|S\rangle$ [34, 30].

Access to a quantum circuit computing U also permits an algorithm to efficiently apply U on inputs that do not produce the state $|S\rangle$, to construct a controlled version of U, etc.

Results

As previously mentioned, Aharonov and Ta-Shma [7] showed that the ability to QSample lends considerable power, including the ability to efficiently solve SZK-complete problems. It is natural to ask just how much power the ability to QSample conveys. In particular, can one extend the result of Aharonov and Ta-Shma [7] from any problem in SZK to any problem in SBP? Equivalently stated, can one solve approximate counting efficiently, using *any* combination of polylog(N) queries and applications of a unitary U that permits QSampling?[6] In this work, we show that the answer is no. We begin by focusing on the slightly simplified setting where the algorithm is only permitted to perform membership queries, QSamples, and reflections about the state $|S\rangle$.

▶ **Theorem 4.** *Let Q be a quantum algorithm that makes T queries to the membership oracle for S, and uses a total of R copies of $|S\rangle$ and reflections about $|S\rangle$. If Q decides whether $|S| = w$ or $|S| = 2w$ with high probability, promised that one of those is the case, then either*

$$T = \Omega\left(\sqrt{\frac{N}{w}}\right) \qquad \text{or} \qquad R = \Omega\left(\min\left\{w^{1/3}, \sqrt{\frac{N}{w}}\right\}\right). \tag{4}$$

This is proved in Section 4.4. So if (for example) we set $w := N^{3/5}$, then any quantum algorithm must either query S, or use the state $|S\rangle$ or reflections about $|S\rangle$, at least $\Omega(N^{1/5})$ times. Put another way, Theorem 4 means that unless w is very small ($w \leq \text{polylog}(N)$))

[6] We thank Paul Burchard (personal communication) for bringing this question to our attention.

or extremely large ($w \geq N/\text{polylog}(N)$), the ability to QSample S, reflect about $|S\rangle$, and determine membership in S is not sufficient to approximately count S efficiently. Efficient quantum algorithms for approximate counting will have to leverage additional structure of S, beyond the ability to QSample, reflect about $|S\rangle$, and determine membership in S.

In Theorem 31 of Section 4.6, we then strengthen Theorem 4 to hold not only against algorithms that can QSample and reflect about $|S\rangle$ (in addition to performing membership queries to S), but also against all algorithms that are given access to a specific unitary U that conveys the power to QSample and reflect about $|S\rangle$.[7]

Finally, we prove that the lower bounds in Theorem 4 and Theorem 31 are optimal. As mentioned before, Brassard et al. [14] gave a quantum algorithm to solve the problem using $T = O(\sqrt{N/w})$ queries alone, which proves the optimality of the lower bound on the number of queries. On the other hand, it's easy to solve the problem using $O(\sqrt{w})$ copies of $|S\rangle$ alone, by simply measuring each copy of $|S\rangle$ in the computational basis and then searching for birthday collisions. Alternately, we can solve the problem using $O\left(\frac{N}{w}\right)$ copies of $|S\rangle$ alone, by projecting onto the state $|\psi\rangle = \frac{1}{\sqrt{N}}(|1\rangle + \cdots + |N\rangle)$ or its orthogonal complement. This measurement succeeds with probability $|\langle S|\psi\rangle|^2 = \frac{|S|}{N}$, so we can approximate $|S|$ by simply counting how many measurements succeed.

In Section 4.2 we improve on these algorithms by using samples *and* reflections, and thereby establish that Theorem 4 and Theorem 31 are tight.

▶ **Theorem 5.** *There is a quantum algorithm that solves* $\mathsf{ApxCount}_{N,w}$ *with high probability using* R *copies of* $|S\rangle$ *and reflections about* $|S\rangle$*, where* $R = O\left(\min\left\{w^{1/3}, \sqrt{\frac{N}{w}}\right\}\right)$.

The Laurent polynomial method

In our view, at least as interesting as Theorem 4 is the technique used to achieve it. In 1998, Beals et al. [8] famously observed that, if a quantum algorithm Q makes T queries to an input X, then Q's acceptance probability can be written as a real multilinear polynomial in the bits of X, of degree at most $2T$. And thus, crucially, if we want to *rule out* a fast quantum algorithm to compute some function $f(X)$, then it suffices to show that any real polynomial p that approximates f pointwise must have high degree. This general transformation, from questions about quantum algorithms to questions about polynomials, has been used to prove many results that were not known otherwise at the time, including the quantum lower bound for the collision problem [1, 6] and the first direct product theorems for quantum search [2, 31].

In our case, even in the simpler model with only queries and samples (and no reflections), the difficulty is that the quantum algorithm starts with many copies of the state $|S\rangle$. As a consequence of this – and specifically, of the $1/\sqrt{|S|}$ normalizing factor in $|S\rangle$ – when we write the average acceptance probability of our algorithm as a function of $|S|$, we find that we get a *Laurent polynomial*: a polynomial that can contain both positive and negative integer powers of $|S|$. The degree of this polynomial (the highest power of $|S|$) encodes the sum of the number of queries, the number of copies of $|S\rangle$, and the number of uses of $\mathcal{R}_S$, while the "anti-degree" (the highest power of $|S|^{-1}$) encodes the sum of the number of copies of $|S\rangle$ and number of uses of $\mathcal{R}_S$. This is described more precisely in Section 4.1. We're thus faced with the task of lower-bounding the degree and the anti-degree of a Laurent polynomial that's bounded in $[0, 1]$ at integer points and that encodes the approximate counting problem.

[7] To be precise, the unitary U to which the lower bound of Theorem 31 applies maps a canonical starting state to $|S\rangle|S\rangle$. As we explain in Section 4.6, such a unitary suffices to implement QSampling, reflections about $|S\rangle$, etc., since the register containing the second copy of $|S\rangle$ can simply be ignored.

We then lower bound the degree of Laurent polynomials that approximate $\mathsf{ApxCount}_{N,w}$, showing that degree $\Omega\big(\min\big\{w^{1/3}, \sqrt{N/w}\big\}\big)$ is necessary. We give two very different lower bound arguments. The first approach, which we call the "explosion argument," is shorter but yields suboptimal lower bounds, whereas the second approach using "dual polynomials" yields the optimal lower bound.

There are two aspects of this that we find surprising: first, that Laurent polynomials appear at all, and second, that they seem to appear in a completely different way than they appear in our other result about QMA (Theorem 3), despite the close connection between the two statements. For Theorem 4, Laurent polynomials are needed just to describe the quantum algorithm's acceptance probability, whereas for Theorem 3, ordinary (bivariate) polynomials sufficed to describe this probability; Laurent polynomials appeared only when we restricted a bivariate polynomial to a hyperbola in the plane. In any case, the coincidence suggests that the "Laurent polynomial method" might be useful for other problems as well.[8]

Before describing our techniques at a high level, observe that there are *rational* functions[9] of degree $O(\log(N/w))$ that approximate $\mathsf{ApxCount}_{N,w}$. This follows, for example, from Aaronson's PostBQP = PP theorem [3], or alternately from the classical result of Newman [41] that for any $k > 0$, there is a rational polynomial of degree $O(k)$ that pointwise approximates the sign function on domain $[-n,-1] \cup [1,n]$ to error $1 - n^{-1/k}$. Thus, our proof relies on the fact that Laurent polynomials are an extremely special kind of rational function.

We also remark that in the randomized classical setting, the complexity of $\mathsf{ApxCount}_{N,w}$ with queries and uniform (classical) samples is easily characterized without such powerful techniques. Either $O(N/w)$ queries or $O(\sqrt{w})$ samples are sufficient, and furthermore either $\Omega(N/w)$ queries or $\Omega(\sqrt{w})$ samples are necessary. For completeness, we provide a sketch of these bounds in Section 4.5.

Overview of the explosion argument

Our first proof (in Section 4.3) uses an "explosion argument" that, as far as we know, is new in quantum query complexity. We separate out the purely positive degree[10] and purely negative degree parts of our Laurent polynomial as $q(|S|) = u(|S|) + v(1/|S|)$, where u and v are ordinary polynomials. We then show that, if u and v both have low enough degree, namely $\deg(u) = o\big(\sqrt{N/w}\big)$ and $\deg(v) = o\left(w^{1/4}\right)$, then we get "unbounded growth" in their values. That is: for approximation theory reasons, either u or v must attain large values, far outside of $[0,1]$, at some integer values of $|S|$. But that means that, for q itself to be bounded in $[0,1]$ (and thus represent a probability), the other polynomial must *also* attain large values. And that, in turn, will force the first polynomial to attain even larger values, and so on forever – thereby proving that these polynomials could not have existed.

Overview of the method of dual polynomials

Our second argument (in Section 4.4) obtains the (optimal) lower bound stated in Theorem 4, via a novel adaptation of the so-called *method of dual polynomials*.

---

[8] Since writing this, a third application of the Laurent polynomial method was discovered by the third author [32]: a simple proof that the AND-OR tree $\mathsf{AND}_m \circ \mathsf{OR}_n$ has approximate degree $\widetilde{\Omega}(\sqrt{mn})$.

[9] A rational function of degree d is of the form $\frac{p(x)}{q(x)}$, where p and q are both real polynomials of degree at most d.

[10] Throughout this paper we allow any "purely positive degree" Laurent polynomial and any "purely negative degree" Laurent polynomial to include a constant (degree zero) term.

With this method, to lower-bound the approximate degree of a Boolean function f, one exhibits an explicit *dual polynomial* ψ for f, which is a dual solution to a certain linear program. Roughly speaking, a dual polynomial ψ is a function mapping the domain of f to $\mathbb{R}$ that is (a) uncorrelated with any polynomial of degree at most d, and (b) well-correlated with f.

Approximating a univariate function g via low-degree Laurent polynomials is also captured by a linear program, but the linear program is more complicated because Laurent polynomials can have negative-degree terms. We analyze the value of this linear program in two steps.

In Step 1, we transform the linear program so that it refers only to ordinary polynomials rather than Laurent polynomials. Although simple, this transformation is crucial, as it lets us bring techniques developed for ordinary polynomials to bear on our goal of proving Laurent polynomial degree lower bounds.

In Step 2, we explicitly construct an optimal dual witness to the transformed linear program from Step 1. We do so by first identifying two weaker dual witnesses: ψ_1, which witnesses that *ordinary* (i.e., purely positive degree) polynomials encoding approximate counting require degree at least $\Omega\big(\sqrt{N/w}\big)$, and ψ_2, which witnesses that purely negative degree polynomials encoding approximate counting require degree $\Omega(w^{1/3})$. The first witness is derived from prior work of Bun and Thaler [18] (who refined earlier work of Špalek [53]), while the second builds on a non-constructive argument of Zhandry [57].

Finally, we show how to "glue together" ψ_1 and ψ_2, to get a dual witness ψ showing that any general Laurent polynomial that encodes approximate counting must have either positive degree $\Omega\big(\sqrt{N/w}\big)$ or negative degree $\Omega(w^{1/3})$.

Overview of the upper bound

To recap, Theorem 4 shows that any quantum algorithm for $\mathsf{ApxCount}_{N,w}$ needs either $\Theta(\sqrt{N/w})$ queries or $\Theta\big(\min\{w^{1/3}, \sqrt{N/w}\}\big)$ samples and reflections. Since we know from the work of Brassard, Høyer, Tapp [14] that the problem can be solved with $O(\sqrt{N/w})$ queries alone, it remains only to show the matching upper bound using samples and reflections, which we describe in Section 4.2.

First we describe a simple algorithm that uses $O(\sqrt{N/w})$ samples and reflections. If we take one copy of $|S\rangle$, and perform a projective measurement onto $|\psi\rangle = \frac{1}{\sqrt{N}}(|1\rangle + \cdots + |N\rangle)$ or its orthogonal complement, the measurement will succeed with probability $|\langle S|\psi\rangle|^2 = |S|/N$. Thus $O(N/w)$ repetitions of this will allow us to distinguish the probabilities w/N and $2w/N$. We can improve this by using amplitude amplification [13] and only make $O(\sqrt{N/w})$ repetitions.

Our second algorithm solves the problem with $O(w^{1/3})$ reflections and samples and is based on the quantum collision-finding algorithm [15]. We first use $O(w^{1/3})$ copies of $|S\rangle$ to learn $w^{1/3}$ distinct elements in S. We now know a fraction of elements in S, and this fraction is either $w^{-2/3}$ or $\frac{1}{2}w^{-2/3}$. We then use amplitude amplification (or quantum counting) to distinguish these two cases, which costs $O(w^{1/3})$ repetitions, where each repetition uses a reflection about $|S\rangle$.

2 Preliminaries

In this section we introduce some definitions and known facts about polynomials and complexity classes.

2.1 Approximation theory

We will use several results from approximation theory, each of which has previously been used (in some form) in other applications of the polynomial method to quantum lower bounds. We start with the basic inequality of A.A. Markov [35].

▶ **Lemma 6** (Markov). *Let p be a real polynomial, and suppose that*

$$\max_{x,y\in[a,b]} |p(x) - p(y)| \leq H. \tag{5}$$

Then for all $x \in [a,b]$, we have

$$|p'(x)| \leq \frac{H}{b-a}\deg(p)^2, \tag{6}$$

where $p'(x)$ is the derivative of p at x.

We'll also need a bound that was explicitly stated by Paturi [43], and which amounts to the fact that, among all degree-d polynomials that are bounded within a given range, the Chebyshev polynomials have the fastest growth outside that range.

▶ **Lemma 7** (Paturi). *Let p be a real polynomial, and suppose that $|p(x)| \leq 1$ for all $|x| \leq 1$. Then for all $x \leq 1+\mu$, we have*

$$|p(x)| \leq \exp\left(2\deg(p)\sqrt{2\mu+\mu^2}\right). \tag{7}$$

We now state a useful corollary of Lemma 7, which says (in effect) that slightly shrinking the domain of a low-degree real polynomial can only modestly shrink its range.

▶ **Corollary 8.** *Let p be a real polynomial of degree d, and suppose that*

$$\max_{x,y\in[a,b]} |p(x) - p(y)| \geq H. \tag{8}$$

Let $\varepsilon \leq \frac{1}{100d^2}$ and $a' := a + \varepsilon(b-a)$. Then

$$\max_{x,y\in[a',b]} |p(x) - p(y)| \geq \frac{H}{2}. \tag{9}$$

Proof. Suppose by contradiction that

$$|p(x) - p(y)| < \frac{H}{2} \tag{10}$$

for all $x, y \in [a', b]$. By affine shifts, we can assume without loss of generality that $|p(x)| < \frac{H}{4}$ for all $x \in [a', b]$. Then by Lemma 7, for all $x \in [a, b]$ we have

$$|p(x)| < \frac{H}{4} \cdot \exp\left(2d\sqrt{2\left(\frac{1}{1-\varepsilon}-1\right) + \left(\frac{1}{1-\varepsilon}-1\right)^2}\right) \leq \frac{H}{2}. \tag{11}$$

But this violates the hypothesis. ◀

We will also need a bound that relates the range of a low-degree polynomial on a discrete set of points to its range on a continuous interval. The following lemma generalizes a result due to Ehlich and Zeller [22] and Rivlin and Cheney [46], who were interested only in the case where the discrete points are evenly spaced.

▶ **Lemma 9.** *Let p be a real polynomial of degree at most $\sqrt{k}$, and let $0 = z_1 < \cdots < z_M = k$ be a list of points such that $z_{i+1} - z_i \leq 1$ for all i (the simplest example being the integers $0, \ldots, k$). Suppose that*

$$\max_{x,y \in [0,k]} |p(x) - p(y)| \geq H. \tag{12}$$

Then

$$\max_{i,j} |p(z_i) - p(z_j)| \geq \frac{H}{2}. \tag{13}$$

Proof. Suppose by contradiction that

$$|p(z_i) - p(z_j)| < \frac{H}{2} \tag{14}$$

for all i, j. By affine shifts, we can assume without loss of generality that $|p(z_i)| < \frac{H}{4}$ for all i. Let

$$c := \max_{x \in [0,k]} \frac{|p(x)|}{H/4}. \tag{15}$$

If $c \leq 1$, then the hypothesis clearly fails, so assume $c > 1$. Suppose that the maximum, $|p(x)| = \frac{cH}{4}$, is achieved between z_i and z_{i+1}. Then by basic calculus, there exists an $x^* \in [z_i, z_{i+1}]$ such that

$$|p'(x^*)| > \frac{2(c-1)}{z_{i+1} - z_i} \cdot \frac{H}{4} \geq \frac{(c-1)H}{2}. \tag{16}$$

So by Lemma 6,

$$\frac{(c-1)H}{2} < \frac{cH/4}{k} \deg(p)^2. \tag{17}$$

Solving for c, we find

$$c < \frac{2k}{2k - \deg(p)^2} \leq 2. \tag{18}$$

But if $c < 2$, then $\max_{x \in [0,k]} |p(x)| < \frac{H}{2}$, which violates the hypothesis. ◀

We also use a related inequality due to Coppersmith and Rivlin [20] that bounds a polynomial on a continuous interval in terms of a bound on a discrete set of points, but now with the weaker assumption that the degree is at most k, rather than $\sqrt{k}$. This gives a substantially weaker bound.

▶ **Lemma 10** (Coppersmith and Rivlin)**.** *Let p be a real polynomial of degree at most k, and suppose that $|p(x)| \leq 1$ for all integers $x \in \{0, 1, \ldots, k\}$. Then there exist universal constants a, b such that for all $x \in [0, k]$, we have*

$$|p(x)| \leq a \cdot \exp\left(b \deg(p)^2 / k\right). \tag{19}$$

2.2 Symmetric polynomials

Univariate symmetrizations

Our starting point is the well-known *symmetrization lemma* of Minsky and Papert [39] (see also Beals et al. [8] for its application to quantum query complexity), by which we can often reduce questions about multivariate polynomials to questions about univariate ones.

▶ **Lemma 11** (Minsky–Papert symmetrization). *Let $p : \{0,1\}^N \to \mathbb{R}$ be a real multilinear polynomial of degree d, and let $q : \{0,1,\ldots,N\} \to \mathbb{R}$ be defined as*

$$q(k) := \mathbb{E}_{|X|=k}[p(X)]. \tag{20}$$

Then q can be written as a real polynomial in k of degree at most d.

We now introduce a different, lesser known notion of symmetrization, which we call the *erase-all-subscripts* symmetrization for reasons to be explained shortly. This symmetrization previously appeared in [51] under the name "linearization," and it is also equivalent to the noise operator used in analysis of Boolean functions [42, Definition 2.46].

▶ **Lemma 12** (Erase-all-subscripts symmetrization). *Let $p : \{0,1\}^N \to \mathbb{R}$ be a real multilinear polynomial of degree d, and for any real number $k \in [0,1]$, let M_k denote the distribution over $\{0,1\}^N$, wherein each coordinate is selected independently to be 1 with probability k. Let $q : [0,1] \to \mathbb{R}$ be defined as*

$$q(k) := \mathbb{E}_{X \sim M_k}[p(X)]. \tag{21}$$

Then q can be written as a real polynomial in k of degree at most d.

Proof. (see, for example, [47, Proof of Theorem 3]). Given the multivariate polynomial expansion of p, we can obtain q easily just by "erasing all the subscripts in each variable". For example, if $p(x_1,x_2,x_3) = 2x_1x_2 + x_2x_3 + x_2$, we replace every x_i with k to obtain $q(k) = 2k \cdot k + k \cdot k + k = 3k^2 + k$. This follows from linearity of expectation along with the fact that M_k is defined to be the product distribution wherein each coordinate has expected value k. ◀

We highlight the following key difference between Minsky–Papert symmetrization and the erase-all-subscripts symmetrization. Let $p : \{0,1\}^N \to [0,1]$ be a real multivariate polynomial whose evaluations at Boolean inputs are in $[0,1]$, i.e., for all $x \in \{0,1\}^n$, we have $p(x) \in [0,1]$. If q is the erase-all-subscripts symmetrization of p, then q takes values in $[0,1]$ at all *real-valued* inputs in $[0,1]$: $q(k) \in [0,1]$ for all $k \in [0,1]$. If q is the Minsky–Papert symmetrization of p, then it is only guaranteed to take values in $[0,1]$ at *integer-valued* inputs in $[0,N]$, i.e., $q(k) \in [0,1]$ is only guaranteed to hold at $k \in \{0,1,\ldots,N\}$. This is the main reason we use erase-all-subscripts symmetrization in this work.

Bivariate symmetrizations

In this paper, it will be convenient to consider bivariate versions of both Minsky–Papert and erase-all-subscripts symmetrization, and their applications to oracle separations. To this end, define $X \in \{0,1\}^N$, the "characteristic string" of the set $S \subseteq [N]$, by $x_i = 1$ if $i \in S$ and $x_i = 0$ otherwise. Let $\mathcal{O}_S$ denote the unitary that performs a membership query to S, defined as

$$\mathcal{O}_S |i\rangle |b\rangle = (1 - 2bx_i) |i\rangle |b\rangle \tag{22}$$

for any index $i \in [N]$ and bit $b \in \{0,1\}$.

Because we study oracle intersection problems, it is often convenient to think of an algorithm as having access to *two* oracles, wherein the first bit in the oracle register selects the choice of oracle. As a consequence, we need a slight generalization of a now well-established fact in quantum complexity: that the acceptance probability of a quantum algorithm with an oracle can be expressed as a polynomial in the bits of the oracle string.

▶ **Lemma 13** (Symmetrization with two oracles). *Let $Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}}$ be a quantum algorithm that makes T queries to a pair of membership oracles for sets $S_0, S_1 \subseteq [N]$. Let D_μ denote the distribution over subsets of $[N]$ wherein each element is selected independently with probability $\frac{\mu}{N}$. Then there exist bivariate real polynomials $q(s,t)$ and $p(x,y)$ of degree at most $2T$ satisfying:*

$$\text{for all real numbers } s,t \in [0,N], \quad q(s,t) = \mathbb{E}_{\substack{S_0\sim D_s,\\ S_1\sim D_t}}\left[\Pr[Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}} \text{ accepts}]\right], \text{ and}$$
$$\text{for all integers } x,y \in \{0,1,\ldots,N\}, \quad p(x,y) = \mathbb{E}_{\substack{|S_0|=x,\\ |S_1|=y}}\left[\Pr[Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}} \text{ accepts}]\right].$$

Proof. Take $X = X_0|X_1$ to be the concatenation of the characteristic strings of the two oracles, and let $S \subseteq [2N]$ be such that X is the characteristic string of S. Then, Lemma 4.2 of Beals et al. [8] tells us that there is a real multilinear polynomial $r(X)$ of degree at most $2T$ in the bits of X such that $r(X) = \Pr[Q^{\mathcal{O}_S} \text{ accepts}]$.

Observe that r has a meaningful probabilistic interpretation over arbitrary inputs in $[0,1]$. A vector $X \in [0,1]^{2N}$ of probabilities corresponds to a distribution over $\{0,1\}^{2N}$ wherein each bit is chosen from a Bernoulli distribution with the corresponding probability. Because r is multilinear, r in fact computes the expectation of the acceptance probability over this distribution. In particular, the polynomial

$$q(s,t) = r\Bigg(\underbrace{\frac{s}{N},\ldots,\frac{s}{N}}_{N \text{ times}},\underbrace{\frac{t}{N},\ldots,\frac{t}{N}}_{N \text{ times}}\Bigg) = \mathbb{E}_{\substack{S_0\sim D_s,\\ S_1\sim D_t}}\left[\Pr[Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}} \text{ accepts}]\right] \tag{23}$$

corresponds to selecting $S_0 \sim D_s$ and $S_1 \sim D_t$. The total degree of q is obviously at most the degree of r, by the same reasoning as in the proof of Lemma 12.

To construct p, we apply the symmetrization lemma of Minsky and Papert [39] to symmetrize r, first with respect to X_0, then with respect to X_1:

$$p_0(x,X_1) = \mathbb{E}_{|S_0|=x}\, r(X_0,X_1) = \mathbb{E}_{|S_0|=x}\left[\Pr[Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}} \text{ accepts}]\right] \tag{24}$$

$$p(x,y) = \mathbb{E}_{|S_1|=y}\, p_0(x,X_1) = \mathbb{E}_{\substack{|S_0|=x,\\ |S_1|=y}}\left[\Pr[Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}} \text{ accepts}]\right] \tag{25}$$

The degree of p is at most the degree of r, due to Lemma 11. ◀

We remark that, as a consequence of their definitions in Lemma 13, p and q satisfy:

$$q(s,t) = \mathbb{E}\left[p(X,Y)\right], \tag{26}$$

where X and Y are drawn from N-trial binomial distributions with means s and t, respectively.

Symmetric Laurent polynomials

Finally, we state a useful fact about Laurent polynomials:

▶ **Lemma 14** (Symmetric Laurent polynomials)**.** *Let $\ell(x)$ be a real Laurent polynomial of positive and negative degree d that satisfies $\ell(x) = \ell(1/x)$. Then there exists a (ordinary) real polynomial q of degree d such that $\ell(x) = q(x + 1/x)$.*

Proof. $\ell(x) = \ell(1/x)$ implies that the coefficients of the x^i and x^{-i} terms are equal for all i, as otherwise $\ell(x) - \ell(1/x)$ would not equal the zero polynomial. Thus, we may write $\ell(x) = \sum_{i=0}^{d} a_i \cdot (x^i + x^{-i})$ for some coefficients a_i. So, it suffices to show that $x^i + x^{-i}$ can be expressed as a polynomial in $x + 1/x$ for all $0 \leq i \leq d$.

We prove by induction on i. The case $i = 0$ corresponds to constant polynomials. For $i > 0$, by the binomial theorem, observe that $(x+1/x)^i = x^i + x^{-i} + r(x)$ where r is a degree $i - 1$ real Laurent polynomial satisfying $r(x) = r(1/x)$. By the induction assumption, r can be expressed as a polynomial in $x+1/x$, so we have $x^i + x^{-i} = (x+1/x)^i - r(x)$ is expressed as a polynomial in $x + 1/x$. ◀

2.3 Complexity classes

▶ **Definition 15.** *The complexity class* QMA *consists of the languages L for which there exists a quantum polynomial time verifier V with the following properties:*

1. *Completeness: if $x \in L$, then there exists a quantum witness state $|\psi\rangle$ on $\mathrm{poly}(|x|)$ qubits such that $\Pr[V(x, |\psi\rangle) \text{ accepts}] \geq \frac{2}{3}$.*
2. *Soundness: if $x \notin L$, then for any quantum witness state $|\psi\rangle$ on $\mathrm{poly}(|x|)$ qubits, $\Pr[V(x, |\psi\rangle) \text{ accepts}] \leq \frac{1}{3}$.*

A quantum verifier that satisfies the above promise for a particular language will be referred to as a QMA verifier or QMA protocol throughout.

Though SBP and SBQP can be defined in terms of counting complexity functions, for our purposes it is easier to work with the following equivalent definitions (see Böhler et al. [11]):

▶ **Definition 16.** *The complexity class* SBP *consists of the languages L for which there exists a probabilistic polynomial time algorithm M and a polynomial σ with the following properties:*

1. *If $x \in L$, then $\Pr[M(x) \text{ accepts}] \geq 2^{-\sigma(|x|)}$.*
2. *If $x \notin L$, then $\Pr[M(x) \text{ accepts}] \leq 2^{-\sigma(|x|)}/2$.*

The complexity class SBQP *is defined analogously, wherein the classical algorithm is replaced with a quantum algorithm.*

A classical (respectively, quantum) algorithm that satisfies the above promise for a particular language will be referred to as an SBP (respectively, SBQP) algorithm throughout. Using these definitions, a query complexity relation between QMA protocols and SBQP algorithms follows from the procedure of Marriott and Watrous [36], which shows that one can exponentially improve the soundness and completeness errors of a QMA protocol without increasing the witness size. This relationship is now standard; see for example [36, Remark 6] or [50, Proposition 4.2] for a proof of the following lemma:

▶ **Lemma 17.** *Suppose there is a* QMA *protocol for some problem that makes T queries and receives an m-qubit witness. Then there is a quantum query algorithm Q for the same problem that makes $O(mT)$ queries, and satisfies the following:*

1. *If $x \in L$, then $\Pr[Q(x) \text{ accepts}] \geq 2^{-m}$.*
2. *If $x \notin L$, then $\Pr[Q(x) \text{ accepts}] \leq 2^{-10m}$.*

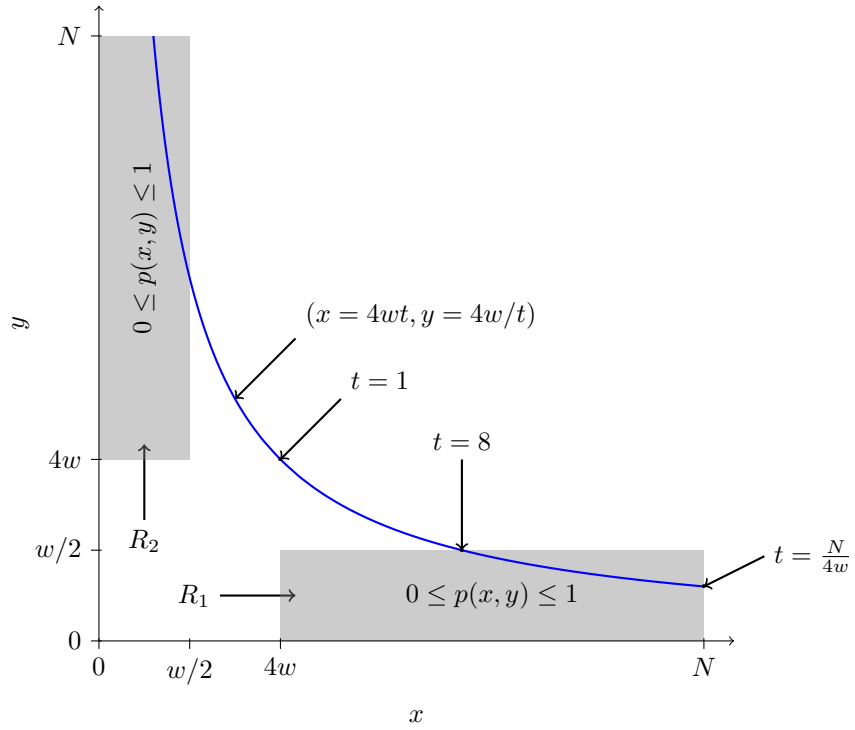


Figure 3 Diagram of Theorem 18 (not drawn to scale).

3 QMA complexity of approximate counting

This section establishes an optimal lower bound on the QMA complexity of approximate counting. We first lower bound the SBQP complexity of the $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ problem (Theorem 3). This implies a QMA lower bound for $\mathsf{ApxCount}_{N,w}$ via Lemma 17, but it is not quantitatively optimal. We prove the optimal QMA lower bound (Theorem 2) via Lemma 19, which leverages additional properties of the SBQP protocol derived via Lemma 17 from any QMA protocol with small witness length. Finally, Corollary 20 describes new oracle separations that are immediate consequences of Theorem 2 and Theorem 3.

3.1 Lower bound for SBQP algorithms

Our lower bound on the SBQP complexity of $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ hinges on the following theorem. The theorem uses Laurent polynomials to prove a degree lower bound for bivariate polynomials that satisfy an upper bound on an "L"-shaped pair of rectangles and a lower bound at a nearby point:

▶ **Theorem 18.** *Let $0 < w < 32w < N$ and $M \ge 1$. Let $R_1 = [4w, N] \times [0, w/2]$ and $R_2 = [0, w/2] \times [4w, N]$ be disjoint rectangles in the plane, and let $L = R_1 \cup R_2$. Let $p(x, y)$ be a real polynomial of degree d with the following properties:*

1. *$p(4w, 4w) \ge 1.5 \cdot M$.*
2. *$0 \le p(x, y) \le 1$ for all $(x, y) \in L$.*

Then $d = \Omega(\sqrt{N/w} \cdot \log M)$.

Proof. Observe that if $p(x, y)$ satisfies the statement of the theorem, then so does $p(y, x)$. This is because the constraints in the statement of the theorem are symmetric in x and y (in particular, because R_1 and R_2 are mirror images of one another along the line $x = y$; see Figure 3). As a result, we may assume without loss of generality that p is symmetric, i.e., $p(x, y) = p(y, x)$. Else, we may replace p by $\frac{p(x,y)+p(y,x)}{2}$ because the set of polynomials that satisfy the inequalities in the statement of the theorem are closed under convex combinations.

Consider the hyperbolic parametric curve $(x = 4wt, y = 4w/t)$ as it passes through R_1 (see Figure 3). We can view the restriction of $p(x, y)$ to this curve as a Laurent polynomial $\ell(t) = p(4wt, 4w/t)$ of positive and negative degree d. The bound of $p(x, y)$ on all of R_1 implies that $|\ell(t)| \leq 1$ when $t \in [8, \frac{N}{4w}]$ and that $\ell(1) \geq 1.5$ (see Figure 3). Moreover, the condition that $p(x, y)$ is symmetric implies that $\ell(t) = \ell(1/t)$.

By Lemma 14 for symmetric Laurent polynomials, $\ell(t)$ can be viewed as a degree d polynomial $q(t + 1/t)$. Under the transformation $s = t + 1/t$, q satisfies $|q(s)| \leq 1$ for $s \in [8+1/8, \frac{N}{4w} + \frac{4w}{N}]$ and $q(2) \geq 1.5M$. Note that the length of the interval $[8+1/8, \frac{N}{4w} + \frac{4w}{N}]$ is $\Theta(N/w)$ because $w < N$. By an appropriate affine transformation of q, we can conclude from Lemma 7 with $\mu = \Theta(w/N)$ that $d = \Omega(\sqrt{N/w} \cdot \log M)$. ◀

Why is Theorem 18 useful? One may be tempted to apply this theorem directly to the polynomial $p(x, y)$ obtained in Lemma 13 to conclude a degree lower bound (and thus a query complexity lower bound), as the "L"-shaped pair of rectangles $L = R_1 \cup R_2$ correspond to "no" instances of $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$, while $(4w, 4w)$ corresponds to a "yes" instance. However, even though $p(x, y)$ is bounded at lattice points in L, it need not be bounded along the entirety of L.[11]

To obtain a lower bound, we instead use the connection between the polynomials $p(x, y)$ and $q(s, t)$ from Lemma 13, and establish Theorem 3 from the introduction, restated for convenience:

▶ **Theorem 3.** *Consider an* SBQP *algorithm for* $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ *that makes* T *queries to membership oracles for the two instances of* $\mathsf{ApxCount}_{N,w}$. *Then* $T = \Omega\left(\min\{w, \sqrt{N/w}\}\right)$.

Proof. Let $N > 32w$ (otherwise the theorem holds trivially). Since Q is an SBQP algorithm, we may suppose that Q accepts with probability at least 2α on a "yes" instance and with probability at most α on a "no" instance (note that α may be exponentially small in N). Take $p(x, y)$ and $q(s, t)$ to be the symmetrized bivariate polynomials of degree at most $2T$ defined in Lemma 13. Define $L' = ([0, w] \times [0, w]) \cup ([0, w] \times [2w, N]) \cup ([2w, N] \times [0, w])$. The conditions on the acceptance probability of Q for all S_0, S_1 that satisfy the $\mathsf{ApxCount}_{N,w}$ promise imply that $p(x, y)$ satisfies these corresponding conditions:

1. $1 \geq p(x, y) \geq 2\alpha$ for all $(x, y) \in ([2w, N] \times [2w, N]) \cap \mathbb{Z}^2$.
2. $0 \leq p(x, y) \leq \alpha$ for all $(x, y) \in L' \cap \mathbb{Z}^2$.

Our strategy is to show that if $T = o(w)$, then these conditions on p imply that the polynomial $q(s, t) \cdot \frac{0.9}{\alpha}$ satisfies the statement of Theorem 18 for all sufficiently large w. This in turn implies $T = \Omega(\sqrt{N/w})$. This allows us conclude that either $T = \Omega(w)$ or $T = \Omega(\sqrt{N/w})$, which proves the theorem.

[11] One can nevertheless use this intuition to obtain a nontrivial (though suboptimal) lower bound by inspecting p alone. Using the Markov brothers' inequality (Lemma 6), if $\deg(p) = o(\sqrt{w})$, then the bounds on $p(x, y)$ at lattice points in L imply that $|p(x, y)| \leq 1 + o_w(1)$ for all $(x, y) \in L$. Thus, Theorem 18 applies if $\deg(p) = o(\sqrt{w})$, so overall we get a lower bound of $\Omega\left(\min\left\{\sqrt{w}, \sqrt{N/w}\right\}\right)$ for the SBQP query complexity of $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$. See arXiv:1902.02398 for details.

Suppose $T = o(w)$, so that $p(x, y)$ and $q(s, t)$ both have degree $d = o(w)$. We begin by upper bounding $p(x, y)$ at the lattice points (x, y) outside of L'. We claim the following:

(a) $|p(x, y)| \leq \alpha \cdot a \cdot \exp(bd^2/w) \leq \alpha \cdot a \cdot \exp(bd)$ whenever $(x, y) \in L'$ and either x or y is an integer, where a and b are the constants from Lemma 10. This follows from Lemma 10 by fixing either x or y to be an integer and viewing the resulting restriction of $p(x, y)$ as a univariate polynomial in the other variable.

(b) $|p(x, y)| \leq \alpha \cdot a \cdot \exp(bd) \cdot \exp(2\sqrt{3}d) = \alpha \cdot a \cdot \exp((b + 2\sqrt{3})d)$ whenever $x \in [w, 2w]$, $y \in [0, w]$, and y is an integer. This follows Lemma 7: consider the univariate polynomial $p(\cdot, y)$ on the intervals $[0, w]$ and $[2w, 3w]$, where it is bounded by (a).

(c) $|p(x, y)| \leq \alpha \cdot a \cdot \exp((b + 2\sqrt{3})d) \cdot a \cdot \exp(bd^2/w) \leq \alpha \cdot a^2 \cdot \exp((2b + 2\sqrt{3})d)$ whenever $x \in [w, 2w]$ and $y \in [0, w]$. This follows from Lemma 10: consider the univariate polynomial $p(x, \cdot)$ on the interval $[0, w]$, where it is bounded at integer points by (b).

(d) $|p(x, y)| \leq \alpha \cdot a^2 \cdot \exp((2b + 2\sqrt{3})d) \cdot \exp(4dy/w) = \alpha \cdot a^2 \cdot \exp((2b + 2\sqrt{3} + 4y/w)d)$ whenever $x \in [0, N]$, $y \in [w + 1, N]$, and x is an integer. This follows from Lemma 7: consider the univariate polynomial $p(x, \cdot)$ on the interval $[0, w]$, where it is bounded by (a) when $x \in [0, w]$ or $x \in [2w, N]$, or bounded by (c) when $x \in [w, 2w]$. By an affine shift, this corresponds to applying Lemma 7 with $\mu = 2y/w - 2$, with the observation that $\sqrt{2\mu + \mu^2} < \mu + 2$.

We now use this to upper bound $q(s, t)$ when $s \in [4w, N]$ and $t \in [0, w/2]$. Let X and Y be drawn from N-trial binomial distributions with means s and t, respectively, so that $q(s, t) = \mathbb{E}[p(X, Y)]$. Using the above bounds and basic probability, we have

$$0 \leq q(s, t) = \mathbb{E}[p(X, Y)] \tag{27}$$

$$\leq \alpha \cdot \Bigg(\Pr[X \geq 2w, Y \leq w] + \Pr[X \leq 2w, Y \leq w] \cdot a \cdot \exp\left(\left(b + 2\sqrt{3}\right) d\right) + \sum_{y=w+1}^{N} \Pr[Y = y] \cdot a^2 \cdot \exp\left(\left(2b + 2\sqrt{3} + 4y/w\right) d\right) \Bigg) \tag{28}$$

$$\leq \alpha \cdot \Bigg(1 + \Pr[X \leq 2w] \cdot a \cdot \exp\left(\left(b + 2\sqrt{3}\right) d\right) + \sum_{y=w+1}^{N} \Pr[Y \geq y] \cdot a^2 \cdot \exp\left(\left(2b + 2\sqrt{3} + 4y/w\right) d\right) \Bigg). \tag{29}$$

The probabilities above are easily bounded with a Chernoff bound:

$$q(s, t) = \mathbb{E}[p(X, Y)] \leq \alpha \cdot \Bigg(1 + a \cdot \exp\left(\left(b + 2\sqrt{3}\right) d - w/2\right) + \sum_{y=w+1}^{N} a^2 \cdot \exp\left(\left(2b + 2\sqrt{3} + 4y/w\right) d - y/6\right) \Bigg). \tag{30}$$

Because a and b are universal constants from Lemma 10, when $d = o(w)$, the first exponential term becomes arbitrarily small for all sufficiently large w. Moreover, for all sufficiently large w, the remaining sum becomes bounded by a geometric sum. For some constant c, we have

$$\begin{aligned} \sum_{y=w+1}^{N} a^2 \cdot \exp\left(\left(2b + 2\sqrt{3} + 4y/w\right) d - y/6\right) &\leq \sum_{y=w+1}^{\infty} c \cdot \exp\left(-y/12\right) \\ &\leq \frac{c}{1 - \exp(-1/12)} \cdot \exp(-w/12) \\ &= o_w(1). \end{aligned}$$

Thus we conclude that $0 \leq q(s,t) \leq \alpha \cdot (1 + o_w(1))$ when $s \in [4w, N]$ and $t \in [0, w/2]$ (i.e., $(s,t) \in R_1$ in the statement of Theorem 18). By symmetry, we can conclude the same bound when $s \in [0, w/2]$ and $t \in [4w, N]$ (i.e., $(s,t) \in R_2$ in the statement of Theorem 18).

Now, we lower bound $q(4w, 4w)$. Let X and Y be drawn from independent N-trial binomial distributions with mean $4w$, so that $q(4w, 4w) = \mathbb{E}[p(X,Y)]$. Then we have

$$\begin{aligned}
\mathbb{E}[p(X,Y)] &\geq 2\alpha \cdot \Pr[X \geq 2w, Y \geq 2w] \\
&\geq 2\alpha \cdot (1 - \Pr[X \leq 2w] - \Pr[Y \leq 2w]) \\
&\geq 2\alpha \cdot (1 - 2\exp(-w/2)) \\
&\geq 2\alpha \cdot (1 - o_w(1))
\end{aligned}$$

We conclude that $q(s,t) \cdot \frac{0.9}{\alpha}$ satisfies the statement of Theorem 18 (with $M = 1$) for all sufficiently large w. ◀

We remark that this lower bound is tight, i.e., there exists an SBQP algorithm that makes $O\left(\min\left\{w, \sqrt{N/w}\right\}\right)$ queries. The $O(\sqrt{N/w})$ upper bound follows from the BQP algorithm of Brassard, Høyer, and Tapp [14]. The $O(w)$ upper bound is in fact an SBP upper bound with the following algorithmic interpretation: first, guess $w+1$ items randomly from each of S_0 and S_1. Then, verify using the membership oracle that the first $w+1$ items all belong to S_0 and that the latter $w+1$ items all belong to S_1, accepting if and only if this is the case. Clearly, this accepts with nonzero probability if and only if $|S_0| \geq w+1$ and $|S_1| \geq w+1$.

3.2 Lower bound for QMA

In this section, we establish the optimal QMA lower bound (Theorem 2). We begin by quantitatively improving the SBQP lower bound for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ of Theorem 3, under the stronger assumption that the parameter α in the SBQP protocol is not smaller than 2^{-w}. (In addition to a stronger conclusion, this assumption also permits a considerably simpler analysis than was required to prove Theorem 3).

▶ **Lemma 19.** *Consider any quantum query algorithm $Q^{\mathcal{O}_{S_0}, \mathcal{O}_{S_1}}$ for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ that makes T queries to the membership oracles $\mathcal{O}_{S_0}$ and $\mathcal{O}_{S_1}$ for the two instances of $\mathsf{ApxCount}_{N,w}$ and satisfies the following. For some $m = o(w)$, $\alpha = 2^{-m}$, and $M \in [1, \alpha^{-1}]$:*
1. *If $x \in L$, then $\Pr[Q(x) \text{ accepts}] \geq \alpha$.*
2. *If $x \notin L$, then $\Pr[Q(x) \text{ accepts}] \leq \alpha/(2M)$.*
Then $T = \Omega\left(\sqrt{N/w} \cdot \log M\right)$

Proof. As in the proof of Theorem 3, define $L' = ([0,w] \times [0,w]) \cup ([0,w] \times [2w,N]) \cup ([2w,N] \times [0,w])$, and take $p(x,y)$ and $q(s,t)$ to be the symmetrized bivariate polynomials of degree at most $2T$ defined in Lemma 13. $p(x,y)$ satisfies the following properties.
(a) $1 \geq p(x,y) \geq \alpha$ for all $(x,y) \in ([2w,N] \times [2w,N]) \cap \mathbb{Z}^2$.
(b) $0 \leq p(x,y) \leq \alpha/(1.5M)$ for all $(x,y) \in L' \cap \mathbb{Z}^2$.
(c) $0 \leq p(x,y) \leq 1$ for all $(x,y) \in ([0,N] \times [0,N]) \cap \mathbb{Z}^2$.

We use these properties to upper bound $q(s,t)$ when $s \in [4w,N]$ and $t \in [0,w/2]$. Let X and Y be drawn from N-trial binomial distributions with means s and t, respectively, so that $q(s,t) = \mathbb{E}[p(X,Y)]$. Using the above bounds and basic probability, we have

$$\begin{aligned}0 \leq q(s,t) &= \mathbb{E}[p(X,Y)] \\ &\leq \alpha/(2M)\Pr[X \geq 2w, Y \leq w] + (1 - \Pr[X \geq 2w, Y \leq w]) \\ &\leq \alpha/(2M) + 2^{-\Omega(w)} \leq (1+o(1))\alpha/(2M)\end{aligned}$$

Here, the first inequality holds by Properties (a)-(c) above, while the second follows from a Chernoff Bound, and the third holds because $\alpha/(2M) \geq 2^{-o(w)}$.

Thus we conclude that $0 \leq q(s,t) \leq \alpha/(2M)\cdot(1+o_w(1))$ when $s \in [4w, N]$ and $t \in [0, w/2]$ (i.e., $(s,t) \in R_1$ in the statement of Theorem 18). By symmetry, we can conclude the same bound when $s \in [0, w/2]$ and $t \in [4w, N]$ (i.e., $(s,t) \in R_2$ in the statement of Theorem 18).

Now, we lower bound $q(4w, 4w)$. Let X and Y be drawn from independent N-trial binomial distributions with mean $4w$, so that $q(4w,4w) = \mathbb{E}[p(X,Y)]$. Then we have

$$\begin{aligned}\mathbb{E}[p(X,Y)] &\geq \alpha \cdot \Pr[X \geq 2w, Y \geq 2w] \\ &\geq \alpha \cdot (1 - \Pr[X \leq 2w] - \Pr[Y \leq 2w]) \\ &\geq \alpha \cdot (1 - 2\exp(-w/2)) \\ &\geq \alpha \cdot (1 - o_w(1))\end{aligned}$$

We conclude that $q(s,t) \cdot \frac{1.8M}{\alpha}$ satisfies the statement of Theorem 18 for all sufficiently large w. Hence, $T = \Omega\left(\sqrt{N/w} \cdot \log M\right)$ as claimed. ◀

We now establish Theorem 2 from the introduction, which quantitatively lower bounds the QMA complexity of $\mathsf{ApxCount}_{N,w}$. The analysis exploits two key properties of the SBQP protocols that result from applying Lemma 17 to a QMA protocol with witness length m: (1) the parameter α of the SBQP protocol is not too small (at least 2^{-m}) and (2) the multiplicative gap between acceptance probabilities when $f(x) = 0$ vs. $f(x) = 1$ is at least 2^m, which may be much greater than 2.

▶ **Theorem 2.** *Consider a* QMA *protocol that solves* $\mathsf{ApxCount}_{N,w}$*. If the protocol receives a quantum witness of length m, and makes T queries to the membership oracle for S, then either $m = \Omega(w)$ or $T = \Omega\left(\sqrt{N/w}\right)$.*

Proof. Consider a QMA protocol for $\mathsf{ApxCount}_{N,w}$ with witness size m and query cost T. If $m = \Omega(w)$, the theorem is vacuous, so suppose that $m = o(w)$. Running the verifier, Arthur, a constant number of times with fresh witnesses to reduce the soundness and completeness errors, one obtains a verifier with soundness and completeness errors $1/6$ that receives an $O(m)$-length witness and makes $O(T)$ queries. Repeating twice with two oracles and computing the AND, one obtains a QMA verifier $V'^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}}$ for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ with soundness and completeness errors $1/3$ that receives an $O(m)$-length witness and makes $O(T)$ queries. Applying Lemma 17 to V', there exists a quantum query algorithm $Q^{\mathcal{O}_{S_0},\mathcal{O}_{S_1}}$ for $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ that makes $O(m \cdot T)$ queries and satisfies the hypothesis of Lemma 19 with $M = 2^{-\Theta(m)}$. Theorem 3 tells us that $m \cdot T = \Omega\left(\sqrt{N/w} \cdot m\right)$. Equivalently, $T = \Omega\left(\sqrt{N/w}\right)$. ◀

Theorem 3 also implies several oracle separations:

▶ **Corollary 20.** *There exists an oracle A and a pair of languages L_0, L_1 such that:*

1. $L_0, L_1 \in \mathsf{SBP}^A$
2. $L_0 \cap L_1 \notin \mathsf{SBQP}^A$.
3. $\mathsf{SBP}^A \not\subset \mathsf{QMA}^A$.

Proof. For an arbitrary function $A : \{0,1\}^* \to \{0,1\}$ and $i \in \{0,1\}$, define $A_i^n = \{x \in \{0,1\}^n : A(i,x) = 1\}$. Define the unary language $L_i^A = \{1^n : |A_i^n| \geq 2^{n/2}\}$. Observe that as long as A satisfies the promise $|A_i^n| \geq 2^{n/2}$ or $|A_i^n| \leq 2^{n/2-1}$ for all $n \in \mathbb{N}$, then $L_i^A \in \mathsf{SBP}^A$. Intuitively, the oracles A that satisfy this promise encode a pair of $\mathsf{ApxCount}_{N,w}$ instances $|A_0^n|$ and $|A_1^n|$ for every $n \in \mathbb{N}$ where $N = 2^n$ and $w = 2^{n/2-1}$.

Theorem 3 tells us that an SBQP algorithm Q that makes $o(2^{n/4})$ queries fails to solve $\mathsf{AND}_2 \circ \mathsf{ApxCount}_{N,w}$ on *some* pair (S_0, S_1) that satisfies the promise. Thus, one can construct an A such that $L_0, L_1 \in \mathsf{SBP}^A$ and $L_0 \cap L_1 \notin \mathsf{SBQP}^A$, by choosing (A_0^n, A_1^n) so as to diagonalize against all SBQP algorithms.

Because QMA^A is closed under intersection for any oracle A, and because $\mathsf{QMA}^A \subseteq \mathsf{SBQP}^A$ for any oracle A, it must be the case that either $L_0 \notin \mathsf{QMA}^A$ or $L_1 \notin \mathsf{QMA}^A$. ◀

4 Approximate counting with quantum samples and reflections

4.1 The Laurent polynomial method

By using Minsky–Papert symmetrization (Lemma 11), we now prove the key fact that relates quantum algorithms, of the type we're considering, to real Laurent polynomials in one variable. The following lemma generalizes the connection between quantum algorithms and real polynomials established by Beals et al. [8].

▶ **Lemma 21.** *Let Q be a quantum algorithm that makes T queries to $\mathcal{O}_S$, uses R_1 copies of $|S\rangle$, and makes R_2 uses of the unitary $\mathcal{R}_S$. Let $R := R_1 + 2R_2$. For $k \in \{1, \ldots, N\}$, let*

$$q(k) := \mathbb{E}_{|S|=k}\left[\Pr\left[Q^{\mathcal{O}_S, \mathcal{R}_S}\left(|S\rangle^{\otimes R_1}\right) \text{ accepts}\right]\right]. \tag{31}$$

Then q can be written a univariate Laurent polynomial, with maximum exponent at most $2T + R$ and minimum exponent at least $-R$.

Proof. Let $|\psi_{\text{initial}}\rangle$ denote the initial state of the algorithm, which we can write as

$$|\psi_{\text{initial}}\rangle = |S\rangle^{\otimes R_1} = \left(\frac{1}{\sqrt{|S|}}\sum_{i \in S}|i\rangle\right)^{\otimes R_1} = \left(\frac{1}{\sqrt{|S|}}\sum_{i \in [N]} x_i|i\rangle\right)^{\otimes R_1}$$
$$= \frac{1}{|S|^{R_1/2}} \sum_{i_1, \ldots, i_{R_1} \in [N]} x_{i_1} \cdots x_{i_{R_1}} |i_1, \ldots, i_{R_1}\rangle.$$

Thus, each amplitude is a complex multilinear polynomial in $X = (x_1, \ldots, x_N)$ of degree R_1, divided by $|S|^{R_1/2}$.

Throughout the algorithm, each amplitude will remain a complex multilinear polynomial in X divided by some power of $|S|$. Since $x_i^2 = x_i$ for all i, we can always maintain multilinearity without loss of generality.

Like Beals et al. [8], we now consider how the polynomial degree of each amplitude and the power of $|S|$ in the denominator change as the algorithm progresses. We have to handle 3 different kinds of unitaries that the quantum circuit may use: the membership query oracle $\mathcal{O}_S$, unitaries independent of the input, and the reflection unitary $\mathcal{R}_S$.

The first two cases are handled as in Beals et al. Since $\mathcal{O}_S$ is a unitary whose entries are degree-1 polynomials in X, each use of this unitary increases a particular amplitude's degree as a polynomial by 1 and does not change the power of $|S|$ in the denominator. Second, input-independent unitary transformations only take linear combinations of existing

polynomials and hence do not increase the degree of the amplitudes or the power of $|S|$ in the denominator. Finally, we consider the reflection unitary $\mathcal{R}_S = \mathbb{1} - 2|S\rangle\langle S|$. The $(i,j)^{\text{th}}$ entry of this operator is $\delta_{ij} - \frac{2x_i x_j}{|S|} = \frac{\delta_{ij}|S| - 2x_i x_j}{|S|}$, where δ_{ij} is the Kronecker delta function. Since $|S| = \sum_i x_i$, this is a degree-2 polynomial divided by $|S|$. Hence applying this unitary will increase the degree of the amplitudes by 2 and increase the power of $|S|$ in the denominator by 1.

In conclusion, we start with each amplitude being a polynomial of degree R_1 divided by $|S|^{R_1/2}$. T queries to the membership oracle will increase the degree of each amplitude by at most T and leave the power of $|S|$ in the denominator unchanged. R_2 uses of the reflection unitary will increase the degree by at most $2R_2$ and the power of $|S|$ in the denominator by R_2. It follows that Q's final state has the form

$$|\psi_{\text{final}}\rangle = \sum_z \alpha_z(X)\,|z\rangle\,, \tag{32}$$

where each $\alpha_z(X)$ is a complex multilinear polynomial in X of degree at most $R_1 + 2R_2 + T = R + T$, divided by $|S|^{R_1/2+R_2} = |S|^{R/2}$. Since X itself is real-valued, it follows that the real and imaginary parts of $\alpha_z(X)$, considered individually, are real multilinear polynomials in X of degree at most $R+T$ divided by $|S|^{R/2}$.

Hence, if we let

$$p(X) := \Pr\left[Q^{\mathcal{O}_S,\mathcal{R}_S}\left(|S\rangle^{\otimes R_1}\right) \text{ accepts}\right], \tag{33}$$

then

$$p(X) = \sum_{\text{accepting } z} |\alpha_z(X)|^2 = \sum_{\text{accepting } z} \left(\text{Re}^2\,\alpha_z(X) + \text{Im}^2\,\alpha_z(X)\right) \tag{34}$$

is a real multilinear polynomial in X of degree at most $2(R+T)$, divided through (in every monomial) by $|S|^R = |X|^R$.

Now consider

$$q(k) := \mathbb{E}_{|X|=k}\left[p(X)\right]. \tag{35}$$

By Lemma 11, this is a real univariate polynomial in $|X|$ of degree at most $2(R+T)$, divided through (in every monomial) by $|S|^R = |X|^R$. Or said another way, it's a real Laurent polynomial in $|X|$, with maximum exponent at most $R + 2T$ and minimum exponent at least $-R$. ◀

4.2 Upper bounds

Before proving our lower bounds on the degree of Laurent polynomials approximating $\mathsf{ApxCount}_{N,w}$, we establish some simpler *upper bounds*. We show upper bounds on Laurent polynomial degree and in the queries, samples, and reflections model.

Laurent polynomial degree of approximate counting

We now describe a *purely negative* degree Laurent polynomial of degree $O(w^{1/3})$ for approximate counting. This upper bound will serve as an important source of intuition when we prove the (matching) lower bound of Theorem 4 (see Section 4.4.3). We are thankful to user "fedja" on MathOverflow for describing this construction.[12]

[12] See https://mathoverflow.net/questions/302113/real-polynomial-bounded-at-inverse-integer-points

▶ **Lemma 22** (fedja). *For all w, there is a real polynomial p of degree $O\left(w^{1/3}\right)$ such that:*
1. $0 \leq p(1/k) \leq \frac{1}{3}$ *for all* $k \in [w]$.
2. $\frac{2}{3} \leq p(1/k) \leq 1$ *for all integers* $k \geq 2w$.
3. $0 \leq p(1/k) \leq 1$ *for all* $k \in \{w+1, w+2, \ldots, 2w-1\}$.

Proof. Assuming for simplicity that w is a perfect cube, consider

$$u(x) := (1-x)(1-2x)\cdots\left(1-w^{1/3}x\right). \tag{36}$$

Notice that $\deg(u) = w^{1/3}$ and $u\left(\frac{1}{k}\right) = 0$ for all $k \in \left[w^{1/3}\right]$. Furthermore, we have $u(x) \in [0,1]$ for all $x \in \left[0, \frac{1}{w^{1/3}}\right]$, and also $u(x) \in \left[1 - O\left(\frac{1}{w^{1/3}}\right), 1\right]$ for all $x \in \left[0, \frac{1}{w}\right]$. Now, let v be the Chebyshev polynomial of degree $w^{1/3}$, affinely adjusted so that $v(x) \in [0,1]$ for all $x \in \left[0, \frac{1}{w^{1/3}}\right]$ (rather than in $[-1,1]$ for all all $|x| \leq 1$), and with a large jump between $\frac{1}{2w}$ and $\frac{1}{w}$. Then the product, $p(x) := u(x)\,v(x)$, has degree $2w^{1/3}$ and satisfies all the requirements, except possibly that the constants $\frac{1}{3}$ and $\frac{2}{3}$ in the first two requirements may be off. Composing with a constant degree polynomial corrects this, and gives a polynomial of degree $O(w^{1/3})$ that satisfies all three requirements. ◀

Interestingly, if we restrict our attention to purely negative degree Laurent polynomials, then a matching lower bound is not too hard to show. In the same MathOverflow post, user fedja also proves the following, which can also be shown using earlier work of Zhandry [57, Proof of Theorem 7.3]):

▶ **Lemma 23.** *Let p be a real polynomial, and suppose that $|p(1/k)| \leq 1$ for all $k \in [2w]$, and that $p\left(\frac{1}{w}\right) \leq \frac{1}{3}$ while $p\left(\frac{1}{2w}\right) \geq \frac{2}{3}$. Then $\deg(p) = \Omega\left(w^{1/3}\right)$.*

Section 4.3 and Section 4.4 below take the considerable step of extending Lemma 23 from purely negative degree Laurent polynomials to general Laurent polynomials.

Upper bounds in the queries, samples, and reflections model

Although we showed that there is a purely negative degree Laurent polynomial of degree $O(w^{1/3})$ for $\mathsf{ApxCount}_{N,w}$, this does not imply the existence of a quantum algorithm in the queries, samples, and reflections model with similar complexity.

We now show that our lower bounds in the queries, samples, and reflections model (in Theorem 4) are tight (up to constants). This is Theorem 5 in the introduction, restated here for convenience:

▶ **Theorem 5.** *There is a quantum algorithm that solves $\mathsf{ApxCount}_{N,w}$ with high probability using R copies of $|S\rangle$ and reflections about $|S\rangle$, where $R = O\left(\min\left\{w^{1/3}, \sqrt{\frac{N}{w}}\right\}\right)$.*

Proof. We describe two quantum algorithms for this problem with the two stated complexities.

The first algorithm uses $O(w^{1/3})$ samples and reflections. This algorithm is reminiscent of the original collision finding algorithm of Brassard, Høyer, and Tapp [15]. We first use $O(w^{1/3})$ copies of $|S\rangle$ to learn a set $M \subset S$ of size $w^{1/3}$ by simply measuring copies of $|S\rangle$ in the computational basis. Now we know that the ratio $|S|/|M|$ is either $w^{2/3}$ or $2w^{2/3}$. Now consider running Grover's algorithm on the set S where the elements in M are considered the "marked" elements. Grover's algorithm alternates reflections about the uniform superposition over the set being searched, S, with an operator that reflects about the marked elements in M. The first reflection is simply $\mathcal{R}_S$, which we have access to. The second unitary can be

constructed since we have an explicit description of the set M. Now Grover's algorithm can be used to distinguish whether the fraction of marked elements is $1/w^{2/3}$ or half of that, and the cost will be $O(w^{1/3})$.

The second algorithm uses $O(\sqrt{N/w})$ reflections only and no copies of $|S\rangle$. Consider running the standard approximate counting algorithm [13] that uses membership queries to S and distinguishes $|S| \leq w$ from $|S| \geq 2w$ using $O(\sqrt{N/w})$ membership queries. Observe that this algorithm starts with the state $|\psi\rangle = \frac{1}{\sqrt{N}}(|1\rangle + \cdots + |N\rangle)$, which is in $\mathrm{span}\{|S\rangle, |\bar{S}\rangle\}$, and only uses reflections about $|\psi\rangle$ and membership queries to $|S\rangle$ in the form of a unitary that maps $|i\rangle$ to $-|i\rangle$ when $i \in S$. This means the state of the algorithm remains in $\mathrm{span}\{|S\rangle, |\bar{S}\rangle\}$ at all times. Within this subspace, a membership query to S is the same as a reflection about $|S\rangle$. Hence we can replace membership queries with the reflection operator to get an approximate counting algorithm that only uses $O(\sqrt{N/w})$ reflections and no copies of $|S\rangle$. ◀

Note that both the algorithms presented above generalize to the situation where we want to distinguish $|S| = w$ from $|S| = (1+\varepsilon)w$. For the first algorithm, we now pick a subset M of size $w^{1/3}/\varepsilon^{2/3}$. Now we want to $(1+\varepsilon)$-approximate the fraction of marked elements, which is either $1/(w\varepsilon)^{2/3}$ or $(1+\varepsilon)^{-1}$ times that. This can be done with approximate counting [13, Theorem 15], and the cost will be $O\left(\frac{1}{\varepsilon}(w\varepsilon)^{1/3}\right) = O\left(\frac{w^{1/3}}{\varepsilon^{2/3}}\right)$. The second algorithm is simpler to generalize, since we simply plug in the query complexity of ε-approximate counting, which is $O\left(\frac{1}{\varepsilon}\sqrt{\frac{N}{w}}\right)$.

4.3 Lower bound using the explosion argument

We now show a weaker version of Theorem 4 using the explosion argument described in the introduction. The difference between the following theorem and Theorem 4 is the exponent of w in the lower bound.

▶ **Theorem 24.** *Let Q be a quantum algorithm that makes T queries to the membership oracle for S, and uses a total of R copies of $|S\rangle$ and reflections about $|S\rangle$. If Q decides whether $|S| = w$ or $|S| = 2w$ with success probability at least $2/3$, promised that one of those is the case, then either*

$$T = \Omega\left(\sqrt{\frac{N}{w}}\right) \qquad \text{or} \qquad R = \Omega\left(\min\left\{w^{1/4}, \sqrt{\frac{N}{w}}\right\}\right). \tag{37}$$

Proof. Since we neglect multiplicative constants in our lower bounds, let us allow the algorithm to use up to R copies of $|S\rangle$ and R uses of $\mathcal{R}_S$. Let

$$q(k) := \mathbb{E}_{|S|=k}\left[\Pr\left[Q^{\mathcal{O}_S, \mathcal{R}_S}\left(|S\rangle^{\otimes R}\right) \text{ accepts}\right]\right]. \tag{38}$$

Then by Lemma 21, we can write q as a Laurent polynomial:

$$q(k) = u(k) + v(1/k), \tag{39}$$

where u is a real polynomial in k with $\deg(u) = O(T+R)$, and v is a real polynomial in $1/k$ with $\deg(v) = O(R)$. So to prove the theorem, it suffices to show that either $\deg(u) = \Omega\left(\sqrt{\frac{N}{w}}\right)$, or else $\deg(v) = \Omega\left(w^{1/4}\right)$. To do so, we'll assume that $\deg(u) = o\left(\sqrt{\frac{N}{w}}\right)$ and $\deg(v) = o\left(w^{1/4}\right)$, and derive a contradiction.

Our high-level strategy is as follows: we'll observe that, if approximate counting is being successfully solved, then either u or v must attain a large first derivative somewhere in its domain. By the approximation theory lemmas that we proved in Section 2.1, this will force that polynomial to have a large range – even on a subset of integer (or inverse-integer) points. But the sum, $u(k) + v(1/k)$, is bounded in $[0,1]$ for all $k \in [N]$. So if one polynomial has a large range, then the other does too. But this forces the *other* polynomial to have a large derivative somewhere in its domain, and therefore (by approximation theory) to have an even larger range, forcing the first polynomial to have an even larger range to compensate, and so on. As long as $\deg(u)$ and $\deg(v)$ are both small enough, this endless switching will force both u and v to attain *unboundedly* large values – with the fact that one polynomial is in k, and the other is in $1/k$, crucial to achieving the desired "explosion." Since u and v are polynomials on compact sets, such unbounded growth is an obvious absurdity, and this will give us the desired contradiction.

In more detail, we will study the following quantities.

$$\begin{array}{ll}
G_u := \max_{x,y\in\left[\sqrt{w},2w\right]} |u(x)-u(y)| & G_v := \max_{x,y\in\left[\frac{1}{N},\frac{1}{w}\right]} |v(x)-v(y)| \\
\Delta_u := \max_{x\in\left[\sqrt{w},2w\right]} |u'(x)| & \Delta_v := \max_{x\in\left[\frac{1}{N},\frac{1}{w}\right]} |v'(x)| \\
H_u := \max_{x,y\in\left[\sqrt{w},N\right]} |u(x)-u(y)| & H_v := \max_{x,y\in\left[\frac{1}{N},\frac{1}{\sqrt{w}}\right]} |v(x)-v(y)| \\
I_u := \max_{x,y\in[w,N]} |u(x)-u(y)| & I_v := \max_{x,y\in\left[\frac{1}{2w},\frac{1}{\sqrt{w}}\right]} |v(x)-v(y)| \\
L_u := \max_{x,y\in\{w,\ldots,N\}} |u(x)-u(y)| & L_v := \max_{x,y\in\left\{\sqrt{w},\ldots,2w\right\}} \left|v\left(\frac{1}{x}\right)-v\left(\frac{1}{y}\right)\right|
\end{array} \tag{40}$$

We have $0 \leq q(k) \leq 1$ for all $k \in [N]$, since in those cases $q(k)$ represents a probability. Since Q solves approximate counting, we also have $q(w) \leq \frac{1}{3}$ and $q(2w) \geq \frac{2}{3}$. This means in particular that either

(i) $u(2w) - u(w) \geq \frac{1}{6}$, and hence $G_u \geq \frac{1}{6}$, or else

(ii) $v\left(\frac{1}{2w}\right) - v\left(\frac{1}{w}\right) \geq \frac{1}{6}$, and hence $G_v \geq \frac{1}{6}$.

We will show that either case leads to a contradiction.

We have the following inequalities regarding u:

$$\begin{array}{ll}
G_u \geq L_v - 1 & \text{by the boundedness of } q \\
\Delta_u \geq \frac{G_u}{2w} & \text{by basic calculus} \\
H_u \geq \frac{\Delta_u\left(N-\sqrt{w}\right)}{\deg(u)^2} & \text{by Lemma 6} \\
I_u \geq \frac{H_u}{2} & \text{by Corollary 8} \\
L_u \geq \frac{I_u}{2} & \text{by Lemma 9}
\end{array} \tag{41}$$

Here the fourth inequality uses the fact that, setting $\varepsilon := \frac{\sqrt{w}}{N}$, we have $\deg(u) = o\left(\frac{1}{\sqrt{\varepsilon}}\right)$ (thereby satisfying the hypothesis of Corollary 8), while the fifth inequality uses the fact that $\deg(u) = o\left(\sqrt{N}\right)$.

Meanwhile, we have the following inequalities regarding v:

$$\begin{array}{ll}
G_v \geq L_u - 1 & \text{by the boundedness of } q \\
\Delta_v \geq G_v w & \text{by basic calculus} \\
H_v \geq \frac{\Delta_v\left(\frac{1}{\sqrt{w}}-\frac{1}{N}\right)}{\deg(v)^2} & \text{by Lemma 6} \\
I_v \geq \frac{H_v}{2} & \text{by Corollary 8} \\
L_v \geq \frac{I_v}{2} & \text{by Lemma 9}
\end{array} \tag{42}$$

Here the fourth inequality uses the fact that, setting $\varepsilon := \frac{1/2w}{1/\sqrt{w}} = \frac{1}{2\sqrt{w}}$, we have $\deg(v) = o\left(\frac{1}{\sqrt{\varepsilon}}\right)$ (thereby satisfying the hypothesis of Corollary 8). The fifth inequality uses the fact that, if we set $V(x) := v(x/w)$, then the situation satisfies the hypothesis of Lemma 9: we

are interested in the range of V on the interval $\left[\frac{1}{2}, \sqrt{w}\right]$, compared to its range on discrete points $\frac{w}{\sqrt{w}}, \frac{w}{\sqrt{w}+1}, \ldots, \frac{w}{2w}$ that are spaced at most 1 apart from each other; and we also have $\deg(V) = \deg(v) = o\left(w^{1/4}\right)$.

All that remains is to show that, if we insert either $G_u \geq \frac{1}{6}$ or $G_v \geq \frac{1}{6}$ into the coupled system of inequalities above, then we get unbounded growth and the inequalities have no solution. Let us collapse the two sets of inequalities to

$$L_u \geq \frac{1}{4}\frac{N-\sqrt{w}}{\deg(u)^2}\frac{G_u}{2w} = \Omega\left(\frac{N}{w\deg(u)^2}G_u\right),$$
$$L_v \geq \frac{1}{4}\frac{\frac{1}{\sqrt{w}}-\frac{1}{N}}{\deg(v)^2}G_v w = \Omega\left(\frac{\sqrt{w}}{\deg(v)^2}G_v\right).$$

Hence

$$G_u \geq L_v - 1 = \Omega\left(\frac{\sqrt{w}}{\deg(v)^2}G_v\right) - 1,$$
$$G_v \geq L_u - 1 = \Omega\left(\frac{N}{w\deg(u)^2}G_u\right) - 1.$$

By the assumption that $\deg(v) = o\left(w^{1/4}\right)$ and $\deg(u) = o\left(\sqrt{\frac{N}{w}}\right)$, we have $\frac{\sqrt{w}}{\deg(v)^2} \gg 1$ and $\frac{N}{w\deg(u)^2} \gg 1$. Plugging in $G_u \geq \frac{1}{6}$ or $G_v \geq \frac{1}{6}$, this is enough to give us unbounded growth. ◀

4.4 Lower bound using dual polynomials

In this section we use the method of dual polynomials to establish our main result, Theorem 4, restated for convenience:

▶ **Theorem 4.** *Let Q be a quantum algorithm that makes T queries to the membership oracle for S, and uses a total of R copies of $|S\rangle$ and reflections about $|S\rangle$. If Q decides whether $|S| = w$ or $|S| = 2w$ with high probability, promised that one of those is the case, then either*

$$T = \Omega\left(\sqrt{\frac{N}{w}}\right) \qquad \text{or} \qquad R = \Omega\left(\min\left\{w^{1/3}, \sqrt{\frac{N}{w}}\right\}\right). \tag{4}$$

Let $p(r)$ be a univariate Laurent polynomial of negative degree D_1 and positive degree D_2. That is, let $p(r)$ be of the form

$$p(r) = a_0/r^{D_1} + a_1/r^{D_1-1} + \cdots + a_{D_1-1}/r + a_{D_1} + a_{D_1+1}\cdot r + \cdots + a_{D_2+D_1}\cdot r^{D_2}. \tag{43}$$

Theorem 4 follows by combining the Laurent polynomial method (Lemma 21) and the following theorem.

▶ **Theorem 25.** *Let $\varepsilon < 1$. Suppose that p has negative degree D_1 and positive degree D_2 and satisfies the following properties.*

- $|p(w) - 1| \leq \varepsilon$
- $|p(2w) + 1| \leq \varepsilon$
- $|p(\ell)| \leq 1 + \varepsilon$ *for all* $\ell \in \{1, 2, \ldots, n\}$

Then either $D_1 \geq \Omega\left(w^{1/3}\right)$ or $D_2 \geq \Omega\left(\sqrt{N/w}\right)$.

In fact, our proof of Theorem 25 will show that the lower bound holds even if $|p(\ell)| \leq 1+\varepsilon$ only for $\ell \in \{w^{1/3}, w^{1/3}+1, \ldots, w\} \cup \{2w, 2w+1, \ldots, N\}$. We refer to a Laurent polynomial p satisfying the three properties of Theorem 25 as an *approximation for approximate counting*.

Proof of Theorem 25

Let p be any Laurent polynomial satisfying the hypothesis of Theorem 25. We begin by transforming p into a (standard) polynomial q in a straightforward manner. This transformation is captured in the following lemma, whose proof is so simple that we omit it.

▶ **Lemma 26.** *If p satisfies the properties of Theorem 25, then the polynomial $q(r) = p(r) \cdot r^{D_1} = a_0 + a_1 r + \cdots + a_{D_1+D_2} r^{D_1+D_2}$ is a (standard) polynomial of degree at most $D_1 + D_2$, and q satisfies the following three properties.*

- $\left|q(w) - w^{D_1}\right| \leq \varepsilon \cdot w^{D_1}$
- $\left|q(2w) + (2w)^{D_1}\right| \leq \varepsilon \cdot (2w)^{D_1}$
- $|q(\ell)| \leq (1+\varepsilon)\,\ell^{D_1}$ *for all* $\ell \in \{1, 2, \ldots, N\}$

We now turn to showing that, for any constant $\varepsilon < 1$, no polynomial q can satisfy the conditions of Lemma 26 unless $D_1 \geq \Omega(w^{1/3})$ or $D_2 \geq \Omega\left(\sqrt{N/w}\right)$.

Consider the following linear program. The variables of the linear program are ε, and the $D_2 + D_1 + 1$ coefficients of q.

$$\begin{array}{ll} \text{minimize} & \varepsilon \\ \text{such that} & \\ & |q(w) - w^{D_1}| \leq \varepsilon \cdot w^{D_1} \\ & |q(2w) + (2w)^{D_1}| \leq \varepsilon \cdot (2w)^{D_1} \\ & |q(\ell)| \leq (1+\varepsilon) \cdot \ell^{D_1} \text{ for all } \ell \in \{1, 2, \ldots, N\} \\ & \varepsilon \geq 0 \end{array} \tag{44}$$

Standard manipulations reveal the dual.

$$\begin{array}{ll} \text{maximize} & \phi(w) \cdot w^{D_1} - \phi(2w) \cdot (2w)^{D_1} - \sum_{\ell \in \{1,\ldots,N\}, \ell \notin \{w, 2w\}} |\phi(\ell)| \cdot \ell^{D_1} \\ \text{such that} & \\ & \sum_{\ell=1}^{N} \phi(\ell) \cdot \ell^j = 0 \text{ for } j = 0, 1, 2, \ldots, D_1 + D_2 \\ & \sum_{\ell=1}^{N} |\phi(\ell)| \cdot \ell^{D_1} = 1 \\ & \phi \colon \mathbb{R} \to \mathbb{R} \end{array} \tag{45}$$

Theorem 25 will follow if we can exhibit a solution ϕ to the dual linear program achieving value $\varepsilon > 0$, for some setting of $D_1 \geq \Omega(w^{1/3})$ and $D_2 \geq \Omega\left(\sqrt{N/w}\right)$.[13] We now turn to this task.

4.4.1 Constructing the dual solution

For a set $T \subseteq \{0, 1, \ldots, N\}$, define

$$Q_T(t) = \prod_{i=0,1,\ldots,N, i \notin T} (t - i). \tag{46}$$

[13] We will alternatively refer to such dual solutions ϕ as *dual witnesses*, since they act as a witness to the fact that any low-degree Laurent polynomial p approximating the approximate counting problem must have large error.

Let $c > 2$ be an integer constant that we will choose later (the bigger we choose c to be, the better the objective value achieved by our final dual witness. But choosing a bigger c will also lower the degrees D_1, D_2 of Laurent polynomials against which our lower bound will hold).

We now define two sets T_1 and T_2. The size of T_1 will be

$$d_1 := \lfloor (w/c)^{1/3} \rfloor = \Theta\left(w^{1/3}\right) \tag{47}$$

and the size of T_2 will be d_2 for

$$d_2 := \lfloor \sqrt{N/(cw)} \rfloor = \Theta\left(\sqrt{N/w}\right). \tag{48}$$

Let

$$T_1 = \left\{ \lfloor w/(ci^2) \rfloor : i = 1, 2, \ldots, d_1 \right\} \tag{49}$$

and

$$T_2 = \left\{ c \cdot i^2 \cdot w : i = 1, 2, \ldots, d_2 := \sqrt{N/(cw)} \right\}. \tag{50}$$

Finally, define

$$T = \{w, 2w\} \cup T_1 \cup T_2. \tag{51}$$

At last, define $\Phi : \{0, 1, \ldots, N\} \to \mathbb{R}$ via

$$\Phi(t) = (-1)^t \cdot \binom{N}{t} \cdot Q_T(t). \tag{52}$$

Our final dual solution ϕ will be a scaled version of Φ. Specifically, Φ itself does not satisfy the second constraint of the dual linear program, that $\sum_{\ell=1}^{N} |\Phi(\ell)| \cdot \ell^{D_1} = 1$. So letting

$$C = \sum_{\ell=1}^{N} |\Phi(\ell)| \cdot \ell^{D_1}, \tag{53}$$

our final dual witness ϕ will be Φ/C.

The sizes of T_1 and T_2

Clearly, under the above definition of T_2, $|T_2| = d_2$ as claimed above. It is not as immediately evident that $|T_1| = d_1$: to establish this, we must show that for distinct $i, j \in \{1, 2, \ldots, d_1\}$, $\lfloor w/(ci^2) \rfloor \neq \lfloor w/(cj^2) \rfloor$. This is handled in the following easy lemma.

▶ **Lemma 27.** *Let $i \neq j$ be distinct numbers in $\{1, \ldots, d_1\}$ and $c > 2$ be a constant. Then as long as $d_1 < (w/c)^{1/3}$, it holds that $\lfloor w/(ci^2) \rfloor \neq \lfloor w/(cj^2) \rfloor$.*

Proof. Assume without loss of generality that $i > j$. Then $w/(cj^2) - w/(ci^2)$ is clearly minimized when $i = d_1$ and $j = i - 1$. For the remainder of the proof, fix $i = d_1$. In this case,

$$\begin{aligned} w/(cj^2) - w/(ci^2) \geq w/\left(c(i-1)^2\right) - w/\left(ci^2\right) &= \frac{wi^2 - w\,(i-1)^2}{c \cdot i^2 \cdot (i-1)^2} \\ &= \frac{w}{c} \cdot \frac{2i-1}{i^2\,(i-1)^2} \geq \frac{w}{c} \cdot \frac{2i-1}{i^4} \geq \frac{w}{ci^3} \geq 1. \end{aligned} \tag{54}$$

Here, the final inequality holds because $i^3 = d_1^3 \leq w/c$.

Equation (54) implies the lemma, as two numbers whose difference is at least 1 cannot have the same integer floor. ◀

Lemma 27 is false for $d_1 = \omega(w^{1/3})$, highlighting on a technical level why one cannot choose d_1 larger than $\Theta(w^{1/3})$ without the entire construction and analysis of Φ breaking down.

4.4.2 Intuition: "gluing together" two simpler dual solutions

Before analyzing the dual witnesses Φ and ϕ constructed in Equation (52) and Equation (53), in this subsection and the next, we provide detailed intuition for why the definitions of Φ and ϕ are natural, and briefly overview their analysis.

A dual witness for purely positive degree (i.e., approximate degree)

Suppose we were merely interested in showing an approximate degree lower bound of $\Omega(\sqrt{N/w})$ for approximate counting (i.e., a lower bound on the degree of traditional polynomials that distinguish input w from $2w$, and are bounded at all other integer inputs in $1, \ldots, N$). This is equivalent to exhibiting a solution to the dual linear program with $D_1 = 0$. A valid dual witness ϕ_1 for this simpler case is to also use Equation (52), but to set

$$T = \{w, 2w\} \cup T_2, \tag{55}$$

rather than $T = \{w, 2w\} \cup T_1 \cup T_2$.

We will explain intuition for why Equation (55) is a valid dual solution for the approximate degree of approximate counting in the next subsection. For now, we wish to explain how this construction relates to prior work. In [18], for any constant $\delta > 0$, a dual witness is given for the fact that the $(1-\delta)$-approximate degree of OR is $\Omega(\sqrt{N})$. This dual witness *nearly* corresponds to the above, with $w = 1$. Specifically, Bun and Thaler [18] use the set $T = \{0, 1\} \cup \{ci^2 \colon i = 1, 2, \ldots, \sqrt{N/c}\}$, and they show that almost all of the "mass" of this dual witness is located on the inputs 0 and 1, i.e.,

$$|\Phi(0)| + |\Phi(1)| \geq (1-\delta) \cdot \sum_{i=2}^{N} |\Phi(i)| \,. \tag{56}$$

Here, the bigger c is chosen to be, the smaller the value of δ for which Equation (56) holds.

In the case of $w = 1$, our dual witness for approximate counting differs from this only in that $\{0, 1\}$ is replaced with $\{1, 2\}$. This is because, in order to show a lower bound for distinguishing input $w = 1$ from input $2w = 2$, we want almost all of the mass to be on inputs $\{1, 2\}$ rather than $\{0, 1\}$ (this is what will ensure that the objective function of the dual linear program is large).

For general w, we want most of the mass of ψ to be concentrated on inputs w and $2w$. Accordingly, relative to the $w = 1$ case, we effectively multiply *all* points in T by w, and one can show that this does not affect the calculation regarding concentration of mass.

A dual witness for purely negative degree

Now, suppose we were merely interested in showing that Laurent polynomials of *purely negative* degree require degree $\Omega(w^{1/3})$ to approximate the approximate counting problem. This is equivalent to exhibiting a solution to the dual linear program with $D_2 = 0$. Then a valid dual witness ϕ_2 for this simpler case is to also use Equation (52), but to set

$$T = \{w, 2w\} \cup T_1. \tag{57}$$

Again, we will give intuition for why this is a valid dual solution in the next subsection (Section 4.4.3). For now, we wish to explain how this construction relates to prior work. Essentially, the $\Omega\left(w^{1/3}\right)$-degree lower bound for Laurent polynomials with *only negative* powers was proved by Zhandry [57, Theorem 7.3]. Translating Zhandry's theorem into our setting is not entirely trivial, and he did not explicitly construct a solution to our dual linear program. However (albeit with significant effort), one can translate his argument to our setting to show that Equation (57) gives a valid dual solution to prove a lower bound against Laurent polynomials with only negative powers.

Gluing them together

The above discussion explains that the key ideas for constructing dual solutions ϕ_1, ϕ_2 witnessing degree lower bounds for Laurent polynomials of *only negative* or *only positive* powers were essentially already known, or at least can be extracted from prior work with enough effort. In this work, we are interested in proving lower bounds for Laurent polynomials with both positive and negative powers. Our dual solution Φ essentially just "glues together" the dual solutions that can be derived from prior work. By this, we mean that the set T of integer points on which our Φ is nonzero is the *union* of the corresponding sets for ϕ_1 and ϕ_2 individually. Moreover, this union is nearly disjoint, as the only points in the intersection of the two sets being unioned are w and $2w$.

Overview of the analysis

To show that we have constructed a valid solution to the dual linear program (Equation (45)), we must establish that (a) Φ is uncorrelated with every polynomial of degree at most $D_1 + D_2$ and (b) Φ is well-correlated with any function g that evaluates to $+1$ on input w, to -1 on input $2w$, and is bounded in $[-1, 1]$ elsewhere. In (b), the correlation is taken with respect to an appropriate weighting of the inputs, that on input $\ell \in [N]$ places mass proportional to ℓ^{D_1}.

The definition of Φ as a "gluing together" of ϕ_1 and ϕ_2 turns out, in a straightforward manner, to ensure that Φ is uncorrelated with polynomials of degree at $D_1 + D_2$. All that remains is to show that Φ is well-correlated with g under the appropriate weighting of inputs. This turns out to be technically demanding, but ultimately can be understood as stemming from the fact that ϕ_1 and ϕ_2 are individually well-correlated with g (albeit, in the case of ϕ_2, under a *different* weighting of the inputs than the weighting that is relevant for Φ).

4.4.3 Intuition via complementary slackness

We now attempt to lend some insight into why the dual witnesses ϕ_1 and ϕ_2 for the purely positive degree and purely negative degree take the form that they do. This section is deliberately slightly imprecise in places, and builds on intuition that has been put forth in prior works proving approximate degree lower bounds via dual witnesses [18, 55, 17].

Notice that ϕ_1 is precisely defined so that $\phi_1(i) = 0$ for any $i \notin \{w, 2w\} \cup T_2$, and similarly $\phi_2(i) = 0$ for any $i \notin \{w, 2w\} \cup T_1$. The intuition for why this is reasonable comes from complementary slackness, which states that an optimal dual witness should equal 0 except on inputs that correspond to primal constraints that are *made tight by an optimal primal solution*. By "constraints made tight by an optimal primal solution", we mean constraints that, for the optimal primal solution, hold with equality rather than (strict) inequality.

Unpacking that statement, this means the following. Suppose that q is an optimal solution to the primal linear program of Section 4.4, meaning it minimizes the error ε amongst all polynomials of the same same degree. The constraints made tight by q are

precisely those inputs ℓ at which q hits its "maximum error" (e.g., an input ℓ such that $|q(\ell)| = (1+\varepsilon)\cdot \ell^{D_1}$). We call these inputs *maximum-error* inputs for q. Complementary slackness says that there is an optimal solution to the dual linear program (Equation (45)) that equals 0 at all inputs that are not maximum-error inputs for q.

In both the purely positive degree case, and the purely negative degree case, we know roughly what primal optimal solutions q look like, and moreover we know what roughly their maximum-error points look like. In the first case, the maximum-error points are well-approximated by the points in T_2, and in the purely negative degree case, the maximum error points are well-approximated by the points in T_1. Let us explain.

Purely positive degree case

Let T_d be the degree d Chebyshev polynomial of the first kind. It can be seen that $P(\ell) = T_{\sqrt{N}}(1 + 2/N - \ell/N)$ satisfies $P(1) \geq 2$, while $|P(\ell)| \leq 1$ for $\ell = 2, 3, \ldots, N$. That is, up to scaling, P approximates the approximate counting problem for $w = 1$, and its known that its degree is within a constant factor of optimal.

It is known that the extreme points of T_d are of the following form, for $k = 1, \ldots, d$:

$$\cos\left(\frac{(2k-1)}{2d}\pi\right) \approx 1 - k^2/(2d^2), \tag{58}$$

where the approximation uses the Taylor expansion of the cosine function around 0. Equation (58) means that the extreme points of P are roughly those inputs ℓ such that $1 + 2/N - \ell/N \approx 1 - k^2/(2d^2)$, where $d = \sqrt{N}$. Such ℓ are roughly of the form $\ell \approx c \cdot i^2$ for some constant c, as i ranges from 1 up to $\Theta(N^{1/2})$.

More generally, when $w \geq 1$, an asymptotically optimal approximation for distinguishing input w from $2w$ is $P(\ell) = T_{\sqrt{N/w}}(1 + 2w/N - \ell/(wN))$. The extreme points of P are roughly of the form $\ell \approx c \cdot i^2 \cdot w$ for some constant c, as i ranges from 1 up to $\Theta(\sqrt{N/w})$, which is exactly the form of the points in our set T_2.

Purely negative degree case

In Lemma 22, we exhibited a simple, purely negative degree Laurent polynomial p (i.e., $p(\ell)$ is a standard polynomial in $1/\ell$) with degree $D_1 = w^{1/3}$ that solves the approximate counting problem (the construction is due to MathOverflow user "fedja"). Roughly speaking, p can be written as a product $p(\ell) = u(\ell) \cdot v(\ell)$, where $u(\ell)$ has the roots $\ell = 1, 2, \ldots, w^{1/3}$, and $v(\ell)$ is (an affine transformation) of a Chebyshev polynomial of degree $w^{1/3}$, applied to $1/\ell$. One can easily look at this construction and see that $p(\ell)$ outputs *exactly* the correct value on inputs $\{1, 2, \ldots, w^{1/3}\}$, so these are not maximum error points for p. Moreover, the analysis of the maximum error points for Chebyshev polynomials above can be applied to show that the maximum error points of p are roughly of the form ℓ such that $1/\ell = c \cdot i^2/w$ for some constant c, with i ranging from 1 up to $\Theta(w^{1/3})$. This means that the extreme points are roughly of the form $\ell \approx \frac{w}{ci^2}$, which is why our set T_1 consists of points of the form $\lfloor \frac{w}{ci^2} \rfloor$ (the floors are required because we are proving lower bounds against polynomials whose behavior is only constrained at integer inputs).

4.4.4 Analysis of the dual solution Φ

▶ **Lemma 28.** *Let $d_1 = |T_1|$ and $d_2 = |T_2|$. Then for any $j = 0, 1, \ldots, d_1 + d_2$, it holds that*

$$\sum_{\ell=1}^{N} \Phi(\ell) \cdot \ell^j = 0.$$

Proof. A basic combinatorial fact is that for any polynomial Q of degree at most $N-1$, the following identity holds:

$$\sum_{\ell=0}^{N} \binom{N}{\ell} (-1)^\ell Q(\ell) = 0. \tag{59}$$

Observe that for any $j \leq d_1 + d_2 + 1$,

$$Q_T(\ell) \cdot \ell^j \text{ is a polynomial in } \ell \text{ of degree at most } N-1. \tag{60}$$

Furthermore, $\Phi(0) = 0$, because $0 \notin T$. Hence

$$\sum_{\ell=0}^{N} \binom{N}{\ell} (-1)^\ell Q_T(\ell) \cdot \ell^j = \sum_{\ell=1}^{N} \binom{N}{\ell} (-1)^\ell Q_T(\ell) \cdot \ell^j. \tag{61}$$

Thus, we can calculate:

$$\begin{aligned}\sum_{\ell=1}^{N} \Phi(\ell) \cdot \ell^j &= \sum_{\ell=1}^{N} (-1)^\ell \cdot \binom{N}{\ell} \cdot Q_T(\ell) \cdot \ell^j \\ &= \sum_{\ell=0}^{N} (-1)^\ell \cdot \binom{N}{\ell} \cdot Q_T(\ell) \cdot \ell^j = 0.\end{aligned}$$

Here, the second equality follows from Equation (61), while the third follows from Equations (59) and (60). ◀

Let us turn to analyzing Φ's value on various inputs. Clearly the following condition holds:

$$\Phi(\ell) = 0 \text{ for all } \ell \notin T. \tag{62}$$

Next, observe that for any $r \in T$,

$$|\Phi(r)| = N! \cdot \frac{1}{\prod_{j \in T, j \neq r} |r - j|}.$$

Consider any quantity $c \cdot i^2 \cdot w \in T_2$. Then

$$\begin{aligned}\left|\Phi(c \cdot w \cdot i^2)\right| / |\Phi(w)| &= \frac{\prod_{j \in T, j \neq w} |w - j|}{\prod_{j \in T, j \neq c \cdot i^2 \cdot w} |w \cdot c \cdot i^2 - j|} \\ &= \frac{|w - 2w| \cdot \left(\prod_{j=1}^{d_2} \left|w - c \cdot j^2 \cdot w\right|\right) \cdot \left(\prod_{j=1}^{d_1} \left(w - \left\lfloor \tfrac{w}{cj^2} \right\rfloor\right)\right)}{|c \cdot i^2 \cdot w - w| \cdot |c \cdot i^2 \cdot w - 2w| \cdot \left(\prod_{j=1, j \neq i}^{d_2} |w \cdot c \cdot i^2 - w \cdot c \cdot j^2|\right) \cdot \left(\prod_{j=1}^{d_1} \left(w \cdot c \cdot i^2 - \left\lfloor \tfrac{w}{c \cdot j^2} \right\rfloor\right)\right)} \\ &= \frac{c^{d_2} \cdot \left(\prod_{j=1}^{d_2} \left(j^2 - \tfrac{1}{c}\right)\right) \cdot \prod_{j=1}^{d_1} \left(w - \left\lfloor \tfrac{w}{c \cdot j^2} \right\rfloor\right)}{(ci^2 - 1) \cdot (ci^2 - 2) \cdot c^{d_2 - 1} \cdot \left(\prod_{j=1, j \neq i}^{d_2} |i^2 - j^2|\right) \cdot \left(\prod_{j=1}^{d_1} \left(w \cdot c \cdot i^2 - \left\lfloor \tfrac{w}{c \cdot j^2} \right\rfloor\right)\right)} \\ &\leq \frac{c \cdot \left(\prod_{j=1}^{d_2} \left(j^2 - \tfrac{1}{c}\right)\right) \cdot \prod_{j=1}^{d_1} \left(w - \left\lfloor \tfrac{w}{c \cdot j^2} \right\rfloor\right)}{(ci^2 - 1) \cdot (ci^2 - 2) \cdot \left(\prod_{j=1, j \neq i}^{d_2} |i^2 - j^2|\right) \cdot \left(\prod_{j=1}^{d_1} \left(w \cdot c \cdot i^2 - \tfrac{w}{c \cdot j^2}\right)\right)}\end{aligned} \tag{63}$$

Now, observe that

$$\prod_{j=1}^{d_1}\left(w-\left\lfloor\frac{w}{c\cdot j^2}\right\rfloor\right)\leq\prod_{j=1}^{d_1}\left(w-\frac{w}{cj^2}+1\right)=\prod_{j=1}^{d_1}w\cdot\left(1-\frac{1}{cj^2}\right)\cdot\left(1+\frac{1}{w\cdot\left(1-\frac{1}{cj^2}\right)}\right)$$
$$\leq\prod_{j=1}^{d_1}w\cdot\left(1-\frac{1}{cj^2}\right)\left(1+\frac{1}{(1-1/c)\cdot w}\right)\leq\left(\prod_{j=1}^{d_1}w\cdot\left(1-\frac{1}{cj^2}\right)\right)\cdot(1+o(1))\,. \tag{64}$$

Hence, we see that Expression (63) is bounded by

$$\begin{aligned}
&\frac{c\cdot\left(\prod_{j=1}^{d_2}\left(j^2-\frac{1}{c}\right)\right)\cdot\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{c\cdot j^2}\right)\right)\cdot(1+o(1))}{(ci^2-1)\cdot(ci^2-2)\cdot\left(\prod_{j=1,j\neq i}^{d_2}|i^2-j^2|\right)\cdot\left(\prod_{j=1}^{d_1}\left(c\cdot i^2-\frac{1}{c\cdot j^2}\right)\right)}\\
&\leq\frac{c\cdot(d_2!)^2\cdot\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{c\cdot j^2}\right)\right)\cdot(1+o(1))}{(ci^2-1)\cdot(ci^2-2)\cdot\left(\prod_{j=1,j\neq i}^{d_2}|i-j|\,|i+j|\right)\cdot(c\cdot i^2)^{d_1}\cdot\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{c^2\cdot i^2\cdot j^2}\right)\right)}\\
&=\frac{c\cdot(d_2!)^2\cdot 2i^2\cdot\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{c\cdot j^2}\right)\right)\cdot(1+o(1))}{(ci^2-1)\cdot(ci^2-2)\cdot(d_2+i)!\,(d_2-i)!\cdot(c\cdot i^2)^{d_1}\cdot\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{c^2\cdot i^2\cdot j^2}\right)\right)}\\
&\leq\frac{c\cdot 2i^2\cdot(d_2!)^2\cdot(1+o(1))}{(ci^2-1)\,(ci^2-2)\cdot(d_2+i)!\,(d_2-i)!\cdot(c\cdot i^2)^{d_1}}\leq\frac{2\,(1+o(1))}{\left(1-\frac{1}{c\cdot i^2}\right)\cdot(c\cdot i^2-2)\cdot(c\cdot i^2)^{d_1}}.
\end{aligned} \tag{65}$$

In the penultimate inequality, we used the fact that $\frac{(d_2!)^2}{(d_2+i)!(d_2-i)!}=\frac{\binom{2d_2}{d_2+i}}{\binom{2d_2}{d_2}}\leq 1$.

Next, consider any quantity $\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\in T_1$. Then

$$\begin{aligned}
&\left|\Phi\left(\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\right)\right|/\left|\Phi(w)\right|\\
&=\frac{|w-2w|\left(\prod_{j=1}^{d_2}|w-cj^2w|\right)\left(\prod_{j=1}^{d_1}\left(w-\left\lfloor\frac{w}{cj^2}\right\rfloor\right)\right)}{\left(w-\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\right)\cdot\left(2w-\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\right)\left(\prod_{j=1}^{d_2}\left(w\cdot c\cdot j^2-\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\right)\right)\prod_{j=1,j\neq i}^{d_1}\left|\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor-\left\lfloor\frac{w}{c\cdot j^2}\right\rfloor\right|}\\
&\leq\frac{|w-2w|\left(\prod_{j=1}^{d_2}|w-cj^2w|\right)\left(\prod_{j=1}^{d_1}\left(w-\left\lfloor\frac{w}{cj^2}\right\rfloor\right)\right)}{\left(w-\frac{w}{c\cdot i^2}\right)\cdot\left(2w-\frac{w}{c\cdot i^2}\right)\left(\prod_{j=1}^{d_2}\left(w\cdot c\cdot j^2-\frac{w}{c\cdot i^2}\right)\right)\prod_{j=1,j\neq i}^{d_1}\left|\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor-\left\lfloor\frac{w}{c\cdot j^2}\right\rfloor\right|}\\
&\leq\frac{|w-2w|\left(\prod_{j=1}^{d_2}|w-cj^2w|\right)\left(\prod_{j=1}^{d_1}\left(w-\frac{w}{cj^2}\right)\right)\cdot(1+o(1))}{\left(w-\frac{w}{c\cdot i^2}\right)\cdot\left(2w-\frac{w}{c\cdot i^2}\right)\left(\prod_{j=1}^{d_2}\left(w\cdot c\cdot j^2-\frac{w}{c\cdot i^2}\right)\right)\prod_{j=1,j\neq i}^{d_1}\left|\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor-\left\lfloor\frac{w}{c\cdot j^2}\right\rfloor\right|}
\end{aligned} \tag{66}$$

Here, the final inequality used Equation (64).

Let us consider the expression $\prod_{j=1,j\neq i}^{d_1}\left|\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor-\left\lfloor\frac{w}{c\cdot j^2}\right\rfloor\right|$. This quantity is *at least*

$$\prod_{j=1,j\neq i}^{d_1}\left(\left|\frac{w}{c\cdot i^2}-\frac{w}{c\cdot j^2}\right|-1\right)=w^{d_1-1}\cdot\prod_{j=1,j\neq i}^{d_1}\frac{\left|j^2-i^2\right|-\frac{ci^2j^2}{w}}{ci^2j^2}$$
$$=w^{d_1-1}\cdot\prod_{j=1,j\neq i}^{d_1}\frac{|j-i|\cdot|j+i|-\frac{ci^2j^2}{w}}{ci^2j^2}$$
$$=\left(\frac{w}{ci^2}\right)^{d_1-1}\cdot\prod_{j=1,j\neq i}^{d_1}\frac{|j-i|\cdot|j+i|-\frac{ci^2j^2}{w}}{j^2} \tag{67}$$

We claim that Expression (67) is at least

$$\left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2}. \tag{68}$$

In the case that $c = 2$ and d_1 is (at most) $w^{1/3}$, this is precisely [57, Claim 4]. We will ultimately take c to be a constant strictly greater than 2 and hence $d_1 = \lfloor (w/c)^{1/3} \rfloor$ is a constant factor smaller than $w^{1/3}$. The proof of [57, Claim 4] works with cosmetic changes in this case. For completeness, we present a derivation of the claim in Appendix A.

Equation (68) implies that Expression (66) is at most:

$$\begin{aligned}
&\frac{|w-2w|\left(\prod_{j=1}^{d_2}|w-cj^2w|\right)\left(\prod_{j=1}^{d_1}\left(w-\frac{w}{cj^2}\right)\right)\cdot(1+o(1))}{\left(w-\frac{w}{c\cdot i^2}\right)\cdot\left(2w-\frac{w}{c\cdot i^2}\right)\left(\prod_{j=1}^{d_2}\left(w\cdot c\cdot j^2-\frac{w}{c\cdot i^2}\right)\right)\left(\frac{w}{ci^2}\right)^{d_1-1}\cdot\frac{1}{2}} \\
&\quad= \frac{2\left(\prod_{j=1}^{d_2}|1-cj^2|\right)\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{cj^2}\right)\right)\cdot(1+o(1))}{\left(1-\frac{1}{c\cdot i^2}\right)\cdot\left(2-\frac{1}{c\cdot i^2}\right)\left(\prod_{j=1}^{d_2}\left(c\cdot j^2-\frac{1}{c\cdot i^2}\right)\right)\left(\frac{1}{ci^2}\right)^{d_1-1}} \\
&\quad= \frac{2\left(\prod_{j=1}^{d_2}(j^2-1/c)\right)\left(\prod_{j=1}^{d_1}\left(1-\frac{1}{cj^2}\right)\right)\cdot(1+o(1))}{\left(1-\frac{1}{c\cdot i^2}\right)\cdot\left(2-\frac{1}{c\cdot i^2}\right)\left(\prod_{j=1}^{d_2}\left(j^2-\frac{1}{c^2\cdot i^2}\right)\right)\left(\frac{1}{ci^2}\right)^{d_1-1}} \\
&\quad\le \frac{2\,(1+o(1))}{\left(1-\frac{1}{c\cdot i^2}\right)\cdot\left(2-\frac{1}{c\cdot i^2}\right)\left(\frac{1}{ci^2}\right)^{d_1-1}} \le 4\cdot\left(ci^2\right)^{d_1-1}.
\end{aligned} \tag{69}$$

Summarizing Equations (65) and (69), we have shown that: for any quantity $c\cdot i^2\cdot w \in T_2$,

$$\left|\Phi(c\cdot w\cdot i^2)\right| / \left|\Phi(w)\right| \le \frac{2\,(1+o(1))}{\left(1-\frac{1}{c\cdot i^2}\right)\cdot(c\cdot i^2-2)\cdot(c\cdot i^2)^{d_1}} \tag{70}$$

and for any quantity $\left\lfloor \frac{w}{c\cdot i^2} \right\rfloor \in T_1$,

$$\left|\Phi\left(\left\lfloor\frac{w}{c\cdot i^2}\right\rfloor\right)\right| / \left|\Phi(w)\right| \le 4\cdot\left(ci^2\right)^{d_1-1}. \tag{71}$$

Let $\phi = \Phi/C$, where C is as in Equation (53). Let $D_1 = d_1$ and $D_2 = d_2$. Lemma 28 implies that ϕ is a feasible solution for the dual linear program of Section 4.4.1. We now show that, for any constant $\delta > 0$, by choosing c to be a sufficiently large constant (that depends on δ), we can ensure that ϕ achieves objective value $1 - 2\delta$.

Let

$$\begin{aligned}
A &= |\Phi(w)|\cdot w^{D_1}, \\
B &= |\Phi(2w)|\cdot(2w)^{D_1},
\end{aligned}$$

and

$$E = \sum_{i=1}^{d_1}|\Phi(\lfloor w/ci^2\rfloor)|\cdot\left(\lfloor w/ci^2\rfloor\right)^{D_1} + \sum_{i=1}^{d_2}|\Phi(\lfloor w\cdot ci^2\rfloor)|\cdot\left(w\cdot c\cdot i^2\right)^{D_1}.$$

By Equation (62), $C = A + B + E$.

Moreover, observe that $\mathrm{sgn}(\Phi(w)) = -\mathrm{sgn}(\Phi(2w))$, so without loss of generality we may assume $\Phi(w) \ge 0$ and $\Phi(2w) \le 0$ (if not, then replace Φ with $-\Phi$ throughout).

We now claim that by choosing c to be a sufficiently large constant, we can ensure that $E \leq \delta \cdot A$. To see this, observe that Equations (70) and (71), along with the fact that $D_1 = d_1$ and $D_2 = d_2$ implies that

$$\begin{aligned} E/A &\leq \frac{1}{w^{D_1}} \left[\left(\sum_{i=1}^{d_1} \left(\lfloor w/ci^2 \rfloor \right)^{D_1} \cdot 4 \cdot \left(ci^2 \right)^{d_1 - 1} \right) + \left(\sum_{i=1}^{d_2} \left(w \cdot c \cdot i^2 \right)^{D_1} \frac{2\left(1 - \frac{1}{c \cdot i^2}\right)(1 + o(1))}{(c \cdot i^2 - 2) \cdot (c \cdot i^2)^{d_1}} \right) \right] \\ &\leq \frac{1}{w^{D_1}} \left[\left(\sum_{i=1}^{d_1} \left(w/ci^2 \right)^{D_1} \cdot 4 \cdot \left(ci^2 \right)^{d_1 - 1} \right) + \left(\sum_{i=1}^{d_2} \left(w \cdot c \cdot i^2 \right)^{D_1} \frac{2\left(1 - \frac{1}{c \cdot i^2}\right)(1 + o(1))}{(c \cdot i^2 - 2) \cdot (c \cdot i^2)^{d_1}} \right) \right] \\ &\leq 4 \left(\sum_{i=1}^{d_1} \frac{1}{c \cdot i^2} \right) + \left(\sum_{i=1}^{d_2} \frac{2(1 + o(1))}{\left(1 - \frac{1}{c \cdot i^2}\right)(c \cdot i^2 - 2)} \right) \end{aligned}$$

Since $\sum_{i=1}^{\infty} 1/(ci^2) \leq \frac{\pi^2}{6c}$, we see that choosing c to be a sufficiently large constant depending on δ ensures that $E/A \leq \delta$ as desired.

Hence, ϕ achieves objective value at least

$$\begin{aligned} &\phi(w) \cdot w^{D_1} - \phi(2w) \cdot (2w)^{D_1} - \sum_{\ell \in \{1,\ldots,N\}, \ell \notin \{w, 2w\}} |\phi(\ell)| \cdot \ell^{D_1} \\ &\qquad \geq \frac{A + B - E}{A + B + E} \geq \frac{(1 - \delta)A + B}{(1 + \delta)A + B} \geq 1 - 2\delta. \end{aligned}$$

4.5 Approximate counting with classical samples

For completeness, in this section, we sketch classical counterparts of Theorem 4 and Theorem 5. That is, we show tight bounds on classical randomized algorithms for $\mathsf{ApxCount}_{N,w}$ that make membership queries and have access to uniform random samples from the set being counted.

▶ **Proposition 29.** *There is a classical randomized algorithm that solves* $\mathsf{ApxCount}_{N,w}$ *with high probability using either* $O(N/w)$ *queries to the membership oracle for* S*, or else using* $O(\sqrt{w})$ *uniform samples from* S*.*

Proof sketch. By reducing approximate counting to the problem of estimating the mean of a biased coin, $O(N/w)$ queries are sufficient.

Alternatively, if we take R samples, then the expected number of birthday collisions is $\binom{R}{2} \cdot \frac{1}{|S|}$ and the variance is $\binom{R}{2} \cdot \frac{1}{|S|}\left(1 - \frac{1}{|S|}\right)$. So, taking $O(\sqrt{w})$ samples and computing the number of birthday collisions is sufficient to distinguish $|S| \leq w$ from $|S| \geq 2w$ with $\frac{2}{3}$ success probability. ◀

▶ **Proposition 30.** *Let* M *be a classical randomized algorithm that makes* T *queries to the membership oracle for* S*, and takes a total of* R *uniform samples from* S*. If* M *decides whether* $|S| = w$ *or* $|S| = 2w$ *with high probability, promised that one of those is the case, then either* $T = \Omega(N/w)$ *or* $R = \Omega(\sqrt{w})$*.*

Proof sketch. Note that without loss of generality, we may assume that the algorithm first takes all of the samples it needs, and then queries random elements of $[N]$ that did not appear in the samples. Suppose the algorithm takes $R = o(\sqrt{w})$ samples and then makes $T = o(N/w)$ queries. Consider what happens when the algorithm tries to distinguish a random subset of size w from a random subset of size $2w$ of $[N]$. By a union bound, the probability that the algorithm sees any collisions in the samples is $o(1)$, and the probability that the algorithm finds any additional elements of S via queries is also $o(1)$. So, if the set

has size either w or $2w$, with $1 - o(1)$ probability, the algorithm's view of the samples is just a random subset of size R of $[N]$ drawn without replacement, and the algorithm's view of the queries is just T "no" answers to membership queries. Hence, the algorithm fails to distinguish random sets of size w and size $2w$ with any constant probability of success. ◀

4.6 Extending the lower bound to QSampling unitarily

So far in this section we have proved upper and lower bounds on the power of quantum algorithms for approximate counting that have access to two resources (in addition to membership queries): copies of $|S\rangle$, and the unitary transformation that reflects about $|S\rangle$. The assumption of access to the reflection unitary is justified by the argument that, if we had access to a unitary that prepared $|S\rangle$, then it could be used to reflect about $|S\rangle$ as well.

Giving the algorithm access to just the two resources above is an appealing model to use for upper bounds, since it does not assume anything about the method by which copies of $|S\rangle$ are prepared. This means algorithms derived in this model work in many different settings. For example, the algorithm may be able to QSample because someone else simply handed the algorithm copies of $|S\rangle$, or perhaps several copies of $|S\rangle$ just happen to be stored in the algorithm's quantum memory as a side effect of the execution of some earlier quantum algorithm. The upper bound given in Theorem 5 applies in any of these settings.

On the other hand, since only permitting access to QSamples and reflections about $|S\rangle$ ties the algorithm's hands, lower bounds for this model (e.g., Theorem 4) could be viewed as weaker than is desirable. In particular, our original justification for allowing access to reflections about $|S\rangle$ was that access to a unitary that prepared the state $|S\rangle$ would in particular allow such reflections to be done. Given this justification, it is very natural to wonder whether our lower bounds extend beyond just QSamples and reflections, to algorithms that are given access to *some* unitary process that permits both QSampling and reflections about $|S\rangle$.

Note that an algorithm with access to such a unitary could potentially exploit the unitary in ways other than QSamples and reflections to learn information about $|S\rangle$. For example, the algorithm could choose to run the unitary on inputs that do not produce $|S\rangle$. More generally, given a quantum circuit that implements a unitary, it is possible to construct, in a completely black-box manner, the inverse of this unitary, and also a controlled version of the unitary. The algorithm may choose to run the inverse on a state other than $|S\rangle$ to learn some additional information that is not captured by access to QSamples and reflections alone.

In summary, in this section we ask whether we can we extend the lower bound of Theorem 4 to work in a model where the algorithm is given access to some unitary operator that conveys the power to both QSample and reflect about $|S\rangle$.[14] Via Theorem 31 below, we explain that the answer is yes.

It may seem convenient to assume that the unitary transformation preparing $|S\rangle$ maps the all-zeros state to $|S\rangle$. But this is not the most general method of preparing $|S\rangle$ by a unitary. A unitary U that maps the all-zeros state to $|S\rangle|\psi\rangle$ would also suffice to create copies of $|S\rangle$, since the register containing $|\psi\rangle$ can simply be ignored for the remainder of the computation. More formally, assume U behaves as

$$U|0^m\rangle = |S\rangle|\psi\rangle, \tag{72}$$

[14] We thank Alexander Belov (personal communication) for raising this question.

where $|S\rangle|\psi\rangle$ is some m-qubit state. Clearly we can use U to create as many copies of $|S\rangle$ as we like, which as a by-product also creates copies of $|\psi\rangle$. This unitary also lets us reflect about $|S\rangle$. To see how, first use this unitary to create a copy of $|\psi\rangle$, and then consider the action of the unitary $U(\mathbb{1} - 2|0^m\rangle\langle 0^m|)U^\dagger$ on the state $|\phi\rangle|\psi\rangle$ for any state $|\phi\rangle$. We claim that this unitary acts as a reflection about $|S\rangle$ when restricted to the first register. This establishes that any U of this form subsumes the power of both QSamples and reflections about $|S\rangle$.

Let us also assume without loss of generality that $|S\rangle|\psi\rangle$ is orthogonal to $|0^m\rangle$ from now on. This can be achieved by adding an additional qubit to the input that is always negated by the unitary. That is, we could instead consider the map $(U \otimes X)|0^m\rangle|0\rangle = |S\rangle|\psi\rangle|1\rangle$, which is orthogonal to the starting state by construction, and only increases the value of m by 1.

Of course, the requirement that $U|0^m\rangle = |S\rangle|\psi\rangle$ does not fully specify U, as it does not prescribe how U behaves on other input states. A reasonable prescription is that U should behave "trivially" on other input states, so that it does not leak information about S by its behavior on other states. In tension with this prescription is the fact the rest of the unitary must depend on S, since the first column of the unitary contains $|S\rangle$, and the rest of the columns have to be orthogonal to this.

Alexander Belov (personal communication) brought to our attention a very simple construction of such a unitary that leaks minimal additional information about S. Consider the unitary U that satisfies $U|0^m\rangle = |S\rangle|\psi\rangle$ and $U|S\rangle|\psi\rangle = |0^m\rangle$, with U acting as identity outside $\operatorname{span}\{|0^m\rangle, |S\rangle|\psi\rangle\}$. U is simply a reflection about the state $\frac{1}{\sqrt{2}}\big(|0^m\rangle - |S\rangle|\psi\rangle\big)$. This state is correctly normalized because we assumed that $|S\rangle|\psi\rangle$ is orthogonal to $|0^m\rangle$. Clearly U is now fully specified on the entire domain (once we have fixed $|\psi\rangle$) and it does not seem to leak any additional information about S.

In order to prove concrete lower bounds on the cost of algorithms for approximate counting given access to U, we need to fix $|\psi\rangle$. To answer the question posed in this section, we only need to establish that there exists *some* choice of $|\psi\rangle$ for which our algorithms cannot be improved. (Note that we cannot hope to establish lower bounds for arbitrary $|\psi\rangle$, since $|\psi\rangle$ could just contain the answer to the problem we are solving.)

To this end we make the specific choice of $|\psi\rangle = |S\rangle$ and consider the unitary V that acts as the unitary U above with $|\psi\rangle = |S\rangle$. In other words, V maps $|0^m\rangle$ to $|S\rangle|S\rangle$, $|S\rangle|S\rangle$ to $|0^m\rangle$, and acts as identity on the rest of the space. We also assume that $|0^m\rangle$ is orthogonal to $|S\rangle|S\rangle$. In other words, V simply reflects about the state $\frac{1}{\sqrt{2}}\big(|0^m\rangle - |S\rangle|S\rangle\big)$.

As previously discussed, granting an algorithm access to this unitary V lends the algorithm at least as much power the ability to QSample and perform reflections about $|S\rangle$. How efficiently can we solve approximate counting with membership queries and uses of the unitary V?

We can use our Laurent polynomial method to establish optimal lower bounds in this model as well and we obtain lower bounds identical to Theorem 4.

▶ **Theorem 31.** *Let Q be a quantum algorithm that makes T queries to the membership oracle for S, and makes R uses of the unitary V defined above (and its inverse and controlled-V). If Q decides whether $|S| = w$ or $|S| = 2w$ with high probability, promised that one of those is the case, then either*

$$T = \Omega\left(\sqrt{\frac{N}{w}}\right) \qquad or \qquad R = \Omega\left(\min\left\{w^{1/3}, \sqrt{\frac{N}{w}}\right\}\right). \tag{73}$$

Proof. We follow the same strategy as in the proof of Theorem 4. Recall that $x \in \{0,1\}^N$ denotes the indicator vector of the set S. We only need to show that such a quantum algorithm gives rise to a Laurent polynomial in $|S| := \sum_{i=1}^{n} x_i$, with maximum exponent $O(T+R)$ and minimum exponent at least $-O(R)$ (as shown in Lemma 21 for the QSamples and reflections model).

We can prove this exactly the same way as Lemma 21 is established. Our quantum algorithm starts out from a canonical starting state that does not depend on the input and hence each entry of the starting state is a degree-0 polynomial. Membership queries involve multiplication with an oracle whose entries are ordinary polynomials of degree at most 1. The only thing that remains is understanding what the entries of the unitary V look like. We claim that the entries of V are given by a polynomial of degree at most 2 in the entries of the input x, with all coefficients of this degree-2 polynomial equal to either a constant, or a constant multiple of $|S|^{-1}$.

To see this, note that V is simply a reflection about the state

$$\frac{1}{\sqrt{2}}\left(|0^m\rangle - |S\rangle|S\rangle\right) = \frac{1}{\sqrt{2}}\left(|0^m\rangle - \frac{1}{|S|}\left(\sum_i x_i|i\rangle\right)\left(\sum_j x_j|j\rangle\right)\right). \tag{74}$$

The coefficient in front of $|0^m\rangle$ is a degree-0 polynomial and the other nonzero coefficients are a polynomial of degree at most 2 in the entries of the input x, with each coefficient of this polynomial equal to a constant multiple of $|S|^{-1}$.

Hence, each entry of the unitary V is also a polynomial of degree at most 2 in the entries of the input x, with each coefficient of this degree-2 polynomial equal to either a constant, or a constant multiple of $|S|^{-1}$. The same also holds for controlled-V, since that unitary is just the direct sum of identity with V. V is also self-inverse, so we do not need to account for that separately.

After the algorithm has made all the membership queries and uses of V, each amplitude of the final quantum state can be expressed as a polynomial of degree $O(T+R)$ in the input x, in which all coefficients are constant multiples of $|S|^{-R}$. The acceptance probability $p(x)$ of this algorithm will be a sum of squares of such polynomials. Exactly as in the proof of Theorem 4, Lemma 11 implies that there is a univariate polynomial q of degree at most $O(T+R)$, with coefficients that are multiples of the coefficients of p, such that for all integers $k \in \{0,\ldots,N\}$,

$$q(k) := \mathbb{E}_{|X|=k}[p(X)]. \tag{75}$$

Since the coefficients of $p(X)$ are constant multiples of $|X|^{-2R}$, q is in fact a real Laurent polynomial in k, with maximum exponent at most $O(R+T)$ and minimum exponent at least $-2R$. The theorem follows by a direct application Theorem 25 to q. ◀

5 Discussion and open problems

5.1 Approximate counting with QSamples and queries only

If we consider the model where we only have membership queries and samples (but no reflections), then the best upper bound we can show is $O\left(\min\left\{\sqrt{w}, \sqrt{N/w}\right\}\right)$, using the sampling algorithm that looks for birthday collisions, and the quantum counting algorithm. It would be interesting to improve the lower bound further in this case, but it is clear that the Laurent polynomial approach cannot do so, since it hits a limit at $w^{1/3}$. Hence a new approach is needed to tackle the model without reflections.

We now give what we think is a viable path to solve this problem. Specifically, we observe that our problem – of lower-bounding the number of copies of $|S\rangle$ *and* the number of queries to $\mathcal{O}_S$ needed for approximate counting of S – can be reduced to a pure problem of lower-bounding the number of copies of $|S\rangle$. To do so, we use a hybrid argument, closely analogous to an argument recently given by Zhandry [58] in the context of quantum money.

Given a subset $S \subseteq [L]$, let $|S\rangle$ be a uniform superposition over S elements. Then let

$$\rho_{L,w,k} := \mathbb{E}_{S \subseteq [L] \,:\, |S|=w} \left[\left(|S\rangle \langle S| \right)^{\otimes k} \right] \tag{76}$$

be the mixed state obtained by first choosing S uniformly at random subject to $|S| = w$, then taking k copies of $|S\rangle$. Given two mixed states ρ and σ, recall also that the *trace distance*, $\|\rho - \sigma\|_{\mathrm{tr}}$, is the maximum bias with which ρ can be distinguished from σ by a single-shot measurement.

▶ **Theorem 32.** *Let $2w \leq L \leq N$. Suppose $\|\rho_{L,w,k} - \rho_{L,2w,k}\|_{\mathrm{tr}} \leq \frac{1}{10}$. Then any quantum algorithm Q requires either $\Omega\left(\sqrt{\frac{N}{L}}\right)$ queries to $\mathcal{O}_S$ or else $\Omega(k)$ copies of $|S\rangle$ to decide whether $|S| = w$ or $|S| = 2w$ with success probability at least $2/3$, promised that one of those is the case.*

Proof. Choose a subset $S \subseteq [N]$ uniformly at random, subject to $|S| = w$ or $|S| = 2w$, and consider S to be fixed. Then suppose we choose $U \subseteq [N]$ uniformly at random, subject to both $|U| = L$ and $S \subseteq U$. Consider the hybrid in which Q is still given R copies of the state $|S\rangle$, but now gets oracle access to $\mathcal{O}_U$ rather than $\mathcal{O}_S$. Then so long as Q makes $o\left(\sqrt{\frac{N}{L}}\right)$ queries to its oracle, we claim that Q cannot distinguish this hybrid from the "true" situation (i.e., the one where Q queries $\mathcal{O}_S$) with $\Omega(1)$ bias. This claim follows almost immediately from the BBBV Theorem [10]. In effect, Q is searching the set $[N] \setminus S$ for any elements of $U \setminus S$ (the "marked items," in this context), of which there are $L - |S|$ scattered uniformly at random. In such a case, we know that $\Omega\left(\sqrt{\frac{N-|S|}{L-|S|}}\right) = \Omega\left(\sqrt{\frac{N}{L}}\right)$ quantum queries are needed to detect the marked items with constant bias.

Next suppose we first choose $U \subseteq [N]$ uniformly at random, subject to $|U| = L$, and consider U to be fixed. We then choose $S \subseteq U$ uniformly at random, subject to $|S| = w$ or $|S| = 2w$. Note that this produces a distribution over (S, U) pairs identical to the distribution that we had above. In this case, however, since U is fixed, queries to $\mathcal{O}_U$ are no longer relevant. The only way to decide whether $|S| = w$ or $|S| = 2w$ is by using our copies of $|S\rangle$ – of which, by assumption, we need $\Omega(k)$ to succeed with constant bias, even after having fixed U. ◀

One might think that Theorem 32 would lead to immediate improvements to our lower bound for the queries and samples model. In practice, however, the best lower bounds that we currently have, even purely on the number of copies of $|S\rangle$, come from the Laurent polynomial method (Theorem 4)! Having said that, we are optimistic that one could obtain a lower bound that beats Theorem 4 at least when w is small, by combining Theorem 32 with a brute-force computation of trace distance.

5.2 Approximate counting to multiplicative factor $1 + \varepsilon$

Throughout, we considered the task of approximating $|S|$ to within a multiplicative factor of 2. But suppose our task was to distinguish the case $|S| \leq w$ from the case $|S| \geq (1 + \varepsilon) w$; then what is the optimal dependence on ε?

In the model with quantum membership queries only, the algorithm of Brassard et al. [13, Theorem 15] makes $O\left(\frac{1}{\varepsilon}\sqrt{\frac{N}{w}}\right)$ queries, which is optimal [40]. The algorithm uses amplitude amplification, the basic primitive of Grover's search algorithm [28]. The original algorithm of Brassard et al. also used quantum phase estimation, in effect *combining* Grover's algorithm with Shor's period-finding algorithm. However, one can remove the phase estimation, and adapt Grover search with an unknown number of marked items to get an approximate count of the number of marked items [5].

One can also show without too much difficulty that in the queries+QSamples model, the problem can be solved with

$$O\left(\min\left\{\frac{\sqrt{w}}{\varepsilon^2}, \frac{1}{\varepsilon}\sqrt{\frac{N}{w}}\right\}\right) \tag{77}$$

queries and copies of $|S\rangle$. As observed after Theorem 5, the problem can also be solved with

$$O\left(\min\left\{\frac{w^{1/3}}{\varepsilon^{2/3}}, \frac{1}{\varepsilon}\sqrt{\frac{N}{w}}\right\}\right) \tag{78}$$

samples and reflections. On the lower bound side, what generalizations of Theorem 4 can we prove that incorporate ε? We note that the explosion argument doesn't automatically generalize; one would need to modify something to continue getting growth in the polynomials u and v after the first iteration. The lower bound using dual polynomials should generalize, but back-of-the-envelope calculations show that the lower bound does not match the upper bound.

5.3 Other questions

Non-oracular example of our result

Is there any interesting real-world example of a class of sets for which QSampling and membership testing are both efficient, but approximate counting is not? (I.e., is there an interesting non-black-box setting that appears to exhibit the behavior that this paper showed can occur in the black-box setting?)

The Laurent polynomial connection

At a deeper level, is there is any meaningful connection between our two uses of Laurent polynomials? And what other applications can be found for the Laurent polynomial method?

6 Followup work

Since this work was completed, Belovs and Rosmanis [9] obtained essentially tight lower bounds on the complexity of approximate counting with access to membership queries, QSamples, reflections, and a unitary transformation that prepares the QSampling state, for all possible tradeoffs between these different resources. Additionally, they resolve the ε-dependence of approximate counting to multiplicative factor $1+\varepsilon$. The techniques involved are quite different from ours: Belovs and Rosmanis use a generalized version of the quantum adversary bound that allows for multiple oracles, combined with tools from the representation theory of the symmetric group.

References

1. Scott Aaronson. Quantum lower bound for the collision problem. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing - STOC 2002*, pages 635–642, 2002. quant-ph/0111102. doi:10.1145/509907.509999.
2. Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005. Earlier version in CCC'2004. quant-ph/0402095. doi:10.4086/toc.2005.v001a001.
3. Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2063):3473–3482, 2005. quant-ph/0412187. doi:10.1098/rspa.2005.1546.
4. Scott Aaronson. Impossibility of succinct quantum proofs for collision-freeness. *Quantum Info. Comput.*, 12(1-2):21–28, January 2012. URL: http://dl.acm.org/citation.cfm?id=2231036.2231039.
5. Scott Aaronson and Patrick Rall. Quantum approximate counting, simplified, 2019. arXiv:1908.10846.
6. Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595–605, 2004. doi:10.1145/1008731.1008735.
7. Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing - STOC 2003*, pages 20–29, 2003. quant-ph/0301023. doi:10.1145/780542.780546.
8. Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'1998, pp. 352-361. quant-ph/9802049. doi:10.1145/502090.502097.
9. Aleksandrs Belovs and Ansis Rosmanis. Tight quantum lower bound for approximate counting with quantum states. *arXiv preprint*, 2020. arXiv:2002.06879.
10. Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. quant-ph/9701001. doi:10.1137/S0097539796300933.
11. Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. *Journal of Computer and System Sciences*, 72(6):1043–1076, 2006. doi:10.1016/j.jcss.2006.05.001.
12. Adam D. Bookatz. QMA-complete Problems. *Quantum Info. Comput.*, 14(5&6):361–383, April 2014. URL: http://dl.acm.org/citation.cfm?id=2638661.2638662.
13. Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation*, volume 305, pages 53–74. American Mathematical Society, 2002. doi:10.1090/conm/305/05215.
14. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *Automata, Languages and Programming*, pages 820–831, 1998. arXiv:quant-ph/9805082. doi:10.1007/bfb0055105.
15. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN'98: Theoretical Informatics*, pages 163–169, 1998. doi:10.1007/BFb0054319.
16. Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21–43, 2002. doi:10.1016/s0304-3975(01)00144-x.
17. Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 297–310, 2018. doi:10.1145/3188745.3188784.
18. Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and Markov-Bernstein inequalities. In *Automata, Languages, and Programming*, pages 303–314, 2013. doi:10.1007/978-3-642-39206-1_26.

19 Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. In *Automata, Languages, and Programming*, pages 268–280, 2015. doi:10.1007/978-3-662-47672-7_22.

20 Don Coppersmith and T. J. Rivlin. The growth of polynomials bounded at equally spaced points. *SIAM J. Math. Anal.*, 23(4):970–983, July 1992. doi:10.1137/0523054.

21 Martin Dyer, Alan Frieze, and Ravi Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38(1):1–17, January 1991. Earlier version in STOC'1989. doi:10.1145/102782.102783.

22 H. Ehlich and K. Zeller. Schwankung von Polynomen zwischen Gitterpunkten. *Mathematische Zeitschrift*, 86:41–44, 1964. doi:10.1007/BF01111276.

23 Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.

24 Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6(2):97–116, 1993.

25 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

26 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016. doi:10.1137/15M103145X.

27 Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. arXiv:quant-ph/0208112.

28 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC 1996*, pages 212–219, 1996. quant-ph/9605043. doi:10.1145/237814.237866.

29 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *Journal of the ACM*, 51(4):671–697, 2004. Earlier version in STOC'2001. doi:10.1145/1008731.1008738.

30 Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J. Yoder. Hamiltonian simulation with optimal sample complexity. *npj Quantum Information*, 3(1), 2017. doi:10.1038/s41534-017-0013-7.

31 Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM Journal on Computing*, 36(5):1472–1493, 2007. Earlier version in FOCS'2004. quant-ph/0402123. doi:10.1137/05063235X.

32 William Kretschmer. Lower bounding the AND-OR tree via symmetrization, 2019. arXiv:1907.06731.

33 Greg Kuperberg. How hard is it to approximate the Jones polynomial? *Theory of Computing*, 11(6):183–219, 2015. doi:10.4086/toc.2015.v011a006.

34 Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014. arXiv:1307.0401. doi:10.1038/nphys3029.

35 Andrei Andreyevich Markov. On a question by DI Mendeleev. *Zapiski Imperatorskoi Akademii Nauk*, 62:1–24, 1890.

36 Chris Marriott and John Watrous. Quantum Arthur–Merlin games. *Computational Complexity*, 14(2):122–152, June 2005. doi:10.1007/s00037-005-0194-x.

37 Daniele Micciancio and Salil P Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *Annual International Cryptology Conference*, pages 282–298. Springer, 2003.

38 Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science*, pages 71–80, October 1999. doi:10.1109/SFFCS.1999.814579.

39 Marvin Minsky and Seymour A. Papert. *Perceptrons (2nd edition)*. MIT Press, 1988. First appeared in 1968.

40 Ashwin Nayak and Felix Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 384–393, New York, NY, USA, 1999. ACM. doi:10.1145/301250.301349.

41 Donald J. Newman. Rational approximation to $|x|$. *The Michigan Mathematical Journal*, 11(1):11–14, 1964. doi:10.1307/mmj/1028999029.

42 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, USA, 2014.

43 Ramamohan Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing - STOC 1992*, pages 468–474, 1992. doi:10.1145/129712.129758.

44 Chris Peikert and Vinod Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *Annual International Cryptology Conference*, pages 536–553. Springer, 2008.

45 Ran Raz and Amir Shpilka. On the power of quantum proofs. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 260–274. IEEE, 2004.

46 T. J. Rivlin and E. W. Cheney. A comparison of uniform approximations on an interval and a finite subset thereof. *SIAM Journal on Numerical Analysis*, 3(2):311–320, 1966. doi:10.1137/0703024.

47 Rocco Servedio, Li-Yang Tan, and Justin Thaler. Attribute-efficient learning andweight-degree tradeoffs for polynomial threshold functions. In *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 14.1–14.19, 2012. URL: http://proceedings.mlr.press/v23/servedio12.html.

48 Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 343–362, Washington, DC, USA, 2009. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=1747597.1748051.

49 Alexander A. Sherstov. Optimal bounds for sign-representing the intersection of two halfspaces by polynomials. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 523–532, New York, NY, USA, 2010. ACM. doi:10.1145/1806689.1806762.

50 Alexander A Sherstov and Justin Thaler. Vanishing-error approximate degree and QMA complexity. *arXiv preprint*, 2019. arXiv:1909.07498.

51 Yaoyun Shi. Approximating linear restrictions of boolean functions. Available at http://web.eecs.umich.edu/~shiyy/mypapers/, 2002.

52 Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133, 1989. doi:10.1016/0890-5401(89)90067-9.

53 Robert Špalek. A dual polynomial for OR. *CoRR*, abs/0803.4516, 2008. arXiv:0803.4516.

54 Larry Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985. doi:10.1137/0214060.

55 Justin Thaler. Lower Bounds for the Approximate Degree of Block-Composed Functions. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.17.

56 Nikolai K. Vereshchagin. On the power of PP. In *Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 138–143, 1992. doi:10.1109/SCT.1992.215389.

57 Mark Zhandry. How to construct quantum random functions. In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 679–687. IEEE, 2012. doi:10.1109/FOCS.2012.37.

58 Mark Zhandry. Quantum lightning never strikes the same state twice. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 408–438. Springer, 2019.

A Establishing Equation 68

A.1 A clean calculation establishing a loose version of equation 68

For clarity of exposition, we begin by presenting a relatively clean calculation that establishes a slightly loose version of Equation (68). Using just this looser bound, we would be able to establish that Equation (68) holds (with the constant 1/2 replaced by a slightly smaller constant) so long as we set d_1 to be $\Theta\left(w^{1/3}/\log w\right)$. A slightly more involved calculation (cf. Appendix A.2) is required to establish Equation (68) for our desired value of $d_1 = \lfloor (w/c)^{1/3} \rfloor$.

Expression (67) equals

$$\left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{i^2}{\left((d_1)!\right)^2} \cdot \prod_{j=1,j\neq i}^{d_1} \left(|j-i| \cdot |j+i| - \frac{ci^2j^2}{w}\right)$$

$$= \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{i^2}{\left((d_1)!\right)^2} \cdot \prod_{j=1,j\neq i}^{d_1} (|j-i| \cdot |j+i|) \cdot \left(1 - \frac{ci^2j^2}{w \cdot |j-i||j+i|}\right)$$

$$= \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{(d_1+i)!(d_1-i)!}{2\left((d_1)!\right)^2} \cdot \prod_{j=1,j\neq i}^{d_1} \left(1 - \frac{ci^2j^2}{w \cdot |j-i||j+i|}\right) \tag{79}$$

$$\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2} \cdot \prod_{j=1,j\neq i}^{d_1} \left(1 - \frac{ci^2j^2}{w \cdot |j-i||j+i|}\right)$$

$$\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2} \cdot \left(1 - \sum_{j=1,j\neq i}^{d_1} \frac{ci^2j^2}{w \cdot |j-i||j+i|}\right)$$

$$\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2} \cdot \left(1 - \frac{ci^2}{w} \sum_{j=1,j\neq i}^{d_1} \frac{j^2}{|j-i||j+i|}\right). \tag{80}$$

Let us consider the expression $\sum_{j=1,j\neq i}^{d_1} \frac{j^2}{|j-i||j+i|}$. If $i^2 \notin [j^2/2, 3j^2/2]$, then the j'th term in this sum is at most 2. Hence, letting H_i denote the ith Harmonic number and using the fact that $H_i \leq \ln(i+1)$,

$$\sum_{j=1,j\neq i}^{d_1} \frac{j^2}{|j-i||j+i|}$$

$$\leq 2 \cdot d_1 + \sum_{j=\lfloor\sqrt{2/3}i\rfloor}^{\lfloor\sqrt{2}i\rfloor} \frac{j^2}{|j-i||j+i|}$$

$$\leq 2 \cdot d_1 + \sum_{j=\lfloor\sqrt{2/3}\cdot i\rfloor}^{\lceil\sqrt{2}\cdot i\rceil} \frac{j}{|j-i|}$$

$$\leq 2d_1 + \sqrt{2} \cdot i \cdot \sum_{j=1}^{(\sqrt{2}-1)\cdot i} 2/j$$

$$\leq 2d_1 + 2\sqrt{2} \cdot i \cdot H_i \leq 2d_1 + 2\sqrt{2}i \ln(i+1). \tag{81}$$

We conclude that if d_1 were set to a value less than $w^{1/3}/(100 \cdot c^2 \cdot \ln(w))$ (rather than to $\lfloor (w/c)^{1/3} \rfloor$), then Expression (80) is at least

$$\left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1-1/c}{2}. \tag{82}$$

A.2 The tight bound

To obtain the tight bound, we need a tighter sequence of inequalities following Expression (79). Specifically, Expression (79) is bounded below by:

$$\begin{aligned}
&\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2}\left(1+\frac{i}{2d_1}\right)^{i} \cdot \prod_{j=1,j\neq i}^{d_1}\left(1-\frac{ci^2j^2}{w\cdot|j-i||j+i|}\right)\\
&\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2}\cdot e^{i^2/(2d_1)} \cdot \prod_{j=1,j\neq i}^{d_1}\left(1-\frac{ci^2j^2}{w\cdot|j-i||j+i|}\right)\\
&\geq \left(\frac{w}{ci^2}\right)^{d_1-1} \cdot \frac{1}{2}\cdot e^{i^2/(2d_1)} \cdot \prod_{j=1,j\neq i}^{d_1}\left(1-\frac{ci^2j^2}{w\cdot|j-i||j+i|}\right) \qquad (83)
\end{aligned}$$

The rough idea of how to proceed is as follows. Equation (81) implies that for $i \ll w^{1/3}/\ln w$, the factor

$$F_1 := \prod_{j=1,j\neq i}^{d_1}\left(1-\frac{ci^2j^2}{w\cdot|j-i||j+i|}\right)$$

is at some a positive constant, and hence Expression (83) is bounded below by the desired quantity. If $i \gtrsim w^{1/3}/\ln w$, then Equation (81) does not yield a good bound on this factor, leaving open the possibility that this factor is subconstant. But in this case, the factor $F_2 := e^{i^2/(2d_1)} \geq e^{\tilde{\Omega}(d_1)}$, and the largeness of F_2 dominates the smallness of F_1.

In more detail, let $x_{i,j} = \frac{ci^2j^2}{w\cdot|i-j|j+i||}$. Then for all $i \neq j$ such that $i, j \leq d_1$,

$$x_{i,j} \leq \frac{c\cdot d_1^2(d_1-1)^2}{(2d_1-1)\cdot w} \leq \frac{c\cdot d_1^3}{2w} \leq 1/2, \qquad (84)$$

where in the final inequality we used the fact that $d_1 \leq (w/c)^{1/3}$.

Using the fact that $1-x \geq e^{-x-x^2}$ for all $x \in [0, 1/2]$, we can write

$$F_1 \geq \prod_{j=1,j\neq i}^{d_1} e^{-x_{i,j}-x_{i,j}^2}.$$

Hence,

$$F_1\cdot F_2 \geq \exp\left(i^2/(2d_1) - \sum_{j=1,j\neq i}^{d_1} -x_{i,j} - x_{i,j}^2\right).$$

From Equations (81) and (84), we know that

$$\sum_{j=1,j\neq i}^{d_1} x_{i,j} + x_{i,j}^2 \leq \frac{ci^2}{w}\cdot\left(3d_1 + 3\sqrt{2}i\ln(i+1)\right) \leq \frac{ci^2}{w}\cdot(4d_1\ln(d_1)).$$

Hence,

$$F_1\cdot F_2 \geq \exp\left(i^2/(2d_1) - \frac{ci^2}{w}\cdot 4c\ln(d_1)\right)$$

$$
\begin{aligned}
&= \exp\left(i^2\left(\frac{1}{2d_1} - \frac{4c^2\ln(d_1)}{w}\right)\right)\\
&\geq \exp\left(i^2 \cdot \frac{1}{2d_1} \cdot (1 - o(1))\right)\\
&\geq 1.
\end{aligned}
$$

Equation (68) follows.

A Generalized Sylvester-Gallai Type Theorem for Quadratic Polynomials

Shir Peleg
Department of Computer Science, Tel Aviv University, Israel
shirpele@tauex.tau.ac.il

Amir Shpilka
Department of Computer Science, Tel Aviv University, Israel
shpilka@tauex.tau.ac.il

Abstract

In this work we prove a version of the Sylvester-Gallai theorem for quadratic polynomials that takes us one step closer to obtaining a deterministic polynomial time algorithm for testing zeroness of $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits. Specifically, we prove that if a finite set of irreducible quadratic polynomials $\mathcal{Q}$ satisfy that for every two polynomials $Q_1, Q_2 \in \mathcal{Q}$ there is a subset $\mathcal{K} \subset \mathcal{Q}$, such that $Q_1, Q_2 \notin \mathcal{K}$ and whenever Q_1 and Q_2 vanish then $\prod_{Q_i \in \mathcal{K}} Q_i$ vanishes, then the linear span of the polynomials in $\mathcal{Q}$ has dimension $O(1)$. This extends the earlier result [33] that showed a similar conclusion when $|\mathcal{K}| = 1$.

An important technical step in our proof is a theorem classifying all the possible cases in which a product of quadratic polynomials can vanish when two other quadratic polynomials vanish. I.e., when the product is in the radical of the ideal generated by the two quadratics. This step extends a result from [33] that studied the case when one quadratic polynomial is in the radical of two other quadratics.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic computation, Computational complexity, Computational geometry

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.8

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.05152`.

Funding The research leading to these results has received funding from the Israel Science Foundation (grant number 552/16) and from the Len Blavatnik and the Blavatnik Family foundation.
Amir Shpilka: Part of this work was done while the author was a visiting professor at NYU.

1 Introduction

This paper studies a problem at the intersection of algebraic complexity, algebraic geometry and combinatorics that is motivated by the polynomial identity testing problem (PIT for short) for depth 4 circuits. The question can also be regarded as an algebraic generalization and extension of the famous Sylvester-Gallai theorem from discrete geometry. We shall first describe the Sylvester-Gallai theorem and some of its many extensions and generalization and then discuss the relation to PIT.

Sylvester-Gallai type theorems

The Sylvester-Gallai theorem asserts that if a finite set of points in $\mathbb{R}^n$ has the property that every line passing through any two points in the set also contains a third point in the set then all the points in the set are colinear. Kelly extended the theorem to points in $\mathbb{C}^n$ and proved that if a finite set of points satisfy the Sylvester-Gallai condition then the points in the set are coplanar. Many variants of this theorem were studied: extensions to higher dimensions, colored versions, robust versions and many more. For a more on the Sylvester-Gallai theorem and some of its variants see [6, 3, 9].

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 8; pp. 8:1–8:33


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

There are two extensions that are of specific interest for our work: The **colored** version, proved by Edelstein and Kelly, states that if three finite sets of points satisfy that every line passing through points from two different sets also contains a point from the third set, then, all the points belong to a low dimensional space. This result was further extended to any constant number of sets. The **robust** version, obtained in [3, 9], states that if a finite set of points satisfy that for every point p in the set a δ fraction of the other points satisfy that the line passing through each of them and p spans a third point in the set, then the set is contained in an $O(1/\delta)$-dimensional space.

Although the Sylvester-Gallai theorem is formulated as a geometric question, it can be stated in algebraic terms: If a finite set of pairwise linearly independent vectors, $\mathcal{S} \subset \mathbb{C}^n$, has the property that every two vectors span a third vector in the set then the dimension of $\mathcal{S}$ is at most 3. It is not very hard to see that if we pick a subspace H of codimension 1, which is in general position with respect to the vectors in the set, then the intersection points $p_i = H \cap \text{span}\{s_i\}$, for $s_i \in \mathcal{S}$, satisfy the Sylvester-Gallai condition. Therefore, $\dim(S) \leq 3$. Another formulation is the following: If a finite set of pairwise linearly independent linear forms, $\mathcal{L} \subset \mathbb{C}[x_1, \ldots, x_n]$, has the property that for every two forms $\ell_i, \ell_j \in \mathcal{L}$ there is a third form $\ell_k \in \mathcal{L}$, so that whenever ℓ_i and ℓ_j vanish then so does ℓ_k, then the linear dimension of $\mathcal{L}$ is at most 3. To see this note that it must be the case that $\ell_k \in \text{span}\{\ell_i, \ell_j\}$ and thus the coefficient vectors of the forms in the set satisfy the condition for the (vector version of the) Sylvester-Gallai theorem, and the bound on the dimension follows.

The last formulation can now be extended to higher degree polynomials. In particular, the following question was asked by Gupta [17].

▶ **Problem 1.** *Can we bound the linear dimension or algebraic rank of a finite set $\mathcal{P}$ of pairwise linearly independent irreducible polynomials of degree at most r in $\mathbb{C}[x_1, \ldots, x_n]$, that has the following property: For any two distinct polynomials $P_1, P_2 \in \mathcal{P}$ there is a third polynomial $P_3 \in \mathcal{P}$, such that whenever P_1, P_2 vanish then so does P_3.*

A robust or colored version of this problem can also be formulated. As we have seen, the case $r = 1$, i.e when all the polynomials are linear forms, follows from the Sylvester-Gallai theorem. For the case of quadratic polynomials, i.e. $r = 2$, [33] gave a bound on the linear dimension for both the non-colored and colored versions. A bound for the robust version is still unknown for $r = 2$ and the entire problem is open for $r \geq 3$. Gupta [17] also raised a more general question of the same form.

▶ **Problem 2.** *Can we bound the linear dimension or algebraic rank of a finite set $\mathcal{P}$ of pairwise linearly independent irreducible polynomials of degree at most r in $\mathbb{C}[x_1, \ldots, x_n]$ that has the following property: For any two distinct polynomials $P_1, P_2 \in \mathcal{P}$ there is a subset $\mathcal{I} \subset \mathcal{P}$, such that $P_1, P_2 \notin \mathcal{I}$ and whenever P_1, P_2 vanish then so does $\prod_{P_i \in \mathcal{I}} P_i$.*

As before this problem can also be extended to robust and colored versions. In the case of linear forms, the bound for Problem 1 carries over to Problem 2 as well. This follows from the fact that the ideal generated by linear forms is prime (see Section 2 for definitions). In the case of higher degree polynomials, there is no clear reduction. For example, let $r = 2$ and

$$P_1 = xy + zw \quad , \quad P_2 = xy - zw \quad , \quad P_3 = xw \quad , \quad P_4 = yz.$$

Then, it is not hard to verify that whenever P_1 and P_2 vanish then so does $P_3 \cdot P_4$, but neither P_3 nor P_4 always vanishes when P_1 and P_2 do. The reason is that the radical of the ideal generated by P_1 and P_2 is not prime. Thus it is not clear whether a bound for Problem 1 would imply a bound for Problem 2. The latter problem was open, prior to this work, for any degree $r > 1$.

The Sylvester-Gallai theorem has important consequences for locally decodable and locally correctable codes [3, 9], for reconstruction of certain depth-3 circuits [32, 22, 35] and for the polynomial identity testing (PIT for short) problem, which we describe next.

Sylvester-Gallai type theorems and PIT

The PIT problem asks to give a deterministic algorithm that given an arithmetic circuit as input determines whether it computes the identically zero polynomial. This is a fundamental problem in theoretical computer science that has attracted a lot of attention because of its intrinsic importance, its relation to other derandomization problems [24, 25, 15, 13, 19, 36] and its connections to lower bounds for arithmetic circuits [20, 1, 21, 11, 16, 7]. Perhaps surprisingly, it was shown that deterministic algorithms for the PIT problem for homogeneous depth-4 circuits or for depth-3 circuits would lead to deterministic algorithms for general circuits [2, 18]. This makes small depth circuit extremely interesting for the PIT problem. We next explain how Sylvester-Gallai type questions are directly related to PIT for such low depth circuits. For more on the PIT problem see [34, 28, 29, 14].

The Sylvester-Gallai theorem is mostly relevant for the PIT problem in the setting when the input is a depth-3 circuit with small top fan-in. Specifically, a homogeneous $\Sigma^{[k]}\Pi^{[d]}\Sigma$ circuit in n variables computes a polynomial of the form

$$\Phi(x_1,\ldots,x_n)=\sum_{i=1}^{k}\prod_{j=1}^{d}\ell_{i,j}(x_1,\ldots,x_n)\ , \tag{1}$$

where each $\ell_{i,j}$ is a linear form. Consider the PIT problem for $\Sigma^{[3]}\Pi^{[d]}\Sigma$ circuits, i.e., Φ is given as in Equation 1 and $k=3$. In particular,

$$\Phi(x_1,\ldots,x_n)=\prod_{j=1}^{d}\ell_{1,j}(x_1,\ldots,x_n)+\prod_{j=1}^{d}\ell_{2,j}(x_1,\ldots,x_n)+\prod_{j=1}^{d}\ell_{3,j}(x_1,\ldots,x_n)\ . \tag{2}$$

If Φ computes the zero polynomial, then for every $j,j'\in[d]$.

$$\prod_{i=1}^{d}\ell_{1,i}\equiv 0 \mod \langle \ell_{2,j},\ell_{3,j'}\rangle\ .^{1}$$

This means that the sets $\mathcal{T}_i=\{\ell_{i,1},\ldots,\ell_{i,d}\}$ satisfy the conditions of the colored version of Problem 2 for $r=1$, and therefore have a small linear dimension. Thus, if $\Phi\equiv 0$ then, assuming that no linear form belongs to all three sets, we can rewrite the expression for Φ using only constantly many variables (after a suitable invertible linear transformation). This gives an efficient PIT algorithms for such $\Sigma^{[3]}\Pi^{[d]}\Sigma$ identities. The case of more than three multiplication gates is more complicated but it also satisfies a similar higher dimensional condition. This rank-bound approach for PIT of $\Sigma\Pi\Sigma$ circuits was raised in [10] and later carried out in [23, 31].[2]

As such rank-bounds found important applications in studying PIT of depth-3 circuits it seemed that a similar approach could potentially work for depth-4 $\Sigma\Pi\Sigma\Pi$ circuits as well.[3] In particular, it seemed most relevant for the case where there are only three multiplication

[2] The best algorithm for PIT of $\Sigma^{[k]}\Pi^{[d]}\Sigma$ circuits was obtained through a different, yet related, approach in [30].

[3] For multilinear $\Sigma\Pi\Sigma\Pi$ circuits Saraf and Volkovich obtained an analogous bound on the sparsity of the polynomials computed by the multiplication gates in a zero circuit [27].

gates and the bottom fan-in is two, i.e. for homogeneous $\Sigma^{[3]}\Pi^{[d]}\Sigma\Pi^{[2]}$ circuits that compute polynomials of the form

$$\Phi(x_1,\ldots,x_n)=\prod_{j=1}^{d}Q_{1,j}(x_1,\ldots,x_n)+\prod_{j=1}^{d}Q_{2,j}(x_1,\ldots,x_n)+\prod_{j=1}^{d}Q_{3,j}(x_1,\ldots,x_n)\,. \quad (3)$$

Both Beecken et al. [4] and Gupta [17] suggested an approach to the PIT problem of such identities based on the colored version of Problem 2 for $r=2$. Both papers described PIT algorithms for depth-4 circuits assuming a bound on the algebraic rank of the polynomials. In fact, Gupta conjectured that the algebraic rank of polynomials satisfying the conditions of Problem 2 depends only on their degree (see Conjectures 1, 2 and 30 in [17]).

▶ **Conjecture 3** (Conjecture 1 in [17]). *Let $\mathcal{F}_1,\ldots,\mathcal{F}_k$ be finite sets of irreducible homogenous polynomials in $\mathbb{C}[x_1,\ldots,x_n]$ of degree $\leq r$ such that $\cap_i\mathcal{F}_i=\emptyset$ and for every $k-1$ polynomials $Q_1,\ldots,Q_{k-1}$, each from a distinct set, there are $P_1,\ldots,P_c$ in the remaining set such that whenever $Q_1,\ldots,Q_{k-1}$ vanish then also the product $\prod_{i=1}^{c}P_i$ vanishes. Then, $trdeg_{\mathbb{C}}(\cup_i\mathcal{F}_i)\leq\lambda(k,r,c)$ for some function λ, where trdeg stands for the transcendental degree (which is the same as algebraic rank).*

Furthermore, using degree arguments Gupta showed that in Problem 2 we can restrict our attention to sets $\mathcal{I}$ such that $|\mathcal{I}|\leq r^{k-1}$. In particular, if the circuit in Equation (3) vanishes identically, then for every $(j,j')\in[d]^2$ there are $i_{1,j,j'},i_{2,j,j'},i_{3,j,j'},i_{4,j,j'}\in[d]$ so that

$$Q_{1,i_{1,j,j'}}\cdot Q_{1,i_{2,j,j'}}\cdot Q_{1,i_{3,j,j'}}\cdot Q_{1,i_{4,j,j'}}\equiv 0 \mod \langle Q_{2,j},Q_{3,j'}\rangle\,.$$

In [4] Beecken et al. conjectured that the algebraic rank of simple and minimal $\Sigma^{[k]}\Pi^{[d]}\Sigma\Pi^{[r]}$ circuits (see their paper for definition of simple and minimal) is $O_k(\log d)$. We note that for $k=3$ this conjecture is weaker than Conjecture 3 as every zero $\Sigma^{[3]}\Pi^{[d]}\Sigma\Pi^{[r]}$ circuit gives rise to a structure satisfying the conditions of Conjecture 3, but the other direction is not necessarily true. Beecken et al. also showed how to obtain a deterministic PIT for $\Sigma^{[k]}\Pi^{[d]}\Sigma\Pi^{[r]}$ circuits, assuming the correctness of their conjecture.

1.1 Our Result

Our main result gives a bound on the linear dimension of polynomials satisfying the conditions of Problem 2 when all the polynomials are irreducible of degree at most 2. Specifically we prove the following theorem.

▶ **Theorem 4.** *There exists a universal constant c such that the following holds. Let $\tilde{\mathcal{Q}}=\{Q_i\}_{i\in\{1,\ldots,m\}}\subset\mathbb{C}[x_1,\ldots,x_n]$ be a finite set of pairwise linearly independent homogeneous polynomials, such that every $Q_i\in\tilde{\mathcal{Q}}$ is either irreducible or a square of a linear form. Assume that, for every $i\neq j$, whenever Q_i and Q_j vanish then so does $\prod_{k\in\{1,\ldots,m\}\setminus\{i,j\}}Q_k$. Then,* $\dim(\mathrm{span}\{\mathcal{Q}\})\leq c$.

While our result still does not resolve Conjecture 3, as we need a colorful version of it, we believe that it is a significant step towards solving the conjecture for $k=3$ and $r=2$, which will yield a PIT algorithm for $\Sigma^{[3]}\Pi^{[d]}\Sigma\Pi^{[2]}$ circuits.

An interesting aspect of our result is that while the conjectures of [4, 17] speak about the algebraic rank we prove a stronger result that bounds that linear dimension (the linear rank is an upper bound on the algebraic rank). As our proof is quite technical it is an interesting question whether one could simplify our arguments by arguing directly about the algebraic rank.

An important algebraic tool in the proof of Theorem 4 is the following result characterizing the different cases in which a product of quadratic polynomials vanishes whenever two other quadratics vanish.

▶ **Theorem 5.** *Let* $\{Q_k\}_{k\in\mathcal{K}}, A, B$ *be homogeneous polynomials of degree* 2 *such that* $\prod_{k\in\mathcal{K}} Q_k \in \sqrt{\langle A, B\rangle}$. *Then one of the following cases hold:*

- **(i)** *There is* $k \in \mathcal{K}$ *such that* Q_k *is in the linear span of* A, B
- **(ii)** *There exists a non trivial linear combination of the form* $\alpha A + \beta B = c \cdot d$ *where* c *and* d *are linear forms.*
- **(iii)** *There exist two linear forms* c *and* d *such that when setting* $c = d = 0$ *we get that* A, B *and one of* $\{Q_k\}_{k\in\mathcal{K}}$ *vanish.*

From now on, to ease notations, we use Theorem 5i, Theorem 5ii or Theorem 5iii to describe different cases of Theorem 5.

The statement of the result is quite similar to Theorem 1.8 of [33] that proved a similar result when $|\mathcal{K}| = 1$. Specifically, in [33] the second item reads "There exists a non trivial linear combination of the form $\alpha A + \beta B = a^2$, where a is a linear form." This "minor" difference in the statements (which is necessary) is also responsible for the much harder work we do in the paper.

The proof of this theorem can be found in the full version of the paper [26].

1.2 Proof Idea

Our proof has a similar structure to the proofs in [33], but it does not rely on any of the results proved there.

Our starting point is the observation that Theorem 5 guarantees that unless one of $\{Q_k\}$ is in the linear span of A and B then A and B must satisfy a very strong property, namely, they must span a reducible quadratic or they have a very low rank (as quadratic polynomials). The proof of this theorem is based on analyzing the resultant of A and B with respect to some variable. We now explain how this theorem can be used to prove Theorem 4.

Consider a set of polynomials $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ satisfying the condition of Theorem 4. First, consider the case in which for every $Q \in \mathcal{Q}$, at least, say, $(1/100) \cdot m$ of the polynomials $Q_i \in \mathcal{Q}$, satisfy that there is another polynomial in $\mathcal{Q}$ in $\text{span}\{Q, Q_i\}$. In this case, we can use the robust version of the Sylvester-Gallai theorem [3, 9] (see Theorem 13) to deduce that the linear dimension of $\mathcal{Q}$ is small.

The second case we consider is when every polynomial $Q \in \mathcal{Q}$ that did not satisfy the first case now satisfies that for at least, say, $(1/100) \cdot m$ of the polynomials $Q_i \in \mathcal{Q}$ there are linear forms a_i and b_i such that $Q, Q_i \in \langle a_i, b_i\rangle$. We prove that if this is the case then there is a bounded dimensional linear space of linear forms, V, such that all the polynomials in $\mathcal{Q}$ that are of rank 2 are in $\langle V\rangle$. Then we argue that the polynomials that are not in $\langle V\rangle$ satisfy the robust version of the Sylvester-Gallai theorem (Theorem 13). Finally we bound the dimension of $\mathcal{Q} \cap \langle V\rangle$.

Most of the work however (Section 4) goes into studying what happens in the remaining case when there is some polynomial $Q_o \in \mathcal{Q}$ for which at least $0.98m$ of the other polynomials in $\mathcal{Q}$ satisfy Theorem 5ii with Q_o. This puts a strong restriction on the structure of these $0.98m$ polynomials. Specificity, each of them is of the form $Q_i = Q_o + a_i b_i$, where a_i and b_i are linear forms. The idea in this case is to show that the set $\{a_i, b_i\}$ is of low dimension. This is done by again studying the consequences of Theorem 5 for pairs of polynomials $Q_o + a_i b_i, Q_o + a_j b_j \in \mathcal{Q}$. After bounding the dimension of these $0.98m$ polynomials we bound the dimension of all the polynomials in $\mathcal{Q}$. The proof of this case is much more involved than the cases described earlier, and in particular we handle differently the case where Q_o is of high rank and the case where its rank is low.

1.3 On the relation to the proof of [33]

In [33] the following theorem was proved.

▶ **Theorem 6** (Theorem 1.7 of [33]). *Let $\{Q_i\}_{i\in[m]}$ be homogeneous quadratic polynomials over $\mathbb{C}$ such that each Q_i is either irreducible or a square of a linear function. Assume further that for every $i \neq j$ there exists $k \notin \{i,j\}$ such that whenever Q_i and Q_j vanish Q_k vanishes as well. Then the linear span of the Q_i's has dimension $O(1)$.*

As mentioned earlier, the steps in our proof are similar to the proof of Theorem 1.7 in [33]. Specifically, [33] also relies on an analog of Theorem 5 and divides the proof according to whether all polynomials satisfy the first case above or not. However, the fact that case ii of Theorem 5 is different than the corresponding case in the statement of Theorem 1.8 of [33], makes our proof is significantly more difficult. The reason for this is that while in [33] we could always pinpoint which polynomial vanishes when Q_i and Q_j vanish, here we only know that this polynomial belongs to a small set of polynomials. This leads to a richer structure in Theorem 5 and consequently to a considerably more complicated proof. To understand the effect of this on our proof we note that the corresponding case to Theorem 5ii was the *simpler* case to analyze in the proof of [33]. The fact that $a_i = b_i$ when $|\mathcal{K}| = 1$ almost immediately implied that the dimension of the span of the a_is is constant (see Claim 5.2 in [33]). In our case however, this is the bulk of the proof, and Section 4 is devoted to handling this case.

In addition to being technically more challenging, our proof gives new insights that may be extended to higher degree polynomials. The first is Theorem 5. While a similar theorem was proved for the simpler setting of [33], it was not clear whether a characterization in the form given in Theorem 5 would be possible, let alone true, in our more general setting. This gives hope that a similar result would be true for higher degree polynomials. Our second contribution is that we show (more or less) that either the polynomials in our set satisfy the robust version of Sylvester-Gallai theorem (Definition 12) or the linear functions composing the polynomials satisfy the theorem. Potentially, this may be extended to higher degree polynomials.

2 Preliminaries

In this section we explain our notation and present some basic algebraic preliminaries.

We will use the following notation. Greek letters $\alpha, \beta, \ldots$ denote scalars from $\mathbb{C}$. Non-capitalized letters $a, b, c, \ldots$ denote linear forms and x, y, z denote variables (which are also linear forms). Bold faced letters denote vectors, e.g. $\vec{x} = (x_1, \ldots, x_n)$ denotes a vector of variables, $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is a vector of scalars, and $\vec{0} = (0, \ldots, 0)$ the zero vector. We sometimes do not use a boldface notation for a point in a vector space if we do not use its structure as vector. Capital letters such as A, Q, P denote quadratic polynomials whereas V, U, W denote linear spaces. Calligraphic letters $\mathcal{I}, \mathcal{J}, \mathcal{F}, \mathcal{Q}, \mathcal{T}$ denote sets. For a positive integer n we denote $[n] = \{1, 2, \ldots, n\}$. For a matrix X we denote by $|X|$ the determinant of X.

A *Commutative Ring* is a group that is abelian with respect to both multiplication and addition operations. We mainly use the multivariate polynomial ring, $\mathbb{C}[x_1, \ldots, x_n]$. An *Ideal* $I \subseteq \mathbb{C}[x_1, \ldots, x_n]$ is an abelian subgroup that is closed under multiplication by ring elements. For $\mathcal{S} \subset \mathbb{C}[x_1, \ldots, x_n]$, we denote with $\langle \mathcal{S} \rangle$, the ideal generated by $\mathcal{S}$, that is, the smallest ideal that contains $\mathcal{S}$. For example, for two polynomials Q_1 and Q_2, the ideal $\langle Q_1, Q_2 \rangle$ is the set $\mathbb{C}[x_1, \ldots, x_n]Q_1 + \mathbb{C}[x_1, \ldots, x_n]Q_2$. For a linear subspace V, we have that $\langle V \rangle$ is the ideal generated by any basis of V. The *radical* of an ideal I, denoted by $\sqrt{I}$, is the set of

all ring elements, r, satisfying that for some natural number m (that may depend on r), $r^m \in I$. Hilbert's Nullstellensatz implies that, in $\mathbb{C}[x_1,\ldots,x_n]$, if a polynomial Q vanishes whenever Q_1 and Q_2 vanish, then $Q \in \sqrt{\langle Q_1, Q_2\rangle}$ (see e.g. [8]). We shall often use the notation $Q \in \sqrt{\langle Q_1, Q_2\rangle}$ to denote this vanishing condition. For an ideal $I \subseteq \mathbb{C}[x_1,\ldots,x_n]$ we denote by $\mathbb{C}[x_1,\ldots,x_n]/I$ the *quotient ring*, that is, the ring whose elements are the cosets of I in $\mathbb{C}[x_1,\ldots,x_n]$ with the proper multiplication and addition operations. For an ideal $I \subseteq \mathbb{C}[x_1,\ldots,x_n]$ we denote the set of all common zeros of elements of I by $Z(I)$.

For $V_1,\ldots,V_k$ linear spaces, we use $\sum_{i=1}^{k} V_i$ to denote the linear space $V_1 + \ldots + V_k$. For two non zero polynomials A and B we denote $A \sim B$ if $B \in \text{span}\{A\}$. For a space of linear forms $V = \text{span}\{v_1,\ldots,v_\Delta\}$, we say that a polynomial $P \in \mathbb{C}[x_1,\ldots,x_n]$ depends only on V if the value of P is determined by the values of the linear forms $v_1,\ldots,v_\Delta$. More formally, we say that P depends only on V if there is a Δ-variate polynomial $\tilde{P}$ such that $P \equiv \tilde{P}(v_1,\ldots,v_\Delta)$. We denote by $\mathbb{C}[v_1,\ldots,v_\Delta] \subseteq \mathbb{C}[x_1,\ldots,x_n]$ the subring of polynomials that depend only on V.

Another notation that we will use throughout the proof is congruence modulo linear forms.

▶ **Definition 7.** *Let $V \subset \mathbb{C}[x_1,\ldots,x_n]$ be a space of linear forms, and $P,Q \in \mathbb{C}[x_1,\ldots,x_n]$. We say that $P \equiv_V Q$ if $P - Q \in \langle V\rangle$.*

▶ **Fact 8.** *Let $V \subset \mathbb{C}[x_1,\ldots,x_n]$ be a space of linear forms and $P,Q \in \mathbb{C}[x_1,\ldots,x_n]$. If $P = \prod_{k=1}^{t} P_k$, and $Q = \prod_{k=1}^{t} Q_k$ satisfy that for all k, P_k and Q_k are irreducible in $\mathbb{C}[x_1,\ldots,x_n]/\langle V\rangle$, and $P \equiv_V Q \not\equiv_V 0$ then, up to a permutation of the indices, $P_k \equiv_V Q_k$ for all $k \in [t]$.*

This follows from the fact that the quotient ring $\mathbb{C}[x_1,\ldots,x_n]/\langle V\rangle$is a unique factorization domain.

2.1 Sylvester-Gallai Theorem and some of its Variants

In this section we present the formal statement the of Sylvester-Gallai theorem and the extensions that we use in this work.

▶ **Definition 9.** *Given a set of points, $v_1,\ldots,v_m$, we call a line that passes through exactly two of the points of the set an* ordinary line*.*

▶ **Theorem 10** (Sylvester-Gallai theorem)**.** *If m distinct points $v_1,\ldots,v_m$ in $\mathbb{R}^n$ are not collinear, then they define at least one ordinary line.*

▶ **Theorem 11** (Kelly's theorem)**.** *If m distinct points $v_1,\ldots,v_m$ in $\mathbb{C}^n$ are not coplanar, then they define at least one ordinary line.*

The robust version of the theorem was stated and proved in [3, 9].

▶ **Definition 12.** *We say that a set of points $v_1,\ldots,v_m \in \mathbb{C}^n$ is a δ-SG configuration if for every $i \in [m]$ there exists at least δm values of $j \in [m]$ such that the line through v_i, v_j contains a third point in the set.*

▶ **Theorem 13** (Robust Sylvester-Gallai theorem, Theorem 1.9 of [9])**.** *Let $V = \{v_1,\ldots,v_m\} \subset \mathbb{C}^n$ be a δ-SG configuration. Then* $\dim(\text{span}\{v_1,\ldots,v_m\}) \leq \frac{12}{\delta} + 1$.

The following is the colored version of the Sylvester-Gallai theorem.

▶ **Theorem 14** (Theorem 3 of [12]). *Let $\mathcal{T}_i$, for $i \in [3]$, be disjoint finite subsets of $\mathbb{C}^n$ such that for every $i \neq j$ and any two points $p_1 \in \mathcal{T}_i$ and $p_2 \in \mathcal{T}_j$ there exists a point p_3 in the third set that lies on the line passing through p_1 and p_2. Then, any such $\mathcal{T}_i$ satisfy that* $\dim(\text{span}\{\cup_i \mathcal{T}_i\}) \leq 3$.

We also state the equivalent algebraic versions of Sylvester-Gallai.

▶ **Theorem 15.** *Let $\mathcal{S} = \{\vec{s}_1, \ldots, \vec{s}_m\} \subset \mathbb{C}^n$ be a set of pairwise linearly independent vectors such that for every $i \neq j \in [m]$ there is a distinct $k \in [m]$ for which $\vec{s}_k \in \text{span}\{\vec{s}_i, \vec{s}_j\}$. Then* $\dim(\mathcal{S}) \leq 3$.

▶ **Theorem 16.** *Let $\mathcal{P} = \{\ell_1, \ldots, \ell_m\} \subset \mathbb{C}[x_1, \ldots, x_n]$ be a set of pairwise linearly independent linear forms such that for every $i \neq j \in [m]$ there is a distinct $k \in [m]$ for which whenever ℓ_i, ℓ_j vanish so does ℓ_k. Then* $\dim(\mathcal{P}) \leq 3$.

In this paper we refer to each of Theorem 11, Theorem 15 and Theorem 16 as the Sylvester-Gallai theorem. We shall also refer to sets of points/vectors/linear forms that satisfy the conditions of the relevant theorem as satisfying the condition of the Sylvester-Gallai theorem.

2.2 Resultant

A tool that will play an important role in the proof of Theorem 5 is the resultant of two polynomials. We will only define the resultant of a a quadratic polynomial and a linear polynomial as this is the case relevant to our work.[4] Let $A, B \in \mathbb{C}[x_1, \ldots, x_n]$. View A and B as polynomials in x_1 over $\mathbb{C}[x_2, \ldots, x_n]$ and assume that $\deg_{x_1}(A) = 2$ and $\deg_{x_1}(B) = 1$, namely,

$$A = \alpha x_1^2 + a x_1 + A_0 \quad \text{and } B = b x_1 + B_0 \,.$$

Then, the resultant of A and B with respect to x_1 is the determinant of their Sylvester matrix

$$\text{Res}_{x_1}(A, B) =: \left| \begin{bmatrix} A_0 & B_0 & 0 \\ a & b & B_0 \\ \alpha & 0 & b \end{bmatrix} \right| .$$

A useful fact is that if the resultant of A and B vanishes then they share a common factor.

▶ **Theorem 17** (See e.g. Proposition 8 in §5 of Chapter 3 in [8]). *Given $F, G \in \mathbb{F}[x_1, \ldots, x_n]$ of positive degree in x_1, the resultant $Res_{x_1}(F, G)$ is an integer polynomial in the coefficients of F and G. Furthermore, F and G have a common factor in $\mathbb{F}[x_1, \ldots, x_n]$ if and only if $Res_{x_1}(F, G) = 0$.*

2.3 Rank of Quadratic Polynomials

In this section we define the rank of a quadratic polynomial, and present some of its useful properties.

▶ **Definition 18.** *For a homogeneous quadratic polynomial Q we denote with* $\text{rank}_s(Q)$ *the minimal r such that there are $2r$ linear forms $\{a_k\}_{k=1}^{2r}$ satisfying $Q = \sum_{k=1}^{r} a_{2k} \cdot a_{2k-1}$. We call such representation a* minimal representation *of Q.*

[4] For the general definition of Resultant, see Definition 2 in §5 of Chapter 3 in [8].

This is a slightly different definition than the usual way one defines rank of quadratic forms,[5] but it is more suitable for our needs. We note that a quadratic Q is irreducible if and only if $\text{rank}_s(Q) > 1$. The next claim shows that a minimal representation is unique in the sense that the space spanned by the linear forms in it is unique.

▷ Claim 19. Let Q be a homogeneous quadratic polynomial and let $Q = \sum_{i=1}^{r} a_{2i-1} \cdot a_{2i}$ and $Q = \sum_{i=1}^{r} b_{2i-1} \cdot b_{2i}$ be two different minimal representations of Q. Then $\text{span}\{a_1, \ldots, a_{2r}\} = \text{span}\{b_1, \ldots, b_{2r}\}$.

Proof. Note that if the statement does not hold then, without loss of generality, a_1 is not contained in the span of the b_i's. This means that when setting $a_1 = 0$ the b_i's are not affected on the one hand, thus Q remains the same function of the b_i's, and in particular $\text{rank}_s(Q|_{a_1=0}) = r$, but on the other hand $\text{rank}_s(Q|_{a_1=0}) = r - 1$ (when considering its representation with the a_i's), in contradiction. ◁

This claim allows us to define the notion of *minimal space* of a quadratic polynomial Q, which we shall denote $\text{Lin}(Q)$.

▶ **Definition 20.** *Let Q be a quadratic polynomial, where $\text{rank}_s(Q) = r$, and let $Q = \sum_{i=1}^{r} a_{2i-1} \cdot a_{2i}$ be some minimal representation of Q. Define $Lin(Q) =: \text{span}\{a_1, \ldots, a_{2r}\}$, also denote $Lin(Q_1, \ldots, Q_k) = \sum_{i=1}^{k} Lin(Q_i)$.*

Claim 19 shows that the minimal space is well defined. The following fact is easy to verify.

▶ **Fact 21.** *Let $Q = \sum_{i=1}^{m} a_{2i-1} \cdot a_{2i}$ be a homogeneous quadratic polynomial, then $Lin(Q) \subseteq \text{span}\{a_1, \ldots, a_{2m}\}$.*

We now give some basic claims regarding rank_s.

▷ Claim 22. Let Q be a homogeneous quadratic polynomial with $\text{rank}_s(Q) = r$, and let $V \subset \mathbb{C}[x_1, \ldots, x_n]$ be a linear space of linear forms such that $\dim(V) = \Delta$. Then $\text{rank}_s(Q|_{V=0}) \geq r - \Delta$.

Proof. Assume without loss of generality $V = \text{span}\{x_1, \ldots, x_\Delta\}$, and consider $Q \in \mathbb{C}[x_{\Delta+1}, \ldots, x_n][x_1, \ldots, x_\Delta]$. There are $a_1, \ldots, a_\Delta \in \mathbb{C}[x_1, \ldots, x_n]$ and $Q' \in \mathbb{C}[x_{\Delta+1}, \ldots, x_n]$ such that $Q = \sum_{i=1}^{\Delta} a_i x_i + Q'$, where $Q|_{V=0} = Q'$. As $\text{rank}_s(\sum_{i=1}^{\Delta} a_i x_i) \leq \Delta$, it must be that $\text{rank}_s(Q|_{V=0}) \geq r - \Delta$. ◁

▷ Claim 23. Let $P_1 \in \mathbb{C}[x_1, \ldots, x_k]$, and $P_2 = y_1 y_2 \in \mathbb{C}[y_1, y_2]$. Then $\text{rank}_s(P_1 + P_2) = \text{rank}_s(P_1) + 1$. Moreover, $y_1, y_2 \in \text{Lin}(P_1 + P_2)$.

Proof. Denote $\text{rank}_s(P_1) = r$ and assume towards a contradiction that there are $a_1, \ldots, a_{2r}$ linear forms in $\mathbb{C}[x_1, \ldots, x_k, y_1, y_2]$ such that $P_1 + P_2 = \sum_{i=1}^{r} a_{2i-1} a_{2i}$. Clearly, $\sum_{i=1}^{r} a_{2i-1} a_{2i} \equiv_{y_1} P_1$. As $\text{rank}_s(P_1) = r$ this is a minimal representation of P_1. Hence, for every i, $a_i|_{y_1=0} \in \text{Lin}(P_1) \subset \mathbb{C}[x_1, \ldots, x_k]$. Moreover, from the minimality of r, $a_i|_{y_1=0} \neq 0$. Therefore, as y_1 and y_2 are linearly independent, we deduce that all the coefficients of y_2 in all the a_i's are 0. By reversing the roles of y_1 and y_2 we can conclude that $a_1, \ldots, a_{2r} \subset \mathbb{C}[x_1, \ldots, x_k]$ which means that Q does not depend on y_1 and y_2 in contradiction. Consider a minimal

[5] $\text{rank}_s(Q)$ is the minimal t such that there are t linear forms $\{a_k\}_{k=1}^{t}$, satisfying $Q = \sum_{k=1}^{t} a_k^2$.

representation $P_1 = \sum_{i=1}^{2r} b_{2i-1}b_{2i}$, from the fact that $\text{rank}_s(P_1 + P_2) = r + 1$ it follows that $P_1 + P_2 = \sum_{i=1}^{2r} b_{2i-1}b_{2i} + y_1y_2$ is a minimal representation of $P_1 + P_2$ and thus $\text{Lin}(P_1 + P_2) = \text{Lin}(P_1) + \text{span}\{y_1, y_2\}$. ◁

▶ **Corollary 24.** *Let a and b be linearly independent linear forms. Then, if c, d, e and f are linear forms such that $ab + cd = ef$ then* $\dim(\text{span}\{a, b\} \cap \text{span}\{c, d\}) \geq 1$.

▷ Claim 25. Let a, b, c and d be linear forms, and V be a linear space of linear forms. Assume $\{0\} \neq \text{Lin}(ab - cd) \subseteq V$ then $\text{span}\{a, b\} \cap V \neq \{0\}$.

Proof. As $\text{Lin}(ab - cd) \subseteq V$ it follows that $ab \equiv_V cd$. If both sides are zero then $ab \in \langle V \rangle$ and without loss of generality $b \in V$ and the statement holds. If neither sides is zero then from Fact 8 there are linear forms $v_1, v_2 \in V$, and $\lambda_1, \lambda_2 \in \mathbb{C}^\times$ such that, $\lambda_1\lambda_2 = 1$ and without loss of generality $c = \lambda_1 a + v_1, d = \lambda_2 b + v_2$. Note that not both v_1, v_2 are zero, as $ab - cd \neq 0$. Thus,

$$ab - cd = ab - (\lambda_1 a + v_1)(\lambda_2 b + v_2) = \lambda_1 a v_2 + \lambda_2 b v_1 + v_1 v_2.$$

As $\text{Lin}(ab - cd) \subseteq V$ it follows that $\text{Lin}(\lambda_1 a v_2 + \lambda_2 b v_1) \subseteq V$ and therefore there is a linear combination of a, b in V and the statement holds. ◁

We end this section with claims that will be useful in our proofs.

▷ Claim 26. Let $V = \sum_{i=1}^{m} V_i$ where V_i are linear subspaces, and for every i, $\dim(V_i) = 2$. If for every $i \neq j \in [m]$, $\dim(V_i \cap V_j) = 1$, then either $\dim(\bigcap_{i=1}^{m} V_i) = 1$ or $\dim(V) = 3$.

Proof. Let $w \in V_1 \cap V_2$. Complete it to basis of V_1 and V_2: $V_1 = \text{span}\{u_1, w\}$ and $V_2 = \text{span}\{u_2, w\}$. Assume that $\dim(\bigcap_{i=1}^{m} V_i) = 0$. Then, there is some i for which $w \notin V_i$. Let $x_1 \in V_i \cap V_1$, and so $x_1 = \alpha_1 u_1 + \beta_1 w$, where $\alpha_1 \neq 0$. Similarly, let $x_2 \in V_i \cap V_2$. Since $w \notin V_i$, $x_2 = \alpha_2 u_2 + \beta_2 w$, where $\alpha_2 \neq 0$. Note that $x_1 \notin \text{span}\{x_2\}$, as $\dim(V_1 \cap V_2) = 1$, and w is already in their intersection. Thus, we have $V_i = \text{span}\{x_1, x_2\} \subset \text{span}\{w, u_1, u_2\}$.

Now, consider any other $j \in [m]$. If V_j does not contain w, we can apply the same argument as we did for V_i and conclude that $V_j \subset \text{span}\{w, u_1, u_2\}$. On the other hand, if $w \in V_j$, then let $x_j \in V_i \cap V_j$, it is easy to see that x_j, w are linearly independent and so $V_j = \text{span}\{w, x_j\} \subset \text{span}\{w, V_i\} \subseteq \text{span}\{w, u_1, u_2\}$. Thus, in any case $V_j \subset \text{span}\{w, u_1, u_2\}$. In particular, $\sum_j V_j \subseteq \text{span}\{w, u_1, u_2\}$ as claimed. ◁

2.4 Projection Mappings

In this section we present and apply a new technique which allows us to simplify the structure of quadratic polynomials. Naively, when we want to simplify a polynomial equation, we can project it on a subset of the variables. Unfortunately, this projection does not necessarily preserve pairwise linear independence, which is a crucial property in our proofs. To remedy this fact, we present a set of mappings, which are somewhat similar to projections, but do preserve pairwise linear independence among polynomials.

▶ **Definition 27.** *Let $V = \text{span}\{v_1, \ldots, v_\Delta\} \subseteq \text{span}\{x_1, \ldots, x_n\}$ be a Δ-dimensional linear space of linear forms, and let $\{u_1, \ldots, u_{n-\Delta}\}$ be a basis for $V^\perp$. For $\vec{\alpha} = (\alpha_1, \ldots, \alpha_\Delta) \in \mathbb{C}^\Delta$ we define $T_{\vec{\alpha},V} : \mathbb{C}[x_1, \ldots, x_n] \mapsto \mathbb{C}[x_1, \ldots, x_n, z]$, where z is a new variable, to be the linear map given by the following action on the basis vectors: $T_{\vec{\alpha},V}(v_i) = \alpha_i z$ and $T_{\vec{\alpha},V}(u_i) = u_i$.*

▶ **Observation 28.** *$T_{\vec{\alpha},V}$ is a linear transformation and is also a ring homomorphism. This follows from the fact that a basis for* $\text{span}\{x_1, \ldots, x_n\}$ *is a basis for* $\mathbb{C}[x_1, \ldots, x_n]$ *as $\mathbb{C}$-algebra.*

▷ Claim 29. Let $V \subseteq \text{span}\{x_1, \ldots, x_n\}$ be a Δ-dimensional linear space of linear forms. Let F and G be two polynomials that share no common irreducible factor. Then, with probability 1 over the choice of $\vec{\alpha} \in [0,1]^{\Delta}$ (say according to the uniform distribution), $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ do not share a common factor that is not a polynomial in z.

Proof. Let $\{u_1, \ldots, u_{n-\Delta}\}$ be a basis for $V^{\perp}$. We think of F and G as polynomials in $\mathbb{C}[v_1, \ldots, v_{\Delta}, u_1, \ldots, u_{n-\Delta}]$. As $T_{\vec{\alpha},V} : \mathbb{C}[v_1, \ldots, v_{\Delta}, u_1, \ldots, u_{n-\Delta}] \to \mathbb{C}[z, u_1, \ldots, u_{n-\Delta}]$, Theorem 17 implies that if $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ share a common factor that is not a polynomial in z, then, without loss of generality, their resultant with respect to u_1 is zero. Theorem 17 also implies that the resultant of F and G with respect to u_1 is not zero. Observe that with probability 1 over the choice of $\vec{\alpha}$, we have that $\deg_{u_1}(F) = \deg_{u_1}(T_{\vec{\alpha},V}(F))$ and $\deg_{u_1}(G) = \deg_{u_1}(T_{\vec{\alpha},V}(G))$. As $T_{\vec{\alpha},V}$ is a ring homomorphism this implies that $Res_{u_1}(T_{\vec{\alpha},V}(G), T_{\vec{\alpha},V}(F)) = T_{\vec{\alpha},V}(Res_{u_1}(G,F))$. The Schwartz-Zippel-DeMillo-Lipton lemma now implies that sending each basis element of V to a random multiple of z, chosen uniformly from $(0,1)$ will keep the resultant non zero with probability 1. This also means that $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ share no common factor. ◁

▶ **Corollary 30.** *Let V be a Δ-dimensional linear space of linear forms. Let F and G be two linearly independent, irreducible quadratics, such that $Lin(F), Lin(G) \not\subseteq V$. Then, with probability 1 over the choice of $\vec{\alpha} \in [0,1]^{\Delta}$ (say according to the uniform distribution), $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ are linearly independent.*

Proof. As F and G are irreducible they share no common factors. Claim 29 implies that $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ do not share a common factor that is not a polynomial in z. The Schwartz-Zippel-DeMillo-Lipton implies that with probability 1, $T_{\vec{\alpha},V}(F)$ and $T_{\vec{\alpha},V}(G)$ are not polynomials in z, and therefore they are linearly independent. ◀

▷ Claim 31. Let Q be an irreducible quadratic polynomial, and V a Δ-dimensional linear space. Then for every $\vec{\alpha} \in \mathbb{C}^{\Delta}$, $\text{rank}_s(T_{\vec{\alpha},V}(Q)) \geq \text{rank}_s(Q) - \Delta$.

Proof. $\text{rank}_s(T_{\vec{\alpha},V}(Q)) \geq \text{rank}_s(T_{\vec{\alpha},V}(Q)|_{z=0}) = \text{rank}_s(Q|_{V=0}) \geq \text{rank}_s(Q) - \Delta$, where the last inequality follows from Claim 22. ◁

▷ Claim 32. Let $\mathcal{Q}$ be a set of quadratics, and V be a Δ-dimensional linear space. Then, if there are linearly independent vectors, $\{\vec{\alpha}^1, \ldots, \vec{\alpha}^{\Delta}\} \subset \mathbb{C}^{\Delta}$, such that, for every i,[6] $\dim(\text{Lin}(T_{\vec{\alpha}^i,V}(\mathcal{Q}))) \leq \sigma$ then $\dim(\text{Lin}(\mathcal{Q})) \leq (\sigma+1)\Delta$.

Proof. As $\dim(\text{Lin}(T_{\vec{\alpha}^i,V}(\mathcal{Q}))) \leq \sigma$, there are $u^i{}_1, \ldots, u^i{}_{\sigma} \subset V^{\perp}$ such that $\text{Lin}(T_{\vec{\alpha}^i,V}(\mathcal{Q})) \subseteq \text{span}\{z, u^i{}_1, \ldots, u^i{}_{\sigma}\}$. We will show that $\text{Lin}(\mathcal{Q}) \subset V + \text{span}\{\{u^i{}_1, \ldots, u^i{}_{\sigma}\}_{i=1}^{\Delta}\}$, which is of dimension at most $\Delta + \sigma\Delta$.

Let $P \in \mathcal{Q}$, then there are linear forms, $a_1, \ldots, a_{\Delta} \subset V^{\perp}$ and polynomials $P_V \in \mathbb{C}[V]$ and $P' \in \mathbb{C}[V^{\perp}]$, such that

$$P = P_V + \sum_{j=1}^{\Delta} a_j v_j + P'.$$

[6] Recall that $\text{Lin}(T_{\vec{\alpha}^i,V}(\mathcal{Q}))$ is the space spanned by $\cup_{Q \in \mathcal{Q}} \text{Lin}(T_{\vec{\alpha}^i,V}(\mathcal{Q}))$.

Therefore, after taking the projection for a specific $T_{\vec{\alpha}^i,V}$, for some $\gamma \in \mathbb{C}$,

$$T_{\vec{\alpha}^i,V}(P) = \gamma z^2 + \left(\sum_{j=1}^{\Delta} \alpha_j^i a_j\right) z + P'.$$

Denote $b_{P,i} = \sum_{j=1}^{\Delta} \alpha_j^i a_j$. By Corollary 30 if $a_1, \ldots, a_\Delta$ are not all zeros, then, with probability 1, $b_{P,i} \neq \vec{0}$.

If $b_{P,i} \notin \mathrm{Lin}(P')$ then from Claim 23 it follows that $\{z, b_{P,i}, \mathrm{Lin}(P')\} \subseteq \mathrm{span}\{\mathrm{Lin}(T_{\vec{\alpha}^i,V}(P))\}$. If, on the other hand, $b_{P,i} \in \mathrm{Lin}(P')$, then clearly $\{b_{P,i}, \mathrm{Lin}(P')\} \subseteq \mathrm{span}\{z, \mathrm{Lin}(T_{\vec{\alpha}^i,V}(P))\}$. To conclude, in either case, $\{b_{P,i}, \mathrm{Lin}(P')\} \subseteq \mathrm{span}\{z, {u^i}_1, \ldots, {u^i}_\sigma\}$.

Applying the analysis above to $T_{\vec{\alpha}^1,V}, \ldots, T_{\vec{\alpha}^\Delta,V}$ we obtain that $\mathrm{span}\{b_{P,1}, \cdots b_{P,\Delta}\} \subseteq \mathrm{span}\{\{{u^i}_1, \ldots, {u^i}_\sigma\}_{i=1}^{\Delta}\}$. As $\vec{\alpha}^1, \ldots \vec{\alpha}^\Delta$ are linearly independent, we have that $\{a_1, \ldots, a_\Delta\} \subset \mathrm{span}\{b_{P,1}, \cdots b_{P,\Delta}\}$, and thus $\mathrm{Lin}(P) \subseteq V + \{a_1, \ldots, a_\Delta\} + LS(P') \subseteq V + \mathrm{span}\{\{{u^i}_1, \ldots, {u^i}_\sigma\}_{i=1}^{\Delta}\}$. ◁

3 Sylvester-Gallai theorem for quadratic polynomials

In this section we prove Theorem 4. For convenience we repeat the statement of the theorem.

▶ **Theorem** (Theorem 4). *There exists a universal constant c such that the following holds. Let $\tilde{\mathcal{Q}} = \{Q_i\}_{i\in\{1,\ldots,m\}} \subset \mathbb{C}[x_1, \ldots, x_n]$ be a finite set of pairwise linearly independent homogeneous polynomials, such that every $Q_i \in \tilde{\mathcal{Q}}$ is either irreducible or a square of a linear form. Assume that, for every $i \neq j$, whenever Q_i and Q_j vanish then so does $\prod_{k\in\{1,\ldots,m\}\setminus\{i,j\}} Q_k$. Then,* $\dim(\mathrm{span}\{\mathcal{Q}\}) \leq c$.

▶ Remark 33. The requirement that the polynomials are homogeneous is not essential as homogenization does not affect the property $Q_k \in \sqrt{\langle Q_i, Q_j\rangle}$.

▶ Remark 34. Note that we no longer demand that the polynomials are irreducible but rather allow some of them to be square of linear forms, but now we restrict all polynomials to be of degree exactly 2. Note that both versions of the theorem are equivalent, as this modification does not affect the vanishing condition.

We use the following claim of [17].

▷ Claim 35 (Claim 11 in [17]). Let $P_1, \ldots, P_d, Q_1, \ldots, Q_k \in \mathbb{C}[x_1, \ldots, x_n]$ be homogeneous and the degree of each P_i is at most r. Then,

$$\prod_{i=1}^{k} Q_i \in \sqrt{\langle P_1, \ldots, P_d\rangle} \Rightarrow \exists\{i_1, \ldots, i_{r^d}\} \subset [k] \text{ such that } \prod_{j=1}^{r^d} Q_{i_j} \in \sqrt{\langle P_1, \ldots, P_d\rangle}\,.$$

▶ Remark 36. Note that from Claim 35 for $r = d = 2$, it follows that for every $i \neq j$ there exists a subset $\mathcal{K} \subseteq [m] \setminus \{i, j\}$ such that $|\mathcal{K}| \leq 4$ and whenever Q_i and Q_j vanish then so does $\prod_{k\in\mathcal{K}} Q_k$.

In what follows we shall use the following terminology. Whenever we say that two quadratics $Q_1, Q_2 \in \tilde{\mathcal{Q}}$ satisfy Theorem 5i we mean that there is a polynomial $Q_3 \in \tilde{\mathcal{Q}} \setminus \{Q_1, Q_2\}$ in their linear span. Similarly, when we say that they satisfy Theorem 5ii (Theorem 5iii) we mean that there is a reducible quadratic in their linear span (they belong to $\langle a_1, a_2\rangle$ for linear forms a_1, a_2).

Proof of Theorem 4. Partition the polynomials to two sets. Let $\mathcal{L}$ be the set of all squares and let $\mathcal{Q}$ be the subset of irreducible quadratics, thus $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$. Denote $|\mathcal{Q}| = m$, $|\mathcal{L}| = r$. Let $\delta = \frac{1}{100}$, and denote

- $\mathcal{P}_1 = \{P \in \mathcal{Q} \mid$ There are at least δm polynomials in $\mathcal{Q}$ such that P satisfies Theorem 5i but not Theorem 5ii with each of them$\}$.
- $\mathcal{P}_3 = \{P \in \mathcal{Q} \mid$ There are at least δm polynomials in $\mathcal{Q}$ such that P satisfies Theorem 5iii with each of them$\}$.

The proof first deals with the case where $\mathcal{Q} = \mathcal{P}_1 \cup \mathcal{P}_3$. We then handle the case that there is $Q \in \mathcal{Q} \setminus (\mathcal{P}_1 \cup \mathcal{P}_3)$.

3.1 The case $\mathcal{Q} = \mathcal{P}_1 \cup \mathcal{P}_3$

Assume that $\mathcal{Q} = \mathcal{P}_1 \cup \mathcal{P}_3$. For our purposes, we may further assume that $\mathcal{P}_1 \cap \mathcal{P}_3 = \emptyset$, by letting $\mathcal{P}_1 = \mathcal{P}_1 \setminus \mathcal{P}_3$.

▷ Claim 37. There exists a linear space of linear forms, V, such that $\dim(V) = O(1)$ and $\mathcal{P}_3 \subset \langle V \rangle$.

The intuition behind the claim is based on the following observation.

▶ **Observation 38.** *If $Q_1, Q_2 \in \mathcal{Q}$ satisfy Theorem 5iii then* $\dim(Lin(Q_1)), \dim(Lin(Q_2)) \leq 4$ *and* $\dim(Lin(Q_1) \cap Lin(Q_2)) \geq 2$.

Thus, we have many small dimensional spaces that have large pairwise intersections and we can therefore expect that such a V may exist.

Proof. We prove the existence of V by explicitly constructing it. Repeat the following process: Set $V = \{\vec{0}\}$, and $\mathcal{P}_3' = \emptyset$. At each step consider any $Q \in \mathcal{P}_3$ such that $Q \notin \langle V \rangle$ and set $V = \mathrm{Lin}(Q) + V$, and $\mathcal{P}_3' = \mathcal{P}_3' \cup \{Q\}$. Repeat this process as long as possible, i.e, as long as $\mathcal{P}_3 \not\subseteq \langle V \rangle$. We show next that this process must end after at most $\frac{3}{\delta}$ steps. In particular, $|\mathcal{P}_3'| \leq \frac{3}{\delta}$. It is clear that at the end of the process it holds that $\mathcal{P}_3 \subset \langle V \rangle$.

▷ Claim 39. Let $Q \in \mathcal{Q}$ and $\mathcal{B} \subseteq \mathcal{P}_3'$ be the subset of all polynomials in $\mathcal{P}_3'$ that satisfy Theorem 5iii with $\mathcal{Q}$, then $|\mathcal{B}| \leq 3$.

Proof. Assume towards a contradiction that $|\mathcal{B}| \geq 4$, and that Q_1, Q_2, Q_3 and Q_4 are the first 4 elements of $\mathcal{B}$ that where added to $\mathcal{P}_3'$. Denote $U = \mathrm{Lin}(Q)$, and $U_i = U \cap \mathrm{Lin}(Q_i)$, for $1 \leq i \leq 4$.

As Q satisfies Theorem 5iii we have that $\dim(U) \leq 4$. Furthermore, for every i, $\dim(U_i) \geq 2$ (by Observation 38). As the Q_is were picked by the iterative process, we have that $U_2 \not\subseteq U_1$. Indeed, since $Q_2 \in \langle U_2 \rangle$, if we had $U_2 \subseteq U_1 \subseteq \mathrm{Lin}(Q_1) \subseteq V$, then this would imply that $Q_2 \in \langle V \rangle$, in contradiction to the fact that $Q_2 \in \mathcal{P}_3'$. Similarly we get that $U_3 \not\subseteq U_1 + U_2$ and $U_4 \not\subseteq U_1 + U_3 + U_3$. However, as the next simple lemma shows, this is not possible.

▶ **Lemma 40.** *Let V be a linear space of dimension ≤ 4, and let $V_1, V_2, V_3 \subset V$ each of dimension ≥ 2, such that $V_1 \not\subseteq V_2$ and $V_3 \not\subseteq V_2 + V_1$ then $V = V_1 + V_2 + V_3$.*

Proof. As $V_1 \not\subseteq V_2$ we have that $\dim(V_1 + V_2) \geq 3$. Similarly we get $4 \leq \dim(V_1 + V_2 + V_3) \leq \dim(V) = 4$. ◀

Thus, Lemma 40 implies that $V = U_1 + U_2 + U_3$ and in particular, $U_4 \subseteq U_1 + U_2 + U_3$ in contradiction. This completes the proof of Claim 39. ◁

For $Q_i \in \mathcal{P}'_3$, define $T_i = \{Q \in \mathcal{Q} \mid Q, Q_i \text{ satisfiy Theorem 5iii}\}$. Since $|T_i| \geq \delta m$, and as by Claim 39 each $Q \in \mathcal{Q}$ belongs to at most 3 different sets, it follows by double counting that $|\mathcal{P}'_3| \leq 3/\delta$. As in each step we add at most 4 linearly independent linear forms to V, we obtain $\dim(V) \leq \frac{12}{\delta}$.

This completes the proof of Claim 37. ◀

So far V satisfies that $\mathcal{P}_3 \subset \langle V \rangle$. Next, we find a small set of polynomials $\mathcal{I}$ such that $\mathcal{Q} \subset \langle V \rangle + \text{span}\{\mathcal{I}\}$.

▷ Claim 41. There exists a set $\mathcal{I} \subset \mathcal{Q}$ such that $\mathcal{Q} \subset \langle V \rangle + \text{span}\{\mathcal{I}\}$ and $|\mathcal{I}| = O(1/\delta)$.

Proof. As before the proof shows how to construct $\mathcal{I}$ by an iterative process. Set $\mathcal{I} = \emptyset$ and $\mathcal{B} = \mathcal{P}_3$. First add to $\mathcal{B}$ any polynomial from $\mathcal{P}_1$ that is in $\langle V \rangle$. Observe that at this point we have that $\mathcal{B} \subset \mathcal{Q} \cap \langle V \rangle$. We now describe another iterative process for the polynomials in $\mathcal{P}_1$. In each step pick any $P \in \mathcal{P}_1 \setminus \mathcal{B}$ such that P satisfies Theorem 5i, but not Theorem 5ii,[7] with at least $\frac{\delta}{3}m$ polynomials in $\mathcal{B}$, and add it to both $\mathcal{I}$ and to $\mathcal{B}$. Then, we add to $\mathcal{B}$ all the polynomials $P' \in \mathcal{P}_1$ that satisfy $P' \in \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$. Note, that we always maintain that $\mathcal{B} \subset \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$.

We continue this process as long as we can. Next, we prove that at the end of the process we have that $|\mathcal{I}| \leq 3/\delta$.

▷ Claim 42. In each step we added to $\mathcal{B}$ at least $\frac{\delta}{3}m$ new polynomials from $\mathcal{P}_1$. In particular, $|\mathcal{I}| \leq 3/\delta$.

Proof. Consider what happens when we add some polynomial P to $\mathcal{I}$. By the description of our process, P satisfies Theorem 5i with at least $\frac{\delta}{3}m$ polynomials in $\mathcal{B}$. Any $Q \in \mathcal{B}$, that satisfies Theorem 5i with P, must span with P a polynomial $P' \in \tilde{\mathcal{Q}}$. Observe that $P' \notin \mathcal{L}$ as Q, P do not satisfy Theorem 5ii, and thus $P' \in \mathcal{Q}$. It follows that $P' \in \mathcal{P}_1$ since otherwise we would have that $P \in \text{span}\{\mathcal{B}\} \subset \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$, which implies $P \in \mathcal{B}$ in contradiction to the way that we defined the process. Furthermore, for each such $Q \in \mathcal{B}$ the polynomial P' is unique. Indeed, if there was a $P \neq P' \in \mathcal{P}_1$ and $Q_1, Q_2 \in \mathcal{B}$ such that $P' \in \text{span}\{Q_1, P\} \cap \text{span}\{Q_2, P\}$ then by pairwise independence we would conclude that $P \in \text{span}\{Q_1, Q_2\} \subset \text{span}\{\mathcal{B}\}$, which, as we already showed, implies $P \in \mathcal{B}$ in contradiction. Thus, when we add P to $\mathcal{I}$ we add at least $\frac{\delta}{3}m$ polynomials to $\mathcal{B}$. In particular, the process terminates after at most $3/\delta$ steps and thus $|\mathcal{I}| \leq 3/\delta$. ◁

Consider the polynomials left in $\mathcal{P}_1 \setminus \mathcal{B}$. As they "survived" the process, each of them satisfies the condition in the definition of $\mathcal{P}_1$ with at most $\frac{\delta}{3}m$ polynomials in $\mathcal{B}$. From the fact that $\mathcal{P}_3 \subseteq \mathcal{B}$ and the uniqueness property we obtained in the proof of Claim 42, we get that $\mathcal{P}_1 \setminus \mathcal{B}$ satisfies the conditions of Definition 12 with parameter $\delta/3$ and thus, Theorem 13 implies that $\dim(\mathcal{P}_1 \setminus \mathcal{B}) \leq O(1/\delta)$. Adding a basis of $\mathcal{P}_1 \setminus \mathcal{B}$ to $\mathcal{I}$ we get that $|\mathcal{I}| = O(1/\delta)$ and every polynomial in $\mathcal{Q}$ is in $\text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$. ◁

We are not done yet as the dimension of $\langle V \rangle$, as a vector space, is not a constant. Nevertheless, we next show how to use Sylvester-Gallai theorem to bound the dimension of $\mathcal{Q}$ given that $\mathcal{Q} \subset \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$. To achieve this we introduce yet another iterative process: For each $P \in \mathcal{Q} \setminus \langle V \rangle$, if there is quadratic L, with $\text{rank}_s(L) \leq 2$, such that $P + L \in \langle V \rangle$, then we set $V = V + \text{Lin}(L)$ (this increases the dimension of V by at most 4). Since this operation increases $\dim(\langle V \rangle \cap \mathcal{Q})$ we can remove one polynomial from $\mathcal{I}$, and thus decrease its size by 1, and still maintain the property that $\mathcal{Q} \subset \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$.

[7] By this we mean that there are many polynomials that together with P span another polynomial in $\mathcal{Q}$ but not in $\mathcal{L}$.

We repeat this process until either $\mathcal{I}$ is empty, or none of the polynomials in $\mathcal{I}$ satisfies the condition of the process. By the upper bound on $|\mathcal{I}|$ the dimension of V grew by at most $4|\mathcal{I}| = O(1/\delta)$ and thus it remains of dimension $O(1/\delta) = O(1)$. At the end of the process we have that $\mathcal{Q} \subset \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$ and that every polynomial in $P \in \mathcal{Q} \setminus \langle V \rangle$ has $\text{rank}_s(P) > 2$, even if we set all linear forms in V to zero.

Consider the map $T_{\vec{\alpha},V}$ as given in Definition 27, for a randomly chosen $\vec{\alpha} \in [0,1]^{\dim(V)}$. Each polynomial in $\mathcal{Q} \cap \langle V \rangle$ is mapped to a polynomial of the form form zb, for some linear form b. From Claim 22, it follows that every polynomial in $\mathcal{Q} \setminus \langle V \rangle$ still has rank larger than 2 after the mapping. Let

$$\mathcal{A} = \{b \mid \text{ some polynomial in } \mathcal{Q} \cap \langle V \rangle \text{ was mapped to } zb\} \cup T_{\vec{\alpha},V}(\mathcal{L}) \ .$$

We now show that, modulo z, $\mathcal{A}$ satisfies the conditions of Sylvester-Gallai theorem. Let $b_1, b_2 \in \mathcal{A}$ such that $b_1 \notin \text{span}\{z\}$ and $b_2 \notin \text{span}\{z, b_1\}$. As $\tilde{\mathcal{Q}}$ satisfies the conditions of Theorem 4 we get that there are polynomials $Q_1, \ldots, Q_4 \in \tilde{\mathcal{Q}}$ such that $\prod_{i=1}^{4} T_{\vec{\alpha},V}(Q_i) \in \sqrt{\langle b_1, b_2 \rangle} = \langle b_1, b_2 \rangle$, where the equality holds as $\langle b_1, b_2 \rangle$ is a prime ideal. This fact also implies that, without loss of generality, $T_{\vec{\alpha},V}(Q_4) \in \langle b_1, b_2 \rangle$. Thus, $T_{\vec{\alpha},V}(Q_4)$ has rank at most 2 and therefore $Q_4 \in \mathcal{L} \cup (\mathcal{Q} \cap \langle V \rangle)$. Hence, $T_{\vec{\alpha},V}(Q_4)$ was mapped to zb_4 or to b_4^2. In particular, $b_4 \in \mathcal{A}$. Claim 29 and Corollary 30 imply that b_4 is neither a multiple of b_1 nor a multiple of b_2, so it must hold that b_4 depends non-trivially on both b_1 and b_2. Thus, $\mathcal{A}$ satisfies the conditions of Sylvester-Gallai theorem modulo z. It follows that $\dim(\mathcal{A}) = O(1)$.

The argument above shows that the dimension of $T_{\vec{\alpha},V}(\mathcal{L} \cup (\mathcal{Q} \cap \langle V \rangle)) = O(1)$. Claim 32 implies that if we denote $U = \text{span}\{\mathcal{L} \cup \text{Lin}(\mathcal{Q} \cap \langle V \rangle)\}$ then $\dim(U)$ is $O(1)$. As $\mathcal{Q} \subseteq \text{span}\{(\mathcal{Q} \cap \langle V \rangle) \cup \mathcal{I}\}$, we obtain that $\dim(\tilde{\mathcal{Q}}) = \dim(\mathcal{L} \cup \mathcal{Q}) = O(1)$, as we wanted to show.

This completes the proof of Theorem 4 for the case $\mathcal{Q} = \mathcal{P}_1 \cup \mathcal{P}_3$.

3.2 The case $\mathcal{Q} \neq \mathcal{P}_1 \cup \mathcal{P}_3$

In this case there is some polynomial $Q_o \in \mathcal{Q} \setminus (\mathcal{P}_1 \cup \mathcal{P}_3)$. In particular, Q_0 satisfies Theorem 5ii with at least $(1-2\delta)m$ of the polynomials in $\mathcal{Q}$; of the remaining polynomials, at most δm satisfy Theorem 5i with Q_o; and, Q_o satisfies Theorem 5iii with at most δm polynomials. Let

- $\mathcal{Q}_1 = \{P \in \mathcal{Q} \mid P, Q_o \text{ satisfiy Theorem 5ii }\} \cup \{Q_o\}$
- $\mathcal{Q}_2 = \{P \in \mathcal{Q} \mid P, Q_o \text{ do not satisfiy Theorem 5ii }\}$
- $m_1 = |\mathcal{Q}_1|$, $m_2 = |\mathcal{Q}_2|$.

As $Q_o \notin \mathcal{P}_1 \cup \mathcal{P}_3$ we have that $m_2 \leq 2\delta m$ and $m_1 \geq (1-2\delta)m$. These properties of Q_o and $\mathcal{Q}$ are captured by the following definition.

▶ **Definition 43.** *Let $\mathcal{Q}_1 = \{Q_o, Q_1, \ldots, Q_{m_1}\}$ and $\mathcal{Q}_2 = \{P_1, \ldots, P_{m_2}\}$ be sets of irreducible homogeneous quadratic polynomials. Let $\mathcal{L} = \{\ell^2{}_1, \ldots, \ell^2{}_r\}$ be a set of squares of homogeneous linear forms. We say that $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ where $\mathcal{Q} = \mathcal{Q}_1 \cup \mathcal{Q}_2$ is a (Q_o, m_1, m_2)-set if it satisfies the following:*

1. *$\tilde{\mathcal{Q}}$ satisfy the conditions in the statement of Theorem 4.*
2. *$m_1 > 5m_2 + 2$.*
3. *For every $j \in [m_1]$, there are linear forms a_j, b_j such that $Q_j = Q_o + a_j b_j$.*
4. *For every $i \in [m_2]$, every non-trivial linear combination of P_i and Q_o has rank at least 2.*
5. *At most m_2 of the polynomials in $\mathcal{Q}$ satisfy Theorem 5iii with Q_o.*

By the discussion above, the following theorem is what we need in order to complete the proof for the case $\mathcal{Q} \neq \mathcal{P}_1 \cup \mathcal{P}_3$.

▶ **Theorem 44.** *Let $\tilde{\mathcal{Q}}$ satisfy the conditions of Definition 43, then* $\dim \tilde{\mathcal{Q}} = O(1)$.

We prove this theorem in Section 4. This concludes the proof of Theorem 17. ◀

4 Proof of Theorem 44

In this section we prove Theorem 44. The proof is divided to two parts according to whether the polynomial Q_o in Definition 43 is of high rank (Claim 46) or of low rank (Claim 60). Each part is also divided to two – first we consider what happens when $m_2 = 0$ and then the general case where $m_2 \neq 0$. The reason for this split is that when Q_o is of high rank then we know, e.g., that it cannot satisfy Theorem 5iii with any other polynomial. Similarly any polynomial satisfying Theorem 5ii with Q_o is also of high rank and cannot satisfy Theorem 5iii with any other polynomial. The reason why we further break the argument to weather $m_2 = 0$ or not, is that when $m_2 = 0$ all the polynomials are of the form $Q_o + ab$ for some linear forms a, b, which means we have fewer cases to analyse. While this seems a bit restrictive, the general case is not much harder and most of the ideas there already appear in the case $m_2 = 0$.

Throughout the proof we use the notation of Definition 43. In particular, each $Q_i \in \mathcal{Q}_1$ is of the form $Q_i = Q_o + a_i b_i$.

4.1 Q_o is of high rank

In this subsection we assume that $\tilde{\mathcal{Q}}$ is a (Q_o, m_1, m_2)-set for some quadratic Q_o of rank at least 100, this constant is arbitrary, as we just need it to be large enough. The following observation says that for our set $\mathcal{Q}$ we will never have to consider Theorem 5iii.

▶ **Observation 45.** *For $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ that satisfy Definition 43 with $\mathrm{rank}_s(Q_o) \geq 100$, for every $j \in [m_1]$ the rank of Q_j is at least $100 - 1 > 2$ and so Q_j never satisfies Theorem 5iii with any other polynomial in $\tilde{\mathcal{Q}}$.*

Our goal in this subsection is to prove the next claim.

▷ Claim 46. Let $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ be a (Q_o, m_1, m_2)-set with $\mathrm{rank_s}(Q_o) \geq 100$. Then $\dim(\mathrm{span}\{\tilde{\mathcal{Q}}\}) = O(1)$.

We break the proof of Claim 46 to two steps. First we handle the case $m_2 = 0$ and then the case $m_2 \neq 0$.

4.1.1 The case $m_2 = 0$

In this subsection we prove the following version of Claim 46 for the case $m_2 = 0$.

▷ Claim 47. Let $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ be a $(Q_o, m_1, 0)$-set with $\mathrm{rank_s}(Q_o) \geq 100$. Then, for a_i, b_i, ℓ_j as in Definition 43, $\dim(\mathrm{span}\{a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}, \ell_1, \ldots, \ell_r\}) \leq 7$. In particular, $\dim(\mathrm{span}\{\mathcal{Q}\}) \leq 8$.

We first show some properties satisfied by the products $\{a_1 b_1, \ldots, a_{m_1} b_{m_1}\}$.

▶ Remark 48. For $\ell_i^2 \in \mathcal{L}$ we can write $\ell_i^2 = 0 \cdot Q_o + \ell_i \ell_i$. Thus, from now on we can assume that every $Q_i \in \tilde{\mathcal{Q}}$ is of the form $Q_i = \alpha_i Q_o + a_i b_i$, for $\alpha_i \in \{0, 1\}$, and when $\alpha_i = 0$ it holds that $a_i = b_i$. We shall use the convention that for $i \in \{m_1 + 1, \ldots, m_1 + r\}$, $a_i = \ell_{i-m_1}$.

▷ Claim 49. Let $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ be a $(Q_o, m_1, 0)$-set with $\mathrm{rank_s}(Q_o) \geq 100$, and let $Q_i = Q_o + a_i b_i$ and $Q_j = Q_o + a_j b_j$ be polynomials in $\mathcal{Q} = \mathcal{Q}_1$.

1. If Q_i and Q_j satisfy Theorem 5i then there exists $k \in [m_1 + r]$ such that for some $\alpha, \beta \in \mathbb{C} \setminus \{0\}$

$$\alpha a_i b_i + \beta a_j b_j = a_k b_k. \tag{4}$$

2. If Q_i and Q_j satisfy Theorem 5ii then there exist two linear forms, c and d such that

$$a_i b_i - a_j b_j = cd. \tag{5}$$

The claim only considers Theorem 5i and Theorem 5ii as by Observation 45 we know that Q_i, Q_j do not satisfy Theorem 5iii. Note that the guarantee of this claim is not sufficient to conclude that the dimension of $a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}$ is bounded. The reason is that c and d are not necessarily part of the set. For example if for every i, $a_i b_i = x_i^2 - x_1^2$. Then every pair, Q_i, Q_j satisfy Theorem 5ii, but the dimension of $a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}$ is unbounded.

Proof of Claim 49. If Q_i, Q_j satisfy Theorem 5i then there are constants $\alpha, \beta \in \mathbb{C}$ and $k \in [m_1 + r] \setminus \{i, j\}$ such that $\alpha(Q_o + a_i b_i) + \beta(Q_o + a_j b_j) = \alpha Q_i + \beta Q_j = Q_k = \alpha_k Q_o + a_k b_k$. Rearranging we get that

$$\alpha a_i b_i + \beta a_j b_j - a_k b_k = (\alpha_k - (\alpha + \beta)) Q_o \ .$$

From the fact that $\mathrm{rank_s}(Q_o) \geq 100$, it must be that $\alpha_k - (\alpha + \beta) = 0$. Hence,

$$\alpha a_i b_i + \beta a_j b_j = a_k b_k \tag{6}$$

and (4) holds. Observe that $\alpha, \beta \neq 0$ as otherwise we will have two linearly dependent polynomials in $\mathcal{Q}$.

If Q_i, Q_j satisfy Theorem 5ii then there are $\alpha, \beta \in \mathbb{C}$ and two linear forms c and d such that $\alpha(Q_o + a_i b_i) + \beta(Q_o + a_j b_j) = cd$, and again, by the same argument, we get that $\beta = -\alpha$, and that, without loss of generality,

$$a_i b_i - a_j b_j = cd. \qquad \blacktriangleleft$$

Let $V_i =: \mathrm{span}\{a_i, b_i\}$. We next show that the different spaces V_i satisfy some non-trivial intersection properties.

▷ Claim 50. Let $\tilde{\mathcal{Q}}$ be a $(Q_o, m_1, 0)$-set such that $\mathrm{rank_s}(Q_o) \geq 100$. If for some $i \in [m_1]$ we have $\dim(V_i) = 2$ then for every $j \in [m_1]$ it holds that $\dim(V_j \cap V_i) \geq 1$. In particular it follows that if $dim(V_j) = 1$ then $V_j \subsetneq V_i$.

Proof. This follows immediately from Claim 49 and Corollary 24. ◁

Next we use this fact to conclude some structure on the set of pairs (a_i, b_i).

▷ Claim 51. Let $\tilde{\mathcal{Q}}$ be as in Claim 47. If $\dim(\mathrm{span}\{a_i, b_i\}) > 3$ then there is a linear space of linear forms, V such that $\dim(V) \leq 4$, and for all $i \in [m_1 + r]$, $b_i \in \mathrm{span}\{a_i, V\}$ or $a_i \in \mathrm{span}\{b_i, V\}$.

Proof. Consider the set of all V_i's of dimension 2. Combining Claim 49 and Claim 26 we get that either $\dim(\bigcup_{i=1}^m V_i) \leq 3$ or $\dim(\bigcap_{i=1}^m V_i) = 1$. If $\dim(\bigcup_{i=1}^m V_i) \leq 3$ then $V = \bigcup_{i=1}^m V_i$ is the linear space promised in the claim. If $\bigcap_{i=1}^m V_i) = 1$ there is a linear form, w, such that $\mathrm{span}\{w\} = \dim(\bigcap_{i=1}^m V_i)$. It follows that for every $i \in [m_1]$ there are constants ϵ_i, δ_i such that, with out loss of generality, $b_i = \epsilon_i a_i + \delta_i w$. Note that if $\dim(V_i) = 1$ this representation also holds with $\delta_i = 0$, and thus $V = \mathrm{span}\{w\}$. is the linear space promised in the claim. ◁

From now on we assume there is a linear space of linear forms, V such that $\dim(V) \leq 4$ and for every $i \in [m_1 + r]$ it holds that $b_i = \epsilon_i a_i + v_i$ (we can do this by replacing the roles of a_i and b_i if needed). Indeed, if $\dim(\mathrm{span}\{a_i, b_i\}) > 3$ then this follows from Claim 51 and

otherwise we can take $V = \text{span}\{a_i, b_i\}$. Thus, following Remark 48, every polynomial in $\mathcal{Q}$ is of the form $\alpha_i Q + a_i(\epsilon_i a_i + v_i)$ and for polynomials in $\mathcal{L}$ we have that $\alpha_i = 0$, $\epsilon_i = 1$ and $v_i = 0$.

The following claim is the crux of the proof of Claim 47. It shows that, modulo V, the set $\{a_1, \ldots, a_{m_1+r}\}$ satisfies the Sylvester-Gallai theorem..

▷ Claim 52. Let $i \neq j \in [m_1 + r]$ be such that $a_i \notin V$ and $a_j \notin \text{span}\{a_i, V\}$. Then, there is $k \in [m_1 + r]$ such that $a_k \in \text{span}\{a_i, a_j, V\}$ and $a_k \notin \text{span}\{a_i, V\} \cup \text{span}\{a_j, V\}$.

Proof. We split the proof to three cases (recall Remark 48): Either

(i) $\alpha_i = \alpha_j = 1$, or
(ii) $\alpha_i = 1, \alpha_j = 0$ (without loss of generality), or
(iii) $\alpha_i = \alpha_j = 0$.

Recall that $\alpha_i = 0$ if and only if $i \in \{m + 1, \ldots, m + r\}$.

(i) $\alpha_i = \alpha_j = 1$. Claim 49 implies that there are two linear forms c and d such that cd is a nontrivial linear combination of $a_j(\epsilon_j a_j + v_j), a_i(\epsilon_i a_i + v_i)$. We next show that without loss of generality c depends non-trivially on both a_i and a_j.

▶ **Lemma 53.** *In the current settings, without lost of generality, $c = \mu a_i + \eta a_j$ where $\mu, \eta \neq 0$.*

Proof. Setting $a_i = 0$ gives that, without loss of generality, $cd \equiv_{a_i} a_j(\epsilon_j a_j + v_j)$ and as $a_j \notin \text{span}\{a_i, V\}$ we have that $cd \not\equiv_{a_i} 0$. Thus, without loss of generality $c \equiv_{a_i} \eta a_j$, for some non-zero η. Let μ and η be such that $c = \mu a_i + \eta a_j$. We will now show that $\mu \neq 0$. Indeed, if this was not the case then we would have that $cd = \eta a_j d$. This means that $a_i(\epsilon_i a_i + v_i) \in \text{span}\{a_j(\epsilon_j a_j + v_j), \eta a_j d\}$ (since the linear dependence was non-trivial) setting $a_j = 0$ we see that either a_i, or $\epsilon_i a_i + v_i$ in $\text{span}\{a_j\}$, which contradicts our assumption. ◀

Equation 4 and Lemma 53 show that if Q_i and Q_j satisfy Theorem 5i, i.e. they span Q_k (for $k \notin \{i, j\}$), then one of $a_k, \epsilon_k a_k + v_k$ is a non-trivial linear combination of a_i and a_j. Thus, modulo V, a_k is in the span of a_i and a_j, which is what we wanted to show.

We next handle the case where Q_i and Q_j satisfy Theorem 5ii. Let cd be a product of linear forms in the span of Q_i and Q_j. From Lemma 53 we can assume that $c = \mu a_i + \eta a_j$ with $\mu\eta \neq 0$. In particular, this means that $\sqrt{\langle Q_i, Q_j \rangle} = \sqrt{\langle cd, Q_j \rangle}$.

The assumption that $\text{rank}_s(Q_o) \geq 100$ implies that Q_j is irreducible even after setting $c = 0$. It follows that if a product of irreducible polynomials satisfy $\prod_i A_i \in \sqrt{\langle cd, Q_j \rangle}$ then, after setting $c = 0$, some A_i is divisible by $Q_j|_{c=0}$. Thus, there is a multiplicand that is equal to $\alpha Q_j + ce$ for some linear form e. In particular, there must be a polynomial Q_k, $k \in [m_1 + r] \setminus \{i, j\}$, such that $Q_k = \alpha Q_j + ce$. If $\alpha = 0$ then it holds that $Q_k = a_k^2 = ce$ and therefore a_k satisfies the claim. Otherwise, as before, the rank condition on Q_o implies that $\alpha = 1$ and thus $a_k(\epsilon_k a_k + v_k) = a_j(\epsilon_j a_j + v_j) + (\mu a_i + \eta a_j)e$. Consider what happens when we set $a_j = 0$. We get that $a_k(\epsilon_k a_k + v_k) \equiv_{a_j} \mu a_i e$. Note that it cannot be the case that $e \equiv_{a_j} 0$ as this would imply that $a_k \in \text{span}\{a_j, v_k\}$ and in turn, this implies that $a_i \in \text{span}\{a_j, V\}$ in contradiction to the choice of a_i and a_j. Thus, we get that either a_k or $\epsilon_k a_k + v_k$ are equivalent to a_i modulo a_j. We next show that if either of them depends only on a_i, then we get a contradiction. Thus,

we are left in the case that $a_k = \lambda a_i$ (the case $\epsilon_k a_k + v_k = \lambda a_i$ is equivalent). Since $Q_k = Q_o + \lambda a_i\,(\epsilon_k \lambda a_i + v_k) = Q_j + ce$ and we have that $Q_i = Q_o + a_i(\epsilon_i a_i + v_i) = Q_j + cd$ we get by subtracting Q_i from Q_k that

$$a_i\left((\lambda^2\epsilon_k - \epsilon_i)a_i + (\lambda v_k - v_i)\right) = \lambda a_i(\epsilon_k \lambda a_i + v_k) - a_i(\epsilon_i a_i + v_i) = Q_k - Q_i = c(e-d)\ ,$$

and clearly neither side of the equation is zero since $Q_i \neq Q_k$. This implies that $c \in \mathrm{span}\{a_i, V\}$, in contradiction. Thus, in this case too we get that a_k satisfies the claim.

(ii) $\alpha_i = 1, \alpha_j = 0$. In this case, Q_i, Q_j must satisfy Theorem 5ii, as $0 \cdot Q_i + Q_j = a_j^2$. As before, the assumption that $\mathrm{rank_s}(Q_o) \geq 100$ implies that Q_i is irreducible even after setting $a_j = 0$. It follows that if a product of irreducible polynomials satisfy $\prod_t A_t \in \sqrt{\langle a_j^2, Q_i\rangle}$ then, after setting $a_j = 0$, some A_t is divisible by $Q_i|_{a_j=0}$. In our case we get that there is a multiplicand that is equal to $\alpha Q_i + a_j e$ for some linear form e. In particular, there must be a polynomial Q_k, for $k \in [m_1 + r] \setminus \{i, j\}$, such that $Q_k = \alpha Q_i + a_j e$. If $\alpha = 0$ it follows that Q_k is reducible and thus of the form $Q_k = a_k^2 = a_j e$ which is a contradiction to pairwise linear independence (as $Q_k \sim Q_j$). Thus $\alpha = \alpha_k = 1$, and $a_k(\epsilon_k a_k + v_k) = a_i(\epsilon_i a_i + v_k) + a_j e$. As before, we can conclude that $a_k \in \mathrm{span}\{a_i, a_j, V\}$ and that it cannot be the case that $a_k \in \mathrm{span}\{a_i, V\} \cup \mathrm{span}\{a_j, V\}$ (as by rearranging the equation we will get a contradiction to the fact that $a_j \notin \mathrm{span}\{a_i, V\}$), which is what we wanted to show.

(iii) $\alpha_i = \alpha_j = 0$. Then $\sqrt{\langle Q_i, Q_j\rangle} = \langle a_i, a_j\rangle$ is a prime ideal. It follows that there is $k \in [m_1 + r] \setminus \{i, j\}$ such that $Q_k \in \langle a_i, a_j\rangle$ the rank condition on Q_o implies that $\alpha_k = 0$ and therefore a_k is a non-trivial linear combination of a_i and a_j, which is what we wanted to show.

This completes the proof of Claim 52. ◁

We can now prove Claim 47.

Proof of Claim 47. Claim 52 implies that any two linear functions in $\{a_1, \ldots, a_{m_1+r}\}$ that are linearly independent modulo V, span (modulo V) a third function in the set. This implies that if we project all the linear functions to the perpendicular space to V then they satisfy the usual condition of the Sylvester-Gallai theorem and thus the dimension of the projection is at most 3. As $\mathrm{span}\{a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}, a_{m_1+1}, \ldots, a_{m_1+r}\} \subseteq \mathrm{span}\{a_1, \ldots, a_{m_1+r}, V\}$, we get that $\dim(\{a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}, a_{m_1+1}, \ldots, a_{m_1+r}\}) \leq 3 + \dim(V) \leq 7$, as claimed. ◁

Thus far we have proved Claim 47 which is a restriction of Claim 46 to the case $m_2 = 0$. In the next subsection we handle the general case $m_2 \neq 0$.

4.1.2 The case $m_2 \neq 0$

In this subsection we prove Claim 46. We shall assume without loss of generality that $m_2 \neq 0$. We first show that each $P_i \in \mathcal{Q}_2$ (recall Definition 43) is either a rank-2 quadratic, or it is equal to Q_o plus a rank-2 quadratic.

▷ Claim 54. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $\mathrm{rank_s}(Q_o) \geq 100$. Then for every $i \in [m_2]$ there exists $\gamma_i \in \mathbb{C}$ such that $\mathrm{rank_s}(P_i - \gamma_i Q_o)) = 2$.

Proof. Fix $i \in [m_2]$. We shall analyse, for each $j \in [m_1]$, which case of Theorem 5 Q_j and P_i satisfy. From Observation 45 we know that P_i does not satisfy Theorem 5iii with any Q_j. We start by analysing what happens when P_i and Q_j satisfy Theorem 5ii. By definition, there exist linear forms a', b' and non zero constants $\alpha, \beta \in \mathbb{C}$, such that $\alpha P_i + \beta Q_j = a'b'$ and thus,

$$P_i = \frac{1}{\alpha}\left(a'b' - \beta\left(Q_o + a_j b_j\right)\right) = \frac{-\beta}{\alpha}Q_o + \left(\frac{1}{\alpha}a'b' - \frac{\beta}{\alpha}a_j b_j\right) . \tag{7}$$

Hence, the statement holds with $\gamma_i = -\frac{\beta}{\alpha}$. Indeed, observe that the rank_s of $(\frac{1}{\alpha}a'b' - \frac{\beta}{\alpha}a_j b_j)$ cannot be 1 as this will contradict item 4 in Definition 43.

Thus, the only case left to consider is when P_i satisfies Theorem 5i alone with all the Q_j's. If for some $j \in [m_1]$ there is $j' \in [m_1]$ such that $Q_{j'} \in \text{span}\{Q_j, P_i\}$, then there are $\alpha, \beta \in \mathbb{C} \setminus \{0\}$, for which $P_i = \alpha Q_j + \beta Q_{j'}$ and then

$$P_i = (\alpha + \beta)Q_o + \alpha a_j b_j + \beta a_{j'} b_{j'},$$

and the statement holds with $\gamma_i = \beta + \alpha$. So, let us assume that for every $j \in [m_1]$, there is $t_j \in [m_2]$ such that $P_{t_j} \in \text{span}\{Q_j, P_i\}$. As $5m_2 + 2 < m_1$ there must be $j' \neq j'' \in [m_1]$ and $t' \in [m_2]$ such that $P_{t'} \in \text{span}\{Q_{j'}, P_i\}$ and $P_{t'} \in \text{span}\{Q_{j''}, P_i\}$. Since $\mathcal{Q}$ is a set of pairwise linearly independent polynomials, we can deduce that $\text{span}\{P_i, P_{t'}\} = \text{span}\{Q_{j'}, Q_{j''}\}$. In particular there exist $\alpha, \beta \in \mathbb{C}$, for which $P_i = \alpha Q_j + \beta Q_{j'}$, which, as we already showed, implies what we wanted to prove. ◁

For simplicity, rescale P_i so that $P_i = \gamma_i Q_o + L_i$ with $\text{rank}_s(L_i) = 2$ and $\gamma_i \in \{0, 1\}$. Clearly $\mathcal{Q}$ still satisfies the conditions of Definition 43 after this rescaling, as it does not affect the vanishing conditions or linear independence. The next claim shows that even in the case $m_2 \neq 0$, the linear forms $\{a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}\}$ "mostly" belong to a low dimensional space (similar to Claim 47).

▷ Claim 55. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $\text{rank}_s(Q_o) \geq 100$. Then, there exists a subspace V of linear forms such that $\dim(V) \leq 4$ and that for at least $m_1 - m_2$ indices $j \in [m_1]$ it holds that $a_j, b_j \in V$. Furthermore, there is a polynomial $P \in \mathcal{Q}_2$ such that $P = \gamma Q_o + L$ and $\text{Lin}(L) = V$.

Proof. Let $P_1 = \gamma_1 Q_o + L_1$ where $\text{rank}_s(L_1) = 2$. To simplify notation we drop the index 1 and only talk of P, L and γ. Set $V = \text{Lin}(L)$. As before, Observation 45 implies that P cannot satisfy Theorem 5iii with any $Q_j \in \mathcal{Q}_1$.

Let $Q_j \in \mathcal{Q}_1 \cup \mathcal{L}$. If Q_j, P satisfy Theorem 5iii, then $\alpha_j = 0$ and $Q_j = a_j^2$. By the rank condition on Q_o it follows that $\gamma = 0$ and therefore $a_j \in \text{Lin}(L) = V$.

Let $Q_j \in \mathcal{Q}_1 \cup \mathcal{L}$ be such that Q_j and P satisfy Theorem 5ii. This means that there are two linear forms e, f, and non zero $\alpha, \beta \in \mathbb{C}$ for which $\alpha P - \beta Q_j = ef$, and so,

$$(\alpha\gamma - \beta\alpha_j)Q_o = -\alpha L + \beta a_j b_j + ef \tag{8}$$

As we assumed that $\text{rank}_s(Q_o) \geq 100$ this implies that $\alpha\gamma - \beta\alpha_j = 0$ and thus $\beta a_j b_j + ef = \beta L$. Claim 19 implies that $e, f, a_j, b_j \in V$.

We have shown that V contains all a_j, b_j that come from polynomials satisfying Theorem 5ii with P.

Let $j \in [m_1]$ be such that P and Q_j satisfy Theorem 5i but not Theorem 5ii, i.e, they span another polynomial in $\tilde{\mathcal{Q}} \setminus \mathcal{L}$. If this polynomial is in $\mathcal{Q}_1$, i.e. there exists $j' \in [m_1]$ such that $Q_{j'} \in \text{span}\{P, Q_j\}$ then $P = \alpha Q_j + \beta Q_{j'}$ and as before we would get that $a_{j'}, b_{j'}, a_j, b_j \in V$.

All that is left is to bound the number of $j \in [m_1]$ so that P and Q_j span a polynomial in $\mathcal{Q}_2$. If there are more than m_2 such indices j then, by the pigeonhole principle, for two of them, say j, j' it must be the case that there is some $i \in [m_2]$ such that $P_i \in \text{span}\{P, Q_j\}$ and $P_i \in \text{span}\{P, Q_{j'}\}$. As our polynomials are pairwise independent this implies that $P \in \text{span}\{Q_j, Q_{j'}\}$, and as before we get that $a_{j'}, b_{j'}, a_j, b_j \in V$.

It follows that the only j's for which we may have $a_j, b_j \notin V$ must be such that Q_j and P span a polynomial in $\mathcal{Q}_2$, and no other $Q_{j'}$ spans this polynomial with P. Therefore, there are at most m_2 such "bad" j's and the claim follows. ◁

▶ Remark 56. The proof of Claim 55 implies that if $Q_i = \alpha_i Q_o + a_i b_i \in \mathcal{Q}_1$ satisfies that $\{a_i, b_i\} \not\subseteq V$ then it must be the case that Q_i and P span a polynomial $P_j \in \mathcal{Q}_2$.

▷ Claim 57. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $\text{rank}_s(Q_o) \geq 100$. Then there exists a 4-dimensional linear space V, such that for every $P_i \in \tilde{\mathcal{Q}}$ either P_i is defined over V, or there is a quadratic polynomial P'_i and a linear form v_i that are defined over V, and a linear form c_i, such that $P_i = Q_o + P'_i + c_i(\epsilon_i c_i + v_i)$, or $P_i = c_i^2$.

Proof. Claim 55 implies the existence of a polynomial $P = \gamma Q_o + L \in \mathcal{Q}_2$ and 4-dimensional linear space $V = \text{Lin}(L)$ such that the set $\mathcal{I} = \{Q_j \mid j \in [m_1] \text{ and } a_j, b_j \in V\}$ satisfies $|\mathcal{I}| \geq m_1 - m_2$. We will prove that V is the space guaranteed in the claim. We first note that every $P_i \in \mathcal{I}$ satisfies the claim with $P'_i = a_i b_i$ and $v_i = c_i = 0$, and clearly for $Q_i \in \mathcal{L}$ the claim trivially holds.

Consider $Q_i \in \mathcal{Q}_1 \setminus \mathcal{I}$. By Remark 56 it must be the case that Q_i and P span a polynomial $P_j \in \mathcal{Q}_2$. Hence, there are $\alpha, \beta \in \mathbb{C} \setminus \{0\}$ such that $P_j = \alpha P + \beta Q_i$. From Claim 54 we get that $P_j = \gamma_j Q_o + L_j$ and thus

$$(\gamma_j - \alpha\gamma - \beta)Q_o = \alpha L + \beta a_i b_i - L_j \ .$$

As $\text{rank}_s(Q_o) \geq 100$ it follows that $(\gamma_j - \alpha\gamma - \beta) = 0$ and $\alpha L + \beta a_i b_i = L_j$. Claim 23 implies that $\text{span}\{a_i, b_i\} \cap V \neq \{\vec{0}\}$ and therefore there is $v_i \in V$ such that, without loss of generality, $b_i = \epsilon_i a_i + v_i$, for some constant ϵ_i. Thus, the claimed statement holds for Q_i with $c_i = a_i$ and $Q'_i = 0$. I.e., $Q_i = Q_o + 0 + a_i(\epsilon_i a_i + v_i)$.

Consider a polynomial $P_i = \gamma_i Q_o + L_i \in \mathcal{Q}_2$.

If $\gamma_i = 0$ then by rank argument we see that P_i cannot satisfy Theorem 5ii nor Theorem 5iii with any polynomial in $\mathcal{Q}_1$. Hence it must satisfy Theorem 5i with all the polynomials in $\mathcal{Q}_1$. Therefore, by the pigeonhole principle P_i must be spanned by two polynomials in $\mathcal{I}$. Note that in this case we get that $P_i = L_i$ is a polynomial defined over V.

Assume then that $\gamma_i = 1$. If P_i is spanned by Q_j and $Q_{j'}$ such that $j, j' \in \mathcal{I}$, then, as before, $\text{Lin}(L_i) \subseteq \text{span}\{a_j b_j, a_{j'} b_{j'}\}$ and hence L_i is a function of the linear forms in V. Thus, the statement holds with $P'_i = L$ and $v_i = c_i = 0$.

The only case left to consider is when $\gamma_i = 1$ and every polynomial Q_j, for $j \in \mathcal{I}$, that satisfies Theorem 5i with P_i, does not span with P_i any polynomial in $\{Q_j \mid j \in \mathcal{I}\} \cup \mathcal{L}$. Note that in such a case it must hold that Q_j spans with P_i a polynomial in $\{Q_j \mid j \in [m_1] \setminus \mathcal{I}\} \cup \mathcal{Q}_2$. Observe that since our polynomials are pairwise linearly independent, if two polynomials from $\mathcal{I}$ span the same polynomial with P_i then P_i is in their span and we are done. From

$$|\{Q_j \mid j \in [m_1] \setminus \mathcal{I}\} \cup \mathcal{Q}_2| \leq (m_1 - |\mathcal{I}|) + m_2 \leq 2m_2 < m_1 - m_2 - 2 \leq |\mathcal{I}| - 2 \ ,$$

we see that for P_i to fail to satisfy the claim it must be the case that it satisfies Theorem 5ii with at least 2 polynomials whose indices are in $\mathcal{I}$. Let $Q_j, Q_{j'} \in \mathcal{I}$ be two such polynomials. In particular, there are four linear forms c, d, e and f and scalars $\epsilon_j, \epsilon_{j'}$, such that

$$P_i - \varepsilon_j Q_j = cd \quad \text{and} \quad P_i - \varepsilon_{j'} Q_{j'} = ef \ . \tag{9}$$

Equivalently,

$$(1-\varepsilon_j)Q_o = cd + \varepsilon_j a_j b_j - L_i \quad \text{and} \quad (1-\varepsilon_{j'})Q_o = ef + \varepsilon_{j'} a_{j'} b_{j'} - L_i \ .$$

As $\text{rank}_s(Q_o) \geq 100$ it must hold that $\varepsilon_j = \varepsilon_{j'} = 1$ and hence

$$L_i = cd + a_j b_j \quad \text{and} \quad L_i = ef + a_{j'} b_{j'} \ .$$

It follows that $cd - ef = a_{j'} b_{j'} - a_j b_j$ and therefore $\text{Lin}(cd - ef) \subseteq V$. Claim 25 implies that without loss of generality $d = \epsilon_i c + v_i$. We therefore conclude that

$$P_i = Q_o + L_i = Q_o + a_j b_j + c(\epsilon_i c + v_i)$$

and the statement holds for $P'_i = a_j b_j$ and $c_i = c$. This completes the proof of the Claim 57. ◁

Consider the representation guaranteed in Claim 57 and let

$$\mathcal{S} = \{c_i \mid \text{there is } P_i \in \mathcal{Q} \text{ such that either } P_i = c_i^2 \text{ or, for some } P'_i \text{ defined over } V, \\ P_i = Q_o + P'_i + c_i(\epsilon_i c_i + v_i)\} \ .$$

Clearly, in order to bound the dimension of $\tilde{\mathcal{Q}}$ it is enough to bound the dimension of $\mathcal{S}$. We do so, by proving that $\mathcal{S}$ satisfies the conditions of Sylvester-Gallai theorem modulo V, and thus have dimension at most $3 + \dim(V) = 7$.

▷ Claim 58. Let $c_i, c_j \in \mathcal{S}$ be such that $c_i \notin V$ and $c_j \notin \text{span}\{c_i, V\}$. Then, there is $c_k \in \mathcal{S}$ such that $c_k \in \text{span}\{c_i, c_j, V\}$ and $c_k \notin \text{span}\{c_i, V\} \cup \text{span}\{c_j, V\}$.

Before proving the claim we prove the following simple lemma.

▶ **Lemma 59.** *Let P_V be a polynomial defined over V and let c_i, c_j as in Claim 58. If there are linear forms e, f such that*

$$c_j(\epsilon_j c_j + v_j) + c_i(\epsilon_i c_i + v_i) + ef = P_V$$

then, without loss of generality, $e \in \text{span}\{c_i, c_j, V\}$ and $e \notin \text{span}\{c_i, V\} \cup \text{span}\{c_j, V\}$.

Proof. First note that $e \notin V$ as otherwise we would have that $c_i \equiv_V c_j$ in contradiction.

By our assumption, $ef = P_V$ modulo c_i, c_j. We can therefore assume without loss of generality that $e \in \text{span}\{c_i, c_j, V\}$. Assume towards a contradiction and without loss of generality that $e = \lambda c_i + v_e$, where $\lambda \neq 0$ and $v_e \in V$. Consider the equation $c_j(\epsilon_j c_j + v_j) + c_i(\epsilon_i c_i + v_i) + ef = P_V$ modulo c_i. We have that $c_j(\epsilon_j c_j + v_j) + v_e f \equiv_{c_i} P_V$ which implies that $\epsilon_j = 0$. Consequently, we also have that $f = \mu c_j + \eta c_i + v_f$, for some $\mu \neq 0$ and $v_f \in V$. We now observe that the product $c_i c_j$ has a non zero coefficient $\lambda\mu$ in ef and a zero coefficient in $P_V - c_j(\epsilon_j c_j + v_j) + c_i(\epsilon_i c_i + v_i)$, in contradiction. ◀

Proof of Claim 58. Following the notation of Claim 57, we either have $Q_i = Q_o + Q'_i + c_i(\epsilon_i c_i + v_i)$ or $Q_i = c_i^2$. Very similarly to Claim 52, we consider which case of Theorem 5 Q_i and Q_j satisfy, and what structure they have.

Assume $Q_i = Q_o + Q'_i + c_i(\epsilon_i c_i + v_i)$ and $Q_j = Q_o + Q'_j + c_j(\epsilon_j c_j + v_j)$. As argued before, since the rank of Q_o is large they can not satisfy Theorem 5iii. We consider the remaining cases:

- Q_i, Q_j satisfy Theorem 5i: there is $Q_k \in \mathcal{Q}$ such that $Q_k \in \text{span}\{Q_i, Q_j\}$.
  By assumption, for some scalars α, β we have that

$$Q_k = \alpha(Q_o + Q'_i + c_i(\epsilon_i c_i + v_i)) + \beta(Q_o + Q'_j + c_j(\epsilon_j c_j + v_j)) \ . \tag{10}$$

  If Q_k depends only on V then we would get a contradiction to the choice of c_i, c_j. Indeed, in this case we have that

$$(\alpha + \beta)Q_o = Q_k - \alpha(Q'_i + c_i(\epsilon_i c_i + v_i)) - \beta(Q'_j + c_j(\epsilon_j c_j + v_j)) \ .$$

  Rank arguments imply that $\alpha + \beta = 0$ and therefore

$$\alpha c_i(\epsilon_i c_i + v_i) + \beta c_j(\epsilon_j c_j + v_j) = Q_k - \alpha Q'_i - \beta Q'_j \ ,$$

  which implies that c_i and c_j are linearly dependent modulo V in contradiction.
  If $Q_k = c_k^2$ then by Lemma 59 it holds that c_k satisfies the claim condition.
  We therefore assume that Q_k is not a function of V alone and denote $Q_k = Q_o + Q'_k + c_k(\epsilon_k c_k + v_k)$. Equation 10 implies that

$$(1 - \alpha - \beta)Q_o = \alpha Q'_i + \beta Q'_j - Q'_k + \alpha c_i(\epsilon_i c_i + v_i) + \beta c_j(\epsilon_j c_j + v_j) - c_k(\epsilon_k c_k + v_k) \ .$$

  As $\alpha Q'_i + \beta Q'_j - Q'_k$ is a polynomial defined over V, its rank is smaller than 4 and thus, combined with the fact that $\text{rank}_s(Q_o) \geq 100$, we get that $(1 - \alpha - \beta) = 0$ and

$$Q'_k - \alpha Q'_i - \beta Q'_j = \alpha c_i(\epsilon_i c_i + v_i) + \beta c_j(\epsilon_j c_j + v_j) - c_k(\epsilon_k c_k + v_k) \ .$$

  We now conclude from Lemma 59 that c_k satisfies the claim.
- Q_i, Q_j satisfy Theorem 5ii: There are linear forms e, f such that for non zero scalars α, β, $\alpha Q_i + \beta Q_j = ef$. In particular,

$$(\alpha + \beta)Q_o = ef - \alpha Q'_i - \beta Q'_j - \alpha c_i(\epsilon_i c_i + v_i) - \beta c_j(\epsilon_j c_j + v_j).$$

  From rank argument we get that $\alpha + \beta = 0$ and from Lemma 59 we conclude that, without loss of generality, $e = \mu c_i + \eta c_j + v_e$ where $\mu, \eta \neq 0$. We also assume without loss of generality that $Q_i = Q_j + ef$.
  By our assumption that $\text{rank}_s(Q_o) \geq 100$ it follows that Q_j is irreducible even after setting $e = 0$. It follows that if a product of irreducible quadratics satisfy

$$\prod_k A_k \in \sqrt{\langle Q_i, Q_j \rangle} = \sqrt{\langle ef, Q_j \rangle}$$

  then, after setting $e = 0$, some A_k is divisible by $Q_j|_{e=0}$. Thus, there is a multiplicand that is equal to $\gamma Q_j + ed$ for some linear form d and scalar γ. In particular, there must be a polynomial $Q_k \in \tilde{\mathcal{Q}} \setminus \{Q_1, Q_2\}$, such that $Q_k = \gamma Q_j + ed$. If $\gamma = 0$ then it must hold that $Q_k = a_k^2 = ed$ and thus $a_k \sim e$, and the statment holds. If $\gamma = 1$ then we can assume without loss of generality that $Q_k = Q_j + ed$. Thus,

$$Q + Q'_k + c_k(\epsilon_k c_k + v_k) = Q_k = Q_j + ed = Q_o + Q'_j + c_j(\epsilon_j c_j + v_j) + (\mu c_i + \eta c_j + v_e)d \ .$$

  Setting $c_j = 0$ we get that

$$Q'_k + c_k(\epsilon_k c_k + v_k) \equiv_{c_j} Q'_j + (\mu c_i + v_e)d \ . \tag{11}$$

  Note that it cannot be the case that $d \equiv_{c_j} 0$. Indeed, if $d = 0$ then we get that Q_j and Q_k are linearly dependent in contradiction. If $d \sim c_j$ then (11) implies that $c_k \in \text{span}\{c_j, V\}$.

From the equality $Q_k = Q_j + ed$ and the fact that e depends non trivially on c_i, it now follows that $c_i \in \text{span}\{c_j, V\}$ in contradiction to the choice of c_i and c_j. As $d \not\equiv_{c_j} 0$, we deduce from (11) that, modulo c_j, $c_k \in \text{span}\{c_i, V\}$. We next show that if c_k depends only on c_i and V then we reach a contradiction and this will conclude the proof. So assume towards a contradiction that $c_k = \lambda c_i + v'_k$, for a scalar λ and $v'_k \in V$. Since

$$Q_j + ed = Q_k = Q_o + Q'_k + c_k(\epsilon_k c_k + v_k) = Q_o + Q'_k + (\lambda c_i + v'_k)\,(\epsilon_k(\lambda c_i + v'_k) + v_k)$$

and

$$Q_j + ef = Q_i = Q_o + Q'_i + c_i(\epsilon_i c_i + v_i)$$

we get by subtracting Q_i from Q_k that

$$e(d-f) = Q_k - Q_i = Q'_k - Q'_i + (\lambda c_i + v'_k)\,(\epsilon_k(\lambda c_i + v'_k) + v_k) - c_i(\epsilon_i c_i + v_i)$$

and clearly neither side of the equation is zero since $Q_i \neq Q_k$. This implies that $e \in \text{span}\{c_i, V\}$. This however contradicts the fact that $e = \mu c_i + \eta c_j + v_e$ where $\mu, \eta \neq 0$.

Now let us consider the case where without loss of generality, $Q_i = Q_o + Q'_i + c_i(\epsilon_i c_i + v_i)$ and $Q_j = c_j^2$. In this case the polynomials satisfy Theorem 5ii as $0 \cdot Q_i + Q_j = c_j^2$. Similarly to the previous argument, it holds that there is Q_k such that $Q_k = \gamma Q_i + c_j e$. If $\gamma = 0$ it holds that Q_k is reducible, and therefore a square of a linear form, in contradiction to pairwise linear independence. Thus $\gamma \neq 0$. If Q_k is defined only on the linear functions in V then it is of rank smaller then $\dim(V) \leq 4$, which will result in a contradiction to the rank assumption on Q_o. Thus $Q_k = Q_o + Q'_k + c_k(\epsilon_k c_k + v_k)$ and $\gamma = 1$. Therefore, we have

$$Q_o + Q'_k + c_k(\epsilon_k c_k + v_k) = Q_k = Q_i + c_j e \; = Q_o + Q'_i + c_i(\epsilon_i c_i + v_i) + c_j e.$$

Hence,

$$Q'_k - Q'_i - c_i(\epsilon_i c_i + v_i) - c_j e = -c_k(\epsilon_k c_k + v_k).$$

Looking at this equation modulo c_j implies that $c_k \in \text{span}\{V, c_i, c_j\}$. and $c_k \notin \text{span}\{V, c_j\}$, or we will get a contradiction to the fact that $c_i \notin \text{span}\{c_j, V\}$. Similarly it holds that $c_k \notin \text{span}\{V, c_i\}$, as we wanted to show.

The last structure we have to consider is the case where $Q_i = c_i^2, Q_j = c_j^2$. In this case, the ideal $\sqrt{\langle c_i^2, c_j^2\rangle} = \langle c_i, c_j\rangle$ is prime and therefore there is $Q_k \in \langle c_i, c_j\rangle$ this means that $\text{rank}_s(Q_k) \leq 2$. If $\text{rank}_s(Q_k) = 1$ then $Q_k = c_k^2$ and the statement holds. $\text{rank}_s(Q_k) = 2$ then Q_k is defined on the linear function of V, which implies $c_i, c_j \in V$ in contradiction to our assumptions. ◁

We are now ready to prove Claim 46.

Proof of Claim 46. Claim 58 implies that if we project the linear forms in $\mathcal{S}$ to $V^\perp$ then, after removing linearly dependent forms, they satisfy the conditions of the Sylvester-Gallai theorem. As $\dim(V) \leq 4$ we obtain that $\dim(\text{span}\{\mathcal{S} \cup V\}) \leq 7$. By Claim 57 every polynomial $P \in \mathcal{Q}$ is a linear combination of Q_o and a polynomial defined over $\text{span}\{\mathcal{S} \cup V\}$ which, by the argument above, implies that $\dim(\text{span}\{\mathcal{Q}\}) \leq 8$. ◁

This completes the proof of Theorem 44 when Q_o has high rank. We next handle the case where Q_o is of low rank.

4.2 Q_o is of Low Rank

In this section we prove the following claim.

▷ Claim 60. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $2 \leq \text{rank}_s(Q_o) < 100$. Then, $\dim(\text{span}\{\tilde{\mathcal{Q}}\}) = O(1)$.

Before we start with the proof of the main claim, let us prove a similar claim but for a more specific structure of polynomials. We will later see that, essentially, this structure holds when $2 \leq \text{rank}_s(Q_o) < 100$.

▷ Claim 61. Let $\tilde{\mathcal{Q}}$ be a set of quadratics polynomials that satisfy the conditions in the statement of Theorem 4. Assume farther that there is a linear space of linear forms, V such that $\dim(V) = \Delta$ and for each polynomial $Q_i \in \tilde{\mathcal{Q}}$ one of the following holds: either $Q_i \in \langle V \rangle$ or there is a linear form a_i such that $\text{Lin}(Q_i) \subseteq \text{span}\{V, a_i\}$. Then $\dim(\tilde{\mathcal{Q}}) \leq 8\Delta^2$.

Proof. Note that by the conditions in the statement of Theorem 4, no two polynomials in $\tilde{\mathcal{Q}}$ share a common factor.

Let $\vec{\alpha} \in \mathbb{C}^\Delta$ be such that if two polynomials in $T_{\vec{\alpha},V}(\tilde{\mathcal{Q}})$ (recall Definition 27) share a common factor then it is a polynomial in z. Note that by Claim 29 such $\vec{\alpha}$ exists. Thus, each $P \in \tilde{\mathcal{Q}}$, satisfies that either $T_{\vec{\alpha},V}(P) = \alpha_P z^2$ or $\text{Lin}(T_{\vec{\alpha},V}(P)) \subseteq \text{span}\{z, a_P\}$ for some linear form a_P independent of z. It follows that every polynomial in $T_{\vec{\alpha},V}(\tilde{\mathcal{Q}})$ is reducible. We next show that $\mathcal{S} = \{a_P \mid P \in \tilde{\mathcal{Q}}\}$ satisfies the conditions of Sylvester-Gallai theorem modulo z.

Let $a_1, a_2 \in \mathcal{S}$ such that $a_2 \notin \text{span}\{z, a_1\}$. Consider Q_1 such that $\text{Lin}(T_{\vec{\alpha},V}(Q_1)) \subseteq \text{span}\{z, a_1\}$ yet $\text{Lin}(T_{\vec{\alpha},V}(Q_1)) \not\subseteq \text{span}\{z\}$. Similarly, let Q_2 be such that $\text{Lin}(T_{\vec{\alpha},V}(Q_2)) \subseteq \text{span}\{z, a_2\}$ and $\text{Lin}(T_{\vec{\alpha},V}(Q_2)) \not\subseteq \text{span}\{z\}$. Then there is a factor of $T_{\vec{\alpha},V}(Q_1)$ of the form $\gamma_1 z + \delta_1 a_1$ where $\delta_1 \neq 0$. Similarly there is a factor of $T_{\vec{\alpha},V}(Q_2)$ of the form $\gamma_2 z + \delta_2 a_2$ where $\delta_2 \neq 0$.

This implies that $\sqrt{\langle T_{\vec{\alpha},V}(Q_1), T_{\vec{\alpha},V}(Q_2)\rangle} \subseteq \langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle$. Indeed, it is clear that for $i \in \{1, 2\}$, $T_{\vec{\alpha},V}(Q_i) \in \langle \gamma_i z + \delta_i a_i \rangle$. Hence, $\sqrt{\langle T_{\vec{\alpha},V}(Q_1), T_{\vec{\alpha},V}(Q_2)\rangle} \subseteq \sqrt{\langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle} = \langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle$, where the equality holds since $\langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle$ is a prime ideal.

We know that, there are $Q_3, Q_4, Q_5, Q_6 \in \mathcal{Q}$ such that

$$Q_3 \cdot Q_4 \cdot Q_5 \cdot Q_6 \in \sqrt{\langle Q_1, Q_2 \rangle}.$$

As $T_{\vec{\alpha},V}$ is a ring homomorphism it follows that,

$$T_{\vec{\alpha},V}(Q_3) \cdot T_{\vec{\alpha},V}(Q_4) \cdot T_{\vec{\alpha},V}(Q_5) \cdot T_{\vec{\alpha},V}(Q_6) \in \sqrt{\langle T_{\vec{\alpha},V}(Q_1), T_{\vec{\alpha},V}(Q_2)\rangle},$$

and

$$\sqrt{\langle T_{\vec{\alpha},V}(Q_1), T_{\vec{\alpha},V}(Q_2)\rangle} \subseteq \langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle .$$

Since $\langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle$ is prime it follows that, without loss of generality, $T_{\vec{\alpha},V}(Q_3) \in \langle \gamma_1 z + \delta_1 a_1, \gamma_2 z + \delta_2 a_2 \rangle$. It cannot be the case that $T_{\vec{\alpha},V}(Q_3) \in \langle \gamma_i z + \delta_i a_i \rangle$ for any $i \in \{1, 2\}$, because otherwise this will imply that $T_{\vec{\alpha},V}(Q_3)$ and $T_{\vec{\alpha},V}(Q_i)$ share a common factor that is not a polynomial in z, in contradiction to our choice of $T_{\vec{\alpha},V}$. This means that there is a factor of $T_{\vec{\alpha},V}(Q_3)$ that is in $\text{span}\{a_1, a_2, z\} \setminus (\text{span}\{a_1, z\} \cup \text{span}\{a_2, z\})$. Consequently, $a_3 \in \text{span}\{a_1, a_2, z\} \setminus (\text{span}\{a_1, z\} \cup \text{span}\{a_2, z\})$ as we wanted to prove. This shows that $\mathcal{S}$ satisfies the conditions of Sylvester-Gallai theorem, and therefore $\dim(\mathcal{S}) \leq 3$. Repeating the analysis above for linearly independent $\vec{\alpha}_1, \ldots, \vec{\alpha}_\Delta$, we can use Claim 32 and obtain that $\dim(\text{Lin}(\tilde{\mathcal{Q}})) \leq (3+1)\Delta$, and thus $\dim(\tilde{\mathcal{Q}}) \leq \binom{4\Delta}{2} + \Delta \leq 8\Delta^2$. ◁

Back to the proof of Claim 60. As before we first prove the claim for the case $m_2 = 0$ and then we prove the general case.

4.2.1 The case $m_2 = 0$

Similarly to the high rank case, in this subsection we prove the following claim.

▷ Claim 62. Let $\tilde{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{L}$ be a$(Q_o, m_1, 0)$-set such that $2 \leq \text{rank}_s(Q_o) < 100$, then $\dim(\text{span}\{a_1, \ldots, a_{m_1}, b_1, \ldots, b_{m_1}, \ell_1, \ldots, \ell_r\}) = O(1)$.

The proof is similar in structure to the proof of Claim 47. As before, we consider a polynomial $\ell_i^2 \in \mathcal{L}$ as $0 \cdot Q_o + \ell_i \ell_i$. We start by proving an analog of Claim 49. The claims are similar but the proofs are slightly different as we cannot rely on Q_o having high rank.

▷ Claim 63. Let $\tilde{\mathcal{Q}}$ satisfy the assumptions of Claim 62. Let $i \in [m_1]$ be such that $dim(a_i, b_i) = 2$ and $\text{span}\{a_i, b_i\} \cap \text{Lin}(Q_o) = \{\vec{0}\}$. Then, for every $j \in [m_1]$ the following holds:

1. Q_i and Q_j do not satisfy Theorem 5iii.
2. If Q_i and Q_j satisfy Theorem 5i then there exists $\alpha, \beta \in \mathbb{C} \setminus \{0\}$ such that for some $k \in [m_1] \setminus \{i, j\}$

$$\alpha a_i b_i + \beta a_j b_j = a_k b_k \ . \tag{12}$$

3. If Q_j is irreducible and Q_i and Q_j satisfy Theorem 5ii then there exist two linear forms, c and d such that

$$a_i b_i - a_j b_j = cd \ . \tag{13}$$

Proof. Assume Q_i and Q_j satisfy Theorem 5i, i.e., there are $\alpha, \beta \in \mathbb{C}$ and $k \in [m_1] \setminus \{i, j\}$ such that

$$\alpha(Q_o + a_i b_i) + \beta(Q_o + a_j b_j) = \alpha Q_i + \beta Q_j = Q_k = \alpha_k Q + a_k b_k$$

This implies that $\alpha a_i b_i + \beta a_j b_j - a_k b_k = (\alpha_k - (\alpha + \beta))Q_o$. We next show that it must be the case that $\alpha_k - (\alpha + \beta) = 0$.

Indeed, if $\alpha_k - (\alpha + \beta) \neq 0$ we get that $\beta a_j b_j - a_k b_k = (\alpha_k - (\alpha + \beta))Q_o - \alpha a_i b_i$. However, as we assumed $\text{span}\{a_i, b_i\} \cap \text{Lin}(Q_o) = \{\vec{0}\}$, we get by Claim 23 that

$$\text{rank}_s(\alpha_k - (\alpha + \beta))Q_o - \alpha a_i b_i) = \text{rank}_s(Q_o) + 1 > 2 \geq \text{rank}_s(\beta a_j b_j - a_k b_k)$$

in contradiction. We thus have that $\alpha_k - (\alpha + \beta) = 0$ and hence

$$\alpha a_i b_i + \beta a_j b_j = a_k b_k \tag{14}$$

and Equation 12 is satisfied. Observe that since our polynomials are pairwise independent $\alpha, \beta \neq 0$.

A similar argument to the one showing $\alpha_k - (\alpha + \beta) = 0$ also implies that Q_i and Q_j do not satisfy Theorem 5iii. If this was not the case then we would have that $\text{rank}_s(Q_o + a_i b_i) = 2$ which would again contradict Claim 23.

If Q_j is irreducible, the only case left is when $Q_o + a_i b_i, Q_o + a_j b_j$ satisfy Theorem 5ii. In this case there are $\alpha, \beta \in \mathbb{C}$ and two linear forms c and d such that $\alpha(Q_o + a_i b_i) + \beta(Q_o + a_j b_j) = cd$, and again, by the same argument we get that $\beta = -\alpha$ and so (after rescaling c)

$$a_i b_i - a_j b_j = cd \ .$$

This completes the proof of Claim 63. ◁

For each $i \in [m_1]$ let $V_i =: \text{span}\{a_i, b_i\}$. The next claim is analogous to Claim 50.

▷ Claim 64. Let $\tilde{\mathcal{Q}}$ satisfy the assumption in Claim 62. If for some $i \in [m_1]$ it holds that $\dim(V_i) = 2$ and $\text{Lin}(Q_o) \cap V_i = \{\vec{0}\}$ then for every $j \in [m_1]$ it is the case that $\dim(V_j \cap V_i) \geq 1$. In particular, if $dim(V_j) = 1$ then $V_j \subsetneq V_i$.

Proof. The proof of this claim follows immediately from Claim 63 and Corollary 24. ◁

the next claim is an analogous to Claim 51.

▷ Claim 65. Under the assumptions of Claim 62 there exists a subspace V of linear forms such that $\dim(V) \leq 2 \cdot 100 + 3$ and for every $i \in [m_1]$ there exists $v_i \in V$ and a constant $\epsilon_i \in \mathbb{C}$ such that $b_i = \epsilon_i a_i + v_i$ (or $a_i = \epsilon_i b_i + v_i$).

Proof. Let $\mathcal{I} = \{i \in [m_1] \mid \dim(V_i) = 2 \text{ and } \text{Lin}(Q_o) \cap V_i = \{\vec{0}\}\}$. If $\dim(\bigcup_{i\in\mathcal{I}} V_i) \leq 3$ then we set $V = \text{span}\{\text{Lin}(Q_o) \cup (\bigcup_{i\in\mathcal{I}} V_i)\}$. Clearly $\dim(V) \leq 2 \cdot \text{rank}_s(Q) + 3 \leq 2 \cdot 100 + 3$. Claim 64 implies that V has the required properties.

If $\dim(\bigcup_{i\in\mathcal{I}} V_i) > 3$ then from Claim 64 and Claim 26 it follows that $\dim(\bigcap_{i\in\mathcal{I}} V_i) = 1$. Let w be such that $\text{span}\{w\} = \bigcap_{i\in\mathcal{I}} V_i$ and set $V = \text{span}\{\text{Lin}(Q_o), w\}$. In this case too it is easy to see that V has the required properties. ◁

From now on we assume, without loss of generality that for every $i \in [m_1]$, $b_i = \epsilon_i a_i + v_i$. This structure also holds for the polynomials in $\mathcal{L}$.

Proof of Claim 62. Claim 65 implies that there is a linear space of linear forms, V, with $\dim(V) \leq 2 \cdot 100 + 3$, with the property that for every $Q_i \in \tilde{\mathcal{Q}}$ there is a linear form a_i such that $\text{Lin}(Q_i) \subseteq \text{span}\{V, a_i\}$. Thus $\tilde{\mathcal{Q}}$ satisfies the conditions of Claim 61, and $\dim(\tilde{\mathcal{Q}}) = O(1)$, as we wanted to show. ◁

We next consider the case $m_2 \neq 0$.

4.2.2 The case $m_2 \neq 0$

In this subsection we prove Claim 60, we can assume without loss of generality that $m_2 \neq 0$, as the case that $m_2 = 0$ was proved in the previous subsection. To handle this case we prove the existence of a subspace V of linear forms, of dimension $O(1)$, such that every polynomial in $\tilde{\mathcal{Q}}$ is in $\langle V \rangle$, and then, like we did before, we bound the dimension of $\tilde{\mathcal{Q}}$. The first step is proving an analog of Claim 54.

▷ Claim 66. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $\text{rank}_s(Q_o) < 100$. Then for every $i \in [m_2]$ there exists $\gamma_i \in \mathbb{C}$ such that $\text{rank}_s(P_i - \gamma_i Q_o) = 2$.

Proof. Consider $i \in [m_2]$. If P_i satisfies Theorem 5iii with any $Q_j \in \mathcal{Q}_1$, then the claim holds with $\gamma_i = 0$. If P_i satisfies Theorem 5ii with any $Q_j \in \mathcal{Q}$ then there exist linear forms c and d and non zero $\alpha, \beta \in \mathbb{C}$, such that $\alpha P_i + \beta Q_j = cd$. Therefore, $P_i = \frac{1}{\alpha}(cd - \beta(Q + a_j b_j))$ and the statement holds with $\gamma_i = -\frac{\beta}{\alpha}$. Observe that the rank of $cd - \beta a_j b_j$ cannot be 1 by Definition 43.

Thus, the only case left to consider is when P_i satisfies Theorem 5i with all the Q_j's in $\mathcal{Q}_1$. We next show that in this case there must exist $j \neq j' \in [m_1]$ such that $Q_{j'} \in \text{span}\{Q_j, P_i\}$. Indeed, since $m_1 > 5m_2 + 2$ there must be $j, j' \in [m_1]$ and $i' \in [m_2]$ such that $P_{i'} \in \text{span}\{Q_{j'}, P_i\}$ and $P_{i'} \in \text{span}\{Q_j, P_i\}$. As we saw before this implies that $P_i \in \text{span}\{Q_j, Q_{j'}\}$, which is what we wanted to show.

Let $j \neq j' \in [m_1]$ be as above and let $\alpha, \beta \in \mathbb{C}$ be such that $P_i = \alpha Q_j + \beta Q_{j'}$. It follows that

$$P_i = (\alpha + \beta)Q_o + \alpha a_j b_j + \beta a_{j'} b_{j'} \ .$$

Let $\gamma_i = \alpha + \beta$. Property 4 in Definition 43 implies that $\text{rank}_s(\alpha a_j b_j + \beta a_{j'} b_{j'}) = 2$ and the claim follows. ◁

As before, whenever $\gamma_i \neq 0$ let us replace P_i with $\frac{1}{\gamma_i} P_i$. Thus, from now on we shall assume $\gamma_i \in \{0, 1\}$. We next prove an analog of Claim 55.

▷ Claim 67. Let $\tilde{\mathcal{Q}}$ be a (Q_o, m_1, m_2)-set such that $\text{rank}_s(Q_o) < 100$. Then there is a subspace V of linear forms such that $\dim(V) \leq 2 \cdot 100 + 4$, $\text{Lin}(Q_o) \subseteq V$ and for at least $m_1 - 2m_2$ of the indices $j \in [m_1]$ it holds that $a_j, b_j \in V$.

Proof. Let $P = P_1$. Claim 66 implies that $P = \gamma Q_o + L$, for some L of rank 2. Set $V = \text{span}\{\text{Lin}(Q_o) \cup \text{Lin}(L)\}$. Clearly $\dim(V) \leq 2 \cdot 100 + 4$.

Let $j \in [m_1]$. If P and Q_j satisfy Theorem 5iii, then there are two linear forms c and d such that $Q_j, P \in \sqrt{\langle c, d \rangle}$, this implies that $\text{span}\{c, d\} \subset \text{Lin}(P) \subseteq V$. If $Q_o = Q_j - a_j b_j$ is not zero modulo c, d, then we obtain that $Q_o \equiv_{c,d} -a_j b_j$. Thus, there are linear forms $v_1, v_2 \in \text{Lin}(Q_o)$ such that $a_j \equiv_{c,d} v_1$ and $b_j \equiv_{c,d} v_2$. In particular, as $\text{Lin}(Q_o) \cup \{c, d\} \subset V$ it follows that $a_j, b_j \in V$. If Q_o is zero modulo c and d, then Q_j, Q_o satisfy Theorem 5iii and from property 5 of Definition 43 we know that there are at most m_2 such Q_j's. Furthermore, as $c, d \in \text{Lin}(Q_o) \subset V$ we obtain that $Q_j \in \langle V \rangle$. Denote by $\mathcal{K}$ the set of all Q_j that satisfy Theorem 5iii with Q_o. As we mentioned, $|\mathcal{K}| \leq m_2$.

If P and Q_j satisfy Theorem 5ii then there are two linear forms c and d, and non zero $\alpha, \beta \in \mathbb{C}$, such that $\alpha P + \beta Q_j = cd$. Hence,

$$\beta Q_o + \alpha P = -\beta a_j b_j + cd \ .$$

As $\beta Q_o + \alpha P$ is a non trivial linear combination of Q_o and P, we get from property 4 of Definition 43 that $2 \leq \text{rank}_s((\alpha\gamma + \beta)Q_o + \alpha L)$. It follows that

$$\text{rank}_s(-\beta a_j b_j + cd) = \text{rank}_s((\alpha\gamma + \beta)Q_o + \alpha L) = 2$$

and therefore by Fact 21,

$$\{a_j, b_j, c, d\} \subset \text{Lin}(-\beta a_j b_j + cd) = \text{Lin}((\alpha\gamma + \beta)Q_o + \alpha L) \subseteq V \ ,$$

and again $a_j, b_j \in V$.

The last case to consider is when P and Q_j satisfy Theorem 5i. If they span a polynomial $Q_{j'} \in \mathcal{Q}_1 \cup \mathcal{L}$, then $P = \alpha Q_j + \beta Q_{j'}$ and as in the previous case we get that $a_j, b_j \in V$.

Let $\mathcal{J}$ be the set of all indices $j \in [m_1]$ such that P and Q_j span a polynomial in $\mathcal{Q}_2$ but no polynomial in $\mathcal{Q}_1 \cup \mathcal{L}$. So far we proved that for every $j \in [m_1] \setminus (\mathcal{J} \cup \mathcal{K})$ we have that $a_j, b_j \in V$. We next show that $|\mathcal{J}| \leq m_2$ which concludes the proof.

Indeed, if this was not the case then by the pigeonhole principle there would exist a polynomial $P_i \in \mathcal{Q}_2$ and two polynomials $Q_j, Q_{j'} \in \mathcal{Q}_1$ such that $P_i \in \text{span}\{Q_j, P\}$ and $P_i \in \text{span}\{Q_{j'}, P\}$. By pairwise independence this implies that $Q_{j'}$ is in the linear span of P and Q_j which contradicts the definition of $\mathcal{J}$. ◁

Our next claim gives more information about the way the polynomials in $\tilde{\mathcal{Q}}$ relate to the subspace V found in Claim 67.

▷ Claim 68. Let $\tilde{\mathcal{Q}}$ and V be as in Claim 67. Then, every polynomial P in $\tilde{\mathcal{Q}}$ satisfies (at least) one of the following cases:

1. $\mathrm{Lin}(P) \subseteq V$ or
2. $P \in \langle V \rangle$ or
3. $P = P' + c(c+v)$ where P' is a quadratic polynomial such that $\mathrm{Lin}(P') \subseteq V$, $v \in V$ and c is a linear form.

Proof. Let $\mathcal{I} = \{j \in [m_1] \mid a_j, b_j \in V\}$. Claim 67 implies that $|\mathcal{I}| \geq m_1 - 2m_2$. Furthermore, by the construction of V we know that $\mathrm{Lin}(Q_o) \subseteq V$. Observe that this implies that for every $j \in \mathcal{I}$, $\mathrm{Lin}(Q_j) \subseteq V$.

Note that every polynomial in $\mathcal{L}$ satisfies the third item of the claim. Let P be any polynomial in $\mathcal{Q}_2 \cup \{Q_j \mid j \in [m_1] \setminus \mathcal{I}\}$. We study which case of Theorem 5 P satisfies with polynomials whose indices belong to $\mathcal{I}$.

If P_i satisfies Theorem 5iii with any polynomial Q_j, for $j \in \mathcal{I}$, then, as $\mathrm{Lin}(Q_j) \subseteq V$, it follows that $P \in \langle V \rangle$.

If P is spanned by two polynomials $Q_j, Q_{j'}$ such that $j, j' \in \mathcal{I}$, then clearly $\mathrm{Lin}(P) \subseteq V$. Similarly, if P is spanned by a polynomial $Q_j, Q_{j'}$ such that $j \in \mathcal{I}$ and $Q_{j'} \in \mathcal{L}$ then $P = \alpha Q_j + \beta a_{j'}^2$, and hence it also satisfies the claim.

Hence, for P to fail to satisfy the claim, it must be the case that every polynomial Q_j, for $j \in \mathcal{I}$, that satisfies Theorem 5i with P, does not span with P any polynomial in $\{Q_j \mid j \in \mathcal{I}\} \cup \mathcal{L}$. Thus, it must span with P a polynomial in $\{Q_j \mid j \in [m_1] \setminus \mathcal{I}\} \cup \mathcal{Q}_2$. As before, observe that by pairwise linear independent, if two polynomials from $\mathcal{I}$ span the same polynomial with P, then P is in their span and we are done. Thus, since

$$|\{Q_j \mid j \in [m_1] \setminus \mathcal{I}\} \cup \mathcal{Q}_2| \leq (m_1 - |\mathcal{I}|) + m_2 \leq 3m_2 < m_1 - 2m_2 - 2 \leq |\mathcal{I}| - 2 \,,$$

for P to fail to satisfy the claim it must be the case that it satisfies Theorem 5ii with at least 2 polynomials whose indices are in $\mathcal{I}$.

Let $Q_j, Q_{j'}$ be two such polynomials. There are four linear forms, c, d, e and f and scalars $\epsilon_j, \epsilon_{j'}$ such that

$$P + \varepsilon_j Q_j = cd \qquad \text{and} \qquad P + \varepsilon_{j'} Q_{j'} = ef \,.$$

Therefore

$$\varepsilon_j Q_j - \varepsilon_{j'} Q_{j'} = cd - ef \,. \tag{15}$$

In particular, $\mathrm{Lin}(cd - ef) \subseteq V$. Claim 25 and Equation (15) imply that, without loss of generality, $d = \epsilon c + v$ for some $v \in V$ and $\epsilon \in \mathbb{C}$. Thus, $P = cd - \varepsilon_j Q_j = c(\epsilon c + v) - \varepsilon_j Q_j$ and no matter whether $\epsilon = 0$ or not. P satisfies the claim. Indeed, if $\epsilon = 0$ then $P \in \langle V \rangle$ and we are done. Otherwise, we can normalize c, v to assume that $\epsilon = 1$ and get that $\mathrm{Lin}(P - c^2) \in V$ as claimed. ◁

We can now complete the proof of Claim 60.

Proof of Claim 60. Claim 68 implies that there is a linear space of linear forms, V, such that $\dim(V) \leq 2 \cdot 100 + 4$ and every polynomial $Q_i \in \tilde{\mathcal{Q}}$ satisfies the following. Either $Q_i \in \langle V \rangle$ or, there is a linear form a_i such that $\mathrm{Lin}(Q_i) \subseteq \mathrm{span}\{V, a_i\}$. (It might be that $\mathrm{Lin}(Q_i) \subseteq V$ or that $\mathrm{Lin}(Q_i) \subseteq \mathrm{span}\{a_i\}$). Thus $\tilde{\mathcal{Q}}$ satisfies the conditions of Claim 61, and $\dim(\tilde{\mathcal{Q}}) = O(1)$, as we wanted to show.

◁

Claim 46 together with Claim 60 completes the proof of Theorem 44. ◀

5 Conclusions and future research

In this work we solved Problem 2 in the case where all the polynomials are irreducible and of degree at most 2. This result directly relates to the problem of obtaining deterministic algorithms for testing identities of $\Sigma^{[3]}\Pi^{[d]}\Sigma\Pi^{[2]}$ circuits. As mentioned in Section 1, in order to obtain PIT algorithms we need a colored version of this result. Formally, we need to prove the following conjecture:

▶ **Conjecture 69.** *Let $\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ be finite sets of homogeneous quadratic polynomials over $\mathbb{C}$ satisfying the following properties:*

- *Each $Q_o \in \cup_i \mathcal{T}_i$ is either irreducible or a square of a linear form.*[8]
- *No two polynomials are multiples of each other (i.e., every pair is linearly independent).*
- *For every two polynomials Q_1 and Q_2 from distinct sets, whenever Q_1 and Q_2 vanish then also the product of all the polynomials in the third set vanishes.*

Then the linear span of the polynomials in $\cup_i \mathcal{T}_i$ has dimension $O(1)$.

We believe that tools similar to the tools developed in this paper should suffice to verify this conjecture. Another interesting question is a robust version of this problem, which is still open.

▶ **Problem 70.** *Let $\delta \in (0, 1]$. Can we bound the linear dimension (as a function of δ) of a set of polynomials $Q_1, \ldots, Q_m \in \mathbb{C}[x_1, \ldots, x_n]$ that satisfy the following property: For every $i \in [m]$ there exist at least δm values of $j \in [m]$ such that for each such j there is $\mathcal{K}_j \subset [m]$, where $i, j \notin \mathcal{K}_j$ and $\prod_{k \in \mathcal{K}_j} Q_k \in \sqrt{\langle Q_i, Q_j \rangle}$.*

In this result, we prove that the dimension of a set of quadratic polynomials satisfying the conditions of Theorem 4 is bounded by a constant c. By carefully examining the proof, we get that $c \leq 20,000$. This is a very loose bound, and we believe it can be improved. Thus, it might be interesting to find a tight bound on the dimension, or even presenting examples for which the dimension is larger then 10.

Extending our approach to the case of more than 3 multiplication gates (or more than 3 sets as in the colored version of the Sylvester-Gallai theorem (Theorem 14)) seems more difficult. Indeed, an analog of Theorem 5 for this case seems harder to prove in the sense that there are many more cases to consider which makes it unlikely that a similar approach will continue to work as the number of gates get larger. Another difficulty is proving an analog of Theorem 5 for higher degree polynomials. Thus, we believe that a different proof approach may be needed in order to obtain PIT algorithms for $\Sigma^{[O(1)]}\Pi^{[d]}\Sigma\Pi^{[O(1)]}$ circuits.

In this paper we only considered polynomials over the complex numbers. However, we believe (though we did not check the details) that a similar approach should work over positive characteristic as well. Observe that over positive characteristic we expect the dimension of the set to scale like $O(\log |\mathcal{Q}|)$, as for such fields a weaker version of Sylvester-Gallai theorem holds.

▶ **Theorem 71** (Corollary 1.3 in [5]). *Let $V = \{\vec{v}_1, \ldots, \vec{v}_m\} \subset \mathbb{F}_p^d$ be a set of m vectors, no two of which are linearly dependent. Suppose that for every $i, j \in [m]$, there exists $k \in [m]$ such that $\vec{v}_i, \vec{v}_j, \vec{v}_k$ are linearly dependent. Then, for every $\epsilon > 0$*

$$\dim(V) \leq \text{poly}(p/\epsilon) + (4 + \epsilon) \log_p m \ .$$

[8] We replace a linear form with its square to keep the sets homogeneous of degree 2.

References

1 Manindra Agrawal. Proving lower bounds via pseudo-random generators. In Ramaswamy Ramanujam and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2005. doi:10.1007/11590156_6.

2 Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.32.

3 Boaz Barak, Zeev Dvir, Avi Wigderson, and Amir Yehudayoff. Fractional sylvester–gallai theorems. *Proceedings of the National Academy of Sciences*, 110(48):19213–19219, 2013.

4 Malte Beecken, Johannes Mittmann, and Nitin Saxena. Algebraic independence and blackbox identity testing. *Inf. Comput.*, 222:2–19, 2013. doi:10.1016/j.ic.2012.10.004.

5 Arnab Bhattacharyya, Zeev Dvir, Shubhangi Saraf, and Amir Shpilka. Tight lower bounds for linear 2-query lccs over finite fields. *Combinatorica*, 36(1):1–36, 2016. doi:10.1007/s00493-015-3024-z.

6 Peter Borwein and William O. J. Moser. A survey of sylvester's problem and its generalizations. *Aequationes Mathematicae*, 40:111–135, 1990. doi:10.1007/BF02112289.

7 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Hardness vs randomness for bounded depth arithmetic circuits. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.13.

8 David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 3rd edition, 2007.

9 Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Improved rank bounds for design matrices and a new proof of kelly's theorem. *CoRR*, abs/1211.0330, 2012. arXiv:1211.0330.

10 Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. doi:10.1137/05063605X.

11 Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. doi:10.1137/080735850.

12 Michael Edelstein and Leroy M. Kelly. Bisecants of finite collections of sets in linear spaces. *Canadian Journal of Mathematics*, 18:375–280, 1966.

13 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. A deterministic parallel algorithm for bipartite perfect matching. *Commun. ACM*, 62(3):109–115, 2019. doi:10.1145/3306208.

14 Michael A. Forbes. *Polynomial identity testing of read-once oblivious algebraic branching programs*. PhD thesis, Massachusetts Institute of Technology, 2014.

15 Michael A. Forbes and Amir Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2013. doi:10.1007/978-3-642-40328-6_37.

16 Michael A. Forbes, Amir Shpilka, and Ben Lee Volk. Succinct hitting sets and barriers to proving lower bounds for algebraic circuits. *Theory of Computing*, 14(1):1–45, 2018. doi:10.4086/toc.2018.v014a018.

17 Ankit Gupta. Algebraic geometric techniques for depth-4 PIT & sylvester-gallai conjectures for varieties. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:130, 2014. URL: http://eccc.hpi-web.de/report/2014/130.

18 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth three. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 578–587, 2013. doi:10.1109/FOCS.2013.68.

19 Rohit Gurjar and Thomas Thierauf. Linear matroid intersection is in quasi-nc. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 821–830. ACM, 2017. doi:10.1145/3055399.3055440.

20 Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 262–272. ACM, 1980. doi:10.1145/800141.804674.

21 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

22 Zohar S. Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 274–285. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.18.

23 Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 198–207. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.67.

24 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *Computational Complexity*, 24(2):295–331, 2015. doi:10.1007/s00037-015-0102-y.

25 Ketan D. Mulmuley. Geometric complexity theory V: Efficient algorithms for Noether normalization. *J. Amer. Math. Soc.*, 30(1):225–309, 2017.

26 Shir Peleg and Amir Shpilka. A generalized sylvester-gallai type theorem for quadratic polynomials. *CoRR*, abs/2003.05152, 2020. arXiv:2003.05152.

27 Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. *Combinatorica*, 38(5):1205–1238, 2018. doi:10.1007/s00493-016-3460-4.

28 Nitin Saxena. Progress on polynomial identity testing. *Bulletin of EATCS*, 99:49–79, 2009. URL: https://eccc.weizmann.ac.il/report/2009/101/.

29 Nitin Saxena. Progress on polynomial identity testing-ii. In M. Agrawal and V. Arvind, editors, *Perspectives in Computational Complexity: The Somenath Biswas Anniversary Volume*, Progress in Computer Science and Applied Logic, pages 131–146. Springer International Publishing, 2014. URL: https://books.google.co.il/books?id=U7ApBAAAQBAJ.

30 Nitin Saxena and Comandur Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012. doi:10.1137/10848232.

31 Nitin Saxena and Comandur Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33, 2013. doi:10.1145/2528403.

32 Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. Comput.*, 38(6):2130–2161, 2009. doi:10.1137/070694879.

33 Amir Shpilka. Sylvester-gallai type theorems for quadratic polynomials. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1203–1214. ACM, 2019. doi:10.1145/3313276.3316341.

34 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

35 Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 31:1–31:53. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.31.

36 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-nc. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.70.

Simultaneous Max-Cut Is Harder to Approximate Than Max-Cut

Amey Bhangale[1]
Department of Computer Science and Engineering, University of California, Riverside, CA, USA
ameyb@ucr.edu

Subhash Khot
Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, NY, USA
khot@cs.nyu.edu

Abstract

A systematic study of simultaneous optimization of constraint satisfaction problems was initiated by Bhangale et al. [ICALP, 2015]. The simplest such problem is the simultaneous Max-Cut. Bhangale et al. [SODA, 2018] gave a .878-minimum approximation algorithm for simultaneous Max-Cut which is *almost optimal* assuming the Unique Games Conjecture (UGC). For single instance Max-Cut, Goemans-Williamson [JACM, 1995] gave an α_{GW}-approximation algorithm where $\alpha_{GW} \approx .87856720...$ which is *optimal* assuming the UGC.

It was left open whether one can achieve an α_{GW}-minimum approximation algorithm for simultaneous Max-Cut. We answer the question by showing that there exists an absolute constant $\varepsilon_0 \geqslant 10^{-5}$ such that it is NP-hard to get an $(\alpha_{GW} - \varepsilon_0)$-minimum approximation for simultaneous Max-Cut assuming the Unique Games Conjecture.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Simultaneous CSPs, Unique Games hardness, Max-Cut

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.9

Supplementary Material `https://github.com/asteric7/simultaneous_maxcut_gadget`

Funding *Amey Bhangale*: Research supported by Irit Dinur's ERC-CoG grant 772839.
Subhash Khot: Supported by the NSF Award CCF-1813438, the Simons Collaboration on Algorithms and Geometry, and the Simons Investigator Award.

Acknowledgements Our numerical calculations involve minor modifications of the prover code [1], written by Austrin et al. [2], which uses interval arithmetic to get a computer generated proof. We are indebted to the authors of [2] for making it available online. We are also thankful to the anonymous reviewers whose comments helped greatly in improving the presentation of the paper.

1 Introduction

Constraint satisfaction problems (CSPs) are among the most fundamental problems in computer science and Max-Cut is the most basic among those. In Max-Cut we are given an undirected (weighted) graph $G(V, E)$ on the vertex set V along with the edge set E. We assume that the total weight of edges is 1 and denote the number of vertices by n. The objective is to partition V into two sets $S, \overline{S}$ so as to maximize the total weight of *crossing* edges i.e. having one endpoint in S and the other in $\overline{S}$. Let us denote the cut value corresponding to the partition $(S, \overline{S})$ by $\mathbf{Cut}_G(S)$. Since Max-Cut is one of the classic

[1] Part of this work was done during a postdoctoral stay at Weizmann Institute of Science, Israel and while visiting the Simons Institute for the Theory of Computing, UC Berkeley, CA, USA.

NP-complete problems, we resort to finding an approximate solution. The seminal result of Goemans-Williamson [9] gave an $\alpha_{GW} \approx .87856720...$ approximation algorithm for Max-Cut. The exact value of the approximation factor is given by the following expression:

$$\alpha_{GW} := \min_{\rho \in [-1,0]} \frac{2\arccos(\rho)}{\pi(1-\rho)}.$$

In [7], the authors initiate the study of simultaneous approximation algorithms for constraint satisfaction problems. In particular, the study of simultaneous Max-Cut which we describe next is the main focus of this paper. In simultaneous Max-Cut the input consists of a collection of weighted undirected graphs $G_1, G_2, \ldots, G_k$ on the same set of vertices V but with different edge weights $E_1, E_2, \ldots, E_k$. The goal is to find a single cut $(S, \overline{S})$ which is *good* for each of G_i. The notion of how *good* the cut is needs to be defined formally. Following are the two notions that [7] considered in their paper:

1. **Pareto approximation:** Suppose $(c_1, c_2, \ldots, c_k) \in [0,1]^k$ is such that there exists a partition $(S, \overline{S})$ such that $\mathbf{Cut}_{G_i}(S) \geqslant c_i$ for all $i \in [k]$. The objective is to find such a partition. An α-*Pareto approximation* algorithm in this context is a polynomial time algorithm, which when given $(c_1, c_2, \ldots, c_k) \in [0,1]^k$ as input, finds a partition $(S, \overline{S})$ such that $\mathbf{Cut}_{G_i}(S) \geqslant \alpha \cdot c_i$ for all $i \in [k]$.
2. **Minimum approximation:** This is the Pareto approximation problem when $c_1 = c_2 = \ldots = c_k$. Define the optimal value of the instance to be

   $$c = \max_{S \subseteq V} \min_{i \in [k]} \mathbf{Cut}_{G_i}(S).$$

   An α-*minimum approximation* algorithm in this context is a polynomial time algorithm which finds a cut $(S, \overline{S})$ such that $\min_{i \in [k]} \mathbf{Cut}_{G_i}(S) \geqslant \alpha \cdot c$.

Note that an α-Pareto approximation gives an α-minimum approximation of simultaneous Max-Cut. For any constant $k \geqslant 1$ and $\varepsilon > 0$, [7] gave $(\frac{1}{2} - \varepsilon)$-Pareto approximation for simultaneous Max-Cut which was improved to .878-Pareto approximation by [6].

▶ **Theorem 1.** *(Pareto approximation algorithm of [6]) Given a collection of graphs $G_i(V, E_i)$ for $1 \leqslant i \leqslant k$ and $c_1, c_2, \ldots, c_k \in [0,1]$ with a guarantee that there exists a partition $(S^\star, \overline{S^\star})$ such that $\mathbf{Cut}_{G_i}(S^\star) \geqslant c_i$ for all i, there exists a randomized algorithm running in time $|V|^{poly(k)}$ which outputs a cut $(S, \overline{S})$ with a guarantee that $\mathbf{Cut}_{G_i}(S) \geqslant .878 \cdot c_i$ for all i.*

In terms of hardness of approximation, the Unique Games Conjecture by [11] gives the tightness of the Goemans-Williamson algorithm for approximating Max-Cut. [12] showed that if approximating a certain optimization problem called the *Unique Games* is NP-hard then it is NP-hard to approximate Max-Cut better than α_{GW} factor. Trivially, the Unique Games Conjecture based hardness (UG-hard henceforth) of approximating Max-Cut within a factor of $(\alpha_{GW} + \varepsilon)$ implies that getting an $(\alpha_{GW} + \varepsilon)$-minimum approximation for simultaneous Max-Cut is *also* UG-hard for all constants $\varepsilon > 0$. As $.878 < \alpha_{GW}$, this leaves an intriguing question of achieving an α_{GW}-minimum approximation for simultaneous Max-Cut.

We answer this question in this paper by proving that there exists an absolute constant $\varepsilon_0 \geqslant 10^{-5}$ such that it is UG-hard to get an $(\alpha_{GW} - \varepsilon_0)$-minimum approximation (and hence $(\alpha_{GW} - \varepsilon_0)$-Pareto approximation) for simultaneous Max-Cut, unlike the single instance Max-Cut.

▶ **Theorem 2** (Main theorem)**.** *There exists an absolute constant $\varepsilon_0 \geqslant 10^{-5}$ such that assuming the Unique Games Conjecture, it is NP-hard to achieve $(\alpha_{GW} - \varepsilon_0)$-minimum approximation for simultaneous Max-Cut.*

One interesting feature of our reduction is that the hard instance involves only *three* graphs! This should be compared with the algorithm of [6] from Theorem 1 which works for any constantly many number of instances of Max-Cut. It will be interesting to know whether one can achieve α_{GW}-minimum approximation for the simultaneous Max-Cut when the number of instances is *two*.

1.1 Organisation

We start with preliminaries in Section 2 where we formally define the simultaneous Max-Cut problem, various distributions on the Boolean hypercube, invariance principle and the Unique Games Conjecture. In Section 3, we present the dictatorship tests for Max-Cut and simultaneous Max-Cut. Finally, in Section 4, we provide our reduction from the Unique Games to the simultaneous Max-Cut.

2 Preliminaries

We first define the main problem that we study. Given an undirected weighted graph $G(V, E)$, the cut value of the partition $(S, \overline{S})$ of V, denoted by $\mathbf{Cut}_G(S)$, is defined to be the total weight of the edges whose endpoints are in different parts. The Max-Cut of a graph G is the maximum cut value over all the partitions of V.

▶ **Definition 3.** *(Simultaneous Max-Cut) An instance of simultaneous Max-Cut is a collection of undirected weighted graphs $G_i(V, E_i)$, $1 \leqslant i \leqslant k$, on the same set of vertices.*

Given an instance $G_i(V, E_i)$, $1 \leqslant i \leqslant k$ of simultaneous Max-Cut and $(c_1, c_2, \ldots, c_k) \in [0, 1]^k$ such that there exists a partition $(S, \overline{S})$ satisfying $\mathbf{Cut}_{G_i}(S) \geqslant c_i$ for all $i \in [k]$. The objective is to find such a partition. An α-Pareto approximation algorithm in this context is a polynomial time algorithm, which when given $(c_1, c_2, \ldots, c_k) \in [0, 1]^k$ as input, finds a partition $(S, \overline{S})$ such that $\mathbf{Cut}_{G_i}(S) \geqslant \alpha \cdot c_i$ for all $i \in [k]$.

We work with the problem of finding α-minimum approximation for simultaneous Max-Cut, which is a special case of the above problem. In this case, the optimum value is given by:

$$\textsc{Opt}(G_1, G_2, \ldots, G_k) := \max_{S \subseteq V} \min_{i \in [k]} \mathbf{Cut}_{G_i}(S).$$

An algorithm is called an α-minimum approximation for simultaneous Max-Cut if given input the graphs $G_1, G_2, \ldots, G_k$, it always outputs a cut $(T, \overline{T})$ such that

$$\min_{i \in [k]} \mathbf{Cut}_{G_i}(T) \geqslant \alpha \cdot \textsc{Opt}(G_1, G_2, \ldots, G_k).$$

For $a, b, c \in \mathbb{R}_{\geqslant 0}$ and a polynomial $P(x_1, x_2, \ldots, x_t)$, we define

$$\underset{x_1, \ldots, x_t \in [a,b]}{\texttt{range}} \{P(x_1, \ldots, x_t) \geqslant c\} := \{(x_1, \ldots, x_t) \,|\, x_i \in [a, b]\ \forall i \in [t] \text{ and } P(x_1, \ldots, x_t)) \geqslant c\}.$$

2.1 Analysis of Boolean functions

We will be working with functions $f : \{0, 1\}^n \to \mathbb{R}$ on the Boolean hypercube. For $q \in [0, 1]$, let μ_q be the distribution of a q-biased bit given as $\mu_q(1) = q$ and $\mu_q(0) = 1 - q$. Let $\mu_q^{\otimes n}$ be the corresponding product distribution on $\{0, 1\}^n$. Let $L^2(\mu_q^{\otimes n})$ be the space of functions $f : \{0, 1\}^n \to \mathbb{R}$ endowed with the distribution $\mu_q^{\otimes n}$. Also, let $\mu_q(f) := \mathbf{E}_{x \sim \mu_q^{\otimes n}}[f(x)]$.

Given x define the ρ-correlated copy y of x as follows:

▶ **Definition 4.** *Given ρ and $x \sim \mu_q^{\otimes n}$ we write $y \sim N_\rho(x)$ to denote the ρ-correlated copy of x where the distribution $N_\rho(x)$ is as follows: Independently for each $i \in [n]$, if $x_i = 1$ then set $y_i = 1$ with probability $q + \rho(1-q)$, and $y_i = 0$ otherwise. If $x_i = 0$ then set $y_i = 1$ with probability $q - \rho q$, and $y_i = 0$ otherwise.*

We will be interested in the setting when $\rho \leqslant 0$. In this case, if we want y to be distributed according to $\mu_q^{\otimes n}$ then ρ *cannot* be arbitrary in $[-1, 0]$. Specifically, for a given $q \in (0, 1)$, ρ must be in the following interval:

$$\rho \in \begin{cases} [-q/(1-q), 0), & \text{if } q < 1/2, \\ (-1, 0), & \text{if } q = 1/2, \\ [-(1-q)/q, 0), & \text{if } q > 1/2. \end{cases}$$

As in [4], we will denote the above interval as $\kappa(q)$ for any given $q \in (0, 1)$. Next we define the noise operator T_ρ over the probability space $L^2(\mu_q^{\otimes n})$.

▶ **Definition 5.** *Let $q \in (0, 1)$ and $\rho \in [-1, 1]$. The noise operator $T_\rho : L^2(\mu_q^{\otimes n}) \to L^2(\mu_q^{\otimes n})$ is given as follows:*

$$T_\rho f(x) = \mathop{\mathbf{E}}_{y \sim N_\rho(x)}[f(y)].$$

▶ **Definition 6** (Influence)**.** *Let $f \in L^2(\mu_q^{\otimes n})$. The influence of the i^{th} variable on f, denoted by $\mathbf{Inf}_i(f)$ is defined as:*

$$\mathbf{Inf}_i(f) = \mathop{\mathbf{E}}_{x \sim \mu_q^{\otimes n}}[\mathbf{Var}_{x_i \sim \mu_q}[f(x) | x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]].$$

The useful property of the operator T_ρ is that if $\mathbf{Var}[f]$ is bounded then the image of f under T_ρ has a bounded number of influential variables. The proof of the lemma can be found in [10, Lemma 3.6]

▶ **Lemma 7.** *Let $q \in (0, 1)$ and $\rho \in \kappa(q)$ and $f \in L^2(\mu_q^{\otimes n})$. Then, for any $\tau > 0$ we have*

$$|\{i \in [n] \,|\, \mathbf{Inf}_i[T_\rho f] \geqslant \tau\}| \leqslant \frac{\mathbf{Var}[f]}{2\tau e \ln(1/|\rho|)}.$$

We have the following definition for functions whose all the influences are low (under the map T_ρ).

▶ **Definition 8.** *Let $q \in (0, 1)$ and $0 < \varepsilon, \delta < 1$. A function $f \in L^2(\mu_q^{\otimes n})$ is called (ε, δ)-quasirandom if for all $i \in [n]$, we have $\mathbf{Inf}_i[T_{1-\delta} f] \leqslant \varepsilon$.*

2.2 Invariance Principle

We need the following definition related to correlated spaces defined by Mossel [13].

▶ **Definition 9.** *Let $(\Omega_1 \times \Omega_2, \mu)$ be a finite correlated space, the correlation between Ω_1 and Ω_2 with respect to μ is defined as*

$$\rho(\Omega_1, \Omega_2; \mu) := \sup_{\substack{f:\Omega_1 \to \mathbb{R}, g:\Omega_2 \to \mathbb{R}, \\ \mathbf{Var}[f]=\mathbf{Var}[g]=1}} \mathbf{Cov}[f, g].$$

We will need the following Gaussian stability measure in our analysis:

▶ **Definition 10.** *Let $\phi : \mathbb{R} \to [0,1]$ be the cumulative distribution function of the standard Gaussian random variable. For a parameter $\rho, \nu_1, \nu_2 \in [0,1]$, we define the following two quantities:*

$$\underline{\Gamma}_\rho(\nu_1,\nu_2) = \Pr[X \leqslant \phi^{-1}(\nu_1), Y \geqslant \phi^{-1}(1-\nu_2)],$$

$$\overline{\Gamma}_\rho(\nu_1,\nu_2) = \Pr[X \leqslant \phi^{-1}(\nu_1), Y \leqslant \phi^{-1}(\nu_2)],$$

where X and Y are two standard Gaussian variables with covariance ρ. We also define $\overline{\Gamma}_\rho(\nu) = \overline{\Gamma}_\rho(\nu,\nu)$ and $\underline{\Gamma}_\rho(\nu) = \underline{\Gamma}_\rho(\nu,\nu)$ for notational convenience.

We are now ready to state a version of invariance principle from [13] which follows from Theorem 3.1 in [8] that we need for our reduction. For variables $\varepsilon_1, \varepsilon_2, \varepsilon_3, \ldots$, by $\varepsilon_1(\varepsilon_2, \varepsilon_3, \ldots)$ we mean ε_1 is a function of $\varepsilon_2, \varepsilon_3, \ldots$ such that $\varepsilon_1 \to 0$ as all $\varepsilon_2, \varepsilon_3, \ldots \to 0$.

▶ **Theorem 11** ([13, 8])**.** *Let $(\Omega_1 \times \Omega_2, \mu)$ be a finite correlated space, the correlation between Ω_1 and Ω_2 with respect to μ is $\rho \in [0,1]$. Then for any $\tau > 0$ there exists $\varepsilon(\tau) > 0, \delta(\tau) > 0$ such that if $f : \Omega_1^n \to [0,1]$ and $g : \Omega_2^n \to [0,1]$ are two functions satisfying*

$$\min(\mathbf{Inf}_i(T_{1-\delta}f), \mathbf{Inf}_i(T_{1-\delta}g)) \leqslant \varepsilon, \tag{1}$$

for all $i \in [n]$, then it holds that

$$\underline{\Gamma}_\rho(\nu_1,\nu_2) - \tau \leqslant \mathop{\mathbf{E}}_{(x,y)\sim\mu^{\otimes n}}[f(x)g(y)] \leqslant \overline{\Gamma}_\rho(\nu_1,\nu_2) + \tau,$$

where $\nu_1 = \mathbf{E}[f]$, $\nu_2 = \mathbf{E}[g]$.

▶ Remark 12. One difference between the versions of invariance principle in Mossel [13] and Dinur et al. [8] is that in [13] instead of a min in (1), it was a max. This improvement was crucial for hardness of graph coloring in [8]. For our hardness result, the difference is not important.

We will be working with correlated spaces $(\{0,1\} \times \{0,1\}, \mu)$ with negative correlation. The following corollary follows from the above theorem.

▶ **Corollary 13.** *Assume the settings in Theorem 11 for a correlated space $(\{0,1\} \times \{0,1\}, \mu)$ except $\rho \in [-1,0)$, then it holds that*

$$\overline{\Gamma}_\rho(\nu_1,\nu_2) - \tau \leqslant \mathop{\mathbf{E}}_{(x,y)\sim\mu^{\otimes n}}[f(x)g(y)].$$

Proof. Define $f'(x) = 1 - f(1-x)$ and let $\rho' = -\rho$. We apply Theorem 11 to f', g and ρ'

$$\begin{aligned}\mathbf{E}[f(x)g(y)] &= \mathbf{E}[g(y)] - \mathbf{E}[f'(-x)g(y)] \\ &\geqslant \nu_2 - \overline{\Gamma}_{\rho'}(1-\nu_1,\nu_2) - \tau \\ &= \nu_2 - \overline{\Gamma}_{\rho'}(1-\nu_1,\nu_2) - \underline{\Gamma}_{\rho'}(\nu_1,\nu_2) + \underline{\Gamma}_{\rho'}(\nu_1,\nu_2) - \tau.\end{aligned}$$

Now, $\overline{\Gamma}_{\rho'}(1-\nu_1,\nu_2) + \underline{\Gamma}_{\rho'}(\nu_1,\nu_2) = \overline{\Gamma}_{\rho'}(\nu_2,1-\nu_1) + \underline{\Gamma}_{\rho'}(\nu_2,\nu_1) = \nu_2$. Therefore,

$$\begin{aligned}\mathbf{E}[f(x)g(y)] &\geqslant \underline{\Gamma}_{\rho'}(\nu_1,\nu_2) - \tau \\ &= \overline{\Gamma}_\rho(\nu_1,\nu_2) - \tau.\end{aligned}$$

◀

2.3 Unique Games

Our hardness result is based on the Unique Games Conjecture. First, we define what the Unique Game is:

▶ **Definition 14** (Unique Games). *An instance $G = (U, V, E, [L], \{\pi_e\}_{e\in E})$ of the Unique Games constraint satisfaction problem consists of a bi-regular bipartite graph (U, V, E), an alphabet $[L]$ and a permutation map $\pi_e : [L] \to [L]$ for every edge $e \in E$. Given a labeling $\ell : U \cup V \to [L]$, , an edge $e = (u, v)$ is said to be satisfied by ℓ if $\pi_e(\ell(v)) = \ell(u)$.*

G is said to be at most δ-satisfiable *if every labeling satisfies at most a δ fraction of the edges.*

The following is a conjecture by Khot [11] which has been used to prove many *tight* inapproximability results.

▶ **Conjecture 15** (Unique Games Conjecture [11]). *For every sufficiently small $\delta > 0$ there exists $L \in \mathbb{N}$ such that the following holds. Given a an instance $(U, V, E, [L], \{\pi_e\}_{e\in E})$ of Unique Games it is NP-hard to distinguish between the following two cases:*

- *YES case: There exist an assignment that satisfies at least $(1 - \delta)$ fraction of the edges.*
- *NO case: Every assignment satisfies at most δ fraction of the edge constraints.*

3 Dictatorship Tests

A function $f : \{0, 1\}^n \to \mathbb{R}$ is called a dictator function if $f(x_1, x_2, \ldots, x_n) = x_i$ for some $i \in [n]$. Dictatorship tests are designed to distinguish between the cases when f is a dictator function and f is an (ε, δ)-quasirandom function for small enough $\varepsilon, \delta > 0$.

3.1 Dictatorship Test for Max-Cut

The α_{GW} Unique Games hardness of Max-Cut relies on the analysis of a certain dictatorship test that we describe next. This will lead us to our dictatorship test for simultaneous Max-Cut. Consider the following test:

Given $f : \{0, 1\}^n \to \{0, 1\}$,

1. Select $x \in \{0, 1\}^n$ uniformly at random.
2. Select a ρ-correlated copy y of x i.e. independently for each $i \in [n]$ set $y_i = x_i$ w.p. $\frac{1+\rho}{2}$ and set $y_i = \overline{x_i}$ w.p. $\frac{1-\rho}{2}$.
3. Check if $f(x) \neq f(y)$.

We have the following completeness property of the dictatorship test, which is easy to show.

▶ **Lemma 16.** *If f is a dictator function, then the test passes with probability $\frac{1-\rho}{2}$.*

The following soundness of the test relies on the "Majority of the Stablest" theorem, which roughly states that among all the Boolean functions with all the influences low, Majority function is the most stable under "positive" perturbation.

▶ **Lemma 17** ([14]). *For $\rho \in [-1, 0)$, if f is (ε, δ)-quasirandom, then the test passes with probability at most $\frac{\arccos(\rho)}{\pi} + \tau(\varepsilon, \delta)$.*

This dictatorship test can be *composed* with Unique Games [12] which gives α_{GW}-hardness of approximation for Max-Cut, where α_{GW} is given by the following expression.

$$\min_{\rho\in[-1,0)} \frac{\frac{\arccos(\rho)}{\pi}}{\frac{1-\rho}{2}} = \alpha_{GW} = .87856720...$$

3.2 Dictatorship Test for simultaneous Max-Cut

In the above dictatorship test, we get a family of graphs parameterized by the quantity ρ. This might give a way to construct multiple instances of Max-Cut, one for each $\rho \in (-1, 1)$. However, this will not work and instead we will construct instances whose vertex set is concentrated around the $q \cdot n^{th}$ slice of the hypercube for some $q \in (0, 1)$. This will give us the family of graphs for each $q \in (0, 1)$ and ρ.

Our final dictatorship test for the simultaneous Max-Cut problem will consist of *three* graphs, G_1 on the qn^{th} slice, G_2 on the $(1-q)n^{th}$ slice and G_3 will be a bipartite graph between the $qn^{th}$and $(1-q)n^{th}$ slice of the Boolean hypercube $\{0,1\}^n$.

▶ **Definition 18** (ρ-correlated μ_q strings). *For every $q \in [0,1]$ and $\rho \in [-1, 0)$, define $\mathcal{A}_{\rho,q}^{\otimes n}$ to be the product distribution on $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ where, $\mathcal{A}_{\rho,q} : \{0,1\}^2 \to \mathbb{R}_{\geqslant 0}$ is defined as follows:*

$$\mathcal{A}_{\rho,q}(0,0) = (1-q) - t,$$
$$\mathcal{A}_{\rho,q}(0,1) = t,$$
$$\mathcal{A}_{\rho,q}(1,0) = t,$$
$$\mathcal{A}_{\rho,q}(1,1) = q - t,$$

where $t = (q - q^2)(1 - \rho)$. As mentioned before, ρ in the above definition must satisfy the following property

$$\rho \in \begin{cases} [-q/(1-q), 0)\,, & \text{if } q < 1/2, \\ [-1, 0)\,, & \text{if } q = 1/2, \\ [-(1-q)/q, 0)\,, & \text{if } q > 1/2. \end{cases}$$

▶ **Definition 19** (ρ-correlated (x, y) where $x \sim \mu_q^{\otimes n}$ and $y \sim \mu_{(1-q)}^{\otimes n}$). *For every $q \in [0,1]$ and $\rho \in [-1, 0)$, define $\mathcal{B}_{\rho,q}^{\otimes n}$ to be the product distribution on $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ where, $\mathcal{B}_{\rho,q} : \{0,1\}^2 \to \mathbb{R}_{\geqslant 0}$ is defined as follows:*

$$\mathcal{B}_{\rho,q}(0,0) = t,$$
$$\mathcal{B}_{\rho,q}(0,1) = (1-q) - t,$$
$$\mathcal{B}_{\rho,q}(1,0) = q - t,$$
$$\mathcal{B}_{\rho,q}(1,1) = t,$$

where $t = (q - q^2)(1 + \rho)$. Note that ρ in the above definition must satisfy the following property:

$$\rho \in \begin{cases} [-1, q/(1-q))\,, & \text{if } q < 1/2, \\ [-1, 0)\,, & \text{if } q = 1/2, \\ [-1, (1-q)/q)\,, & \text{if } q > 1/2. \end{cases}$$

We will define a simultaneous Max-Cut instance on the vertex set $\{0,1\}^n$. The instance consists of three weighted graphs G_1, G_2 and G_3. We fix $q_\star = .58$, $\rho_1 = -\frac{1-q_\star}{q_\star}$ and $\rho_2 = \frac{2q_\star^2 - 1}{2q_\star(1-q_\star)}$.

- G_1 is concentrated around the $q_\star n^{th}$ slice of the hypercube. More formally, the edge distribution of this graph is given by the distribution $\mathcal{A}_{\rho_1,q_\star}^{\otimes n}$.
- G_2 is concentrated around the $(1-q_\star)n^{th}$ slice of the hypercube. Formally, the edge distribution of this graph is given by the distribution $\mathcal{A}_{\rho_1,(1-q_\star)}^{\otimes n}$.
- G_3 is roughly a bipartite graph between the $q_\star n^{th}$ and $(1-q_\star)n^{th}$ slices of the hypercube. The edge distribution is given by the distribution $\mathcal{B}_{\rho_2,q_\star}^{\otimes n}$.

A few remarks about the choice of parameters: We arrive at the choice of $q_\star = .58$ by doing numerical calculations. Setting $\rho_1 = -\frac{1-q_\star}{q_\star}$ is a natural choice as it is the maximum negative correlation that the two $q_\star$-biased bits can have. Finally, $\rho_2 = \frac{2q_\star^2-1}{2q_\star(1-q_\star)}$ is chosen such that the following is satisfied:

$$\Pr_{(x_i,y_i)\sim\mathcal{A}_{\rho_1,q_\star}}[x_i \neq y_i] = \Pr_{(x_i,y_i)\sim\mathcal{B}_{\rho_2,q_\star}}[x_i \neq y_i].$$

3.2.1 Completeness

▶ **Lemma 20.** *If f is a dictator function then the value of the cut induced by f is $2(1-q_\star)$ for all G_1, G_2, G_3.*

Proof. The proof is easy in this case. Suppose f is an i^{th} dictator for some $i \in [n]$. This induces a cut $(S_f, \overline{S}_f)$ where $S_f = \{x \in \{0,1\}^n \,|\, x_i = 0\}$. In this case, $\mathbf{Cut}_{G_1}(S_f)$ is equal to the probability that (x_i, y_i) sampled from $\mathcal{A}_{\rho_1,q_\star}$ are not equal. This is precisely $2(q_\star - q_\star^2)(1-\rho_1)$ which is equal to $2(1-q_\star)$ by the choice of $\rho_1 = -\frac{1-q_\star}{q_\star}$.

Similarly, $\mathbf{Cut}_{G_2}(S_f)$ is equal to the probability that (x_i, y_i) sampled from $\mathcal{A}_{\rho_1,(1-q_\star)}$ are not equal. This is also $2(1-q_\star)$.

For G_3,

$$\mathbf{Cut}_{G_3}(S_f) = \Pr_{(x_i,y_i)\sim\mathcal{B}_{\rho_2,q_\star}}[x_i \neq y_i] = 1 - 2(q_\star - q_\star^2)(1+\rho_2).$$

By our choice of ρ_2, this also equals to $2(1-q_\star)$. ◀

3.2.2 Soundness

▶ **Lemma 21.** *Let $f : \{0,1\}^n \to \{0,1\}$ be an (ε,δ)-quasirandom function and let $(S_f, \overline{S}_f)$ be the cut induced by f. Then*

$$\min_{i\in[3]} \mathbf{Cut}_{G_i}(S_f) \leqslant (\alpha_{GW} - 10^{-5}) \cdot 2(1-q_\star) + \tau(\varepsilon,\delta).$$

Proof. The proof is as follows:

1. We have an (ε,δ)-quasirandom function $f : \{0,1\}^n \to \{0,1\}$. Invariance principle says that in order to get at least $(\alpha_{GW} - 10^{-5})$ approximation for G_1, the density of function $\mu_{q_\star}(f)$ must be in some range. This essentially follows from the analysis of Austrin et al. [3, 4]. Furthermore, the invariance principle precisely tells us that this is similar to what approximation ratio the *biased hyperplane rounding* algorithm of [6] gives us on a pair of vectors with SDP biases $q_\star$ when rounded using rounding bias $\mu_{q_\star}(f)$. (See [6] for the formal definitions of SDP bias and rounding bias). More formally, if the $\mu_{q_\star}(f) = \nu_1$ then the cut value is bounded as follows:

$$\begin{aligned}
\mathbf{Cut}_{G_1}(S_f) &= \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}_{\rho_1,q_\star}^{\otimes n}}\left[\frac{1-(1-2f(x))(1-2f(y))}{2}\right] \\
&= \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}_{\rho_1,q_\star}^{\otimes n}}[f(x)+f(y)-2f(x)f(y)] \\
&= \nu_1 + \nu_1 - 2\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}_{\rho_1,q_\star}^{\otimes n}}[f(x)f(y)] \\
&\leqslant 2\nu_1 - 2\overline{\Gamma}_{\rho_1}(\nu_1) + \tau_1(\varepsilon,\delta),
\end{aligned}$$

where the last inequality follows from Corollary 13. Let us define the following range:

$$R_1(\varepsilon,\delta) := \underset{\nu_1\in[0,1]}{\texttt{range}} \left\{ \frac{2\nu_1 - 2\overline{\Gamma}_{\rho_1}(\nu_1) + \tau_1(\varepsilon,\delta)}{2(1-q_\star)} \geqslant (\alpha_{GW} - 10^{-5}) \right\}.$$

$R_1(\varepsilon,\delta)$ is the set of all biases $\mu_{q_\star}(f)$ that gives $\mathbf{Cut}_{G_1}(S_f)$ which is at least $(\alpha_{GW} - 10^{-5})$ factor greater than $2(1-q_\star)$. For a sufficiently small $\varepsilon,\delta > 0$ and our given values of $q_\star$ and ρ_1, numerical calculations show that

$$R_1(\varepsilon,\delta) \subseteq [.43676765, .56323235].$$

2. Same is true for G_2. More formally, if the $\mu_{1-q_\star}$ measure of f is ν_2 then the cut value is bounded above by $2\nu_2 - 2\overline{\Gamma}_{\rho_1}(\nu_2)$ and we have

$$R_2(\varepsilon,\delta) := \underset{\nu_2\in[0,1]}{\texttt{range}} \left\{ \frac{2\nu_2 - 2\overline{\Gamma}_{\rho_1}(\nu_2) + \tau_2(\varepsilon,\delta)}{2(1-q_\star)} \geqslant (\alpha_{GW} - 10^{-5}) \right\}.$$

3. This fixes possible densities of f with respect to the $\mu_{q_\star}^{\otimes n}$ and $\mu_{(1-q_\star)}^{\otimes n}$ distributions. Both these densities should lie in $[.43676765, .56323235]$ if we want $\mathbf{Cut}_{G_1}(S_f) \geqslant (\alpha_{GW} - 10^{-5}) \cdot 2(1-q_\star)$ and $\mathbf{Cut}_{G_2}(S_f) \geqslant (\alpha_{GW} - 10^{-5}) \cdot 2(1-q_\star)$. Now we use the full power of the invariance principle to claim that the value of the cut given by such an f is similar to what the biased hyperplane rounding gives us on the graph G_3.

$$\begin{aligned}
\mathbf{Cut}_{G_3}(S_f) &= \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{B}^{\otimes n}_{\rho_2,q_\star}} \left[\frac{1-(1-2f(x))(1-2f(y))}{2}\right] \\
&= \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{B}^{\otimes n}_{\rho_2,q_\star}} [f(x) + f(y) - 2f(x)f(y)] \\
&= \nu_1 + \nu_2 - 2 \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{B}^{\otimes n}_{\rho_2,q_\star}} [f(x)f(y)] \\
&\leqslant \nu_1 + \nu_2 - \overline{\Gamma}_{\rho_2}(\nu_1,\nu_2) + \tau_3(\varepsilon,\delta).
\end{aligned}$$

Here again, the last inequality follows from Corollary 13. By doing numerical calculations, we show that for the following range

$$R(\varepsilon,\delta) := \underset{\nu_1,\nu_2\in[0,1]}{\texttt{range}} \left\{ \frac{\nu_1+\nu_2 - 2\overline{\Gamma}_{\rho_2}(\nu_1,\nu_2) + \tau_3(\varepsilon,\delta)}{2(1-q_\star)} \geqslant (\alpha_{GW} - 10^{-5}) \right\},$$

$R(\varepsilon,\delta) \cap (R_1(\varepsilon,\delta) \times R_2(\varepsilon,\delta)) = \emptyset$ for sufficiently small $\varepsilon,\delta > 0$.

Therefore, no matter which f we start with, if it is (ε,δ)-quasirandom for sufficiently small $\varepsilon,\delta > 0$, then there exists an $i \in [3]$ such that the cut guaranteed by S_f on G_i is strictly less that $(\alpha_{GW} - 10^{-5}) \cdot 2(1-q_\star) + \tau(\varepsilon,\delta)$. ◀

4 Actual Reduction

In this section we give a reduction from Unique Games to the simultaneous Max-Cut problem. Given an instance $G = (U, V, E, [L], \{\pi_e\}_{e\in E})$ of the Unique Games, we reduce it to a simultaneous Max-Cut instance $\mathcal{I}$ on the vertex set $\mathcal{V} = V \times 2^{[L]} = \{(v,x) \mid v \in V, x \in \{0,1\}^L\}$.

The instance will involve three weighted graphs $\mathcal{G}_1(\mathcal{V},\mathcal{E}_1), \mathcal{G}_2(\mathcal{V},\mathcal{E}_2)$ and $\mathcal{G}_3(\mathcal{V},\mathcal{E}_3)$ on the common vertex set $\mathcal{V}$. We fix the following parameters: $q_\star = .58, \rho_1 = -\frac{1-q_\star}{q_\star}$ and $\rho_2 = \frac{2q_\star^2-1}{2q_\star(1-q_\star)}$. For a string $x \in \{0,1\}^L$ and a permutation $\pi : [L] \to [L]$, define $x\circ\pi \in \{0,1\}^L$ such that $(x\circ\pi)_i = x_{\pi(i)}$ for all $i \in [L]$. The respective edge weights are given by the following distributions:

1. $\mathcal{E}_1$: Select $u \in U$ uniformly at random and $v_1, v_2 \sim N(u)$ independently and uniformly at random. Select (x, y) according to $\mathcal{A}^{\otimes L}_{\rho_1, q_\star}$ and output $(v_1, x \circ \pi^{-1}_{uv_1}), (v_2, y \circ \pi^{-1}_{uv_2})$.
2. $\mathcal{E}_2$: Select $u \in U$ uniformly at random and $v_1, v_2 \sim N(u)$ independently and uniformly at random. Select (x, y) according to $\mathcal{A}^{\otimes L}_{\rho_1, (1-q_\star)}$ and output $(v_1, x \circ \pi^{-1}_{uv_1}), (v_2, y \circ \pi^{-1}_{uv_2})$.
3. $\mathcal{E}_3$: Select $u \in U$ uniformly at random and $v_1, v_2 \sim N(u)$ independently and uniformly at random. Select (x, y) according to $\mathcal{B}^{\otimes L}_{\rho_2, q_\star}$ and output $(v_1, x \circ \pi^{-1}_{uv_1}), (v_2, y \circ \pi^{-1}_{uv_2})$.

We now prove the completeness and the soundness of the reduction.

▶ **Lemma 22** (Completeness). *If the Unique Games instance G is $(1 - \frac{\eta}{2})$-satisfiable then there exists a cut $(\mathcal{S}, \overline{\mathcal{S}})$ such that*

$$\min_{i \in [3]} \mathbf{Cut}_{\mathcal{G}_i}(\mathcal{S}) \geqslant 2(1 - q_\star) - \eta.$$

▶ **Lemma 23** (Soundness). *There exist absolute constants $\varepsilon_0 \geqslant 10^{-5}$ and $0 < \eta_0 < 1$ such that for all $0 < \eta \leqslant \eta_0$ and $\varepsilon(\eta/2), \delta(\eta/2)$ from Theorem 11, if there exists a cut $(S, \overline{S})$ such that*

$$\min_{i \in [3]} \mathbf{Cut}_{\mathcal{G}_i}(S) \geqslant (\alpha_{GW} - \varepsilon_0)(2(1 - q_\star) - \eta),$$

then there exists an assignment to the Unique Games instance G which satisfies at least $\eta' = \eta \cdot \frac{\varepsilon^2 \cdot e \cdot \ln(1/(1-\delta))}{2}$ fraction of the constraints.

The above two lemmas along with Conjecture 15 show that assuming the Unique Games Conjecture, it is NP-hard to get an α-minimum approximation for simultaneous Max-Cut where $\alpha \leqslant \alpha_{GW} - 10^{-5}$. This proves Theorem 2. We now prove the completeness and soundness of the reduction.

Proof of Lemma 22. Let $\sigma : U \cup V \to [L]$ be an assignment to the Unique Games instance G which satisfies at least $(1 - \eta)$ fraction of the constraints. Consider the following partition $(\mathcal{S}, \overline{\mathcal{S}})$ of $\mathcal{V}$ where

$$\mathcal{S} = \{(v, x) \mid v \in V, x_{\sigma(v)} = 0\}.$$

Let us analyze the value of this cut for the graph $\mathcal{G}_1$:

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &= \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{v_1, v_2 \in N(u)} \Pr_{(x,y) \sim \mathcal{A}^{\otimes L}_{\rho_1, q_\star}} [(v_1, x \circ \pi^{-1}_{uv_1}), (v_2, y \circ \pi^{-1}_{uv_2}) \text{ in different parts}] \\
&= \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{v_1, v_2 \in N(u)} \Pr_{(x,y) \sim \mathcal{A}^{\otimes L}_{\rho_1, q_\star}} [((x \circ \pi^{-1}_{uv_1}))_{\sigma(v_1)} \neq (y \circ \pi^{-1}_{uv_2})_{\sigma(v_2)}] \\
&= \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{v_1, v_2 \in N(u)} \Pr_{(x,y) \sim \mathcal{A}^{\otimes L}_{\rho_1, q_\star}} [x_{\pi^{-1}_{uv_1}(\sigma(v_1))} \neq y_{\pi^{-1}_{uv_2}(\sigma(v_2))}] \\
&\geqslant (1 - \eta) \Pr_{(x,y) \sim \mathcal{A}^{\otimes L}_{\rho_1, q_\star}} [x_{\sigma(u)} \neq y_{\sigma(u)}] \\
&= (1 - \eta) \cdot 2(q_\star - q_\star^2)(1 - \rho_1) \\
&= (1 - \eta) \cdot 2(1 - q_\star) \\
&\geqslant 2(1 - q_\star) - \eta,
\end{aligned}$$

where the first inequality uses the fact that with probability at least $1 - \eta$, both the constraints on the edges (u, v_1) and (u, v_2) are satisfied by the assignment σ. Using similar calculations, we can show that

$$\mathbf{Cut}_{\mathcal{G}_2}(\mathcal{S}) \geqslant (1 - \eta) \cdot 2(q_\star - q_\star^2)(1 - \rho_1) \geqslant 2(1 - q_\star) - \eta$$
$$\mathbf{Cut}_{\mathcal{G}_3}(\mathcal{S}) \geqslant (1 - \eta) \cdot (1 - 2(q_\star - q_\star^2)(1 + \rho_2)) \geqslant 2(1 - q_\star) - \eta.$$

Thus, we have

$$\min_{i\in[3]} \mathbf{Cut}_{\mathcal{G}_i}(\mathcal{S}) \geqslant 2(1-q_\star) - \eta. \qquad \blacktriangleleft$$

We now prove the main soundness lemma:

Proof of Lemma 23. Suppose the value of the Unique Games instance is at most η'. Let $f : V \times 2^{[L]} \to \{0,1\}$ be the indicator function of the cut $(\mathcal{S}, \overline{\mathcal{S}})$. We will show that

$$\min_{i\in[3]} \mathbf{Cut}_{\mathcal{G}_i}(S) \leqslant (\alpha_{GW} - \varepsilon_0)(2(1-q_\star) - \eta).$$

We start with analysing the value $\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S})$:

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &= \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{v_1,v_2\in N(u)} \Pr_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} [f(v_1, x\circ\pi^{-1}_{uv_1}) \neq f(v_2, y\circ\pi^{-1}_{uv_2})] \\
&= \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{v_1,v_2\in N(u)} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} \left[\frac{1}{2} - \frac{(1-2f(v_1, x\circ\pi^{-1}_{uv_1}))(1-2f(v_2, y\circ\pi^{-1}_{uv_2}))}{2}\right] \\
&= \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{v_1,v_2\in N(u)} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} \left[\begin{array}{c} f(v_1, x\circ\pi^{-1}_{uv_1}) + f(v_2, y\circ\pi^{-1}_{uv_2}) - \\ 2f(v_1, x\circ\pi^{-1}_{uv_1}) f(v_2, y\circ\pi^{-1}_{uv_2}) \end{array}\right].
\end{aligned}$$

Define $f_v(x) := f(v,x)$ for $v \in V$ and $f_u(x) := \mathbf{E}_{v\sim N(u)}\left[f_v(x\circ\pi^{-1}_{uv})\right]$ for $u \in U$. Let $\nu^u_q(f) = \mathbf{E}_{x\sim\mu_q^{\otimes L}}[f_u(x)]$ be the q-biased measure of the function f_u and $\nu_q(f) = \mathbf{E}_{u\in U}[\nu^u_q(f)]$ be the average q-biased measure of f. Since we sample $v_1, v_2 \in N(u)$ independently, we have

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &= \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} [f_u(x) + f_u(y) - 2f_u(x)f_u(y)] \\
&= 2\cdot\nu_{q_\star}(f) - 2 \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} [f_u(x)f_u(y)].
\end{aligned}$$

We now show that the expectation in the above expression is lower bounded by the quantity $\overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f)) - \frac{\eta'}{2}$ unless the value of the Unique Games instance is at least η'.

▷ Claim 24. For at least $(1-\eta)$ fraction of $u \in U$,

$$\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} [f_u(x)f_u(y)] \geqslant \overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f)) - \frac{\eta}{2}.$$

Proof. Consider $f_u \in L^2(\mu_{q_\star}^{\otimes n})$ and suppose the claim is not true and we have for at least η fraction of $u \in U$,

$$\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{A}^{\otimes L}_{\rho_1,q_\star}} [f_u(x)f_u(y)] \leqslant \overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f)) - \frac{\eta}{2}.$$

Then using Corollary 13, there exists $\varepsilon(\eta/2), \delta(\eta/2) > 0$ such that for at least η fraction of f_u, we have that $\mathbf{Inf}_i(T_{1-\delta}f_u) \geqslant \varepsilon$ for some $i \in [L]$. Since $f_u(x) := \mathbf{E}_{v\sim N(u)}\left[f_v(x\circ\pi^{-1}_{uv})\right]$ and $\mathbf{Inf}_i$ is a convex function, we have

$$\mathop{\mathbf{E}}_{v\sim N(u)}\left[\mathbf{Inf}_i(T_{1-\delta}(f_v(x\circ\pi^{-1}_{uv})))\right] \geqslant \varepsilon \implies \mathop{\mathbf{E}}_{v\sim N(u)}\left[\mathbf{Inf}_{\pi_{uv}(i)}(T_{1-\delta}f_v)\right] \geqslant \varepsilon.$$

Thus, if $\mathbf{Inf}_i(T_{1-\delta}f_u) \geqslant \varepsilon$, then by an averaging argument, for at least $\varepsilon/2$ fraction of $v \in N(u)$ we have that $\mathbf{Inf}_{\pi_{uv}(i)}(T_{1-\delta}f_v) \geqslant \varepsilon/2$. Let

$$L_v = \{j \in [L] \mid \mathbf{Inf}_j(T_{1-\delta}f_v) \geqslant \varepsilon/2\}.$$

We know that $|L_v| \leqslant \frac{1}{\varepsilon \cdot e \cdot \ln(1/(1-\delta))}$ using Lemma 7. Consider the following randomized labeling to the Unique Games instance. For each $u \in U$, if there exists $i \in [L]$ such that $\mathbf{Inf}_i(T_{1-\delta} f_u) \geqslant \varepsilon$ then assign label i to u. Otherwise, assign a random label from $[L]$ to u. For each $v \in V$, pick a random label from L_v if it is non-empty. If $|L_v| = 0$ then pick a random label from $[L]$. The randomized labeling satisfies at least $\eta \cdot \frac{\varepsilon}{2} \cdot \frac{1}{|L_v|} \geqslant \eta \cdot \frac{\varepsilon}{2} \cdot \frac{\varepsilon \cdot e \cdot \ln(1/(1-\delta))}{1} = \eta'$ fraction of the edges in expectation, which is a contradiction. ◁

Let $U' \subseteq U$ be the set of $u \in U$ for which the above claim holds. Using the above claim, we have

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &= 2 \cdot \nu_{q_\star}(f) - 2 \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{(x,y) \sim \mathcal{A}^{\otimes L}_{\rho_1, q_\star}} [f_u(x) f_u(y)] \\
&\leqslant 2 \cdot \nu_{q_\star}(f) - 2 \left((1-\eta) \mathop{\mathbf{E}}_{u \in U'} \left[\overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f)) - \frac{\eta}{2} \right] + \eta \cdot 0 \right) \\
&\leqslant 2 \cdot \nu_{q_\star}(f) - 2 \mathop{\mathbf{E}}_{u \in U'} [\overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f))] + \eta.
\end{aligned}$$

Now using the convexity of the function $\overline{\Gamma}_\rho(x, y)$, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{u \in U'} \left[\overline{\Gamma}_{\rho_1}(\nu^u_{q_\star}(f), \nu^u_{q_\star}(f)) \right] &\geqslant \overline{\Gamma}_{\rho_1} \left(\mathop{\mathbf{E}}_{u \in U'} (\nu^u_{q_\star}(f)), \mathop{\mathbf{E}}_{u \in U'} (\nu^u_{q_\star}(f)) \right) \\
&\geqslant \overline{\Gamma}_{\rho_1} \left(\nu_{q_\star}(f) - \eta, \nu_{q_\star}(f) - \eta \right),
\end{aligned}$$

where the last inequality follows from $\left| \mathbf{E}_{u \in U}[\nu^u_{q_\star}(f)] - \mathbf{E}_{u \in U'}[\nu^u_{q_\star}(f)] \right| \leqslant \eta$ and the fact that $\overline{\Gamma}_\rho(x, y)$ is an increasing function of x and y. Thus, we have

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &\leqslant 2 \cdot \nu_{q_\star}(f) - 2 \cdot \overline{\Gamma}_{\rho_1} \left(\nu_{q_\star}(f) - \eta, \nu_{q_\star}(f) - \eta \right) + \eta \\
&\leqslant 2 \cdot \nu_{q_\star}(f) - 2 \cdot \overline{\Gamma}_{\rho_1} \left(\nu_{q_\star}(f), \nu_{q_\star}(f) \right) + 3\eta. \qquad (2)
\end{aligned}$$

The exact same calculation shows that

$$\mathbf{Cut}_{\mathcal{G}_2}(\mathcal{S}) \leqslant 2 \cdot \nu_{(1-q_\star)}(f) - 2 \cdot \overline{\Gamma}_{\rho_1} \left(\nu_{(1-q_\star)}(f), \nu_{(1-q_\star)}(f) \right) + 3\eta. \qquad (3)$$

We now analyze the value of the cut given by f in $\mathcal{G}_3$:

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_3}(\mathcal{S}) &= \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{(x,y) \sim \mathcal{B}^{\otimes L}_{\rho_2, q_\star}} [f_u(x) + f_u(y) - 2 f_u(x) f_u(y)] \\
&= \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2 \mathop{\mathbf{E}}_{u \in U} \mathop{\mathbf{E}}_{(x,y) \sim \mathcal{B}^{\otimes L}_{\rho_2, q_\star}} [f_u(x) f_u(y)].
\end{aligned}$$

Similar to Claim 24, we have,

▷ Claim 25. For at least $(1-\eta)$ fraction of $u \in U$,

$$\mathop{\mathbf{E}}_{(x,y) \sim \mathcal{B}^{\otimes L}_{\rho_1, q_\star}} [f_u(x) f_u(y)] \geqslant \overline{\Gamma}_{\rho_2} \left(\nu^u_{q_\star}(f), \nu^u_{(1-q_\star)}(f) \right) - \frac{\eta}{2}.$$

Proof. The proof is similar to the proof of Claim 24 once we conclude, using Corollary 13 that there exists $\varepsilon, \delta > 0$ such that for at least η fraction of f_u we have that $\mathbf{Inf}_i(T_{1-\delta} f_u) \geqslant \varepsilon$ for some $i \in [L]$. ◁

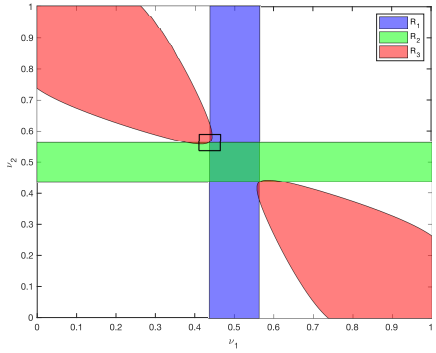


Figure 1 Plots of $R_1(0)$, $R_2(0)$ and $R_3(0)$.

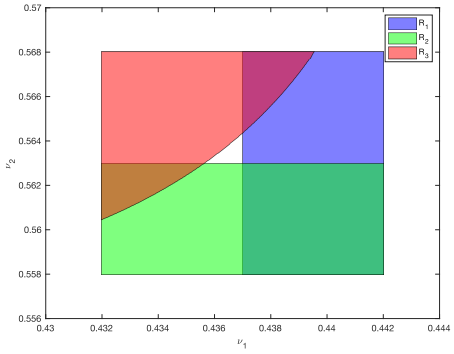


Figure 2 Zooming in to the black box in Figure 1 shows $R_1(0) \cap R_2(0) \cap R_3(0) = \emptyset$.

Let $U'' \subseteq U$ be the set of $u \in U$ for which the above claim holds. Using the above claim, we have

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_3}(\mathcal{S}) &= \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2 \mathop{\mathbf{E}}_{u\in U} \mathop{\mathbf{E}}_{(x,y)\sim \mathcal{B}^{\otimes L}_{\rho_2,q_\star}} [f_u(x) f_u(y)] \\
&\leqslant \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2\left((1-\eta) \mathop{\mathbf{E}}_{u\in U''} \left[\overline{\Gamma}_{\rho_2}\left(\nu^u_{q_\star}(f), \nu^u_{(1-q_\star)}(f)\right) - \frac{\eta}{2}\right] + \eta \cdot 0\right) \\
&\leqslant \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2 \mathop{\mathbf{E}}_{u\in U} \left[\overline{\Gamma}_{\rho_2}\left(\nu^u_{q_\star}(f), \nu^u_{(1-q_\star)}(f)\right)\right] + \eta.
\end{aligned}$$

Again, using the convexity of $\overline{\Gamma}_{\rho_2}$,

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_3}(\mathcal{S}) &\leqslant \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2\overline{\Gamma}_{\rho_2}\left(\nu_{q_\star}(f) - \eta, \nu_{(1-q_\star)}(f) - \eta\right) + \eta \\
&\leqslant \nu_{q_\star}(f) + \nu_{(1-q_\star)}(f) - 2\overline{\Gamma}_{\rho_2}\left(\nu_{q_\star}(f), \nu_{(1-q_\star)}(f)\right) + 3\eta. \qquad (4)
\end{aligned}$$

Now, let us compare the solution w.r.t $2(1-q_\star) - \eta$. For the notational convenience, let $\nu_1 = \nu_{q_\star}(f)$ and $\nu_2 = \nu_{(1-q_\star)}(f)$. Then,

$$\begin{aligned}
\mathbf{Cut}_{\mathcal{G}_1}(\mathcal{S}) &\leqslant 2 \cdot \nu_1 - 2\overline{\Gamma}_{\rho_1}(\nu_1, \nu_1) + 3\eta \\
\mathbf{Cut}_{\mathcal{G}_2}(\mathcal{S}) &\leqslant 2 \cdot \nu_2 - 2\overline{\Gamma}_{\rho_1}(\nu_2, \nu_2) + 3\eta \\
\mathbf{Cut}_{\mathcal{G}_3}(\mathcal{S}) &\leqslant \nu_1 + \nu_2 - 2\overline{\Gamma}_{\rho_2}(\nu_1, \nu_2) + 3\eta.
\end{aligned}$$

In this case, ν_1, ν_2 are the free parameters which come from the indicator function f of the cut we started with. Define the following ranges:

$$\begin{aligned}
R_1(\eta) &= \mathop{\mathtt{range}}_{\nu_1,\nu_2\in[0,1]} \left\{ \frac{2\nu_1 - 2\overline{\Gamma}_{\rho_1}(\nu_1,\nu_1) + 3\eta}{2(1-q_\star) - \eta} \geqslant (\alpha_{GW} - 10^{-5}) \right\}, \\
R_2(\eta) &= \mathop{\mathtt{range}}_{\nu_1,\nu_2\in[0,1]} \left\{ \frac{2\nu_2 - 2\overline{\Gamma}_{\rho_1}(\nu_2,\nu_2) + 3\eta}{2(1-q_\star) - \eta} \geqslant (\alpha_{GW} - 10^{-5}) \right\}, \\
R_3(\eta) &= \mathop{\mathtt{range}}_{\nu_1,\nu_2\in[0,1]} \left\{ \frac{\nu_1 + \nu_2 - 2\overline{\Gamma}_{\rho_2}(\nu_1,\nu_2) + 3\eta}{2(1-q_\star) - \eta} \geqslant (\alpha_{GW} - 10^{-5}) \right\}.
\end{aligned}$$

If we want to get a cut with values $(\alpha_{GW} - 10^{-5}) \cdot (2(1-q_\star) - \eta)$ in all the graphs $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_3$ then we must have the $R_1(\eta) \cap R_2(\eta) \cap R_3(\eta) \neq \emptyset$.

By performing numerical calculations, we show that there exists an absolute constant $\eta_0 > 0$ such that for all $0 < \eta \leqslant \eta_0$, $R_1(\eta) \cap R_2(\eta) \cap R_3(\eta)$ is in fact $\emptyset$. This is depicted in Figure 1 and Figure 2.[2] Thus, no matter which densities $\nu_1 = \nu_{q_\star}(f)$ and $\nu_2 = \nu_{(1-q_\star)}(f)$ we choose, there exists an $i \in [3]$ such that the value of the cut in graph $\mathcal{G}_i$ given by f will be less than $(\alpha_{GW} - \varepsilon_0)(2(1-q_\star) - \eta)$ for some fixed constant $\varepsilon_0 \geqslant 10^{-5}$. ◀

References

1 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Max-bisection analysis - prover code. https://github.com/austrin/max-bisection-analysis/. Accessed: 08-May-2020.

2 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better balance by being biased: A 0.8776-approximation for max bisection. *ACM Trans. Algorithms*, 13(1):1–27, 2016. doi:10.1145/2907052.

3 Per Austrin, Subhash Khot, and Muli Safra. Inapproximability of vertex cover and independent set in bounded degree graphs. *Theory of Computing*, 7(3):27–43, 2011. doi:10.4086/toc.2011.v007a003.

4 Per Austrin and Aleksa Stankovic. Global cardinality constraints make approximating some max-2-csps harder. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019*, pages 24:1–24:17, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.24.

5 Amey Bhangale and Subhash Khot. Simultaneous max-cut dictatorship gadget - prover code. https://github.com/asteric7/simultaneous_maxcut_gadget. Accessed: 08-May-2020.

6 Amey Bhangale, Subhash Khot, Swastik Kopparty, Sushant Sachdeva, and Devanathan Thiruvenkatachari. Near-optimal approximation algorithm for simultaneous max-cut. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 1407–1425, 2018. doi:10.1137/1.9781611975031.93.

7 Amey Bhangale, Swastik Kopparty, and Sushant Sachdeva. Simultaneous approximation of constraint satisfaction problems. In *Automata, Languages, and Programming - 42nd International Colloquium, (ICALP), Proceedings, Part I*, pages 193–205, 2015. doi:10.1007/978-3-662-47672-7_16.

8 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM Journal on Computing*, 39(3):843–873, 2009. doi:10.1137/07068062X.

9 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995. doi:10.1145/227683.227684.

10 Venkatesan Guruswami, Johan HÅstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering csp is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011. doi:10.1137/090756144.

11 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th Annual ACM symposium on Theory of computing (STOC)*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

12 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. doi:10.1137/S0097539705447372.

13 Elchanan Mossel. Gaussian bounds for noise correlation of functions. *Geometric and Functional Analysis*, 19(6):1713–1756, 2010. doi:10.1007/s00039-010-0047-x.

[2] We give a rigorous proof of this fact using a computer generated proof that uses interval arithmetic. The code can be found at [5]. A similar method was used in [15, 16, 2] for getting computer generated proofs of certain inequalities.

14 Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 21–30. IEEE, 2005. doi:10.1109/SFCS.2005.53.

15 Henrik Sjögren. Rigorous analysis of approximation algorithms for max 2-csp. *Master's thesis*, 2009.

16 Uri Zwick. Computer assisted proof of optimal approximability results. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 496–505, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545448.

Hitting Sets Give Two-Sided Derandomization of Small Space

Kuan Cheng
Department of Computer Science, University of Texas at Austin, TX, USA
https://sites.google.com/site/ckkcdh/home
ckkcdh@hotmail.com

William M. Hoza
Department of Computer Science, University of Texas at Austin, TX, USA
https://williamhoza.com
whoza@utexas.edu

Abstract

A *hitting set* is a "one-sided" variant of a pseudorandom generator (PRG), naturally suited to derandomizing algorithms that have one-sided error. We study the problem of using a given hitting set to derandomize algorithms that have *two-sided* error, focusing on space-bounded algorithms. For our first result, we show that if there is a log-space hitting set for polynomial-width read-once branching programs (ROBPs), then not only does $\mathbf{L} = \mathbf{RL}$, but $\mathbf{L} = \mathbf{BPL}$ as well. This answers a question raised by Hoza and Zuckerman [16].

Next, we consider constant-width ROBPs. We show that if there are log-space hitting sets for constant-width ROBPs, then given black-box access to a constant-width ROBP f, it is possible to deterministically estimate $\mathbb{E}[f]$ to within $\pm\varepsilon$ in space $O(\log(n/\varepsilon))$. Unconditionally, we give a deterministic algorithm for this problem with space complexity $O(\log^2 n + \log(1/\varepsilon))$, slightly improving over previous work.

Finally, we investigate the limits of this line of work. Perhaps the strongest reduction along these lines one could hope for would say that for every explicit hitting set, there is an explicit PRG with similar parameters. In the setting of constant-width ROBPs over a large alphabet, we prove that establishing such a strong reduction is at least as difficult as constructing a good PRG outright. Quantitatively, we prove that if the strong reduction holds, then for every constant $\alpha > 0$, there is an explicit PRG for constant-width ROBPs with seed length $O(\log^{1+\alpha} n)$. Along the way, unconditionally, we construct an improved hitting set for ROBPs over a large alphabet.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Complexity classes

Keywords and phrases hitting sets, derandomization, read-once branching programs

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.10

Funding *Kuan Cheng*: Supported by a Simons Investigator Award (#409864, David Zuckerman). *William M. Hoza*: Supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin.

Acknowledgements We thank David Zuckerman, Dean Doron, and Pooya Hatami for helpful discussions and for comments on drafts of this paper. We thank Adam Klivans for bringing his work with Gopalan and Meka [13] to our attention.

1 Introduction

Suppose some decision problem can be solved by an efficient randomized algorithm. That's good, but an efficient deterministic algorithm would be even better. We would therefore like to deterministically analyze the acceptance probability of the randomized algorithm on a given input. An ambitious approach to derandomization is to try to design a suitable *pseudorandom generator* (PRG).

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 10; pp. 10:1–10:25

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


▶ **Definition 1.1.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An* ε-PRG *for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, $\left|\mathbb{E}[f] - \mathbb{E}_{X\in\{0,1\}^s}[f(G(X))]\right| \leq \varepsilon$.*

Let n be the number of random bits used by the randomized algorithm, and ensure that $\mathcal{F}$ can compute the action of the randomized algorithm on its random bits. By iterating over all "seeds" $x \in \{0,1\}^s$ and plugging $G(x)$ into the randomized algorithm, we can get an estimate of its acceptance probability with additive error ε.

Unfortunately, designing efficient PRGs has proved to be extremely difficult. Constructing a *hitting set* is sometimes less difficult.

▶ **Definition 1.2.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An* ε-hitting set *for $\mathcal{F}$ is a set $H \subseteq \{0,1\}^n$ such that for every $f \in \mathcal{F}$ with $\mathbb{E}[f] \geq \varepsilon$, there is some $x \in H$ such that $f(x) = 1$.*

The image of any PRG is clearly a hitting set. By iterating over all strings in a hitting set, we can at least distinguish acceptance probability 0 from acceptance probability $\geq \varepsilon$. This is already sufficient for derandomizing some algorithms (namely, those with "one-sided error"). In this paper, we investigate the possibility of using a hitting set in a nontrivial way to obtain an estimate of the acceptance probability with a small additive error, just like what a PRG would have provided.

This possibility was previously studied in the context of derandomizing time-bounded algorithms. Several proofs have been discovered showing that if there is a polynomial-time hitting set for size-n circuits, then $\mathbf{P} = \mathbf{BPP}$ [3, 9, 4, 11]. In Appendix A we provide yet another proof of this theorem; our short proof is arguably simpler than all previous proofs. However, the focus of our paper is derandomizing space-bounded algorithms.

1.1 Derandomizing Log-Space Algorithms

The behavior of a small-space algorithm as a function of its random bits can be modeled by a *read-once*[1] *branching program* (ROBP). A width-w length-n ROBP is a directed graph consisting of $n+1$ layers with w vertices per layer. There is a designated "start vertex" v_{start} in the first layer. Every vertex not in the last layer has two outgoing edges labeled 0 and 1 leading to the next layer. An n-bit input string naturally identifies a path through the graph by reading from left to right. The program accepts or rejects this string depending on whether the path ends at the designated "accept vertex" v_{acc} in the last layer.

Recall that **BPL** and **RL** are the classes of languages that can be decided by randomized log-space algorithms that always halt with two-sided and one-sided error respectively. A log-space hitting set[2] for polynomial-width ROBPs would immediately imply $\mathbf{L} = \mathbf{RL}$. For our first result, we show that such a hitting set would also imply $\mathbf{L} = \mathbf{BPL}$.

▶ **Theorem 1.3.** *Assume that for every $n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set for width-n, length-n ROBPs that can be computed in space $O(\log n)$. Then $\mathbf{L} = \mathbf{BPL}$.*

---

[1] Because space-bounded algorithms only have read-once access to their random bits, it does not seem possible to adapt the existing derandomizations of **BPP** using a hitting set to the **BPL** case.

[2] When we say "a log-space hitting set," we mean a family of hitting sets $H_n \subseteq \{0,1\}^n$ such that given input $n \in \mathbb{N}$, the set H_n can be enumerated in space $O(\log n)$. For such a family, $|H_n| \leq \text{poly}(n)$.

1.2 Motivation: Recent Work on Hitting Sets

Theorem 1.3 is especially interesting in light of recent constructions of improved hitting sets for ROBPs [8, 16, 10]. The best known PRG for polynomial-width ROBPs is still Nisan's PRG [22], which has seed length

$$O(\log^2 n + \log n \log(1/\varepsilon)).$$

Until recently, Nisan's PRG also provided the best hitting set for polynomial-width ROBPs. Using sophisticated and novel techniques, Braverman, Cohen, and Garg obtained a hitting set with space complexity

$$\widetilde{O}(\log^2 n + \log(1/\varepsilon)),$$

which is an improvement when ε is very small [8].

Actually, Braverman, Cohen, and Garg constructed something better than a hitting set, called a *pseudorandom pseudodistribution* (PRPD).

▶ **Definition 1.4** ([8]). *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An* ε-PRPD *for $\mathcal{F}$ is a function $D\colon \{0,1\}^n \to \mathbb{R}$ such that for every $f \in \mathcal{F}$,*

$$\left| \sum_{x \in \{0,1\}^n} f(x)D(x) - \mathbb{E}[f] \right| \leq \varepsilon.$$

A PRPD can be used to estimate $\mathbb{E}[f]$ to within $\pm\varepsilon$, provided there is an efficient algorithm that enumerates all $x \in \operatorname{supp}(D)$ and computes $D(x)$. The concept of a PRPD generalizes the concept of a PRG, because given a PRG G with seed length s, one can set $D(x) = |G^{-1}(x)| \cdot 2^{-s}$. In turn, if D is a PRPD, then $\operatorname{supp}(D)$ is a hitting set. So a PRPD is *intermediate* between a hitting set and a genuine PRG. (Independently of our work, Chattopadhyay and Liao recently gave an improved PRPD construction with space complexity $\widetilde{O}(\log^2 n) + O(\log(1/\varepsilon))$ [10].)

After Braverman, Cohen, and Garg's work [8], Hoza and Zuckerman gave a simpler construction of an ε-hitting set for polynomial-width ROBPs, with the slightly improved space complexity $O(\log^2 n + \log(1/\varepsilon))$ [16]. Their construction is weaker in that it does not provide a PRPD. Theorem 1.3 bridges the gap between the two concepts somewhat: by Theorem 1.3, any generic hitting set can be used for two-sided derandomization, which was the main strength of a PRPD over a hitting set in the first place.

1.3 The Constant-Width Setting

However, there is a weakness of Theorem 1.3. A PRG or a PRPD would provide a *black-box* derandomization, whereas the algorithm of Theorem 1.3 is not black-box. This weakness is especially acute when we consider the constant-width case. Given a constant-width ROBP f directly as input, it is trivial to compute $\mathbb{E}[f]$ with high accuracy, so the algorithm of Theorem 1.3 is meaningless. Nevertheless, constant-width ROBPs can compute many interesting functions, and it is a major open challenge to design improved PRGs, PRPDs, or hitting sets for constant-width ROBPs. (For width 2, optimal PRGs are known [7]. For width 3, the current best PRG has seed length $\widetilde{O}(\log n \log(1/\varepsilon))$ [21]. The best hitting sets for width 3 are superior, with space complexity $\widetilde{O}(\log(n/\varepsilon))$ for small ε [14] or $O(\log n)$ for $\varepsilon \approx 1$ [25]. For width 4, the state of the art is simply the best results for polynomial-width ROBPs.)

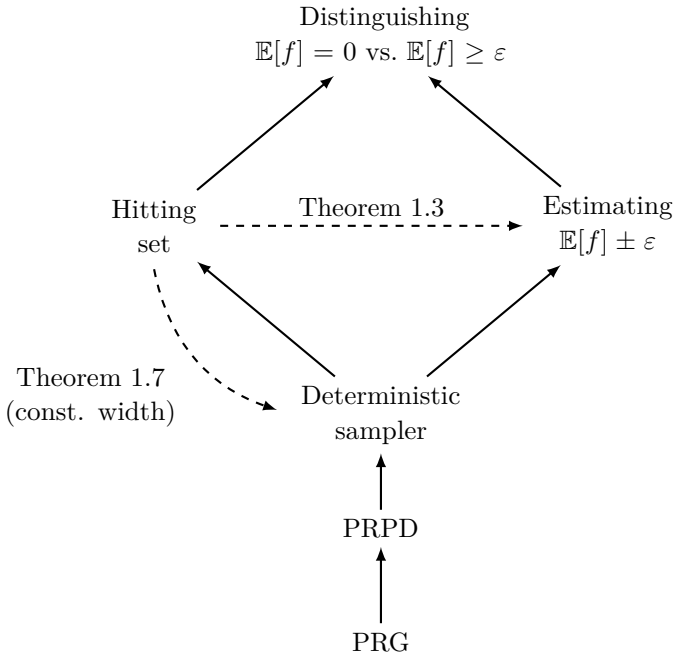


Figure 1 The relationships between different derandomization goals. The solid arrows are implications that are immediate from the definitions and hold for essentially any class $\mathcal{F}$ (possibly with some loss in ε). The dashed arrows are theorems in this paper, holding for ROBPs specifically.

To address this weakness of Theorem 1.3, we abstract the "black-box" feature of PRGs and PRPDs in the following definition.

▶ **Definition 1.5.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. A* deterministic ε-sampler *for $\mathcal{F}$ is a deterministic oracle algorithm A that outputs a real number such that for every $f \in \mathcal{F}$,*

$$|A^f - \mathbb{E}[f]| \leq \varepsilon.$$

The concept of a deterministic sampler generalizes that of a PRPD, because given a PRPD D, one can set $A^f = \sum_x f(x)D(x)$. In the other direction, deterministic samplers imply hitting sets.

▶ **Proposition 1.6.** *Identify 0 with the constant 0 function on $\{0,1\}^n$, and assume $0 \in \mathcal{F}$. Let A be a deterministic ε-sampler for $\mathcal{F}$, and let $H \subseteq \{0,1\}^n$ be the set of points where A^0 queries its oracle. Then for every $\varepsilon' > 2\varepsilon$, H is an ε'-hitting set for $\mathcal{F}$.*

Proof. Let $f \in \mathcal{F}$ satisfy $\mathbb{E}[f] > 2\varepsilon$. Since $|A^0 - 0| \leq \varepsilon$ and $|A^f - \mathbb{E}[f]| \leq \varepsilon$, $A^0 \neq A^f$. Therefore, A^f must query f at some point $x \in f^{-1}(1)$. The *first* such query must be at a point $x \in H$. ◀

All known derandomizations of **BPP** using a hitting set [3, 9, 4, 11], including our new derandomization in Appendix A, are black-box. That is, one can generically "upgrade" a polynomial-time hitting set for size-n circuits into a polynomial-time deterministic sampler for size-n circuits. For our second result, we prove the analogous reduction for constant-width ROBPs. (See Figure 1.)

▶ **Theorem 1.7.** *Assume that for every constant w, for all $n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set for width-w length-n ROBPs that can be computed in space $O(\log n)$. Then for every constant w, for all $n \in \mathbb{N}$ and all $\varepsilon > 0$, there is a deterministic ε-sampler for width-w length-n ROBPs that runs in space $O(\log(n/\varepsilon))$.*

The proof of Theorem 1.7 uses different techniques than that of Theorem 1.3. The space complexity of our deterministic sampler is proportional to the width parameter w (see Theorem 3.1), so the sampler becomes meaningless when w is large. Thus, Theorems 1.3 and 1.7 are incomparable.

We also obtain a new *unconditional* deterministic sampler. When ε is moderate, the best deterministic sampler for constant-width ROBPs is simply from Nisan's PRG [22], which gives a sampler with space complexity $O(\log^2 n + \log n \log(1/\varepsilon))$. When ε is small, using prior work, the best deterministic sampler for constant-width ROBPs was from Braverman, Cohen, and Garg's PRPD [8] (space complexity $\widetilde{O}(\log^2 n + \log(1/\varepsilon))$). The concurrent work by Chattopadhyay and Liao [10] gives a slightly better PRPD, and hence a slightly better deterministic sampler (space complexity $\widetilde{O}(\log^2 n) + O(\log(1/\varepsilon))$). By applying the reduction underlying Theorem 1.7 to the hitting set of Hoza and Zuckerman [16], we achieve a slightly better bound.

▶ **Theorem 1.8** (Unconditional sampler)**.** *For every constant w, for all $n \in \mathbb{N}$ and all $\varepsilon > 0$, there is a deterministic ε-sampler for width-w length-n ROBPs running in space $O(\log^2 n + \log(1/\varepsilon))$.*

In light of Theorem 1.8, when it comes to deterministic samplers, there is now a slight gap between the state of the art for polynomial-width ROBPs vs. the state of the art for width-w ROBPs with w a large constant. In other words, Theorem 1.8 is a case where we can take advantage of narrowness. There is no such gap when it comes to PRGs, PRPDs, or hitting sets.

1.4 Negative Result

Theorem 1.7 raises the question of whether we can go even further and upgrade any hitting set into a genuine PRG. In the time-bounded setting, this is indeed possible via the "hardness vs. randomness" paradigm. (If for every n there is a hitting set for size-n circuits computable in $\mathrm{poly}(n)$ time, then there is a language in $\mathbf{E}$ that requires circuits of size $2^{\Omega(n)}$. A major achievement in complexity theory was to show that assuming such a language exists, for every n, there is a polynomial-time logarithmic-seed PRG for size-n circuits [18].) Also, in the context of low-degree polynomials, Bogdanov showed how to convert any hitting set with a certain density property into a PRG [6]. Can a similar reduction be proven for small-space models?

We focus on the setting of constant-width ROBPs over a large alphabet. (An *ROBP over the alphabet* Σ computes a function $f\colon \Sigma^n \to \{0,1\}$; each vertex not in the last layer has $|\Sigma|$ outgoing edges labeled with the symbols in Σ.) We prove that *if* for every explicit hitting set in this setting, there is an explicit PRG with similar parameters, then there is in fact an explicit PRG for constant-width binary ROBPs with seed length $O(\log^{1+\alpha} n)$, where $\alpha > 0$ is an arbitrarily small constant. See Theorem 4.3 for the precise statement.

Our result is similar to a theorem by Hoza and Umans [15]. Like us, Hoza and Umans showed that if PRGs are equivalent to a seemingly weaker notion, then the equivalence itself can be used to construct a good PRG. Hoza and Umans focused on the distinction between PRGs and non-black-box derandomization, whereas we focus on the distinction between PRGs and hitting sets.

Interpretation

Like any conditional theorem, Theorem 4.3 has both a positive and a negative interpretation.[3] According to the negative interpretation, Theorem 4.3 shows that it would be difficult to establish a general reduction from PRGs to hitting sets. After all, it's as difficult as constructing a good PRG for constant-width ROBPs, which is a challenge that researchers have been struggling with for decades. In this sense, Theorem 4.3 provides an "excuse" for the fact that Theorems 1.3 and 1.7 do not provide genuine PRGs.

We feel that the negative interpretation is more realistic, but there is also a sensible positive interpretation. According to the positive interpretation, our work provides a new approach to constructing improved PRGs or hitting sets for constant-width ROBPs. One "merely" needs to bridge the gap between deterministic samplers and PRGs. This could be done in one of two ways. One could improve Theorem 1.7 so that it concludes with a PRG instead of a deterministic sampler. Alternatively, one could improve the construction of Theorem 4.3 so that rather than relying on the equivalence of hitting sets and PRGs, it merely relies on the equivalence of hitting sets and deterministic samplers. (In exchange, presumably the conclusion would merely be a deterministic sampler rather than a true PRG, but that would still be a breakthrough.)

1.5 Overview of Techniques

Let us first fix some notation. Let U_n denote the uniform distribution over $\{0,1\}^n$. For two strings x, y, let $x \circ y$ denote the concatenation of x with y. Suppose an ROBP f is clear from context. If u and v are vertices, let $p_{u\to v}$ be the probability that a random walk starting at u reaches v. We use the shorthand $p_{\to v} = p_{v_{\text{start}}\to v}$ and $p_{u\to} = p_{u\to v_{\text{acc}}}$. We use V_i to denote the set of vertices in the i-th layer of the ROBP, where $i \in \{0, 1, \ldots, n\}$.

1.5.1 Techniques for Theorem 1.3

We begin by outlining the proof of Theorem 1.3 (on derandomizing **BPL**). To derandomize **BPL**, it suffices to show that given a width-n length-n ROBP f, one can estimate $\mathbb{E}[f]$ to within a small additive error in log space. We do this using a hitting set H for width-(n^c) length-(n^c) ROBPs, where c is a large enough constant.

Each $x \in H$ is a string of length n^c. We think of it as a list of many shorter strings. Specifically, for every vertex v in f, the string x provides $\text{poly}(n)$ "sample inputs" associated with v. We compute the fraction $\widehat{p}_{\to v}$ of those sample inputs that lead to v. The hope is that

$$\forall v,\ \widehat{p}_{\to v} \approx p_{\to v}. \tag{1}$$

Of course we cannot directly verify Equation (1), since we do not know the values $p_{\to v}$. Instead, our algorithm looks for an $x \in H$ such that the estimates $\widehat{p}_{\to v}$ are *locally consistent*, i.e., for every $i \in [n]$ and every $v \in V_i$,

$$\widehat{p}_{\to v} \approx \sum_{u \in V_{i-1}} \widehat{p}_{\to u} \cdot p_{u\to v}.$$

Having found such an $x \in H$, we output the corresponding value $\widehat{p}_{\to v_{\text{acc}}}$.

[3] Throughout this discussion, we will ignore the issue of alphabet size, to simplify matters. The proof of Theorem 1.7 does generalize well to the large-alphabet case.

To establish the correctness of our algorithm, we must show two assertions. First, if the estimates $\widehat{p}_{\to v}$ pass the local consistency test, then $\mathbb{E}[f] \approx \widehat{p}_{\to v_{\text{acc}}}$. Second, there is always a string $x \in H$ that passes the local consistency test.

To show the first assertion, we bound $\sum_{v \in V_i} |\widehat{p}_{\to v} - p_{\to v}|$ by induction on i. Because of the structure of the ROBP, the error accumulates mildly, only blowing up by a factor that is approximately the size of f.

For the second assertion, we use the hitting property of H. At first glance, it might seem that the assertion is immediate. After all, a random x certainly passes the local consistency test with high probability, and the local consistency test can be computed in small space. Unfortunately, however, that computation involves reading the bits of x multiple times, whereas H is merely guaranteed to hit *read-once* branching programs.

To deal with this issue, we notice that if x were chosen at random, then with high probability, it would satisfy Equation (1). Furthermore, there exists a width-(n^c) length-(n^c) ROBP f' that determines whether its input x satisfies Equation (1). The values $p_{\to v}$ are all hard-coded into f'. There is no need to algorithmically construct f'; the mere fact that it exists implies the existence of an $x \in H$ that satisfies Equation (1). Satisfying Equation (1) readily implies that x also passes the local consistency test, completing the proof of the second assertion.

1.5.2 Techniques for Theorem 1.7

The proof of Theorem 1.7 (on deterministic samplers) uses different techniques. Let f be a constant-width ROBP. To estimate $\mathbb{E}[f]$, we attempt to work our way backward through the branching program, computing the acceptance probability $p_{v\to}$ from each vertex v. This plan is complicated by the fact that we only have black-box access to f. At a high level, for each layer, we use the assumed hitting set H to approximately compute the transitions at that layer, which allows us to continue computing the values $p_{v\to}$.

In more detail, the hitting set assists us in two different ways. First, we identify each prefix of a string in H with the vertex that is reached when f reads the prefix. In this way we are able to "find" all the vertices of f – or at least, all non-negligible vertices.

However, we are now effectively dealing with a width-$|H|$ branching program, because we have a copy of v for each string in H that leads to v. This interferes with our plan, because $|H| = \text{poly}(n)$ and hence we cannot afford to store the acceptance probabilities of all vertices in a single layer. The second way we use H is to determine which of these vertices are redundant. If there is some string in H that leads to accept from one vertex and reject from another, then the two vertices are not equivalent. Otherwise, the two vertices can be safely merged, because they must be two copies of the same vertex in f – or at least, they must correspond to two very similar vertices in f. The merging condition can be checked by making queries to f. By merging vertices, we effectively bring the width back down to a constant. (This merging operation is similar to a randomized learning algorithm by Gopalan, Klivans, and Meka [13]. Note that their algorithm is not space-efficient.)

Unfortunately, the fact that two vertices are equivalent does not imply that their out-neighbors are equivalent, so it is not immediately clear how to "merge" the outgoing edges. We show that it suffices to retain the outgoing edges from whichever vertex has the higher acceptance probability.

1.5.3 Techniques for Theorem 4.3

Recall that to prove Theorem 4.3, we must (conditionally) construct a PRG with seed length $O(\log^{1+\alpha} n)$, where $\alpha > 0$ is an arbitrarily small constant. For simplicity, in this overview, we will focus on the case $\alpha = 1/2$, i.e., seed length $O(\log^{3/2} n)$. Recall also that we are focusing on the constant-width case.

The starting point of the construction is the INW PRG, which ε-fools constant-width ROBPs over the alphabet $\{0,1\}^t$ with seed length $O(t+\log(n/\varepsilon)\log n)$ [17]. (Nisan's PRG [22] does not achieve the same optimal dependence on t.) Next, we present a reduction, showing how to convert a PRG with moderate error into a hitting set with very small threshold (Theorem 4.4). Hoza and Zuckerman gave a similar reduction [16], but their reduction only applies to binary ROBPs (the case $t = 1$). Our reduction is based on a more sophisticated variant of a key lemma in Hoza and Zuckerman's work [16].

Applying our new reduction to the INW generator, we unconditionally obtain an improved hitting set. The best previous hitting sets had space complexity $O(t + \log^2 n + \log(1/\varepsilon)\log n)$ [17] or $O(t \log n + \log^2 n + \log(1/\varepsilon))$ [16]. Our new hitting set (Corollary 4.8) achieves the "best of both worlds," with space complexity $O(t + \log^2 n + \log(1/\varepsilon))$.

The next step in the proof of Theorem 4.3 is to apply the assumption of Theorem 4.3, converting our hitting set into a PRG. The final step is to use traditional "seed recycling" techniques to trade the excellent dependence on ε for an improved dependence on n. Briefly, starting with a length-n ROBP over the alphabet $\{0,1\}^t$, we first use a randomized sampler [12] to reduce the alphabet size to $\mathrm{poly}(n)$. Then we divide our length-n ROBP of interest into blocks of length $m = 2^{\sqrt{\log n}}$. We can fool each chunk to within error $1/\mathrm{poly}(n)$ using a seed of length $O(\log^2 m + \log n) = O(\log n)$. Using the randomized sampler again, this allows us to effectively pay $O(\log n)$ truly random bits and reduce the length of the branching program by a factor of m. After repeating this process $\sqrt{\log n}$ times, the length is reduced to a constant, and we have paid a total of $O(\log^{3/2} n)$ truly random bits.

(To achieve seed length $O(\log^{1+\alpha} n)$, we start the whole process over again and iterate roughly $1/\alpha$ times. This iterative strategy is similar to the work of Hoza and Umans [15], but the specific reductions are different.)

1.6 Related Work

We have already referenced most of the work related to this paper, such as work on derandomizing **BPP** using a hitting set [3, 9, 4, 11]. However, a couple additional papers deserve mention.

1.6.1 BPL $\subseteq$ ZP*L

Our derandomization of **BPL** given a hitting set is similar to Nisan's unconditional proof that **BPL** $\subseteq$ **ZP*****L** [23]. To estimate the acceptance probability of a width-n length-n ROBP f, Nisan, like us, interprets a string $x \in \{0,1\}^{\mathrm{poly}(n)}$ as a list of sample inputs, which he uses to compute estimates of $p_{\to v}$ for each vertex v. Nisan's algorithm picks x at random, and then in a similar fashion as our algorithm, performs certain "local tests" at each vertex to verify that the sample inputs are trustworthy. Nisan's local tests can be computed in small space given two-way access to x, and passing the local tests implies that the estimates are close to the corresponding true probabilities. Our local consistency test also satisfies these properties, and indeed, one can obtain an alternative proof that **BPL** $\subseteq$ **ZP*****L** from our analysis. However, a technical point is that we use *fresh samples* for each vertex, whereas Nisan uses one set of

n-bit sample inputs for all the vertices. This crucial distinction is how we are able to ensure the existence of a polynomial-width ROBP that verifies Equation (1). Unfortunately, using fresh samples breaks Nisan's local tests, hence our new local consistency test.

1.6.2 Deterministically Simulating BPL with Very Low Error

The current best hitting sets for polynomial-width ROBPs [8, 16, 10] are superior to the best known PRGs [22] when ε is very small. One might hope that by plugging in the recent hitting sets, our reductions could provide a new unconditional deterministic algorithm for estimating the acceptance probability of a **BPL** algorithm to within $\pm\varepsilon$, with an improved space complexity when ε is very small. Unfortunately, this idea doesn't get off the ground, because to estimate the acceptance probability to within $\pm\varepsilon$, we rely on a $\frac{1}{2}$-hitting set for ROBPs of length $\text{poly}(n/\varepsilon)$ rather than an ε-hitting set for ROBPs of length n. The good news is that Ahmadinejad et al. recently tackled this same problem with different techniques. They designed an algorithm that runs in space $O(\log^{3/2} n + \log n \log\log(1/\varepsilon))$ [1].

1.7 Outline of This Paper

In Section 2, we present our derandomization of **BPL** given a hitting set for polynomial-width ROBPs. In Section 3, we present our deterministic sampler for constant-width ROBPs given a hitting set. Finally, in Section 4, we present our theorem on the limitations of this line of work.

2 Derandomizing BPL Given a Hitting Set

In this section, we show that the acceptance probability of an arbitrary polynomial width ROBP can be approximated within a small bias in small space, given a certain hitting set. Theorem 1.3 will follow from this.

▶ **Theorem 2.1.** *Assume there is a $\frac{1}{2}$-hitting set H for width-w' length-n' ROBPs that can be computed in space s. Then the acceptance probability of a given width-w length-n ROBP f can be approximated within a bias $\pm\varepsilon$, in space $O(s + \log \frac{wn}{\varepsilon})$.*

Here $w' = \left\lceil 9\frac{w^3 n^2 \log(wn)}{\varepsilon^2} \right\rceil, n' = \left\lceil 5\frac{w^3 n^4 \log(wn)}{\varepsilon^2} \right\rceil$.

Strictly speaking, Theorem 2.1 ought to be phrased in terms of *families* of ROBPs, to make the space bounds meaningful. That is, we assume there is an algorithm that constructs a $\frac{1}{2}$-hitting set for width-w length-n ROBPs, given w and n as inputs, running in space $s(w, n)$. Then given inputs f, ε, Theorem 2.1 should be understood to say that we can estimate $\mathbb{E}[f]$ to within $\pm\varepsilon$ in space $O(s(w', n') + \log(wn/\varepsilon))$.

We are most interested in the case that ε is a small constant, but we remark that when ε is very small, the parameters of Theorem 2.1 could be improved by applying the recent amplification technique by Ahmadinejad et al. [1].

We first give the derandomization and then give the analysis.

2.1 Derandomization Based on a Local Consistency Test

For $x \in \{0,1\}^{n'}$, we interpret it as a concatenation of wn segments. For each $i \in [n]$ and each $v \in V_i$, there is a segment corresponding to v consisting of a concatenation of t sample strings of length i, where t is a power of two satisfying $t \geq 4(\frac{wn}{\varepsilon})^2 \log(wn)$. Let $\widehat{p}_{\to v}(x)$ be

the fraction of strings that lead to v from the start vertex, among these t sample strings for v. When x is clear, we simply denote it as $\widehat{p}_{\to v}$. Also, for $v \in V_0$, we let $\widehat{p}_{\to v} = 1$ if $v = v_{\text{start}}$ and $\widehat{p}_{\to v} = 0$ otherwise.

The derandomization conducts a local consistency test $\mathsf{Test} : \{0,1\}^{n'} \to \{0,1\}$ for every $x \in H$ as follows. For all $i \in [n]$, for all $v \in V_i$, check if

$$\left| \widehat{p}_{\to v} - \left(\sum_{u \in V_{i-1}} \widehat{p}_{\to u} \cdot p_{u \to v} \right) \right| \leq \left(1 + \sum_{u \in V_{i-1}} p_{u \to v} \right) \varepsilon', \tag{2}$$

where $\varepsilon' = \frac{\varepsilon}{2wn}$. If x passes the checks for all v, then $\mathsf{Test}(x) = 1$, otherwise it is 0.

Finally we find an $x \in H$ that passes Test, and output $\widehat{p}_{\to v_{\text{acc}}}(x)$ as the approximation of $\mathbb{E}[f]$.

2.2 Analysis

We now define the "sample verification" function f' of f. For each $x \in \{0,1\}^{n'}$, we set $f'(x) = 1$ if and only if for every vertex v in f,

$$|\widehat{p}_{\to v} - p_{\to v}| \leq \varepsilon'. \tag{3}$$

We stress that our derandomization algorithm does not require computing f'; we define f' only for the sake of analysis.

▶ **Lemma 2.2.** *f' can be computed by a width-w' length-n' ROBP.*

Proof. For each vertex v of f, we construct an ROBP f'_v which simulates f on each sample string and counts how many lead to v. It stores a state of f and a counter value, for a total width of $w \cdot (t+1)$ and a total length $i \cdot t$. f'_v accepts if and only if the counter value is in $[p_{\to v}t - \frac{\varepsilon}{2wn}t, p_{\to v}t + \frac{\varepsilon}{2wn}t]$.

To construct f', we take the conjunction of f'_v, over all v in f. Note that this is a conjunction of ROBPs over disjoint variables. So we can easily see that f' can be computed by an ROBP with width at most $w(t+1)+1 \leq \left\lceil 9\frac{w^3n^2\log(wn)}{\varepsilon^2} \right\rceil$, length at most $tw\sum_{i=1}^{n} i \leq \left\lceil 5\frac{w^3n^4\log(wn)}{\varepsilon^2} \right\rceil$. ◀

▶ **Lemma 2.3.** *The acceptance probability of f' is at least $\frac{1}{2}$.*

Proof. By the construction of f', for each v of f, there are t uniform random samples. For each sample string, the probability that it leads to v from v_{start} in f is $p_{\to v}$. Hence the expected number of samples leading to v from v_{start} is $p_{\to v}t$. So by Hoeffding's inequality, $\Pr[|\widehat{p}_{\to v}t - p_{\to v}t| \geq \frac{\varepsilon}{2wn}t] \leq 2 \cdot 2^{-2\log(wn)} \leq \frac{2}{(wn)^2}$. There are wn vertices that need to be tested in f. (For $v \in V_0$, the estimate $\widehat{p}_{\to v}$ is always exactly correct.) Thus by a union bound,

$$\Pr\left[\forall v, |\widehat{p}_{\to v} - p_{\to v}| \leq \frac{\varepsilon}{2wn}\right] \geq 1 - \frac{2}{wn}.$$

This is at least $\frac{1}{2}$ when considering n to be at least some large enough constant. So by the definition of f', its acceptance probability is at least $\frac{1}{2}$. ◀

▶ **Lemma 2.4.** *For every $x \in \{0,1\}^{n'}$, if $f'(x) = 1$ then $\mathsf{Test}(x) = 1$.*

Proof. For every $i \in [n]$, every $v \in V_i$,

$$p_{\to v} = \sum_{u \in V_{i-1}} p_{\to u} p_{u \to v}, \tag{4}$$

by the structure of ROBP. So

$$\begin{aligned}
&\left| \widehat{p}_{\to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| \\
&= \left| \widehat{p}_{\to v} - p_{\to v} + p_{\to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| \\
&= \left| \widehat{p}_{\to v} - p_{\to v} + \sum_{u \in V_{i-1}} p_{\to u} p_{u \to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| && \text{(Equation (4))} \\
&\leq |\widehat{p}_{\to v} - p_{\to v}| + \sum_{u \in V_{i-1}} |p_{\to u} - \widehat{p}_{\to u}| \, p_{u \to v} && \text{(Triangle Inequality)} \\
&\leq \left(1 + \sum_{u \in V_{i-1}} p_{u \to v} \right) \varepsilon'. && \text{(Equation (3))}
\end{aligned}$$

◀

▶ **Lemma 2.5.** *For every $x \in \{0,1\}^{n'}$, if $\mathsf{Test}(x) = 1$ then $|\widehat{p}_{\to v_{\mathrm{acc}}} - p_{\to v_{\mathrm{acc}}}| \leq \varepsilon$.*

Proof. We use induction to show that for the i-th layer of f,

$$\sum_{v \in V_i} |\widehat{p}_{\to v} - p_{\to v}| \leq 2wi\varepsilon'.$$

For the base case, when $i = 0$, it's trivially true since we set $\widehat{p}_{\to v} = p_{\to v}$ for each $v \in V_0$. For the induction case, assume the hypothesis is true for layer i. Consider layer $i + 1$.

$$\begin{aligned}
&\sum_{v \in V_{i+1}} |\widehat{p}_{\to v} - p_{\to v}| \\
&= \sum_{v \in V_{i+1}} \left| \widehat{p}_{\to v} - \sum_{u \in V_i} p_{\to u} p_{u \to v} \right| && \text{(Equation (4))} \\
&= \sum_{v \in V_{i+1}} \left| \widehat{p}_{\to v} - \sum_{u \in V_i} \widehat{p}_{\to u} p_{u \to v} + \sum_{u \in V_i} \widehat{p}_{\to u} p_{u \to v} - \sum_{u \in V_i} p_{\to u} p_{u \to v} \right| \\
&\leq \sum_{v \in V_{i+1}} \left(\left| \widehat{p}_{\to v} - \sum_{u \in V_i} \widehat{p}_{\to u} p_{u \to v} \right| + \left| \sum_{u \in V_i} \widehat{p}_{\to u} p_{u \to v} - \sum_{u \in V_i} p_{\to u} p_{u \to v} \right| \right) \\
&\leq \sum_{v \in V_{i+1}} \left| \widehat{p}_{\to v} - \sum_{u \in V_i} \widehat{p}_{\to u} p_{u \to v} \right| + \sum_{v \in V_{i+1}} \sum_{u \in V_i} p_{u \to v} \, |\widehat{p}_{\to u} - p_{\to u}| \\
&\leq \sum_{v \in V_{i+1}} \left(1 + \sum_{u \in V_i} p_{u \to v} \right) \varepsilon' + \sum_{v \in V_{i+1}} \sum_{u \in V_i} p_{u \to v} \, |\widehat{p}_{\to u} - p_{\to u}| && (\mathsf{Test}(x) = 1) \\
&= 2w\varepsilon' + \sum_{u \in V_i} |\widehat{p}_{\to u} - p_{\to u}| && (5) \\
&\leq 2w\varepsilon' + 2wi\varepsilon' && \text{(Induction)} \\
&= 2w \cdot (i+1) \cdot \varepsilon'.
\end{aligned}$$

Here Equation (5) is due to structures of ROBPs. Note that

$$\sum_{v\in V_{i+1}}\sum_{u\in V_i} p_{u\to v} = \sum_{u\in V_i}\sum_{v\in V_{i+1}} p_{u\to v} = w,$$

since for every pair (u,v), $p_{u\to v}$ appears and only appears once in the summation. Also due to the same reasoning,

$$\sum_{v\in V_{i+1}}\sum_{u\in V_i} p_{u\to v}|\widehat{p}_{\to u} - p_{\to u}| = \sum_{u\in V_i}\sum_{v\in V_{i+1}} p_{u\to v}|\widehat{p}_{\to u} - p_{\to u}| = \sum_{u\in V_i}|\widehat{p}_{\to u} - p_{\to u}|.$$

As a result, for the last layer,

$$|\widehat{p}_{\to v_{\mathrm{acc}}} - p_{\to v_{\mathrm{acc}}}| \le \sum_{v\in V_n}|\widehat{p}_{\to v} - p_{\to v}| \le 2wn\varepsilon' = \varepsilon. \qquad \blacktriangleleft$$

▶ **Lemma 2.6.** *The derandomization is in space* $O(s + \log\frac{wn}{\varepsilon})$.

Proof. Since H is computable in space s, for every $x \in H$ we can output any specified bit of it in space $O(s + \log n')$. So when considering the space for computing $\mathsf{Test}(x)$ and $\widehat{p}_{\to v}(x)$, we can just regard x as an input string and only consider working space.

Given vertex v in f, we first consider the space for computing $\widehat{p}_{\to v}$. By the definition of $\widehat{p}_{\to v}$, we can locate the starting position of the t samples for v, taking space $O(\log\frac{wn}{\varepsilon})$. From there, we read the t samples one by one. For each sample, we run f from v_{start} to the layer of v to test if the sample leads to v. We use a counter c to record the number of samples leading to v. Then compute $\widehat{p}_{\to v}$ as c/t. Since t is a power of two, we can store this number exactly, with no rounding errors. So this step takes space $O(\log(wn)) + O(\log t) = O(\log\frac{wn}{\varepsilon})$. Thus the whole computation is in space $O(\log\frac{wn}{\varepsilon})$.

Next we consider Test. By the definition of Test, for every $i \in [n]$, for each vertex $v \in V_i$, we only need to compute $\widehat{p}_{\to v}$, $\sum_{u\in V_{i-1}}\widehat{p}_{\to u}\cdot p_{u\to v}$ and then test the inequality (2). This again takes space $O(\log\frac{wn}{\varepsilon})$. Note that computing $\mathsf{Test}(x)$ requires *two-way* access to x.

So the overall space of the derandomization is $O(s + \log\frac{wn}{\varepsilon})$. $\blacktriangleleft$

Proof of Theorem 2.1. Given a width-w length-n ROBP f, by Lemma 2.2, the function f' can be computed by a width-w' length-n' ROBP. By Lemma 2.3, the acceptance probability of f' is at least $1/2$. Since H is a $\frac{1}{2}$-hitting set for width-w' length-n' ROBPs, there exists $x \in H$ s.t. $f'(x) = 1$. So by Lemma 2.4, there is an $x \in H$ s.t. $\mathsf{Test}(x) = 1$. Hence we can exhaustively search though H to find an x which passes Test. Further, by Lemma 2.5, for this x, $|\widehat{p}_{\to v_{\mathrm{acc}}} - p_{\to v_{\mathrm{acc}}}| \le \varepsilon$. This shows the derandomization outputs the desired approximation for $p_{\to v_{\mathrm{acc}}}$.

By Lemma 2.6, the derandomization can be done in space $O(s + \log\frac{wn}{\varepsilon})$. $\blacktriangleleft$

Theorem 1.3 is directly implied from Theorem 2.1. The proof is straightforward by applying the well known transformation between logspace computations and ROBPs.

3 Deterministic Samplers for Constant-Width ROBPs

In this section, we will show how to use hitting sets to construct deterministic samplers for constant-width ROBPs, thereby proving Theorem 1.7. Most of the work will go toward establishing the following reduction, which is meaningful even for slightly super-constant width.

▶ **Theorem 3.1.** *Let $w, n \in \mathbb{N}$ and let $\varepsilon > 0$. Assume there is an $(\frac{\varepsilon}{2n})$-hitting set H for width-$(\binom{w}{2}+1)$ length-n ROBPs computable in space s. Then there is a deterministic ε-sampler for width-w length-n ROBPs that runs in space $O(s + w\log(n/\varepsilon))$.*

Like Theorem 2.1, Theorem 3.1 technically ought to be phrased in terms of families of ROBPs. We should also clarify the model of space-bounded oracle algorithm. We assume that the sampler has write-only access to a "query tape" where it can write down an n-bit query string (the query string does not count against the sampler's space complexity). The sampler can then enter a special "query" state, which returns the result of the query into the algorithm's state and clears the query tape. This simple model was perhaps first studied by Ladner and Lynch [19].

3.1 Setting Up the Reduction

Toward proving Theorem 3.1, we begin by setting up some notation. For any ROBP f and a string $x \in \{0,1\}^{\leq n}$, let $v_f(x)$ be the vertex reached when f reads x. Furthermore, define

$$p_f(x) = \mathbb{E}[f(x \circ U_{n-|x|})],$$

i.e., $p_f(x) = p_{v_f(x)\to}$.

Now, let $H \subseteq \{0,1\}^n$ be an ε_H-hitting set for width-$(\binom{w}{2}+1)$ ROBPs. For $i \leq n$, let H_i be the set of i-bit prefixes of strings in H, i.e., $H_i = \{x_1x_2\dots x_i : x \in H\}$. One can verify that H_i is an ε_H-hitting set for width-$(\binom{w}{2}+1)$ length-i ROBPs.

Let f be the width-w ROBP to which we have oracle access. Let λ denote the empty string. Our goal is to estimate $p_f(\lambda)$. For each $i \leq n$, define an equivalence relation $\sim$ on $\{0,1\}^i$ by the rule

$$x \sim y \iff \forall z \in H_{n-i}, f(x \circ z) = f(y \circ z).$$

▶ **Lemma 3.2.** *If $x \sim y$, then $|p_f(x) - p_f(y)| < \varepsilon_H$.*

Proof. Let $i = |x| = |y|$. Define $g\colon \{0,1\}^{n-i} \to \{0,1\}$ by

$$g(z) = f(x \circ z) \oplus f(y \circ z).$$

The function $g(z)$ can be computed by an ROBP of width $\binom{w}{2}+1$: we have one state in g for each unordered pair of states in f to run the computations $f(x \circ z), f(y \circ z)$ in parallel, along with one additional $\perp$ state in g to indicate that the two computations converged to the same state. If $|p_f(x) - p_f(y)| \geq \varepsilon_H$, then H_{n-i} hits g, hence $x \not\sim y$. ◀

Let $[x]$ denote the equivalence class of x, so $[x] \subseteq \{0,1\}^{|x|}$. Our deterministic sampler will be based on numbers $\widetilde{p}_f([x]) \in [0,1]$ for each equivalence class $[x]$. The definition of $\widetilde{p}_f$ will ensure that $\widetilde{p}_f([x]) \approx p_f(x)$ for typical values of x, although there might be some anomalous values of x where $\widetilde{p}_f([x]) \not\approx p_f(x)$.

The definition of $\widetilde{p}_f([x])$ is inductive. For the base case, when $x \in \{0,1\}^n$, define $\widetilde{p}_f([x]) = f(x)$. This is well-defined, because $x \sim y \implies f(x) = f(y)$. For the inductive step, suppose $x \in \{0,1\}^i$ with $i < n$. Define

$$\widetilde{p}_f([x]) = \max_{x' \in H_i \cap [x]} \left(\frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) \right), \tag{6}$$

with the convention that $\widetilde{p}_f([x]) = 0$ if $H_i \cap [x] = \emptyset$. Our sampler will output[4] $\widetilde{p}_f([\lambda])$. (In Section 3.3, we will explain in more detail how to efficiently compute $\widetilde{p}_f([\lambda])$.)

[4] Actually the sampler's output differs slightly from $\widetilde{p}_f([\lambda])$ due to rounding errors.

3.2 Correctness

The upper bound on $\widetilde{p}_f([x])$ is straightforward:

▷ Claim 3.3. For every i, for every $x \in \{0,1\}^{n-i}$,

$$\widetilde{p}_f([x]) \leq p_f(x) + i\varepsilon_H.$$

Proof. We proceed by induction on i. In the base case $i = 0$, $\widetilde{p}_f([x]) = f(x) = p_f(x)$. For the inductive step $i > 0$, we consider two cases. If $H_i \cap [x] = \emptyset$, then $\widetilde{p}_f([x]) = 0$ and the claim is trivial. Otherwise, there is some $x' \in H_i \cap [x]$ such that

$$\begin{aligned}
\widetilde{p}_f([x]) &= \frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) && \text{(Equation (6))} \\
&\leq \frac{1}{2}p_f(x' \circ 0) + \frac{1}{2}p_f(x' \circ 1) + (i-1)\varepsilon_H && \text{(Induction)} \\
&= p_f(x') + (i-1)\varepsilon_H \\
&< p_f(x) + i\varepsilon_H && \text{(Lemma 3.2.)}
\end{aligned}$$

◁

The lower bound is a little more subtle. If u is a vertex in layer i of f, we say that u is *H-reachable* if there is some $x \in H_i$ with $v_f(x) = u$. Otherwise, we say that u is *H-unreachable*. Let $\widetilde{f}$ be a width-$(w+1)$ ROBP obtained from f by replacing all H-unreachable nodes with reject nodes.[5]

▷ Claim 3.4. For every i, for every $x \in \{0,1\}^{n-i}$,

$$\widetilde{p}_f([x]) \geq p_{\widetilde{f}}(x).$$

Proof. We proceed by induction on i. In the base case $i = 0$, $\widetilde{p}_f([x]) = f(x) \geq \widetilde{f}(x) = p_{\widetilde{f}}(x)$. For the inductive step $i > 0$, we consider two cases. If f visits some H-unreachable node when it reads x, then $p_{\widetilde{f}}(x) = 0$ and the claim is trivial. Therefore, assume that when f reads x, every node visited is H-reachable. Then there is some $x' \in H_{n-i}$ such that $v_f(x) = v_f(x')$. Of course when f reads x', every node visited is H-reachable, so

$$v_{\widetilde{f}}(x') = v_f(x') = v_f(x) = v_{\widetilde{f}}(x).$$

Therefore,

$$\begin{aligned}
p_{\widetilde{f}}(x) &= p_{\widetilde{f}}(x') \\
&= \frac{1}{2}p_{\widetilde{f}}(x' \circ 0) + \frac{1}{2}p_{\widetilde{f}}(x' \circ 1) \\
&\leq \frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) && \text{(Induction)} \\
&\leq \widetilde{p}_f(x) && \text{(Equation (6)).}
\end{aligned}$$

(The last inequality uses the fact that $v_f(x) = v_f(x')$ and hence $x \sim x'$.) ◁

[5] More precisely, add an extra node to each layer labeled $\perp$. In layers prior to the final layer, both outgoing edges from $\perp$ lead to $\perp$, and both outgoing edges from H-unreachable nodes lead to $\perp$. In the final layer, H-unreachable nodes and $\perp$ are reject nodes.

▶ **Corollary 3.5.** $|\widetilde{p}_f([\lambda]) - \mathbb{E}[f]| \leq n \cdot \varepsilon_H$.

Proof. By Claim 3.3,

$$\widetilde{p}_f([\lambda]) \leq p_f(\lambda) + n \cdot \varepsilon_H = \mathbb{E}[f] + n \cdot \varepsilon_H.$$

In the other direction, by Claim 3.4,

$$\widetilde{p}_f([\lambda]) \geq p_{\widetilde{f}}(\lambda) = \mathbb{E}\left[\widetilde{f}\right].$$

Define $g\colon \{0,1\}^n \to \{0,1\}$ by

$$g(x) = 1 \iff \text{when } f \text{ reads } x, \text{ an } H\text{-unreachable node is visited.}$$

Then g can be computed by a width-$(w+1)$ ROBP by a construction very similar to that of $\widetilde{f}$. By construction, g rejects every string in H. Therefore, $\mathbb{E}[g] < \varepsilon_H$. Furthermore, $g(x) = 0 \implies f(x) = \widetilde{f}(x)$. Therefore, $\left|\mathbb{E}\left[\widetilde{f}\right] - \mathbb{E}[f]\right| < \varepsilon_H$, so $\widetilde{p}_f([\lambda]) > \mathbb{E}[f] - \varepsilon_H$. ◀

3.3 Efficiently Computing $\widetilde{p}_f([\lambda])$

To complete the proof of Theorem 3.1, we just need to show how to efficiently compute $\widetilde{p}_f([\lambda])$. This is fairly straightforward from the definitions; the details follow.

Proof of Theorem 3.1. Say a string $x \in H_i$ is a *representative* if it is the lexicographically first element of $[x] \cap H_i$. Let $x^{(i,1)}, x^{(i,2)}, \ldots$ be an enumeration of the representatives in H_i in lexicographic order. Given i, j, and oracle access to f, one can compute $x^{(i,j)}$ in space $O(s)$.

Our sampler works its way backward through the branching program, starting at layer n and ending with layer 0. The sampler stores data about layer i and uses it when processing layer $i-1$. Specifically, the data stored regarding layer i consists of a list of numbers $p_{i,1}, p_{i,2}, \ldots$, with the interpretation $p_{i,j} = \widetilde{p}_f([x^{(i,j)}])$, or rather $p_{i,j} \approx \widetilde{p}_f([x^{(i,j)}])$ due to rounding error.

For layer n, we can compute this value exactly by setting $p_{i,j} = f(x^{(i,j)})$. Given these values for layer $i+1$, we compute $p_{i,j}$ by the rule

$$p_{i,j} := \max_{\substack{x' \in H_i \cap [x^{(i,j)}] \\ x^{(i+1,j_0)} \sim x' \circ 0 \\ x^{(i+1,j_1)} \sim x' \circ 1}} \left(\frac{1}{2} p_{i+1,j_0} + \frac{1}{2} p_{i+1,j_1}\right), \tag{7}$$

with the convention $p_{i,j} = 0$ if there is no suitable triple (x', j_0, j_1).

The sampler performs the arithmetic in Equation (7) to within $\lceil \log(2n/\varepsilon) \rceil$ bits of precision. This ensures that the rounding error is not too large in each step; by induction, $|p_{i,j} - \widetilde{p}_f([x^{(i,j)}])| \leq \frac{\varepsilon(n-i)}{2n}$. The sampler outputs $p_{0,1}$, which is within ε of $\mathbb{E}[f]$ by Corollary 3.5, since $\varepsilon_H = \frac{\varepsilon}{2n}$.

The number of vertices in each layer of f is at most w, so the number of equivalence classes in $\{0,1\}^i$ is also at most w. Therefore, there are at most w representatives in H_i, and hence there are only w numbers $p_{i,j}$ being stored for each layer. Storing those numbers for the layer currently being processed and the layer most recently processed takes $O(w \log(n/\varepsilon))$ bits of space, so overall, the space complexity of the sampler is $O(s + w \log(n/\varepsilon))$ as claimed. ◀

Interestingly, the sampler of Theorem 3.1 can be implemented to be non-adaptive, because it only queries f at strings of the form $x \circ y$ or $x \circ b \circ z$, where $x \in H_i$, $y \in H_{n-i}$, $b \in \{0,1\}$, and $z \in H_{n-i-1}$.

3.4 Applying the Reduction

Proof of Theorem 1.8. Hoza and Zuckerman constructed an $\left(\frac{\varepsilon}{2n}\right)$-hitting set H even for *polynomial*-width ROBPs that can be computed in space $O(\log^2 n + \log(1/\varepsilon))$ [16]. Combining this result with Theorem 3.1 immediately proves Theorem 1.8. ◀

To prove Theorem 1.7, we must first amplify the assumed $\frac{1}{2}$-hitting set to get an $\left(\frac{\varepsilon}{2n}\right)$-hitting set. This is straightforward, although we must pay a small penalty in terms of width, length, and cardinality.

▶ **Lemma 3.6.** *Suppose H is a $\frac{1}{2}$-hitting set for width-$(w+1)$ length-(nm) ROBPs. Divide each string $x \in H$ into blocks of length n, $x = x^{(1)} \circ x^{(2)} \circ \cdots \circ x^{(m)}$. Let $H' = \{x^{(i)} : x \in H, i \in [m]\}$. Then H' is a $(\frac{1}{m})$-hitting set for width-w length-n ROBPs.*

Proof. Let f be a width-w length-n ROBP with $\mathbb{E}[f] \geq 1/m$. Define $g\colon (\{0,1\}^n)^m \to \{0,1\}$ by

$$g(x^{(1)} \circ \cdots \circ x^{(m)}) = \bigvee_{i \in [m]} f(x^{(i)}).$$

Then g can be computed by a width-$(w+1)$ ROBP. Furthermore,

$$\mathbb{E}[g] = 1 - (1 - \mathbb{E}[f])^m \geq 1 - \left(1 - \frac{1}{m}\right)^m > \frac{1}{2}.$$

Therefore, H hits g, hence H' hits f. ◀

Proof of Theorem 1.7. Combine Lemma 3.6 with Theorem 3.1. ◀

4 Negative Result: A Barrier for Upgrading Hitting Sets to PRGs

To directly compare hitting sets and PRGs, it is convenient to address the strings in the hitting set using a *hitting set generator* (HSG).

▶ **Definition 4.1.** *An ε-*HSG *for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that $G(\{0,1\}^s)$ is an ε-hitting set for $\mathcal{F}$.*

In our theorem statements so far, we have been somewhat informal with the distinction between an individual generator vs. a family of generators. Since our negative result is more "meta" than our other results, we will make a precise definition for clarity's sake.

▶ **Definition 4.2.** *Let $s(n,t,\varepsilon)$ be a space-constructible*[6] *function. An* explicit PRG (HSG) family *for width-w large-alphabet ROBPs with seed length s is a uniform algorithm G that takes as input the parameters n, t, ε and a string $y \in \{0,1\}^{s(n,t,\varepsilon)}$ and outputs a string $G_{n,t,\varepsilon}(y) \in \{0,1\}^{tn}$. The algorithm runs in space $O(s(n,t,\varepsilon))$, and for each fixed n, t, ε, we require that $G_{n,t,\varepsilon}$ is an ε-PRG (ε-HSG) for width-w length-n ROBPs over the alphabet $\{0,1\}^t$.*

The assumption of Theorem 4.3 says that hitting sets can be upgraded into PRGs with essentially no loss: the width parameter remains the same, and the seed length only increases by a constant factor, for any arbitrary setting of n, t, ε. This is only for simplicity's sake. The proof would still go through even if the parameters deteriorated a little when moving from hitting sets to PRGs.

[6] I.e., given n, t, ε, the value $s(n,t,\varepsilon)$ can be computed in space $O(s(n,t,\varepsilon))$.

▶ **Theorem 4.3.** *Let w be a constant. Assume that for every $s(n,t,\varepsilon)$, if there exists an explicit HSG family for width-w large-alphabet ROBPs with seed length s, then there exists an explicit PRG family for width-w large-alphabet ROBPs with seed length $O(s)$. Then for every constant $\alpha > 0$, there exists an explicit PRG family for width-w ROBPs with seed length*

$$O(t + \log(n/\varepsilon)\log^{\alpha} n).$$

4.1 From PRGs with Moderate Error to HSGs with Tiny Threshold

As outlined in Section 1.5.3, the proof of Theorem 4.3 is based on two reductions. For the first reduction, we show how to convert any PRG with inverse polynomial error into an ε-HSG for any ε. In the regime $n \geq w$, our reduction is a generalization of Hoza and Zuckerman's reduction [16] to the large-alphabet case $t \gg 1$.

▶ **Theorem 4.4.** *Let $w, n, t \in \mathbb{N}$ and let $\varepsilon > 0$. Assume there is a $(\frac{1}{2w^3n^2})$-PRG G for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, with seed length and space complexity bounded by s. Then there is an ε-hitting set H for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, computable in space $O(s + t + \log(wn/\varepsilon))$.*

(Just like Theorems 2.1 and 3.1, Theorem 4.4 technically ought to be phrased in terms of families of ROBPs.)

4.1.1 Construction of the Hitting Set H

Our hitting set H relies on a hitting set H_{rect} for *combinatorial rectangles* [20]. Recall that a combinatorial rectangle over alphabet Γ of dimension r is a function $g\colon \Gamma^r \to \{0,1\}$ of the form $g(x_1,\dots,x_r) = g_1(x_1) \wedge \cdots \wedge g_r(x_r)$. Without loss of generality, assume $\varepsilon < \frac{1}{w^2n^2}$ and $s \geq t$. The algorithm to enumerate H is as follows.

1. For all $r \in \left\{1, 2, \dots, \left\lfloor \frac{\log(1/\varepsilon)}{\log(wn)} \right\rfloor\right\}$:
   a. Let $H_{\text{rect}} \subseteq (\{0,1\}^s)^{2r-1}$ be an ε^4-hitting set for combinatorial rectangles over alphabet $\{0,1\}^s$ of dimension $2r-1$.
   b. For all sequences $(x_1, y_1, x_2, y_2, \dots, x_{r-1}, y_{r-1}, x_r) \in H_{\text{rect}}$ and for all sequences of nonnegative integers $(n_1,\dots,n_r)$ satisfying $n_1 + n_2 + \cdots + n_r = n - r$, output the (nt)-bit string

$$(G(x_1)|_{n_1 t}) \circ (y_1|_t) \circ (G(x_2)|_{n_2 t}) \circ (y_2|_t) \circ \cdots \circ (y_{r-1}|_t) \circ (G(x_r)|_{n_r t}). \tag{8}$$

In Equation (8), the notation $y|_t$ denotes the t-bit prefix of the bitstring y. The key difference between our construction and Hoza and Zuckerman's original hitting set construction [16] is the presence of the strings y_i, which do not pass through the PRG G.

4.1.2 Proof of Correctness

Hoza and Zuckerman's reduction was based on a simple structural lemma for ROBPs [16, Lemma 1]. Toward proving the correctness of H, we will now prove a new variant of that lemma, applicable to ROBPs over a large alphabet. For two vertices u, v in an ROBP f, write $u \rightsquigarrow v$ if there is an edge from u to v. Let $\rightsquigarrow^*$ be the reflexive transitive closure of $\rightsquigarrow$, i.e., $u \rightsquigarrow^* v$ if $u = v$ or there is a path from u to v.

The way to think about Lemma 4.5 is to suppose that one is choosing a route from u to v_{acc}. Lemma 4.5 suggests two vertices $v \rightsquigarrow u'$ that one could visit on the way. Item 2 says that it is not difficult to find v. Item 3 says that if one can make it to u', it will be quite a bit easier to find v_{acc} from there. Item 4 says that overall, visiting v and u' is only a mild detour.

In general, in any ROBP over the alphabet Σ, if $v \rightsquigarrow u'$, then $p_{v\to u'} \geq 1/|\Sigma|$. In Hoza and Zuckerman's lemma [16, Lemma 1], they assume $\Sigma = \{0,1\}$, and they use the fact that therefore $p_{v\to u'} \geq \Omega(1)$. At a high level, the reason we need a new structural lemma is that if Σ is large, $p_{v\to u'}$ might be small. Indeed, observe that Lemma 4.5 does not guarantee any lower bound on $p_{v\to u'}$.

▶ **Lemma 4.5.** *Let f be a width-w, length-n ROBP over any alphabet. Let u be a vertex in f, and assume $0 < p_{u\to} \leq \frac{1}{wn}$. Then there is a pair of vertices (v, u') in f such that:*

1. $u \rightsquigarrow^* v \rightsquigarrow u'$.
2. $p_{u\to v} \geq \frac{1}{w^3n^2}$.
3. $p_{u'\to} \geq wn \cdot p_{u\to}$.
4. $p_{u\to v} \cdot p_{v\to u'} \cdot p_{u'\to} \geq \frac{p_{u\to}}{w^2n}$.

Proof. Suppose some pair (v, u') satisfies Item 1, but it violates Item 4. For such a pair, if we take a random walk from u, the probability that we visit v, u', and v_{acc} is less than $\frac{p_{u\to}}{w^2n}$. The number of such pairs is at most w^2n, so by the union bound, when we start at u and read random bits, the probability that we visit *any* such pair and v_{acc} is less than $p_{u\to}$. Therefore, there is some path from u to v_{acc} that never visits such a pair.

Let u' be the first vertex along that path that satisfies Item 3. (Such a u' exists, because if nothing else we can let $u' = v_{\text{acc}}$.) Let v be the vertex immediately preceding u' in the path. (This makes sense, because $p_{u\to} < wn \cdot p_{u\to}$, so $u' \neq u$.) This pair clearly satisfies Items 1, 3 and 4; all that remains is to verify Item 2. Indeed,

$$\begin{aligned}\frac{p_{u\to}}{p_{u\to v}} &\leq w^2n \cdot p_{v\to u'} \cdot p_{u'\to} && \text{(Item 4)}\\ &\leq w^2n \cdot p_{v\to}\\ &< w^2n \cdot wn \cdot p_{u\to},\end{aligned}$$

where the last inequality holds because u' is the *first* vertex in the path satisfying Item 3, and v precedes u', so v must not satisfy Item 3. Rearranging completes the proof. ◀

▶ **Corollary 4.6.** *Let $0 < \varepsilon \leq \frac{1}{wn}$. Let f be a width-w, length-n ROBP over any alphabet with $\mathbb{E}[f] \geq \varepsilon$. Then there is a sequence of vertices*

$$v_{\text{start}} = u_1 \rightsquigarrow^* v_1 \rightsquigarrow u_2 \rightsquigarrow^* v_2 \rightsquigarrow \cdots \rightsquigarrow u_r \rightsquigarrow^* v_r = v_{\text{acc}}$$

such that:

1. *For every i, $p_{u_i\to v_i} \geq \frac{1}{w^3n^2}$.*
2. $r \leq \frac{\log(1/\varepsilon)}{\log(wn)}$.
3. $p_{u_1\to v_1} \cdot p_{v_1\to u_2} \cdot p_{u_2\to v_2} \cdots p_{v_{r-1}\to u_r} \cdot p_{u_r\to v_r} \geq \varepsilon^3$.

Proof. We define the sequence inductively, starting with $u_1 = v_{\text{start}}$. Assume we've defined $u_1, v_1, u_2, v_2, \ldots, u_i$. If $p_{u_i\to} \geq \frac{1}{w^3n^2}$, then set $r = i$, set $v_i = v_{\text{acc}}$, and terminate the sequence. Otherwise, let (v_i, u_{i+1}) be the vertices provided by plugging $u = u_i$ into Lemma 4.5.

Item 1 of Lemma 4.5 implies that $u_i \rightsquigarrow^* v_i$ and $v_i \rightsquigarrow u_{i+1}$. Item 1 is guaranteed by Item 2 of Lemma 4.5 and the termination condition. By Item 3 of Lemma 4.5, $p_{u_{i+1}\to} \geq wn \cdot p_{u_i\to}$, which implies Item 2. Finally, iteratively applying Item 4 of Lemma 4.5 shows that

$$p_{u_1\to v_1} \cdot p_{v_1\to u_2} \cdot p_{u_2\to v_2} \cdots p_{v_{r-1}\to u_r} \cdot p_{u_r\to v_r} \geq \frac{p_{u_1\to}}{(w^2n)^r} \geq \varepsilon^3,$$

i.e., Item 3 holds. ◀

We are now ready to complete the proof of correctness of our hitting set H.

▷ Claim 4.7. If f is a width-w length-n ROBP over the alphabet $\{0,1\}^t$ with $\mathbb{E}[f] \geq \varepsilon$, then $f^{-1}(1) \cap H \neq \emptyset$.

Proof. Let $u_1 \rightsquigarrow^* v_1 \rightsquigarrow \cdots \rightsquigarrow u_r \rightsquigarrow^* v_r$ be the sequence of vertices guaranteed by Corollary 4.6. Let n_i be the distance from u_i to v_i. Let $g\colon (\{0,1\}^s)^{2r-1} \to \{0,1\}$ be the following combinatorial rectangle:

$$g(x_1, y_1, x_2, y_2, \ldots, x_{r-1}, y_{r-1}, x_r) = 1 \iff$$
$$\forall i \in [r],\ G(x_i)|_{n_i t} \text{ leads from } u_i \text{ to } v_i \ \text{ and } \ \forall i \in [r-1],\ y_i|_t \text{ leads from } v_i \text{ to } u_{i+1}.$$

By Item 1 of Corollary 4.6, $p_{u_i \to v_i} \geq \frac{1}{w^3 n^2}$. Since G has error $\frac{1}{2w^3n^2}$,

$$\Pr[G(U) \text{ leads from } u_i \text{ to } v_i] \geq \frac{1}{2} p_{u_i \to v_i}.$$

Therefore, by Item 3 of Corollary 4.6, $\mathbb{E}[g] \geq \varepsilon^3 \cdot 2^{-r} \geq \varepsilon^4$. Therefore, there is some sequence $(x_1, y_1, \ldots, y_{r-1}, x_r) \in H_{\text{rect}}$ that hits g. By construction, the corresponding element of H is accepted by f. ◁

4.1.3 Efficiency

Proof of Theorem 4.4. To complete the proof of Theorem 4.4, let us analyze the space complexity of H. The number r can be stored using $O(\log\log(1/\varepsilon))$ bits of space. Using a construction by Linial, Luby, Saks, and Zuckerman [20], because of our chosen value of r, we can enumerate H_{rect} in space $O(s + \log(1/\varepsilon))$. The integers $n_1, \ldots, n_r$ can be straightforwardly stored using $O(r \log n) = O(\log(1/\varepsilon))$ bits of space. Thus, overall, the space complexity is $O(s + \log(1/\varepsilon))$. (Recall that we assumed without loss of generality that $\varepsilon < \frac{1}{w^2 n^2}$ and $s \geq t$.) ◀

4.2 Application: Unconditional Improved Hitting Sets for Large-Alphabet ROBPs

As outlined in Section 1.5.3, plugging the class INW generator [17] into the reduction of Theorem 4.4 already gives something interesting: an improved hitting set for large-alphabet ROBPs, even of polynomial width.

▶ **Corollary 4.8.** *Let $w, n, t \in \mathbb{N}$ and let $\varepsilon > 0$. There is an ε-hitting set H for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, computable in space $O(t + \log(wn)\log n + \log(1/\varepsilon))$.*

4.3 Trading a Good Dependence on ε for a Good Dependence on n

Recall that to prove Theorem 4.3, we must (conditionally) construct a PRG with a good dependence on n. So far, unconditionally, Theorem 4.4 has provided us with an HSG with a good dependence on ε. The assumption of Theorem 4.3 allows us to convert that HSG into a PRG with the same seed length, $O(t + \log^2 n + \log(1/\varepsilon))$ (for width w, a constant). In this section, we show how to convert that PRG into another PRG with seed length $O(t + \log^{3/2} n + \log(1/\varepsilon)\sqrt{\log n})$, i.e., we improve the dependence on n at the expense of a worse dependence on ε. That follows from setting $\alpha = 1/2$ in the following more general reduction.

▶ **Lemma 4.9.** *Let $\alpha \in (0,1)$ be a constant. Let $w, n, t \in \mathbb{N}$ and $\varepsilon > 0$. Define $m = \left\lceil 2^{(\log n)^{1-\alpha}} \right\rceil$ and $d = \lceil C \log(n/\varepsilon) \rceil$, where C is an appropriate constant. Assume there is an $(\frac{\varepsilon}{4n})$-PRG G for width-w length-m ROBPs over the alphabet $\{0,1\}^d$ with seed length and space complexity bounded by s. Then there is an ε-PRG G' for width-w length-n ROBPs over the alphabet $\{0,1\}^t$ with seed length and space complexity $O(t + s \cdot \log^{\alpha} n + \log(wn/\varepsilon))$.*

As usual, Lemma 4.9 should technically be phrased in terms of families of ROBPs. As suggested in Section 1.5.3, the proof of Lemma 4.9 is not particularly novel. It is an application of traditional seed-recycling techniques, similar to classic constructions of PRGs for space-bounded computation [22, 17, 24]. Our construction and analysis are especially similar to Armoni's work [5].

One difference is that we use randomized samplers rather than extractors for convenience; in this respect, our construction is similar to a variant of the INW generator [17] described by Braverman, Cohen, and Garg [8] as a warm-up to their main construction. In particular, we rely on the following randomized sampler by Goldreich and Wigderson [12].

▶ **Theorem 4.10** ([12, Lemma 6.6])**.** *For all $t \in \mathbb{N}$, $\delta > 0$, there exists a function $\mathsf{Samp}\colon \{0,1\}^t \times \{0,1\}^{O(\log(1/\delta))} \to \{0,1\}^t$ such that for any[7] function $f\colon \{0,1\}^t \to [0,1]$,*

$$\Pr_x \left[\left| \mathop{\mathbb{E}}_y [f(\mathsf{Samp}(x,y))] - \mathbb{E}[f] \right| \le \delta \right] \ge 1 - \delta.$$

Furthermore, given t, δ, x, y as inputs, $\mathsf{Samp}(x,y)$ can be computed in space $O(t)$.

We will recursively use the following basic PRG, which stretches $t + dn$ bits to tn bits. It might be helpful to think of the case $t = 100d$.

▶ **Lemma 4.11.** *Let t, δ be arbitrary, and let $\mathsf{Samp}\colon \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^t$ be the randomized sampler of Theorem 4.10. Define $G_0\colon \{0,1\}^t \times (\{0,1\}^d)^n \to (\{0,1\}^t)^n$ by*

$$G_0(x, z_1 \circ \cdots \circ z_n) = \mathsf{Samp}(x, z_1) \circ \cdots \circ \mathsf{Samp}(x, z_n).$$

Then G_0 fools width-w length-n ROBPs over the alphabet $\{0,1\}^t$ with error $\delta w^2 n$.

The proof of Lemma 4.11 is straightforward, and we omit it. When reading the proof of Lemma 4.9, it might be helpful to keep in mind that all "x" variables are strings of length t, all "y" variables are strings of length s, and all "z" variables are strings of length d.

Proof of Lemma 4.9. Define $n_i = n/m^i$. For simplicity, we ignore rounding issues, i.e., we assume that n_i is an integer and that $m = 2^{(\log n)^{1-\alpha}}$ exactly. Let $\delta = \frac{\varepsilon}{4w^2 n}$, and let $d = O(\log(wn/\varepsilon))$ be the length of the second input to the function Samp of Theorem 4.10. We will recursively define a sequence of PRGs

$$G_i\colon \{0,1\}^t \times (\{0,1\}^s)^i \times (\{0,1\}^d)^{n_i} \to (\{0,1\}^t)^n.$$

The base case $i = 0$ is the basic PRG of Lemma 4.11:

$$G_0(x, z_1 \circ \cdots \circ z_n) = \mathsf{Samp}(x, z_1) \circ \cdots \circ \mathsf{Samp}(x, z_n).$$

[7] Goldreich and Wigderson analyze the case that f is $\{0,1\}$-valued, but the $[0,1]$-valued case automatically follows with only a quadratic loss in δ.

For the inductive step $i > 0$, we define[8]

$$G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i}) \\ = G_{i-1}(x, y_1 \circ \cdots \circ y_{i-1}, G(\mathsf{Samp}(y_i, z_1)) \circ \cdots \circ G(\mathsf{Samp}(y_i, z_{n_i}))),$$

where G is the given PRG. To analyze these generators, let f be a width-w length-n ROBP over the alphabet $\{0,1\}^t$. For each i and each fixing of $x, y_1, \ldots, y_i$, define

$$g^{(x,y_1,\ldots,y_i)}(z_1 \circ \cdots \circ z_{n_i}) = f(G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i}))$$
$$h^{(x,y_1,\ldots,y_i)}(y'_1 \circ \cdots \circ y'_{n_{i+1}}) = f(G_i(x, y_1 \circ \cdots \circ y_i, G(y'_1) \circ \cdots \circ G(y'_{n_{i+1}})).$$

These functions are related to one another by the rules

$$h^{(x,y_1,\ldots,y_i)}(y'_1 \circ \cdots \circ y'_{n_{i+1}}) = g^{(x,y_1,\ldots,y_i)}(G(y'_1) \circ \cdots \circ G(y'_{n_{i+1}})) \tag{9}$$
$$g^{(x,y_1,\ldots,y_i)}(z_1 \circ \cdots \circ z_{n_i}) = h^{x,y_1,\ldots,y_{i-1}}(\mathsf{Samp}(y_i, z_1) \circ \cdots \circ \mathsf{Samp}(y_i, z_{n_i})). \tag{10}$$

This shows by induction on i that each g function can be computed by a width-w ROBP over the alphabet $\{0,1\}^d$ and each h function can be computed by a width-w ROBP over the alphabet $\{0,1\}^s$.

Let us now show by induction on i that G_i fools f with error $(\delta w^2 + \varepsilon_G) \cdot \sum_{j=0}^{i} n_j$, where ε_G is the error of G. The base case $i = 0$ is already established by Lemma 4.11. For the inductive step, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_i \\ z_1,\ldots,z_{n_i}}} [f(G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i}))] \\
&= \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_i \\ z_1,\ldots,z_{n_i}}} [g^{(x,y_1,\ldots,y_i)}(z_1 \circ \cdots \circ z_{n_i})] \\
&= \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_i \\ z_1,\ldots,z_{n_i}}} [h^{(x,y_1,\ldots,y_{i-1})}(\mathsf{Samp}(y_i, z_1) \circ \cdots \circ \mathsf{Samp}(y_i, z_{n_i}))] && \text{(Equation (10))} \\
&\le \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_{i-1} \\ y'_1,\ldots,y'_{n_i}}} [h^{(x,y_1,\ldots,y_{i-1})}(y'_1 \circ \cdots \circ y'_{n_i})] + \delta w^2 n_i && \text{(Lemma 4.11)} \\
&= \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_{i-1} \\ y'_1,\ldots,y'_{n_i}}} [g^{(x,y_1,\ldots,y_{i-1})}(G(y'_1) \circ \cdots \circ G(y'_{n_i}))] + \delta w^2 n_i && \text{(Equation (9))} \\
&\le \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_{i-1} \\ z_1,\ldots,z_{n_{i-1}}}} [g^{(x,y_1,\ldots,y_{i-1})}(z_1 \circ \cdots \circ z_{n_{i-1}})] + (\delta w^2 + \varepsilon_G) \cdot n_i \\
&= \mathop{\mathbb{E}}_{\substack{x \\ y_1,\ldots,y_{i-1} \\ z_1,\ldots,z_{n_{i-1}}}} [f(G_{i-1}(x, y_1 \circ \cdots \circ y_{i-1}, z_1 \circ \cdots \circ z_{n_{i-1}}))] + (\delta w^2 + \varepsilon_G) \cdot n_i.
\end{aligned}$$

The lower bound follows the same argument. Let $r = \log^{\alpha} n$ and $G' = G_r$. Then G' fools f with error

$$(\delta w^2 + \varepsilon_G) \cdot \sum_{i=0}^{r} n_i \le (\delta w^2 + \varepsilon_G) \cdot n \cdot \sum_{i=0}^{\infty} m^{-i} \le 2n \cdot (\delta w^2 + \varepsilon_G) \le \varepsilon.$$

Furthermore, the seed length of G' is $t + rs + d$ as claimed, and the space complexity of G' is clearly also $O(t + rs + d)$. ◀

[8] Note that strictly speaking, we are using *two* instantiations of Samp. In the base case, Samp has output length t, whereas in the inductive step, Samp has output length s. Hopefully, using the same name Samp for both samplers will not cause confusion.

4.4 Putting Things Together to Prove Theorem 4.3

Proof of Theorem 4.3. We will show by induction on a that for each constant $a \in \mathbb{N}$, there is an explicit PRG family for width-w ROBPs with seed length $O(t + \log(n/\varepsilon)\log^{1/a} n)$. The base case $a = 1$ holds unconditionally – this is the seed length of the classic INW generator [17].

For the inductive step, suppose $a > 1$. Let t, n, ε be arbitrary; we will construct an ε-PRG for width-w length-n ROBPs over the alphabet $\{0,1\}^t$. Define $\alpha = 1/a$. Let $m = 2^{(\log n)^{1-\alpha}}$ and $d = C\log(n/\varepsilon)$, as in Lemma 4.9.

By induction, there is a $(\frac{1}{2w^3m^2})$-PRG G for width-w length-m ROBPs over the alphabet $\{0,1\}^d$, with seed length and space complexity bounded by $O(d + \log^{1+\frac{1}{a-1}} m)$. Now,

$$\log^{1+\frac{1}{a-1}} m = (\log n)^{(1-\frac{1}{a})\cdot(1+\frac{1}{a-1})} = \log n,$$

so G has seed length and space complexity bounded by $O(\log(n/\varepsilon))$. Plugging G into Theorem 4.4, we get an $(\frac{\varepsilon}{4n})$-hitting set H for width-w length-m ROBPs over the alphabet $\{0,1\}^d$, computable in space $O(\log(n/\varepsilon))$. Now we use our assumption to convert H into a PRG G' with exactly the same parameters. Finally, plugging G' into Lemma 4.9 gives the desired PRG. ◀

5 Directions for Further Research

In this paper, we have shown that hitting sets for **RL** would derandomize **BPL**. Constructing a hitting set is the most natural way to prove $\mathbf{L} = \mathbf{RL}$, but there are also other approaches. In general, does $\mathbf{L} = \mathbf{RL}$ imply $\mathbf{L} = \mathbf{BPL}$? In the polynomial-time setting, the "promise" variant of this question has been answered in the affirmative, i.e., $\mathbf{prP} = \mathbf{prRP} \implies \mathbf{P} = \mathbf{BPP}$ [9]. Does $\mathbf{prL} = \mathbf{prRL}$ imply $\mathbf{L} = \mathbf{BPL}$? Or relaxing the challenge even further, does $\mathbf{L} = \mathbf{NL}$ imply $\mathbf{L} = \mathbf{BPL}$?

We gave two different algorithms for estimating the expectation of an ROBP given a hitting set, one suited for $w = \text{poly}(n)$ (Theorem 2.1) and one suited for $w = O(1)$ (Theorem 3.1). What about the case $n = \text{polylog}\, w$? Unconditionally, there are optimal hitting sets known in this regime [2, 16]. Given such an ROBP f as input, is it possible to compute $\mathbb{E}[f] \pm \frac{1}{w}$ in space $O(\log w)$? (The Nisan-Zuckerman PRG [24] achieves seed length $O(\log w)$ in this regime, but only for moderate error $\varepsilon \gg \frac{1}{w}$.) An affirmative answer would imply that any space-s decision algorithm that uses n random bits could be simulated by another space-$O(s)$ algorithm using only $O(n/s^c)$ random bits, where c is an arbitrarily large constant.

Recently, Meka, Reingold, and Tal constructed a PRG for width-3 ROBPs with seed length $\widetilde{O}(\log n \log(1/\varepsilon))$ [21]. This is near-optimal when ε is not too small, but for $\varepsilon = 1/n$ it is worse than Nisan's PRG [22]. On the other hand, there is an explicit hitting set for width-3 ROBPs with near-optimal seed length $\widetilde{O}(\log(n/\varepsilon))$ [14]. Can one construct an explicit deterministic sampler for width-3 ROBPs with near-optimal seed length? Unfortunately, to produce a deterministic sampler for width-3 ROBPs, Theorem 3.1 would require a hitting set for width-4 ROBPs.

Assuming the existence of a log-space hitting set for polynomial-width ROBPs, is it possible to construct a log-space deterministic sampler for polynomial-width ROBPs?

Recall that PRPDs are superior to deterministic samplers (see Figure 1). Is it possible to improve Theorem 1.7 so that it concludes with a PRPD rather than a mere deterministic sampler?

References

1 AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. *arXiv preprint*, 2019. arXiv:1912.04524.

2 Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the 19th Annual Symposium on Theory of Computing (STOC)*, pages 132–140. ACM, 1987.

3 Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. Hitting sets derandomize BPP. In *Automata, languages and programming (Paderborn, 1996)*, volume 1099 of *Lecture Notes in Computer Science*, pages 357–368. Springer, Berlin, 1996. doi:10.1007/3-540-61440-0_142.

4 Alexander E. Andreev, Andrea E. F. Clementi, José D. P. Rolim, and Luca Trevisan. Weak random sources, hitting sets, and BPP simulations. *SIAM Journal on Computing*, 28(6):2103–2116, 1999. doi:10.1137/S0097539797325636.

5 Roy Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd International Workshop on Randomization and Computation (RANDOM)*, volume 1518 of *Lecture Notes in Computer Science*, pages 47–59. Springer, Berlin, 1998. doi:10.1007/3-540-49543-6_5.

6 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2005.

7 Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9:283–292, 2013. doi:10.4086/toc.2013.v009a007.

8 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM Journal on Computing*, 0(0):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.

9 Harry Buhrman and Lance Fortnow. One-sided versus two-sided error in probabilistic computation. In Christoph Meinel and Sophie Tison, editors, *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 100–109, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. doi:10.1007/3-540-49116-3_9.

10 Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, 2020. To appear.

11 Oded Goldreich, Salil Vadhan, and Avi Wigderson. Simplified derandomization of BPP using a hitting set generator. In *Studies in complexity and cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 59–67. Springer, Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_8.

12 Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4<315::AID-RSA3>3.0.CO;2-1.

13 Parikshit Gopalan, Adam R. Klivans, and Raghu Meka. Learning functions of halfspaces using prefix covers. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, volume 23 of *Proceedings of Machine Learning Research*, pages 15.1–15.10, Edinburgh, Scotland, 25–27 Jun 2012. PMLR. URL: http://proceedings.mlr.press/v23/gopalan12.html.

14 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–129. IEEE, 2012.

15 William M. Hoza and Chris Umans. Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. In *Proceedings of the 49th Annual Symposium on Theory of Computing (STOC)*, pages 629–640. ACM, New York, 2017.

16 William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018.

17 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 356–364. ACM, 1994.

18 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229, New York, NY, USA, 1997. ACM.

19 Richard E. Ladner and Nancy A. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10(1):19–32, 1976. doi:10.1007/BF01683260.

20 Nathan Linial, Michael Luby, Michael Saks, and David Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997. doi:10.1007/BF01200907.

21 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 626–637. ACM, New York, 2019.

22 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

23 Noam Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993. doi:10.1016/0304-3975(93)90258-U.

24 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.

25 Jiří Šíma and Stanislav Žák. Almost k-wise independent sets establish hitting sets for width-3 1-branching programs. In *Computer science – theory and applications*, volume 6651 of *Lecture Notes in Computer Science*, pages 120–133. Springer, Heidelberg, 2011. doi:10.1007/978-3-642-20712-9_10.

A Derandomizing BPP Given a Hitting Set

▶ **Theorem A.1** ([3]). *Assume that for every $s, n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set $H_{s,n}$ for size-s circuits on n input bits that can be computed in time* $\mathrm{poly}(s, n)$. *Then* $\mathbf{P} = \mathbf{BPP}$.

Proof. By naïve amplification, we may assume that the randomized algorithm has failure probability 2^{-N}, where N is the *input* length. Let C be a size-n circuit on n input bits describing the action of this algorithm on its random bits, so $n = \mathrm{poly}(N)$ and we are trying to distinguish the cases $\mathbb{E}[C] \leq 2^{-N}$ vs. $\mathbb{E}[C] \geq 1 - 2^{-N}$. Our algorithm accepts if and only if there exists $x \in H_{n^c,n}$ such that for all $y \in H_{3n,n}$, $C(x \oplus y) = 1$. Here, c is a suitable constant that will become clear later. The runtime is clearly $\mathrm{poly}(N)$.

For the correctness proof, first suppose $\mathbb{E}[C] \leq 2^{-N}$. For any fixed x, the function $y \mapsto \neg C(x \oplus y)$ has expectation at least $1 - 2^{-N}$ and can be computed by a circuit of size $3n$. Therefore, there is some $y \in H_{3n,n}$ such that $C(x \oplus y) = 0$, and hence our algorithm rejects. Conversely, suppose $\mathbb{E}[C] \geq 1 - 2^{-N}$. Consider sampling $x \in \{0,1\}^n$ and $y \in H_{3n,n}$ uniformly at random. Since x is uniform, $\mathbb{E}_{x,y}[\neg C(x \oplus y)] \leq 2^{-N}$. By Markov's inequality,

$$\Pr_{x \in \{0,1\}^n}\left[\mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] < 2 \cdot 2^{-N}\right] > 1/2.$$

Since $H_{3n,n}$ can be computed in polynomial time, $|H_{3n,n}| \leq \mathrm{poly}(N)$. Therefore, when N is sufficiently large,

$$\mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] < 2 \cdot 2^{-N} \implies \mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] = 0.$$

Therefore,

$$\Pr_{x\in\{0,1\}^n}[\forall y \in H_{3n,n}, C(x \oplus y) = 1] > 1/2.$$

Given input x, the predicate $\forall y \in H_{3n,n}, C(x \oplus y) = 1$ can be computed by a circuit of size n^c for some suitable constant c. Therefore, there is some $x \in H_{n^c,n}$ that hits that circuit. ◀

Palette-Alternating Tree Codes

Gil Cohen
Department of Computer Science, Tel Aviv University, Israel
gil@tauex.tau.ac.il

Shahar Samocha
Department of Computer Science, Tel Aviv University, Israel
samocha@mail.tau.ac.il

Abstract

A tree code is an edge-coloring of the complete infinite binary tree such that every two nodes of equal depth have a fraction–bounded away from 0–of mismatched colors between the corresponding paths to their least common ancestor. Tree codes were introduced in a seminal work by Schulman [29] and serve as a key ingredient in almost all deterministic interactive coding schemes. The number of colors effects the coding scheme's rate.

It is shown that 4 is precisely the least number of colors for which tree codes exist. Thus, tree-code-based coding schemes cannot achieve rate larger than 1/2. To overcome this barrier, a relaxed notion called *palette-alternating tree codes* is introduced, in which the number of colors can depend on the layer. We prove the existence of such constructs in which most layers use 2 colors–the bare minimum. The distance-rate tradeoff we obtain matches the Gilbert-Varshamov bound.

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity. To analyze our protocol, we prove a structural result on the location of failed communication-rounds induced by the error pattern enforced by the adversary. Our coding scheme is efficient given an explicit palette-alternating tree code and serves as an alternative to the scheme obtained by [13].

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Tree Codes, Coding Theory, Interactive Coding Scheme

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.11

Funding The research leading to these results has received funding from the Israel Science Foundation (grant number 1569/18) and from the Azrieli Faculty Fellowship.

Acknowledgements We wish to thank Leonard J. Schulman for many insightful discussions over the years regarding tree codes and interactive coding schemes.

1 Introduction

Tree codes are a powerful combinatorial structure, defined and proven to exist in [29] in order to serve as a key ingredient for achieving a constant rate interactive coding scheme. Tree codes are the central object for encoding information in the interactive coding theory which developed from the initial papers. They remain a crucial building block in almost all interactive coding schemes [26, 10, 8, 13, 3, 5, 2, 4, 16, 17, 22, 1, 14, 7, 19, 32].

We turn to formally define tree codes. Let T be a rooted binary tree that is endowed with an edge coloring from some ambient color set (or alphabet) Σ. Let u, v be a pair of vertices in T with equal depth and a least common ancestor w. Let ℓ be the distance, in edges, from u to w. Let $p_u, p_v \in \Sigma^\ell$ be the sequences of colors on the path from w to u and to v, respectively. We define $h(u, v)$ to be the relative Hamming distance between p_u and p_v.

▶ **Definition 1** (Tree codes [29]). *Let T be the complete rooted infinite binary tree. The tree T, together with an edge-coloring of T by a color set Σ is called a* tree code with distance δ *if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. When there is no $\delta > 0$ for which T is a tree code with distance δ we say that T has* vanishing distance.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 11; pp. 11:1–11:29

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

Schulman [29] proved that for every distance parameter $\delta < 1$ tree codes with a constant number of colors $c = c(\delta)$ exist. Although tree codes are used in different ways by different interactive coding schemes, one aspect is common to all: When a party wishes to send a bit, a suitable color from Σ is sent instead. Thus, the rate of all tree-code-based coding schemes is bounded above by $1/\log_2|\Sigma|$. One is led to ask a natural combinatorial question–what is the least number of colors in a tree code with non-vanishing distance?

1.1 Tree codes: 4 colors suffice and are necessary

We first observe that 3 colors do not suffice and, as a result, the rate of every tree-code-based coding scheme cannot exceed 1/2, let alone approach capacity. Consider any 3-color tree code. First, we may assume that every two siblings are connected to their parent with edges having distinct colors as otherwise the distance of the tree code is 0. Let u, v be any two vertices. Out of u, v go four edges and so by the pigeonhole principal in every 3-coloring, two of these edges share the same color. By the above, one of these edges goes out of u and the other goes out of v. This implies that, starting from the two sons of the root, one can construct two paths of any desired length $n \geq 1$ with the same color pattern, establishing that the tree has vanishing distance.

Based on the ideas Schulman introduced to prove the existence of tree codes with a constant number of colors, we complement the above observation and establish that 4 colors suffice for a tree code with non-vanishing distance.

▶ **Theorem 2.** *There exists a* 4*-color tree code with distance* 0.136*.*

The proof of Theorem 2 appears in Section 3. As Schulman's original proof for the existence of tree codes, Theorem 2 is nonconstructive. Coming up with explicit constructions of non-vanishing distance tree codes with a constant number of colors is one of the most challenging problems in this field [30, 15, 6, 25, 23, 13, 11, 24]. The currently best known result [11] guarantees any designated distance $\delta < 1$ when using $(\log n)^{O_\delta(1)}$ colors at depth n. This work, however, concerns with the information-theoretic aspect of the channel capacity, and the computational aspects are left for future work.

While our proof of Theorem 2 closely follows Schulman's proof, and the observation that 4 colors are necessary is easy to prove, to the best of our knowledge, this basic combinatorial result was not known and, furthermore, we find it surprising that merely 4 colors suffice to guarantee such a strong combinatorial structure. Still, even if 4 is a surprisingly small number of colors, an interactive coding scheme that uses a 4-color tree code would have rate bounded above by 1/2.

1.2 Palette-alternating tree codes

To save on communication, one might hope to avoid the use of the tree code "every now and then". However, if one sends a bit in the clear without encoding it, and that bit is flipped by the adversary, the simulation seems doomed to fail without some way of generating an unpredictable verification (which can be done when considering randomized schemes). Perhaps a better idea would be to try and apply puncturing–a standard tool from classic error correcting codes used for improving the rate of a code. However, the distance of a tree code is far more sensitive than the distance of a standard error correcting code. In particular, changing the color of a single edge can cause the distance to vanish. It is thus not clear how one can "puncture" a tree code without vanish its distance.

Our key insight is to consider a variant of tree codes we call *palette-alternating tree codes* in which the number of colors is allowed to depend on the depth. A good first example to have in mind is a coloring that uses 4 colors in even layers and 2 colors in odd layers. To our surprise, such palette-alternating tree codes with non-vanishing distance exist! To calculate the rate-overhead incurred by using this palette-alternating tree code, observe that the number of bits sent when using an (even) depth-n palette-alternating tree code is

$$\frac{n}{2}\log_2 2 + \frac{n}{2}\log_2 4 = \frac{3}{2}n,$$

and so the rate incurred by the encoding is $2/3$, improving upon the $1/2$ rate one would get by using the best available tree code. Note that this even beats the rate of a 3-color tree code–had it existed–since $\log_2 3 > 3/2$. Put differently, in an amortized sense, the palette-alternating tree code above requires only $2^{3/2} \approx 2.83$ colors.

One can get greedy and ask whether a palette-alternating tree code that uses, say, 4 colors at layers $0, 3, 6, ...$ and 2 colors in the remaining layers exist. If so, one can potentially improve the scheme's rate to $3/4$. We prove the existence of such palette-alternating tree codes. In fact, we show that one can use 4 colors as seldom as she please and 2 colors–the bare minimum–in most layers. We turn to give a formal treatment of the above discussion.

▶ **Definition 3** (Palette-alternating tree codes). *Let $\Sigma_0, \ldots, \Sigma_{c-1}$ be (not necessarily distinct) sets. Let T be the complete rooted infinite binary tree. A* palette-alternating tree code *is an edge-coloring of T where at layer $t \in \mathbb{N}$ the colors are taken from the set $\Sigma_{t \pmod c}$. T is said to have distance δ if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. We define the* rate *ρ of T to be the number satisfying*

$$\frac{1}{\rho} = \frac{1}{c}\sum_{i=0}^{c-1}\log_2 |\Sigma_i|.$$

We suggest that the flexibility introduced by palette-alternating tree codes allows one to better capture the notion of rate in the online setting. Indeed, the importance of rate is only significant when "long" messages are being sent and so, informally, using a big palette of colors only once in a while should not be considered as an indication of poor rate. Our definition of rate formalizes that property. Note that we still insist on having the distance measured in terms of worst-case–a must as we wish to replace tree codes with palette-alternating tree codes in interactive coding schemes. It is only the rate that is being, in a sense, amortized.

As mentioned, we prove that palette-alternating tree codes can have rate approaching arbitrarily close to 1 while maintaining non-vanishing distance, thus bypass the $1/2$ bound proven for (standard) tree codes.

▶ **Theorem 4.** *For every $\varepsilon > 0$ there exists a palette-alternating tree code with rate $1 - \varepsilon$ and distance $\delta = \Omega(\varepsilon \cdot \log^{-1}(1/\varepsilon))$.*

Comparison with the Gilbert-Varshamov bound

Observe that the distance-rate trade-off obtained in Theorem 4 is the same as the one obtained by the Gilbert-Varshamov bound for standard offline binary error correcting codes, and in particular is optimal (up to constant factors). Interestingly, while it is known that the channel capacity in the online setting is $1 - \Theta(\sqrt{\varepsilon \log(1/\varepsilon)})$–significantly lower than in the offline setting [20], the online requirement on the encoding function itself does not cost more in terms of the distance-rate trade-off. Rather, it is the additional overhead incurred by the mechanism required for synchronization that is responsible for the lower channel capacity in the online setting. We elaborate more on this in Section 1.3.

The proof of Theorem 4, which can be found in Section 4, is based on a variant of the construction we use in Theorem 2. There, the alphabet symbols are taken from the field of four elements, $\mathbb{F}_4$. The key idea in obtaining the savings in the alphabet size is to trace the $\mathbb{F}_4$ field elements down to $\mathbb{F}_2$ in most layers. Interestingly, we cannot afford to work over the field $\mathbb{F}_3$ as we crucially rely on the fact that the characteristic of the fields is 2 as well as on the smaller field being a subfield of the larger one.

1.2.1 Palette-alternating tree codes: further discussion and generalization

We remark that it is not clear if one can start from an arbitrary 4-color tree code and change some of the layers to have only 2 colors (in a sense, effectively puncturing the 4-color tree code) while maintaining non-vanishing distance. Our proof seems to have the effect of "correlating" the colors in the 4-color layers with the paths that contain them. To emphasize this point, note that a 2-color layer does not immediately "buy" us redundancy. Nevertheless, the 2-color layers have the important task of making sure that the 4-color layers do. Indeed, by switching the colors of siblings in the 2-color layers one can potentially vanish the distance.

It is also interesting to compare palette-alternating tree codes that use 2 colors in most layers with some of the known probabilistic schemes [20, 18] that take the following strategy: in most rounds simulate the protocol as is (namely, assuming no errors occur) and only rarely verify the transcript using hash functions. It is tempting to compare the 2-color layers in a palette-alternating tree code with the error-free part of the simulation and the 4-color layers with the verification rounds. Indeed, at the very least, both the 2-color layers and the error-free part cost nothing in terms of rate. The crucial difference, however, lies in the fact that while the error-free simulation does not carry any weight in terms of error correction, the 2-color layers do.

We end this section by proposing a more general, and arguable more natural, definition than palette-alternating tree codes which allows for different palettes used at different layers without being necessarily periodical. While our proof of Theorem 4 yields a palette-alternating tree code, we believe that the more general definition is worth presenting here. For simplicity, we identify a finite color set Σ with $\{1, 2, \ldots, |\Sigma|\}$.

▶ **Definition 5** (Dynamic-Palette Tree Codes). *Let $c : \mathbb{N} \to \mathbb{N}$. Let T be the complete rooted infinite binary tree. A* dynamic-palette tree code *is an edge-coloring of T where at layer $t \in \mathbb{N}$ the colors are taken from the set $\{1, 2, \ldots, c(t)\}$. T is said to have distance δ if for every pair of vertices u, v with equal depth it holds that $h(u, v) \geq \delta$. We define the rate ρ of T to be the number satisfying*

$$\frac{1}{\rho} = \inf_{\ell \in \mathbb{N}} \frac{1}{\ell} \sum_{i=1}^{\ell} \log_2 c(i).$$

1.3 Interactive coding schemes

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity. Our coding scheme is efficient given an explicit construction of palette-alternating tree codes and serves as an alternative to the scheme obtained by Gelles *et al.* [13]. In this section we describe our result and proof technique. We start by reviewing basic notions in interactive coding schemes.

Communication complexity

Communication complexity addresses a basic question: If several parties wish to compute a function of the information they jointly possess, how long does their conversation need to be? In its most basic form, one considers two parties, Alice and Bob, that would like to jointly compute a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ of their respective inputs $x, y \in \{0,1\}^n$. The parties can communicate over a channel, and their goal is to compute $f(x,y)$ by exchanging as few bits as possible.

An interactive computation as above is performed via a communication protocol π which consists of a pair of algorithms π_A and π_B run by Alice and Bob, respectively. In this paper we focus on deterministic protocols, that is, π_A and π_B are deterministic algorithms. Informally, the communication is performed in rounds where the protocol dictates what is sent in each round based on the round number, the input of the party, and the bits received so far. After some number of rounds $r = r(x,y)$ the protocol terminates, at which point both parties know $f(x,y)$. The *(deterministic) communication complexity* of the protocol π is given by $\mathsf{CC}(\pi) = \max_{x,y} r(x,y)$. The *communication complexity* of f, denoted by $\mathsf{CC}(f)$, is the minimum of $\mathsf{CC}(\pi)$ over all protocols π that compute f.

Interactive coding schemes

One aspect that is always an issue when considering communication are errors in transmission introduced by imperfect or compromised channels. The research field of coding for interactive communication that addresses this issue was initiated in a sequence of seminal papers by Schulman [28, 29, 31], and is by now an active and exciting research field (see Gelles's excellent survey [12]). There are several models one can consider. For examples, transmitted bits can be erased (replaced with a senseless symbol $\perp$) or worse–flipped–leaving no trace to the occurred error. In this paper we focus on perhaps the most well-studied model in which bits can be flipped. Further, we consider the most difficult setting of *adversarial errors* in which any ε-fraction of the bits might be flipped.

A protocol π is said to be *ε-resilient* if the protocol preserves its functionality even at the presence of ε-fraction of adversarial errors. The *ε-resilient communication complexity* of f, denoted by $\mathsf{CC}_\varepsilon(f)$, is the minimum of $\mathsf{CC}(\pi)$ over all ε-resilient protocols π that compute f. For any fixed function f it is clear that $\mathsf{CC}_\varepsilon(f)$ is non-decreasing as ε increases. In the extreme cases $\mathsf{CC}_0(f) = \mathsf{CC}(f)$ whereas $\mathsf{CC}_1(f) = \infty$, namely, $\mathsf{CC}_1(f)$ is unbounded.

Resilient protocols are typically obtained by devising an *interactive coding scheme* which, informally, is a compiler CS_ε that is parameterized by the resiliency parameter ε. Given a protocol π, the interactive coding scheme produces an ε-resilient protocol $\mathsf{CS}_\varepsilon(\pi) = \pi_\varepsilon$ that computes the same function as π. The goal is to design an interactive coding scheme with low overhead in communication. Namely, one would like to maximize $\rho(\pi) = \mathsf{CC}(\pi)/\mathsf{CC}(\pi_\varepsilon)$. The *rate* of the interactive coding scheme $\rho(\mathsf{CS}_\varepsilon)$ is the infimum of $\rho(\pi)$ over all protocols π.

Channel capacity

Focusing on the channel itself, rather than on any specific function f, one can define the *channel capacity* $\mathsf{Cap} : [0,1] \to [0,1]$ by

$$\mathsf{Cap}(\varepsilon) = \inf_f \left(\frac{\mathsf{CC}(f)}{\mathsf{CC}_\varepsilon(f)} \right),$$

where the infimum is taken over all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ for all $n \geq 1$. Note that $\mathsf{Cap}(0) = 1$ whereas $\mathsf{Cap}(1) = 0$. A fundamental problem in interactive coding theory, and the focus of this work, is the study of the channel capacity $\mathsf{Cap}(\varepsilon)$.

We remark that the channel capacity can be defined with respect to other models and a huge body of work is devoted to the study of the channel capacity in our setting as well as for other channels, most notably binary symmetric channels (BSC) in which every bit is flipped independently with probability ε. Moreover, one needs to specify other properties of the protocols so as to formalize the problem. For example, is the turn of speak predetermined by the protocol or can it depend on the exchanged bits? In case of such "adaptive" protocols, what happens if both parties send a message at the same round?

As in most works, we focus on non-adaptive protocols in which the turn of speak is fixed in advance. For concreteness, we focus on *alternating protocols* where Alice speaks at even rounds and Bob speaks at odd rounds. We made this choice mostly for convenience and our results can be straightforwardly generalized. We also assume that the channel is binary. This is the most difficult setting and allowing for channels over a larger alphabet, especially one that can depend on the error parameter ε, only makes the problem of devising protocols easier.

In his seminal work [29], Schulman proved that $\mathsf{Cap}(\varepsilon) > 0$ for some $\varepsilon > 0$. In a tour de force result, Kol and Raz [20] gave a tight bound of $\mathsf{Cap}(\varepsilon) = 1 - \Theta(\sqrt{\varepsilon \log 1/\varepsilon})$ on the channel capacity in this setting for non-adaptive probabilistic protocols. Their upper bound clearly holds for adversarial errors as well. Gelles *et al.* [13] gave the first deterministic coding scheme against adversarial errors, derandomizing Haeupler's protocol [18], that approaches capacity, namely, their coding scheme has rate $1 - O(\sqrt{\varepsilon \log 1/\varepsilon})$.

1.4 Capacity approaching coding schemes via palette-alternating tree codes

Based on palette-alternating tree codes, we devise a deterministic interactive coding scheme against adversarial errors that approaches capacity and thus matches the rate obtained by [13]. The advantage of our coding scheme is that given an explicit construction of palette-alternating tree codes, our scheme is efficient. We believe that the recent progress on tree code constructions [11, 24] may eventually lead to constructions of palette-alternating tree codes. The coding scheme suggested in [13], on the other hand, relies on a certain counting argument, and it is not clear to us how to obtain an efficient scheme based on these ideas.

▶ **Theorem 6.** *Let $\varepsilon > 0$. Assume there exists an explicit palette-alternating tree code with rate $1 - \varepsilon$ and distance $\delta = \Omega(\varepsilon \cdot \log^{-1}(1/\varepsilon))$ (which, computational aspects aside, we know exists by Theorem 4). Then, there exists an efficient deterministic coding scheme against ε-fraction of adversarial errors with rate $1 - O(\sqrt{\varepsilon \log(1/\varepsilon)})$.*

1.4.1 Proof idea

In the remaining of this section, we elaborate on some of the ideas that go into our construction and analysis of Theorem 6.

1.4.1.1 Synchronization

Interactive coding schemes that make use of tree codes do not simply encode the bits that are meant to be sent by the non-resilient protocol π using the tree code. These schemes also need to implement a mechanism for making sure that both parties are, in a sense, synchronized. Indeed, informally, the errors have the effect of causing the parties to transmit data with respect to information that was never sent to them. Without a way to synchronize, even with no additional errors, the parties will not be able to make progress on simulating the protocol as the information they exchange is irrelevant.

Thus, on top of the bits that the parties would have communicate without the presence of errors, some meta data used for synchronization must be maintained and transmitted. Both the "data bits" as well as the "sync bits" are encoded using a tree code before sent over the channel. Thus, the rate of deterministic interactive coding schemes is determined both by the rate of the tree code as well as by the overhead required for synchronization.

To obtain interactive coding schemes with rate approaching 1 we need, on top of replacing a tree code with a palette-alternating tree code, to have a low overhead in synchronization. There are two main obstacles for accomplishing that:

1. One must argue that not too many sync bits are needed to successfully maintain synchronization; and
2. One needs to distinguish between sync bits and data bits which in previous works was effectively done by sending a bit indicating the bit "type" (more precisely, a larger alphabet was used followed by an alphabet reduction).

The first issue is fairly straightforward to handle. Indeed, it is intuitive that in a sensible scheme, the amount of synchronization required is proportional to the fraction of errors and this is true for both Schulman's coding scheme [29] and for Braverman-Rao's scheme [9]. The second issue requires more care. Braverman-Rao's scheme is very dynamic and on any given round the bit type depends on the error pattern enforced by the adversary. Although most bits are data bits, it seems difficult to argue that their scheme can be made to have high rate. Luckily, we are able to devise a coding scheme based on some adaptation of Schulman's original ideas. The coding scheme obtained, however, does not approach capacity, and has rate $1 - \tilde{O}(\sqrt[3]{\varepsilon})$ (see Section 5.3). Further ideas are required to prove Theorem 6 which we discuss next.

1.4.1.2 Clusters of failed decoding rounds

In order to approach capacity, we examine more closely the effect that adversarial errors have on (palette-alternating) tree codes. Schulman's analysis is based on bounding the number of rounds in which decoding fails. More precisely, it was shown [29] that if one encodes using a tree code with distance $\delta_{\mathcal{TC}}$ then at most $O(\varepsilon/\delta_{\mathcal{TC}})$ fraction of rounds would result in failed decoding. We prove a structural result, refining the quantitative one, regarding *where* these "bad" rounds may occur as a function of the locations of the adversarial errors. We show that the bad rounds are, in a sense, clustered around the errors that are introduced. We exploit this structure to obtain a tighter analysis of our protocol, and achieve the stated, optimal, rate.

1.5 Organization

In Section 2 we give the formal definitions of protocols and interactive coding schemes, as well as setting notation and state some known results we use. In Section 3 we prove Theorem 2 which asserts that 4-color tree codes exist. While not directly applicable to our proof of Theorem 6, we encourage the reader to read the proof (including Section 3.1) as ideas from the proof will be used for proving the existence of palette-alternating tree codes (Theorem 4). In Section 4 we prove Theorem 4. Lastly, in Section 5, we prove Theorem 6 where first, in Section 5.3, we give a sub-optimal analysis.

2 Preliminaries

Unless otherwise stated, all logarithms are taken to the base 2. We denote by $\mathbb{N}$ the set of natural numbers (of course, including 0), and write $\mathbb{N}_1$ for $\mathbb{N} \setminus \{0\}$. For integers $a \leq b$ we write $[a, b]$ for all integers in this interval. For an integer $c \geq 1$, we let $[c] = \{1, 2, \ldots, c\}$. We follow the convention that strings are indexed starting from 1. For two strings $x, y \in \Sigma_1 \times \cdots \times \Sigma_n$, we denote by $\Delta(x, y)$ their hamming distance. We make use of the following standard inequalities.

▶ **Lemma 7.** *For every integers $1 \leq k \leq n$ with $\frac{k}{n} = \delta \leq \frac{1}{2}$ it holds that*

$$\sum_{i=0}^{k} \binom{n}{i} \leq 2^{H(\delta)n}.$$

▶ **Lemma 8.** *For every $0 < x < \frac{1}{2}$ it holds that*

$$\frac{x}{2\log_2(6/x)} \leq H^{-1}(x) \leq \frac{x}{\log_2(1/x)}.$$

2.1 Coding for interactive communication

2.1.1 Communication protocols

In this section we briefly recall some basic definitions from communication complexity. For more details we refer the reader to [21, 27]. Let $T = (V, E)$ be a complete finite rooted binary tree. Given an internal vertex v in T, define $\mathsf{son}(v, 0)$, $\mathsf{son}(v, 1)$ to be the left son and the right son of v in T, respectively. Extend son for bit strings of length $n \geq 1$ in the natural way and denote by path the function that given $x \in \{0, 1\}^n$, returns the edges on the unique rooted path to $\mathsf{son}(\mathsf{root}(T), x)$. A *communication protocol* π consists of:

- A function $f_v : \{0, 1\}^n \to \{0, 1\}$ for every internal node v in T.
- A label $\mathsf{player}(v) \in \{A, B\}$ for each internal node v.
- A label $\mathsf{value}(v) \in \{0, 1\}$ for every leaf v.

The protocol π induces a function $f = f(\pi) : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ in the following natural way. Given $x, y \in \{0, 1\}^n$, for every internal node $v \in V$, if $\mathsf{player}(v) = A$ let $d = f_v(x)$ and otherwise let $d = f_v(y)$. Let u be the left son of v if $d = 0$ and otherwise let u be the right son of v. Thus, given x, y, from every internal node v goes out exactly one edge $e_v(x, y) = (v, u(x, y))$. Let $E(x, y) = \{e_v(x, y) \mid v \text{ internal node}\}$ be the set of these edges. Observe that the edge set $E(x, y)$ induces a unique root to leaf path in T. Let $v(x, y)$ be that unique leaf that is reachable from the root. We define $f(x, y) = \mathsf{value}(v(x, y))$. We write $\mathsf{depth}(\pi)$ for the depth of T.

The computation above of $f(x, y)$ can be made by two parties, Alice that holds x and Bob that holds y, that can communicate over a channel, in the natural way. Namely, at node v, if $\mathsf{player}(v) = A$ then Alice sends to Bob $f_v(x)$ wheres at a node v with $\mathsf{player}(v) = B$ Bobs sends $f_v(y)$ to Alice. It is clear that the number of bits communicated is the depth of the tree. We say that a protocol is *alternating* if $\mathsf{player}(v) = A$ if and only if v is at even depth. From here on we focus only on alternating protocols.

2.1.2 The pointer jumping game

The pointer jumping game is, in a sense, a complete problem for interactive protocols. Let $T = (V, F)$ be a complete finite rooted binary tree. The depth of a vertex v is the distance, measured in edges, from the root to v. In particular, the depth of the root is 0. We partition the internal nodes of T to $V = V_A \cup V_B$, where V_A contains all nodes at even depth and V_B all nodes at odd depth. We partition the edge set $F = X \cup Y$ with X being the edges going out of V_A and Y the edges leaving V_B. We call a subset of edges $E \subseteq F$ *consistent* if every internal node has exactly one outgoing edge in E. Given a consistent set of edges E, we partition $E = E_A \cup E_B$ where $E_A = E \cap X$ and $E_B = E \cap Y$. It is convenient to represent E_A and E_B by functions $\pi_A : V_A \to \{0,1\}$, $\pi_B : V_B \to \{0,1\}$ as follows: for $v \in V_A$, $\pi_A(v) = 0$ if and only if the edge in E_A that goes out of v is to the left son of v, and similarly for π_B.

Note that in any consistent set of edges E there is a unique root to leaf path. The *pointer jumping game* is a function that given a consistent set of edges E returns the unique leaf reachable from the root using the edge set E. Consider a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and a protocol π for f. Note that for any fixed x, y the task of computing the value $f(x, y)$ is an instance of the pointer jumping game. In that sense, the pointer jumping game is complete. Given a function f as above, it is sometimes convenient to consider a corresponding pointer jumping game of depth $R > n$ in which the edge leaving every vertex of depth larger than n points to its left son (this choice is of course arbitrary and any fixed choice will do).

2.1.3 Resilient protocols and interactive coding schemes

A protocol π is said to be ε-*resilient* if on any pair $x, y \in \{0,1\}^n$, in the above two party computation, both Alice and Bob compute $f(x, y)$ correctly even if at most ε-fraction of the communicated bits are flipped. An *interactive coding scheme* (coding scheme for short) is a function CS_ε, parameterized by $\varepsilon \in [0, 1]$, that gets as input a protocol π and outputs an ε-resilient protocol $\pi_\varepsilon = \mathsf{CS}_\varepsilon(\pi)$ with $f(\pi_\varepsilon) = f(\pi)$. The *rate* of the coding scheme CS_ε is defined by

$$\rho(\mathsf{CS}_\varepsilon) = \inf_\pi \frac{\mathsf{depth}(\pi)}{\mathsf{depth}(\pi_\varepsilon)}.$$

Observe that for the purpose of devising a coding scheme CS_ε one may assume that the inputs x, y are fixed. Thus, it suffices to focus on the problem of devising a coding scheme for the pointer jumping game.

3 Binary Tree Codes: Four Colors Suffice

In this section we prove Theorem 2. We start by setting some notation. Let T be the infinite complete rooted binary tree. We identify length-n paths in T that starts at the root with length-n binary strings in the natural way. Namely, we identify left son and right son with 0 and 1, respectively. Given a node v at depth $n \geq 1$ we define $p_v \in \{0,1\}^n$ to be the string that encodes the (unique) path from the root to v.

An edge-coloring of T by a color set Σ is given by a function, which for ease of readability, we slightly abuse notation and also denote by $T : \{0,1\}^{\mathbb{N}_1} \to \Sigma^{\mathbb{N}_1}$, where the color of an edge $e = u \to v$ is $T(p_v)_{\mathsf{depth}(v)}$. Note that T is an online function, namely, for every $x \in \{0,1\}^{\mathbb{N}_1}$ and $i \in \mathbb{N}_1$, the value $T(x)_i$ is determined by $x_1, \ldots, x_i$.

The (probabilistic) construction

Let $\{R_i\}_{i\in\mathbb{N}_1}$ be a sequence of independent random variables, each is uniformly distributed over $\mathbb{F}_4$–the field of 4 elements. Let $\mathbb{F}_2$ be the (unique) subfield of $\mathbb{F}_4$ of size 2. Define the (random) coloring function $T : \mathbb{F}_2^{\mathbb{N}_1} \to \mathbb{F}_4^{\mathbb{N}_1}$ (where we identify $\mathbb{F}_2$ and $\{0,1\}$ in the natural way) as follows: for every $t \in \mathbb{N}_1$

$$T(x)_t = \sum_{i=1}^{t} R_{t+1-i} x_i. \tag{1}$$

▶ **Definition 9.** *Let v be a depth-n vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. For $k = 1, \ldots, \ell$ we define the random variable*

$$a_v(x, y, k) = T(p_v \circ 1 \circ y)_{n+k} - T(p_v \circ 0 \circ x)_{n+k}.$$

Note that $a_v(x, y, k)$ is a (random) element in $\mathbb{F}_4$. We define the integral random variable

$$h_v(x, y) = \sum_{k=1}^{\ell} I_k,$$

where I_k is the indicator random variable that equals 1 when $a_v(x, y, k) \neq 0$. Note that $h_v(x, y) \in \{0, 1, \ldots, \ell\}$ is the Hamming distance between $T(p_v \circ 0 \circ x)_{[n+1,n+\ell]}$ and $T(p_v \circ 1 \circ y)_{[n+1,n+\ell]}$.

▷ Claim 10. Let v be a vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, for every $k \in \{1, \ldots, \ell\}$ it holds that

$$a_v(x, y, k) = R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i.$$

Proof. Denote the depth of v by n. Fix $k \in \{1, \ldots, \ell\}$. By Equation (1),

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i.$$

Thus,

$$\begin{aligned} a_v(x, y, k) &= \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i - \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i. \end{aligned}$$

◁

▷ Claim 11. Let v be a vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, the random variables $a_v(x, y, 1), \ldots, a_v(x, y, \ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$.

Proof. By Claim 10, $a_v(x,y,k) = R_k + L_k$ where L_k is some $\mathbb{F}_4$-linear combination of $R_1,\ldots,R_{k-1}$. Therefore, $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1)$, $\ldots, a_v(x,y,k-1)$. As this holds for every k we have that $a_v(x,y,1),\ldots,a_v(x,y,\ell)$ are independent. To conclude the proof, note that for every fixing of $R_1,\ldots,R_{k-1}$, $a_v(x,y,k) = R_k + \ell_k$ for some fixed $\ell_k \in \mathbb{F}_4$ and so $a_v(x,y,k)$ is uniform over $\mathbb{F}_4$. ◁

▷ Claim 12. For every two vertices u,v in T and every $x,y \in \mathbb{F}_2^{\ell-1}$,

$$h_v(x,y) = h_u(x,y),$$
$$h_v(x,y) = h_v(0^{\ell-1}, y-x).$$

Proof. The first equality follows immediately by Claim 10 as, for every $k \in \{1,\ldots,\ell\}$, the expression obtained for $a_v(x,y,k)$ is independent of the choice of v. As for the second asserted equality, again by Claim 10,

$$\begin{aligned} a_v(x,y,k) &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y-x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}((y-x)-0)_i \\ &= a_v(0^{\ell-1}, y-x, k), \end{aligned}$$

where observe that for the last equality we are using the fact that $\mathbb{F}_2$ is a subfield of $\mathbb{F}_4$ and so $y-x \in \mathbb{F}_2^{\ell-1}$. Indeed, recall that a_v's second argument is a binary string and so the equality above would have been meaningless otherwise. The above equation implies $h_v(x,y) = h_v(0^{\ell-1}, y-x)$, proving the claim. ◁

Given Claim 12 we can simplify our notation as follows. Let r denote the root of T. For $x \in \{0,1\}^{\ell-1}$ and $k \in \{1,\ldots,\ell\}$ we define the random variables

$$a(x,k) = a_r(0^\ell, 1 \circ x, k),$$
$$h(x) = h_r(0^{\ell-1}, x).$$

Note that $h(x) = \sum_{k=1}^{\ell} a(x,k)$.

▶ **Theorem 13.** *There exists a fixing of the sequence $\{R_i\}_i$ such that the function T is a tree code with distance* 0.05.

Proof. First note that for every fixing of the sequence $\{R_i\}_i$, T is an online function. Observe that, for a fixing of $\{R_i\}_i$, T is a tree code with distance δ if and only if for every $\ell \geq 1$ and $x \in \{0,1\}^{\ell-1}$ it holds that $h(x) \geq \delta\ell$. Indeed, recall that by definition, T is a tree code with distance δ if and only if for every vertex v in T, $\ell \geq 1$, and for every $x,y \in \{0,1\}^{\ell-1}$ it holds that $h_v(x,y) \geq \delta\ell$. However, by Claim 12, $h_v(x,y) = h(y-x)$.

For $x \in \{0,1\}^{\ell-1}$ denote by $E(x)$ the event $h(x) < \delta\ell$. By the above discussion, it suffices to prove, for $\delta = 0.05$, that

$$\Pr\left[\bigcup_{x\in\{0,1\}^{\mathbb{N}}} E(x)\right] < 1.$$

To this end, by the union bound, it suffices to prove that

$$\sum_{x\in\{0,1\}^{\mathbb{N}}} \Pr[E(x)] < 1.$$

Consider any $x \in \{0,1\}^{\ell-1}$ with $\ell \geq 1$. Note that the event $E(x)$ holds if and only if there exists a set $T \subseteq \{1,\ldots,\ell\}$ of size $|T| \geq \lceil (1-\delta)\ell \rceil$ such that for every $k \in T$, $a(x,k) = 0$. By taking the union bound over all such sets T, and using that $a(x,1),\ldots,a(x,\ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$ (Claim 11), we get

$$\Pr[E(x)] \leq \binom{\ell}{\lceil (1-\delta)\ell \rceil} 4^{-\lceil (1-\delta)\ell \rceil}. \tag{2}$$

By Lemma 7, we have that

$$\frac{1}{\ell} \cdot \log_2 \binom{\ell}{\lceil (1-\delta)\ell \rceil} \leq H\left(\frac{\lceil (1-\delta)\ell \rceil}{\ell}\right).$$

As $\delta < \frac{1}{2}$ and since the entropy function H decreases in $[\frac{1}{2}, 1]$ we have that

$$H\left(\frac{\lceil (1-\delta)\ell \rceil}{\ell}\right) \leq H(1-\delta) = H(\delta).$$

Substitute to Equation (2), we get that

$$\Pr[E(x)] \leq 2^{(H(\delta)-2(1-\delta))\ell}.$$

Thus,

$$\begin{aligned}\sum_{x \in \{0,1\}^{\mathbb{N}}} \Pr[E(x)] &\leq \sum_{\ell=1}^{\infty} 2^{\ell-1} \cdot 2^{(H(\delta)-2(1-\delta))\ell} \\ &= \frac{1}{2} \sum_{\ell=1}^{\infty} 2^{(H(\delta)+2\delta-1)\ell}.\end{aligned}$$

One can verify that for $\delta = 0.05$ the above geometric sum is strictly smaller than 1, and the theorem follows. ◀

3.1 Improving the distance

We now show a method for improving the distance. We illustrate it to obtain a bound of 0.136 on the distance, which proves Theorem 2, though we believe that the method can be used to push the bound further. It is fairly easy to show that the distance of a 4-color tree code cannot be larger than 1/2.

▶ **Theorem 14.** *There exists a fixing of the sequence $\{R_i\}_i$ such that the function T is a tree code with distance* 0.136*.*

Proof. For the proof it will be convenient to consider a specific representation of $\mathbb{F}_4$. We make use of the standard construction of $\mathbb{F}_4$ as a quotient of the polynomial ring over $\mathbb{F}_2$ with respect to an ideal generated by a degree 2 irreducible element as follows. Note that $t^2+t+1 \in \mathbb{F}_2[t]$ is irreducible, and so $K = \mathbb{F}_2[t]/\langle t^2+t+1 \rangle$ is a field of 4 elements which we will take as the construction for $\mathbb{F}_4$. Let α be the class of t in K. In this representation, the field $\mathbb{F}_4$ consists of the elements $0, 1, \alpha, \alpha+1$ where $\alpha^2+\alpha+1 = 0$.

Consider the sequence $\{R_i\}_{i \in \mathbb{N}}$ as in the beginning of the section but with the fixings $R_1 = 1$ and $R_2 = \alpha$. Observe that for every $x \in \mathbb{F}_2^{\ell-1}$ with $\ell \geq 2$ it holds that $a(x,1) = 1$ and $a(x,2) = \alpha + x_1$. In particular, $a(x,1), a(x,2)$ are both non-zeros and so $h(x) \geq 2$. Let

$\ell_0 \geq 2$ be an integer parameter to be chosen later on. By the above, we have that for every $x \in \mathbb{F}_2^{\ell-1}$ with $\ell \leq \ell_0$ it holds that

$$\frac{h(x)}{\ell} \geq \frac{2}{\ell_0}. \tag{3}$$

For $x \in \{0,1\}^{\ell-1}$ denote by $E_{1,\alpha}(x)$ the event $h(x) < \delta\ell$ with the $\{R_i\}_{i\in\mathbb{N}}$ as defined above, namely, $R_1 = 1$, $R_2 = \alpha$ and the rest of the random variables $\{R_i \mid i \geq 3\}$ are independent and uniformly distributed over $\mathbb{F}_4$. Once we establish a bound of

$$\Pr\left[\bigcup_{|x|\geq\ell_0} E_{1,\alpha}(x)\right] < 1 \tag{4}$$

for some choice of δ then, combined with Equation (3), we will establish the existence of a tree code with distance at least

$$\min\left(\frac{2}{\ell_0}, \delta\right).$$

Consider any $x \in \{0,1\}^{\ell-1}$ with $\ell \geq \ell_0 + 1$. The event $E_{1,\alpha}(x)$ holds if and only if there exists a set $T \subseteq \{3,\ldots,\ell\}$ of size $|T| \geq \lceil(1-\delta)\ell\rceil$ such that for every $k \in T$, $a(x,k) = 0$. By taking the union bound over all such sets T, and using that $a(x,3),\ldots,a(x,\ell)$ are independent and each is uniformly distributed over $\mathbb{F}_4$, we get that

$$\begin{aligned}\Pr[E_{1,\alpha}(x)] &\leq \binom{\ell-2}{\lceil(1-\delta)\ell\rceil} 4^{-\lceil(1-\delta)\ell\rceil} \\ &\leq \binom{\ell}{\lceil(1-\delta)\ell\rceil} 4^{-\lceil(1-\delta)\ell\rceil}\end{aligned}$$

By Lemma 7, we have that

$$\frac{1}{\ell} \cdot \log_2 \binom{\ell}{\lceil(1-\delta)\ell\rceil} \leq H\left(\frac{\lceil(1-\delta)\ell\rceil}{\ell}\right).$$

As we will choose $\delta < \frac{1}{2}$ and the entropy function H decreases in $[\frac{1}{2}, 1]$ we have that

$$H\left(\frac{\lceil(1-\delta)\ell\rceil}{\ell}\right) \leq H(1-\delta) = H(\delta).$$

Thus,

$$\Pr[E_{1,\alpha}(x)] \leq 2^{(H(\delta)-2(1-\delta))\ell}.$$

By substituting the above equation to Equation (4), we get that

$$\sum_{|x|\geq\ell_0} \Pr[E_{1,\alpha}(x)] \leq \sum_{\ell=\ell_0+1}^{\infty} 2^{\ell-1} \cdot 2^{(H(\delta)-2(1-\delta))\ell}.$$

Write $\beta = 2^{H(\delta)+2\delta-1}$. Then, the above is bounded by

$$\frac{1}{2}\sum_{\ell=\ell_0+1}^{\infty} \beta^\ell = \frac{\beta^{\ell_0+1}}{2(1-\beta)}.$$

Consider the real polynomial

$$f_{\ell_0}(x) = x^{\ell_0+1} - 2(1-x).$$

We have that

$$f'_{\ell_0}(x) = (\ell_0+1)x^{\ell_0} + 2$$

Since $\ell_0 \geq 2$, $f'_{\ell_0}(x) > 0$ for all $x \geq 0$. Further, $f_{\ell_0}(0) = -2$ and $f_{\ell_0}(1) = 1$. Thus, $f_{\ell_0}(x)$ has a single root $\beta_{\ell_0} \in [0,1]$ (in fact, β_{ℓ_0} is monotone-increasing as a function of ℓ_0, and $\beta_{\ell_0} \to 1$ as $\ell_0 \to \infty$). For a fixed choice of ℓ_0, by choosing $\beta < \beta_{\ell_0}$ and solving for δ (recall $\beta = 2^{H(\delta)+2\delta-1}$) to obtain δ_{ℓ_0}, we get that there exists a fixing of $\{R_i \,|\, i \geq 3\}$ such that the obtained tree code has distance at least $\min(\delta_{\ell_0}, \frac{2}{\ell_0})$. Thus, the obtained bound is

$$\max_{\ell_0 \geq 2} \min\left(\delta_{\ell_0}, \frac{2}{\ell_0}\right).$$

One can verify that $\ell_0 = 14$ maximizes the above equation to get distance larger than 0.136. ◀

4 Palette-Alternating Tree Codes

In this section we prove Theorem 4. To this end we recall the definition of the (field) trace function $\mathsf{Tr} : \mathbb{F}_4 \to \mathbb{F}_2$ that is given by $\mathsf{Tr}(x) = x + x^2$. Observe that the trace function is an $\mathbb{F}_2$-linear map whose image and kernel are $\mathbb{F}_2$. In particular, if X is uniform over $\mathbb{F}_4$, then $\mathsf{Tr}(X)$ is uniform over $\mathbb{F}_2$.

Let ε be a given parameter and define $b = \lceil 1/\varepsilon \rceil$. Let $\{R_i\}_{i\in\mathbb{N}}$ be a sequence of independent random variables, each is uniformly distributed over $\mathbb{F}_4$ except that R_1 is fixed to $R_1 = 1$. We define a palette-alternating tree code with b palette sets $\Sigma_0, \ldots, \Sigma_{b-1}$ such that $\Sigma_0 = \mathbb{F}_4$ and $\Sigma_i = \mathbb{F}_2$ for $i > 0$. Let $x \in \mathbb{F}_2^{\mathbb{N}}$. For every $k \in \mathbb{N}$, define

$$S_k(x) = \sum_{i=1}^{k} R_{k+1-i} x_i,$$

where addition and multiplication are performed in $\mathbb{F}_4$ and, as usual, $\mathbb{F}_2$ is identified with the unique subfield of two elements in $\mathbb{F}_4$. The coloring function is given by

$$T(x)_k = \begin{cases} S_k(x), & k \equiv_b 0; \\ \mathsf{Tr}(S_k(x)), & \text{otherwise.} \end{cases}$$

▶ **Theorem 15.** *The function T above is a palette-alternating tree code with rate $1-\varepsilon$ and distance $\delta = \Omega(\varepsilon \log^{-1}(1/\varepsilon))$.*

Proof. First, observe that T is indeed an online function with rate larger than $1-\varepsilon$. Further Definition 9 can be carried over to the more general case of palette-alternating tree codes. We turn to prove an analog to Claim 10.

▷ Claim 16. Let v be a depth-n vertex in T. Let $\ell \geq 1$ and $x, y \in \mathbb{F}_2^{\ell-1}$. Then, for every $k \in \{1, \ldots, \ell\}$ it holds that

$$a_v(x, y, k) = \begin{cases} R_k + S_{k-1}(y - x), & n + k \equiv_b 0; \\ \mathsf{Tr}(R_k + S_{k-1}(y - x)), & \text{otherwise.} \end{cases}$$

Proof. Fix $k \in \{1, \ldots, \ell\}$. Assume first that $n + k \equiv_b 0$. Then,

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=n+1}^{n+k} R_{n+k+1-i}(0 \circ x)_{i-n} \\ &= \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i + \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i.$$

Thus,

$$\begin{aligned} a_v(x, y, k) &= \sum_{i=1}^{k} R_{k+1-i}(1 \circ y)_i - \sum_{i=1}^{k} R_{k+1-i}(0 \circ x)_i \\ &= R_k + \sum_{i=1}^{k-1} R_{k-i}(y - x)_i \\ &= R_k + S_{k-1}(y - x). \end{aligned}$$

Assume now that $n + k \not\equiv_b 0$. Using that Tr is $\mathbb{F}_2$-linear,

$$\begin{aligned} T(p_v \circ 0 \circ x)_{n+k} &= \mathsf{Tr}\left(\sum_{i=1}^{n+k} R_{n+k+1-i}(p_v \circ 0 \circ x)_i\right) \\ &= \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=n+1}^{n+k} \mathsf{Tr}\left(R_{n+k+1-i}\right)(0 \circ x)_{i-n} \\ &= \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=1}^{k} \mathsf{Tr}(R_{k+1-i})(0 \circ x)_i. \end{aligned}$$

Similarly

$$T(p_v \circ 1 \circ y)_{n+k} = \mathsf{Tr}\left(\sum_{i=1}^{n} R_{n+k+1-i}(p_v)_i\right) + \sum_{i=1}^{k} \mathsf{Tr}(R_{k+1-i})(1 \circ y)_i.$$

Thus, again by $\mathbb{F}_2$-linearity of Tr,

$$\begin{aligned} a_v(x, y, k) &= \mathsf{Tr}(R_k) + \sum_{i=1}^{k-1} \mathsf{Tr}(R_{k-i})(y - x)_i \\ &= \mathsf{Tr}(R_k + S_{k-1}(y - x)). \end{aligned}$$

◁

▷ Claim 17. Let v be a depth-n vertex and $x, y \in \mathbb{F}_2^{\ell-1}$ distinct. Then, the random variables $a_v(x, y, 1), \ldots, a_v(x, y, \ell)$ are independent. Moreover, let $k \in [\ell]$. If $n + k \equiv_b 0$ then $a_v(x, y, k)$ is uniformly distributed over $\mathbb{F}_4$ and otherwise it is uniform over $\mathbb{F}_2$.

Proof. By Claim 16, if $n+k \equiv_b 0$ then $a_v(x,y,k) = R_k + L_k$ where L_k is a linear combination of $R_1, \ldots, R_{k-1}$. Thus, in this case, $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1), \ldots, a_v(x,y,k-1)$. Otherwise, namely $n+k \not\equiv_b 0$, we have that $a_v(x,y,k) = \mathsf{Tr}(R_k + L_k) = \mathsf{Tr}(R_k) + \mathsf{Tr}(L_k)$. Since for every fixing of L_k, $a_v(x,y,k)$ is uniform over $\mathbb{F}_2$, we have that $a_v(x,y,k)$ is independent of the joint distribution of $a_v(x,y,1), \ldots, a_v(x,y,k-1)$. As this holds for every $k \in [\ell]$ we have that $a_v(x,y,1), \ldots, a_v(x,y,\ell)$ are independent and their marginal distributions are as stated. ◁

▷ Claim 18. Let u, v be two vertices with depth n, m, respectively such that $n \equiv_b m$. Let $x, y \in \mathbb{F}_2^{\ell-1}$. Then,

$$h_v(x,y) = h_u(x,y),$$
$$h_v(x,y) = h_v(0^{\ell-1}, y-x).$$

Proof. Let $C_k = R_k + S_{k-1}(y-x)$. By Claim 16,

$$a_u(x,y,k) = \begin{cases} C_k, & n+k \equiv_b 0; \\ \mathsf{Tr}(C_k), & \text{otherwise.} \end{cases}$$

As C_k is independent of the choice of u and $n \equiv_b m$ we have that $a_u(x,y,k)$ is the same random variable as $a_v(x,y,k)$. Since this holds for every k, we have that $h_v(x,y) = h_u(x,y)$.

We turn to prove the the second asserted equality. Assume first that $k \in [\ell]$ is such that $n + k \equiv_b 0$. By Claim 16,

$$\begin{aligned} a_u(x,y,k) &= R_k + S_{k-1}(y-x) \\ &= R_k + S_{k-1}((y-x) - 0^{\ell-1}) \\ &= a_u(0^{\ell-1}, y-x, k), \end{aligned}$$

where observe that for the last equality we are using the fact that $\mathbb{F}_2$ is a subfield of $\mathbb{F}_4$ and so $y - x \in \mathbb{F}_2^{\ell-1}$. Indeed, recall that a_v's second argument is a binary string and so the equality above would have been meaningless otherwise. The case $n + k \not\equiv_b 0$ follows by a similar argument and using the $\mathbb{F}_2$-linearity of Tr. ◁

Given Claim 18, we can simplify our notation as follows. Let v_0 denote the root of the tree. For $i = 1, \ldots, b-1$ let v_i denote the left son of v_{i-1}. For every $i \in \{0, 1, \ldots, b-1\}$ and $x \in \{0,1\}^{\ell-1}$ we define the random variables

$$a_i(x,k) = a_{v_i}(0^\ell, 1 \circ x, k),$$
$$h_i(x) = h_{v_i}(0^{\ell-1}, x).$$

Define

$$\delta = c_1 \varepsilon \log^{-1}(1/\varepsilon),$$
$$\ell_0 = 12\lceil \varepsilon^{-1} \log(1/\varepsilon) \rceil,$$

for some constant $c_1 \in [0,1]$ to be set later on. Observe that for every fixing of the sequence $\{R_i\}$, T is a palette-alternating tree code with distance δ if and only if for every $x \in \{0,1\}^{\ell-1}$ and $i \in \{0, 1, \ldots, b-1\}$ it holds that $h_i(x) \geq \delta\ell$. Indeed, by definition, T is a palette-alternating tree code with distance δ if and only if for every vertex v, $\ell \geq 1$, and every distinct $x, y \in \{0,1\}^{\ell-1}$ it holds that $h_v(x,y) \geq \delta\ell$. However, by Claim 18, the random variable $h_v(x,y)$ is the same as the random variable $h_i(y-x)$ for $i = \mathsf{depth}(v) \bmod b$.

For $x \in \{0,1\}^{\ell-1}$ and $i \in \{0,1,\ldots,b-1\}$ denote by $E_i(x)$ the event $h_i(x) < \delta\ell$. Note that as $R_1 = 1$ and since $\mathsf{Tr}(1) = 1$ we have that $h_i(x) \geq 1$ for every x. Thus, for $|x| < \ell_0$ we have that

$$\frac{h(x)}{|x|+1} \geq \frac{1}{\ell_0}.$$

Therefore, in order to prove Theorem 15 it suffices to prove that

$$\Pr\left[\bigcup_{|x|\geq\ell_0}\bigcup_{i=0}^{b-1} E_i(x)\right] < 1.$$

Indeed, this will give a bound of $\min\left(\frac{1}{\ell_0}, \delta\right) = \Omega(\varepsilon \log^{-1}(1/\varepsilon))$ on the distance.

Fix $x \in \{0,1\}^{\ell-1}$ and $i \in \{0,1,\ldots,b-1\}$. Observe that $E_i(x)$ holds if and only if there exists a set $T \subseteq [\ell]$ of size $\lceil(1-\delta)\ell\rceil$ such that for every $k \in T$, $a_i(x,k) = 0$. For ease of readability we ignore the ceiling in the calculations below. Recall that $a_i(x,1),\ldots,a_i(x,\ell)$ are independent. Further, $1 - \frac{1}{b}$ fraction of them are uniform over $\mathbb{F}_2$ whereas the remaining $\frac{1}{b}$ fraction are uniform over $\mathbb{F}_4$. Note that by our choice of parameters, $\delta < 1/b$. Thus, for any $\gamma \geq 0$ and a fixed T, we have that

$$\begin{aligned}\Pr\left[\forall k \in T\ a_i(x,k) = 0\right] &\leq 2^{-\left(1-\frac{1}{b}-\gamma\right)\ell} 4^{-\left(\frac{1}{b}-\delta+\gamma\right)\ell} \\ &\leq 2^{-\left(1-\frac{1}{b}\right)\ell} 4^{-\left(\frac{1}{b}-\delta\right)\ell} \\ &= 2^{-\left(1+\frac{1}{b}-2\delta\right)\ell}.\end{aligned}$$

By taking the union bound over the choice of T, and using Lemma 7, we get that

$$\begin{aligned}\Pr[E_i(x)] &\leq \binom{\ell}{\lceil(1-\delta)\ell\rceil} 2^{-\left(1+\frac{1}{b}-2\delta\right)\ell} \\ &\leq 2^{-\left(1+\frac{1}{b}-2\delta-H(\delta)\right)\ell}.\end{aligned}$$

By the union bound,

$$\begin{aligned}\Pr\left[\bigcup_{|x|\geq\ell_0}\bigcup_{i=0}^{b-1} E_i(x)\right] &\leq \sum_{|x|\geq\ell_0}\sum_{i=0}^{b-1}\Pr[E_i(x)] \qquad (5)\\ &\leq b\cdot\sum_{\ell=\ell_0}^{\infty} 2^{\ell-1}\cdot 2^{-\left(1+\frac{1}{b}-2\delta-H(\delta)\right)\ell} \\ &= \frac{b}{2}\cdot\sum_{\ell=\ell_0}^{\infty} 2^{\left(H(\delta)+2\delta-\frac{1}{b}\right)\ell}.\end{aligned}$$

By taking c_1 sufficiently small and using Lemma 8, we get that $H(\delta) + 2\delta - 1/b \leq -\varepsilon/3$. Therefore, Equation (5) is bounded above by

$$\begin{aligned}b\cdot\sum_{\ell=\ell_0}^{\infty} 2^{-\varepsilon\ell/3} &= b\cdot\frac{2^{-\varepsilon\ell_0/3}}{1-2^{-\varepsilon/3}} \\ &\leq \frac{b\varepsilon^4}{1-2^{-\varepsilon/3}} \\ &\leq \frac{2\varepsilon^3}{1-2^{-\varepsilon/3}},\end{aligned}$$

where the penultimate inequality follows by our choice of ℓ_0 and the last inequality follows since $b = \lceil 1/\varepsilon\rceil$. One can verify that the above is strictly bounded by 1 for any $\varepsilon < 1/3$. ◀

5 The Interactive Coding Scheme

In this section we prove Theorem 6. In the first section we set up the framework over which our coding scheme will be defined. In Section 5.2 we present our coding scheme and Sections 5.3, 5.4 contain the analysis.

5.1 Setting up the framework

Round types

Throughout the scheme Alice and Bob send information in an alternating manner. More precisely, at even rounds Alice would decide on a bit to be sent and at odd rounds, Bob will decide what bit to send. Let $t \geq 0$. If t is even we say that it is an *Alice's round* and otherwise it is a *Bob's round*.

Epochs

We further partition the rounds as follows. Let c be a parameter to be set later on. The protocol is divided to *epochs* where each epoch consists of $2c+2$ rounds. The first epoch starts from round 0 to round $2c+1$ and is denoted by $e_0 = [0, 2c+2)$. The second epoch is denoted by $e_1 = [2c+2, 4c+4)$ and, generally, the k'th epoch consists of rounds $[k(2c+2), (k+1)(2c+2))$. Let t be an Alice's round and consider $m = t \bmod (2c+2)$. If $m = 2c$, then t is referred to as *Alice's bit sync round*, and otherwise, t is called an *Alice's edge round*. Similarly, for t a Bob's round, let $m = t \bmod (2c+2)$. If $m = 2c+1$, then t is a *Bob's bit sync round*. Otherwise, t is called *Bob's edge round*.

We denote by $\mathsf{edges}(e)$ the sequence of $2c$ bits sent throughout the edge rounds during epoch e, and define $\mathsf{sync}_A(e), \mathsf{sync}_B(e)$ the bits sent by Alice and Bob during their sync rounds, respectively.

Rewinding mechanism

As the adversary introduce some fraction of errors, the coding scheme should incorporate a "regret mechanism" using which the parties can revert back parts of the already exchanged messages. To formalize that, we will make use of the pair of functions

$$\mathsf{rewind} : \{S, X\}^* \to \{S, X\}^*,$$
$$\mathsf{survive} : \{S, R, X\}^* \to \{S, X\}^*,$$

which are defined as follows. Let $n \geq 1$. We define $\mathsf{rewind}(X^n) = X^n$. Let $v \in \{S, X\}^n \setminus \{X^n\}$ and denote $i \in [n]$ the largest index such that $v_i = S$. Then,

$$\mathsf{rewind}(v)_j = \begin{cases} v_j & j \neq i; \\ X & j = i. \end{cases}$$

We define the function $\mathsf{survive}$ recursively as follows. Let $v \in \{S, R, X\}^n$,

$$\mathsf{survive}(v) = \begin{cases} \mathsf{rewind}(\mathsf{survive}(v_0, \ldots, v_{n-1})) \circ X & v_n = R; \\ \mathsf{survive}(v_0, \ldots, v_{n-1}) \circ v_n & v_n \neq R. \end{cases}$$

Decoding the pointer jumping path

We describe now how to decode a rooted path in the pointer jumping game from the bits that were sent during a sequence of epochs. To formalize that, we define the function PJPath that given a sequence of epochs $(e_0, \ldots, e_n)$, computes a rooted path in the depth-n tree T as follows. Define $h : e \to \{S, R\}$ by

$$h(e) = R \quad \iff \quad \mathsf{sync}_A(e) \vee \mathsf{sync}_B(e) = 1$$

where an epoch is initialized with $\mathsf{sync}_A(e) = \mathsf{sync}_B(e) = 0$. Denote $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and let $i_1 < \cdots < i_\ell$ be the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. Finally, set

$$\mathsf{PJPath}(e_0, \ldots, e_n) = \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})).$$

where path is defined in the preliminaries.

▷ Claim 19. Let $e_0, \ldots, e_{n+1}$ be a sequence of epochs such that $\mathsf{sync}_A(e_{n+1}) \vee \mathsf{sync}_B(e_{n+1}) = 0$, then

$$v(\mathsf{PJPath}(e_0, \ldots, e_n)) = \mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_{n+1})), 2c).$$

If on the other hand $\mathsf{sync}_A(e_{n+1}) \vee \mathsf{sync}_B(e_{n+1}) = 1$, then

$$\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1})).$$

Proof. For the first direction of the claim, note that as $h(e_{n+1}) = S$ it follows that

$$\mathsf{survive}(h(e_0), \ldots, h(e_{n+1})) = \mathsf{survive}(h(e_0), \ldots, h(e_n)) \circ S.$$

Let $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and $0 \leq i_1 < \cdots < i_\ell \leq n$ where $\ell \geq 0$, the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. Thus, the set of indices that corresponds to an S symbol in $\mathsf{survive}(h(e_0), \ldots, h(e_n)) \circ S$ is exactly $\{i_1, \ldots, i_\ell, n+1\}$. Hence,

$$\begin{aligned} \mathsf{PJPath}(e_0, \ldots, e_n) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})); \\ \mathsf{PJPath}(e_0, \ldots, e_{n+1}) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell}) \circ \mathsf{edges}(e_{n+1})), \end{aligned}$$

and so $\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1}))$.

For the other direction, by definition, $h(e_{n+1}) = R$ and so

$$\mathsf{survive}(h(e_0), \ldots, h(e_{n+1})) = \mathsf{rewind}(\mathsf{survive}(h(e_0), \ldots, h(e_n))) \circ X.$$

Let $(m_0, \ldots, m_n) = \mathsf{survive}(h(e_0), \ldots, h(e_n))$ and $i_1 < \cdots < i_\ell$ the indices such that $m_{i_1} = \cdots = m_{i_\ell} = S$. By the definition of the rewind function, if $\ell > 0$ then the indices $i_1, \ldots, i_{\ell-1}$ correspond to an S symbol in $\mathsf{rewind}(\mathsf{survive}(h(e_0), \ldots, h(e_n)))$. Thus,

$$\begin{aligned} \mathsf{PJPath}(e_0, \ldots, e_n) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_\ell})); \\ \mathsf{PJPath}(e_0, \ldots, e_{n+1}) &= \mathsf{path}(\mathsf{edges}(e_{i_1}) \circ \cdots \circ \mathsf{edges}(e_{i_{\ell-1}})). \end{aligned}$$

Therefore, $\mathsf{ancestor}(v(\mathsf{PJPath}(e_0, \ldots, e_n)), 2c) = v(\mathsf{PJPath}(e_0, \ldots, e_{n+1}))$ as stated. In the case that $\ell = 0$, recall that $\mathsf{root}(T) = \mathsf{ancestor}(\mathsf{root}(T), m)$ for all $m \in \mathbb{N}$ concluding the proof. ◁

Transcript notations

Let $\mathcal{TC}$ be the palette-alternating tree code from Theorem 15 set with distance parameter $\delta_{\mathcal{TC}}$ whose value will be set later on. Denote by $\mathsf{TCEnc}, \mathsf{TCDec}$ the encoding and decoding functions of $\mathcal{TC}$, respectively where we decode to minimize the distance from the received word to a codeword. At every round, one of the parties would decide on a bit to be sent. That bit is not sent over the channel as is but rather is encoded using a palette-alternating tree code. For an even integer $t \geq 0$ we denote by $(a_0, a_2, \ldots, a_t)$ those bits that Alice would "like" to send from round 0 until round t. As mentioned above, the actual symbols that Alice sends are obtained by encoding these bits using $\mathcal{TC}$. Similarly, for an odd $t \geq 1$ we denote $(b_1, b_3, \ldots, b_t)$ the bits Bob would like to send. For an even integer $t \geq 0$ we define $\tilde{a}(t) = (\tilde{a}(t)_0, \tilde{a}(t)_2, \ldots, \tilde{a}(t)_t)$ to be the bits that are decoded, via TCDec, given the received transmission to Bob at round t. Note that $\tilde{a}(t)_i$ may not equal $\tilde{a}(t')_i$ for distinct times t, t', and certainly may not equal a_i.

For an odd t, we define $r_A(t) = (a_0, \tilde{b}(t)_1, a_2, \tilde{b}(t)_3, \ldots, \tilde{b}(t)_t)$ and similarly for an even t, $r_B(t) = (\tilde{a}(t)_0, b_1, \tilde{a}(t)_2, b_3, \ldots, \tilde{a}(t)_t)$. We further define $r(t) = (a_0, b_1, a_2, \ldots, b_t)$ for odd t and $r(t) = (a_0, b_1, a_2, \ldots, a_t)$ for even t. Recall that for a given set of edges E', we defined $v(E')$ to be the unique vertex in T with largest depth that is reachable from the root using the edge set E'. We define

$$p_A(t) = \mathsf{PJPath}(r_A(t));$$
$$\gamma_A(t) = v(p_A(t));$$
$$\alpha_A(t) = v(p_A(t) \cap (E_A \cup Y)).$$

Similarly,

$$p_B(t) = \mathsf{PJPath}(r_B(t));$$
$$\gamma_B(t) = v(p_B(t));$$
$$\alpha_B(t) = v(p_B(t) \cap (E_B \cup X)).$$

5.2 The coding scheme

The coding scheme is composed of two parts. The first consists of R rounds and the second of additional $2\tau R$ rounds where τ is a parameter to be chosen later on. We turn to describe the first part of the scheme. The second part is described in Section 5.2.2.

5.2.1 Part 1 of the coding scheme

We present the scheme from Alice's point of view. The scheme from Bob's point of view can be easily inferred. As mentioned, Alice's algorithm is partitioned to epochs. At the first round of epoch $e_k = [k(2c+2), (k+1)(2c+2))$ Alice computes $v_A = \gamma_A(k(2c+2)-1)$. We will make sure to maintain the invariant that at odd times t, $\gamma_A(t) \in V_A$. In particular, $v_A \in V_A$. For each round type, Alice proceeds as follows:

5.2.1.1 Alice's edge round

Let t be an Alice's edge round, namely, t is an even integer with $t \not\equiv 2c \pmod{2c+2}$.

1. At the edge rounds, Alice maintains v_A in order to choose a_t which is the bit that she would like to send at round t. Alice sets $a_t \leftarrow \pi_A(v_A)$. This operation is well-defined as we will be making sure also to maintain the invariant that in Alice's edge rounds $v_A \in V_A$.
2. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$.
3. Update $v_A \leftarrow \mathsf{son}(v_A, a_t)$.

5.2.1.2 Bob's edge round

Let t be Bob's edge round, namely, t is odd with $t \not\equiv 2c+1 \pmod{2c+2}$. At this round Bob sent a bit to Alice who, in turn, proceeds by updating v_A as follows:

1. $v_A \leftarrow \mathsf{son}(v_A, \tilde{b}(t)_t)$, where, recall $\tilde{b}(t)$ is the bit-string Alice decoded from the received transcript at round t.

5.2.1.3 Alice's bit sync round

Let $t \equiv 2c \pmod{2c+2}$. Notice that $\alpha_A(t-1)$ is an ancestor of $\gamma_A(t-1)$. We consider the following cases according to $\alpha_A(t-1)$, $\gamma_A(t-1)$ locations:

1. If $\alpha_A(t-1) = \gamma_A(t-1)$, then
   a. $a_t \leftarrow 0$ (0 encodes "hold")
   b. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$
2. If $\alpha_A(t-1)$ is a strict ancestor of $\gamma_A(t-1)$ then
   a. $a_t \leftarrow 1$ (1 encodes "revert")
   b. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$

5.2.2 Part 2 of the coding scheme

Recall that the coding scheme is divided to two parts. We now present the second part which take place during rounds $[R, (1+2\tau)R]$. This part is not partitioned to epochs and we describe it per round. We define the function $\mathsf{counter}_A : V \to \mathbb{N}$ that is initialized to 0. Recall that n denotes the depth of the tree T. More precisely, our convention is that edges leaving vertices of depth larger than n always point to their left son.

5.2.2.1 Alice's edge round

Let t be an Alice's round, namely, t is an even integer.

1. Alice sets $a_t \leftarrow 0$.
2. Transmit $\mathsf{TCEnc}(a_0, a_2, \ldots, a_t)_{t/2}$.

5.2.2.2 Bob's edge round

Let t be a Bob's round, namely, t is odd. At this round, Bob sent a bit to Alice who, in turn, proceeds by updating $\mathsf{counter}_A$ as follows:

1. Alice computes $\gamma_A(t)$.
2. If $\mathsf{depth}(\gamma_A(t)) \geq n$, denote by v the unique ancestor of $\gamma_A(t)$ of depth n. Alice sets $\mathsf{counter}_A(v) = \mathsf{counter}_A(v) + 1$.

5.2.2.3 Final round

Alice returns the vertex v that maximizes $\mathsf{counter}_A(v)$. The analysis will show that such vertex exists and is unique.

5.2.2.4 Remark

Note that in most rounds, TCEnc outputs a symbol in $\mathbb{F}_2$ which corresponds to a single bit transmitted. At the rounds in which the symbol is an $\mathbb{F}_4$-element, we send the information in two rounds and the round of the other party in between is ignored. For simplicity, we make this issue transparent to the coding scheme.

5.3 A simpler analysis with sub-optimal rate

In this section we prove that the coding scheme above, when set with suitable parameters $\delta_{\mathcal{TC}}$, c, τ, has rate $1-\tilde{O}(\sqrt[3]{\varepsilon})$. Many of the ideas and results used in this section will be used for the proof of Theorem 6, to be presented in Section 5.4, which requires additional ideas. We assume R is an integral multiple of $2c+2$ and let k be the number of epochs, namely, $R=(2c+2)k$.

Good rounds

We say that $t \in [R]$ is *good* if the decoding at round t succeeds. More precisely, when t is even, round t is good if

$$(a_0, a_2, \ldots, a_t) = (\tilde{a}(t)_0, \tilde{a}(t)_2, \ldots, \tilde{a}(t)_t).$$

Similarly, an odd t is good if

$$(b_1, b_3, \ldots, b_t) = (\tilde{b}(t)_1, \tilde{b}(t)_3, \ldots, \tilde{b}(t)_t).$$

We make use of the following lemma proved by Schulman [29] (see also Section 2.1.3 in [12]).

▶ **Lemma 20** ([29]). *Let $\mathcal{TC}$ be a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$. Assume the channel has at most ε-fraction errors. Then, at most*

$$\mu \triangleq 2\varepsilon/\delta_{\mathcal{TC}}$$

fraction of rounds are bad.

Good epochs

We say that epoch $e=[t, t+2c+2)$ is *good* if each round $r \in [t-1, t+2c]$ is good and otherwise we call it *bad*. Note that for an epoch to be good we require that the last round of the previous epoch is good though do not require the last round of the current epoch to be good. Note further that at least $1-(2c+2)\mu$ fraction of the epochs are good. We wish to define vertices analog to $\gamma_A(t), \alpha_A(t)$ and $\gamma_B(t), \beta_B(t)$ that are defined according to what was actually sent by the parties in the first t rounds rather than according to what was received. Formally, define

$$\gamma(t) = v(\mathsf{PJPath}(r(t)));$$
$$\alpha(t) = v(\mathsf{PJPath}(r(t)) \cap (E_A \cup Y));$$
$$\beta(t) = v(\mathsf{PJPath}(r(t)) \cap (E_B \cup X)),$$

where recall that $r(t)$ is defined in the paragraph presenting our transcript notations in Section 5.1. Let $v(t)$ be the least common ancestor of $\alpha(t), \beta(t)$ in T. Observe that $v(t)$ is equal to either $\alpha(t)$ or $\beta(t)$ and in particular is an ancestor of $\gamma(t)$.

▷ Claim 21. Let $e=[t, t+2c+2)$ be a good epoch such that $v(t-1) \neq \gamma(t-1)$. Then,

$$\mathsf{sync}_A(e) = 1 \vee \mathsf{sync}_B(e) = 1.$$

Proof. Observe that the hypothesis of the claim implies that $v(t-1)$ is a strict ancestor of $\gamma(t-1)$. As $\gamma(t-1)$ is a strict ancestor of $\gamma(t+2c-1)$ and since $v(t+2c-1) = v(t-1)$ it follows that $v(t+2c-1) \neq \gamma(t+2c-1)$. As round $t+2c-1$ is good, it holds that

$$\gamma_A(t+2c-1) = \gamma(t+2c-1) = \gamma_B(t+2c-1).$$

Furthermore, by the definition of α_A and β_B it follows that

$$\alpha_A(t+2c-1) = \alpha(t+2c-1);$$
$$\beta_B(t+2c-1) = \beta(t+2c-1).$$

Thus, as $v(t+2c-1) \neq \gamma(t+2c-1)$, at least one of the following holds $\alpha_A(t+2c-1) \neq \gamma_A(t+2c-1)$ or $\beta_B(t+2c-1) \neq \gamma_B(t+2c-1)$. Hence, at least one of the parties set its sync bit to 1. ◁

Short-split epochs

We define the indicator function

$$\mathsf{nearAncestor}(v(t), \gamma(t)) = \begin{cases} 1 & \mathsf{dist}(v(t), \gamma(t)) \in (0, 2c); \\ 0 & \text{otherwise.} \end{cases}$$

A good epoch $e = [t, t+2c+2)$ is called a *short-split* epoch if

$$\mathsf{nearAncestor}(v(t-1), \gamma(t-1)) = 1.$$

▷ Claim 22. The number of short-split epochs is bounded above by the number of bad epochs.

Proof. Consider any two short-split epochs $e = [t, t+2c+2), e' = [t', t'+2c+2)$ with $t < t'$. Since e is a short-split epoch, then e is good and also $v(t-1) \neq \gamma(t-1)$. By Claim 21, Alice or Bob set their sync bit to 1. By Claim 19 it holds that $\gamma(t+2c+1) = \mathsf{ancestor}(\gamma(t-1), 2c)$. Observe that as $d < 2c$, this results in $v(t+2c+1) = \gamma(t+2c+1)$.

Observe further that, until the arrival of a bad epoch, at epoch $e'' = [t'', t''+2c+2)$ we have that $v(t''-1) = \gamma(t''-1)$. Since e' is a short-split epoch, $v(t'-1) \neq \gamma(t'-1)$. It then follows that there exists a bad epoch preceding e'. Since the first epoch is not short-split, the claim follows. ◁

Potential function for the progress

For an integer $i \geq 0$ and $t = (2c+2)i - 1$, consider the following potential function

$$\Phi(t) = 2\mathsf{depth}(v(t)) - \mathsf{depth}(\gamma(t)).$$

Recall that $\mathsf{depth}(\gamma(t)) \geq \mathsf{depth}(v(t))$ and so when $\Phi(t) \geq n$ it holds that $\mathsf{depth}(v(t)) \geq n$.

▷ Claim 23. If $e = [t, t+2c+2)$ is a good epoch that is not short-split, then $\Phi(t+2c+1) = \Phi(t-1) + 2c$. Otherwise, $\Phi(t+2c+1) \geq \Phi(t-1) - 6c$.

Proof. By Claim 19, $\mathsf{dist}(\gamma(t+2c+1), \gamma(t-1)) \leq 2c$. Observe that by Claim 19 and by the definition of v it follows that $\mathsf{dist}(v(t+2c+1), v(t-1)) \leq 2c$ as well. Thus, the assertion $\Phi(t+2c+1) \geq \Phi(t-1) - 6c$ follows. Let then e be a good epoch that is not short-split, and consider the following cases:

1. First assume that $v(t-1) = \gamma(t-1)$. As epoch e is good, it follows that at the edge rounds, Alice and bob extends the same (correct) path, and so

$$v(t+2c-1) = \gamma(t+2c-1). \tag{6}$$

Since round $t+2c-1$ is good,

$$\gamma_A(t+2c-1) = \gamma(t+2c-1) = \gamma_B(t+2c-1).$$

The above equation together with Equation (6) implies that

$$\alpha_A(t+2c-1) = \gamma_A(t+2c-1);$$
$$\beta_B(t+2c-1) = \gamma_B(t+2c-1).$$

By the algorithm both Alice and Bob sets their sync bit to 0, namely, $\mathsf{sync}_A(e) = \mathsf{sync}_B(e) = 0$. Thus, together with Claim 19 and Equation (6),

$$\mathsf{depth}(\gamma(t+2c+1)) = \mathsf{depth}(\gamma(t-1)) + 2c,$$
$$\mathsf{depth}(v(t+2c+1)) = \mathsf{depth}(v(t-1)) + 2c,$$

and it follows that $\Phi(t+2c+1) = \Phi(t-1) + 2c$.

2. Consider now the case that $v(t-1)$ is a strict ancestor of $\gamma(t-1)$. By Claim 21 it follows that $\mathsf{sync}_A(e) = 1$ or $\mathsf{sync}_B(e) = 1$. Then, by Claim 19 it holds that $\mathsf{ancestor}(\gamma(t-1), 2c) = \gamma(t+2c+1)$. Since e is not a short-split epoch, $v(t-1)$ is an ancestor of $\gamma(t+2c+1)$, and by the definition of v this implies $v(t+2c+1) = v(t-1)$. Thus, it holds that

$$\mathsf{depth}(\gamma(t+2c+1)) = \mathsf{depth}(\gamma(t-1)) - 2c,$$
$$\mathsf{depth}(v(t+2c+1)) = \mathsf{depth}(v(t-1)),$$

and

$$\Phi(t+2c+1) = \Phi(t-1) + 2c,$$

concluding the proof. ◁

By Claim 22, there are at least $(1-2(2c+2)\mu)k$ good epochs which are not short-split. By Claim 23, Φ increases by at least $2c$ in every such epoch. In the remaining epochs, Φ decreases by at most $6c$. Since $\Phi(-1) = 0$ we have that

$$\begin{aligned}\Phi(R) &\geq ((1-2(2c+2)\mu)2c + 2(2c+2)\mu \cdot (-6c))k \\ &= (1-8(2c+2)\mu) \cdot 2ck \\ &= \left(1 - \left(\frac{4}{2c+2} + 16c\mu\right)\right) R.\end{aligned}$$

By setting c to be an integer $c = \Theta(1/\sqrt{\mu})$, we get $\Phi(R) = \left(1 - \Theta(\sqrt{\mu})\right) R$. Now setting $R = (1+\Theta(\sqrt{\mu}))n$, the first part of the scheme assures that $\mathsf{depth}(v(R)) \geq n$.

Analysis of part 2 of the scheme

Let v_{pj} be the unique ancestor of $v(R)$ of depth n in T, it is well defined as the analysis of Part 1 of the scheme assures that $\mathsf{depth}(v(R)) \geq n$. Recall that the second part of the scheme contains $2\tau R$ rounds. By Lemma 20, there are at most

$$(1+2\tau)\mu R$$

bad rounds. By the algorithm, for every good odd round in Part 2 of the scheme, Alice increases $\mathsf{counter}_A(v_{\mathrm{pj}})$ by 1. Observe that in the final round Alice returns the vertex that maximizes $\mathsf{counter}_A$ and so the assertion $\mathsf{counter}_A(v_{\mathrm{pj}}) > (\tau R)/2$ implies that the simulation will terminates successfully. By setting $\tau = 6\mu$, we get

$$\frac{\tau}{2}R > (1+2\tau)\mu R$$

which guarantees that the majority of both Alice's and Bob's rounds in the second part of the coding scheme are good. This concludes the proof of the theorem.

Calculating the rate

At each round of the simulation, a palette-alternating tree code symbol is sent instead of a single bit. By Theorem 4 $\mathcal{TC}$ has rate $1 - O(\delta_{\mathcal{TC}} \log(1/\delta_{\mathcal{TC}}))$. Setting $\delta_{\mathcal{TC}} = \sqrt[3]{\varepsilon}$, we get that the simulation uses

$$\left(1 + O\left(\sqrt{\frac{\varepsilon}{\delta_{\mathcal{TC}}}}\right)\right)\left(1 + O\left(\delta_{\mathcal{TC}} \log\left(\frac{1}{\delta_{\mathcal{TC}}}\right)\right)\right) n = \left(1 + O\left(\sqrt[3]{\varepsilon} \log\left(\frac{1}{\varepsilon}\right)\right)\right) n$$

bits. Thus, the coding scheme rate is $1 - \tilde{O}(\sqrt[3]{\varepsilon})$ as stated.

5.4 Optimal analysis

In this section we prove Theorem 6. We make use of the same coding scheme analyzed in Section 5.3. The improved analysis follows by applying a more delicate analysis of the bad rounds locations as a function of the errors introduced by the adversary.

Let $\mathcal{TC}$ be a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$. Denote by $\mathcal{E} = \{e_1, \ldots, e_{\varepsilon R}\}$ the set of rounds at which the adversary has introduced errors, where $0 \le e_1 < \cdots < e_{\varepsilon R} \le R$. A set of consecutive errors $C = \{e_j, \ldots, e_{j+r-1}\}$ is called a *cluster of errors* (with respect to $\mathcal{TC}$ or more precisely $\delta_{\mathcal{TC}}$) if

$$\forall \ell \in [r-1] \quad e_{j+\ell} - e_j \le \frac{2\ell}{\delta_{\mathcal{TC}}}\,.$$

We define the *cluster interval* of C by $\mathcal{I}(C) = [e_j, e_j + 2r/\delta_{\mathcal{TC}}]$. We denote by $\mathcal{C}$ the set of all clusters (with respect to $\mathcal{E}$).

▷ Claim 24. Let $C_1, C_2 \in \mathcal{C}$ with $C_1 \subseteq C_2$. Then, $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$.

Proof. Let $C_1 = \{e_i, \ldots, e_j\}, C_2 = \{e_m, \ldots, e_k\}$ with $m \le i \le j \le k$. By definition, it holds that $\mathcal{I}(C_1) = [e_i, e_i + 2(j-i+1)/\delta_{\mathcal{TC}}]\,, \mathcal{I}(C_2) = [e_m, e_m + 2(k-m+1)/\delta_{\mathcal{TC}}]$. As $e_i \in C_2$ we have that $e_i \le e_m + 2(i-m)/\delta_{\mathcal{TC}}$, and so

$$\begin{aligned} e_i + \frac{2(j-i+1)}{\delta_{\mathcal{TC}}} &\le e_m + \frac{2(j-m+1)}{\delta_{\mathcal{TC}}} \\ &\le e_m + \frac{2(k-m+1)}{\delta_{\mathcal{TC}}}\,, \end{aligned}$$

which, together with $e_m \le e_i$, concludes the proof. ◁

We will be interested to study clusters on sub-intervals of $[0, R]$ and in particular we wish to consider clusters that are, in a sense, maximal in the sub-interval. To formalize that, let $[a, b]$ be a sub-interval of $[0, R]$. A cluster $C \in \mathcal{C}$ with $C \subseteq [a, b]$ is called $[a, b]$-*maximal* if for

every cluster $C' \subseteq [a,b]$ such that $C \subseteq C'$ it holds that $C' = C$. A $[0,R]$-maximal cluster is simply called maximal. We denote by $\mathcal{M}_{[a,b]}$ the set of all $[a,b]$-maximal clusters, and by $\mathcal{M}$ the set of all maximal clusters.

▷ Claim 25. Every $C_1, C_2 \in \mathcal{M}_{[a,b]}$ are either equal or disjoint.

The proof of the above claim is straightforward. Indeed, by adapting the proof of Claim 24, if false $C_1 \cup C_2 \in \mathcal{C}$ in contradiction to the maximality.

▷ Claim 26. Let $C_1, C_2 \in \mathcal{M}_{[a,b]}$ distinct. Then, $\mathcal{I}(C_1) \cap \mathcal{I}(C_2) = \emptyset$.

Proof. By Claim 25 we have that $C_1 \cap C_2 = \emptyset$, and so we may denote $C_1 = \{e_i, \ldots, e_{i+j}\}$, $C_2 = \{e_m, \ldots, e_{m+n}\}$ with $i+j < m$. Assume toward a contradiction that $\mathcal{I}(C_1) \cap \mathcal{I}(C_2) \neq \emptyset$, and so $e_m \in [e_i, e_i + 2(j+1)/\delta_{\mathcal{TC}})$. Observe that this would imply that $C' = \{e_i, \ldots, e_m\} \in \mathcal{C}$, which together with $C' \subseteq [a,b]$, stands in contradiction to $C_1 \in \mathcal{M}_{[a,b]}$. ◁

▷ Claim 27.

$$\left| \bigcup_{M \in \mathcal{M}} \mathcal{I}(M) \right| \leq \frac{2\varepsilon R}{\delta_{\mathcal{TC}}} .$$

Proof. By Claim 26, and since $|\mathcal{I}(C)| = 2|C|/\delta_{\mathcal{TC}}$ for every $C \in \mathcal{C}$,

$$\left| \bigcup_{M \in \mathcal{M}} \mathcal{I}(M) \right| = \sum_{M \in \mathcal{M}} \frac{2|M|}{\delta_{\mathcal{TC}}} .$$

As all maximal clusters are disjoint (Claim 25),

$$\sum_{M \in \mathcal{M}} |M| \leq \varepsilon R,$$

which concludes the proof. ◁

▶ **Lemma 28.** *Let $r \in [0,R]$. If $r \notin \bigcup_{C \in \mathcal{C}} \mathcal{I}(C)$ then r is a good round.*

Proof. Denote by σ_t the palette-alternating tree code symbol that is sent at round t, and let $\tilde{\sigma}_t$ be the received symbol at that round. Denote by $(\mu_1, \ldots, \mu_r)$ the path on $\mathcal{TC}$ that corresponds to the decoded codeword . Assume toward a contradiction that r is bad, namely, $(\sigma_1, \ldots, \sigma_r) \neq (\mu_1, \ldots, \mu_r)$. Let $\ell \in [r]$ be the largest integer such that $\mu_{r-\ell} \neq \sigma_{r-\ell}$. As $\mathsf{TCDec}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_r)$ returns the codeword that minimizes the distance, and since $\mu_i = \sigma_i$ for every $i < r - \ell$, we have that

$$\Delta((\mu_{r-\ell}, \ldots, \mu_r), (\tilde{\sigma}_{r-\ell}, \ldots, \tilde{\sigma}_r)) \leq \Delta((\tilde{\sigma}_{r-\ell}, \ldots, \tilde{\sigma}_r), (\sigma_{r-\ell}, \ldots, \sigma_r)). \tag{7}$$

Since $\mathcal{TC}$ is a palette-alternating tree code with distance $\delta_{\mathcal{TC}}$,

$$\Delta((\mu_{r-\ell}, \ldots, \mu_r), (\sigma_{r-\ell}, \ldots, \sigma_r)) \geq (\ell+1)\delta_{\mathcal{TC}} . \tag{8}$$

Let $I = \mathcal{E} \cap [r-\ell, r]$, i.e the set of all rounds i such that $\sigma_i \neq \tilde{\sigma}_i$ in the interval $[r-\ell, r]$. Denote $|I| = k$. As $\mathcal{M}_{[r-\ell,r]} \subseteq \mathcal{C}$ and by the hypothesis of the lemma, it follows that

$$r \notin \bigcup_{C \in \mathcal{M}_{[r-\ell,r]}} \mathcal{I}(C).$$

Observe that

$$\bigcup_{C\in\mathcal{M}_{[r-\ell,r]}} \mathcal{I}(C) \subseteq [r-\ell, r).$$

Claim 26 states that the intervals of any two maximal clusters are disjoint, hence,

$$\sum_{C\in\mathcal{M}_{[r-\ell,r]}} |\mathcal{I}(C)| \leq \ell.$$

As $|\mathcal{I}(C)| = 2|C|/\delta_{\mathcal{TC}}$ for every $C \in \mathcal{C}$ and since $\mathcal{M}_{[r-\ell,r]}$ forms a partition of I, it follows that

$$\sum_{C\in\mathcal{M}_{[r-\ell,r]}} |\mathcal{I}(C)| = \frac{2k}{\delta_{\mathcal{TC}}}.$$

By the above two equations, we have that $\ell \geq 2k/\delta_{\mathcal{TC}}$. Substituting to Equation (8), we have that $\Delta((\mu_{r-\ell},\ldots,\mu_r),(\sigma_{r-\ell},\ldots,\sigma_r)) > 2k$. Since $\Delta((\tilde{\sigma}_{r-\ell},\ldots,\tilde{\sigma}_r),(\sigma_{r-\ell},\ldots,\sigma_r)) = k$, we have that $\Delta((\mu_{r-\ell},\ldots,\mu_r),(\tilde{\sigma}_{r-\ell},\ldots,\tilde{\sigma}_r)) > k$ in contradiction to Equation (7). ◀

Using the above, we obtain a better bound on the fraction of bad epochs compared to the bound $O(\varepsilon c/\delta_{\mathcal{TC}})$ established in Section 5.3.

▶ **Lemma 29.** *At most* $(4\varepsilon/\delta_{\mathcal{TC}} + \varepsilon(2c+2))$ *fraction of the epochs are bad.*

Proof. Observe that for every $C \in \mathcal{C}$ there exists a maximal cluster $M \in \mathcal{M}$ such that $C \subseteq M$. By Claim 24 it then follows that $\mathcal{I}(C) \subseteq \mathcal{I}(M)$, and so

$$\bigcup_{C\in\mathcal{C}} \mathcal{I}(C) = \bigcup_{M\in\mathcal{M}} \mathcal{I}(M).$$

Claim 27 implies that

$$\sum_{M\in\mathcal{M}} |\mathcal{I}(M)| \leq \frac{2\varepsilon R}{\delta_{\mathcal{TC}}}. \tag{9}$$

Notice that each cluster M intersect with at most $\lceil |\mathcal{I}(M)|/(c+1) \rceil$ bad epochs. By Claim 28, if $r \notin \mathcal{I}(M)$ for every $M \in \mathcal{M}$ then r is good. Hence there are at most

$$\sum_{M\in\mathcal{M}} \left\lceil \frac{|\mathcal{I}(M)|}{c+1} \right\rceil$$

bad epochs. Since the maximal clusters form a partition of $\mathcal{E}$, it follows that $|\mathcal{M}| \leq \varepsilon R$. This, together with Equation (9) yields

$$\begin{aligned}\sum_{M\in\mathcal{M}} \left\lceil \frac{|\mathcal{I}(M)|}{c+1} \right\rceil &\leq \varepsilon R + \sum_{M\in\mathcal{M}} \frac{|\mathcal{I}(M)|}{c+1} \\ &\leq \varepsilon R + \frac{2\varepsilon R}{\delta_{\mathcal{TC}}(c+1)} \\ &= \left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}} + \varepsilon(2c+2)\right) k.\end{aligned}$$

So, at most $(4\varepsilon/\delta_{\mathcal{TC}} + \varepsilon(2c+2))$ fraction of the epochs are bad as stated. ◀

By Claim 22 and Lemma 29 there are at least $(1-2(4\varepsilon/\delta_{\mathcal{TC}}+\varepsilon(2c+2)))\,k$ good epochs that are not short-split. By Claim 23, in each such epoch, Φ increases by at least $2c$. In the remaining epochs, Φ decreases by at most $6c$. Since $\Phi(-1)=0$ we have that

$$\begin{aligned}\Phi(R) &\geq \left(\left(1-2\left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}}+\varepsilon(2c+2)\right)\right)2c+2\left(\frac{4\varepsilon}{\delta_{\mathcal{TC}}}+\varepsilon(2c+2)\right)\cdot(-6c)\right)k\\ &= \left(1-\frac{32\varepsilon}{\delta_{\mathcal{TC}}}-8\varepsilon(2c+2)\right)\cdot 2ck\\ &\geq \left(1-\frac{2}{c}-\frac{32\varepsilon}{\delta_{\mathcal{TC}}}-16c\varepsilon\right)R.\end{aligned}$$

By setting c to be an integer with $c=\Theta(\frac{1}{\sqrt{\varepsilon}})$ and $\delta_{\mathcal{TC}}=\sqrt{\varepsilon/\log(1/\varepsilon)}$, we get that

$$\Phi(R)\geq\left(1-\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})\right)R.$$

By setting $R=(1+\Theta(\sqrt{\varepsilon\log(1/\varepsilon)}))n$, and since $\mathcal{TC}$ has rate $1-\Theta(\delta_{\mathcal{TC}}\log(1/\delta_{\mathcal{TC}}))=1-\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})$, the first part of the scheme assures that $\mathsf{depth}(v(R))\geq n$. Similarly to the analysis of Part 2 from Section 5.3, by setting $\tau=\Theta(\mu)=\Theta(\sqrt{\varepsilon\log(1/\varepsilon)})$, Theorem 6 follows.

References

1. S. Agrawal, R. Gelles, and A. Sahai. Adaptive protocols for interactive communication. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 595–599, 2016.
2. N. Alon, M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler. Reliable communication over highly connected noisy networks. In *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 165–173, 2016.
3. Z. Brakerski and Y. T. Kalai. Efficient interactive coding against adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–166, 2012.
4. Z. Brakerski, Y. T. Kalai, and M. Naor. Fast interactive coding against adversarial noise. *Journal of the ACM (JACM)*, 61(6):35:1–30, 2014.
5. Z. Brakerski and M. Naor. Fast algorithms for interactive coding. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 443–456, 2013.
6. M. Braverman. Towards deterministic tree code constructions. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 161–167, 2012.
7. M. Braverman and K. Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 236–245, 2014.
8. M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017.
9. M. Braverman and A. Rao. Towards coding for maximum errors in interactive communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 159–166, 2011.
10. M. Braverman and A. Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, 2014.
11. G. Cohen, B. Haeupler, and L. J. Schulman. Explicit binary tree codes with polylogarithmic size alphabet. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 535–544. ACM, 2018.
12. R. Gelles. Coding for interactive communication: A survey. *Foundations and Trends in Theoretical Computer Science*, 13(1-2):1–157, 2017. doi:10.1561/0400000079.
13. R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson. Towards optimal deterministic coding for interactive communication, 2016.

14 R. Gelles, A. Moitra, and A. Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.

15 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 768–777. IEEE, 2011.

16 M. Ghaffari and B. Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, 2014.

17 M. Ghaffari, B. Haeupler, and M. Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 794–803, 2014.

18 B. Haeupler. Interactive Channel Capacity Revisited. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 226–235, 2014.

19 A. Jain, Y. T. Kalai, and A. B. Lewko. Interactive coding for multiparty protocols. In *Proc. ACM Innovations in Theoretical Computer Science Conference (ITCS)*, pages 1–10, 2015.

20 G. Kol and R. Raz. Interactive channel capacity. In *STOC*, volume 13, pages 715–724, 2013.

21 E. Kushilevitz and N. Nisan. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

22 A. Lewko and E. Vitercik. Balancing communication for multi-party interactive coding. *CoRR*, abs/1503.06381, 2015. `arXiv:1503.06381`.

23 C. Moore and L. J. Schulman. Tree codes and a conjecture on exponential sums. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 145–154, 2014.

24 Anand Kumar Narayanan and Matthew Weidner. On decoding Cohen-Haeupler-Schulman tree codes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1337–1356. SIAM, 2020.

25 P. Pudlák. Linear tree codes and the problem of explicit constructions. *Linear Algebra and its Applications*, 490:124–144, 2016.

26 S. Rajagopalan and L. J. Schulman. A coding theorem for distributed computation. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 790–799, 1994.

27 A. Rao and A. Yehudayoff. Communication complexity (early draft), 2018.

28 L. J. Schulman. Communication on noisy channels: a coding theorem for computation. *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1992.

29 L. J. Schulman. Deterministic coding for interactive communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 747–756, 1993.

30 L. J. Schulman. Postscript of 21 September 2003 to Coding for Interactive Communication. http://users.cms.caltech.edu/∼schulman/Papers/intercodingpostscript.txt, 1994.

31 L. J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

32 A. A. Sherstov and P. Wu. Optimal interactive coding for insertions, deletions, and substitutions. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017.

Search Problems in Algebraic Complexity, GCT, and Hardness of Generators for Invariant Rings

Ankit Garg
Microsoft Research, Bangalore, India
garga@microsoft.com

Christian Ikenmeyer
University of Liverpool, UK
christian.ikenmeyer@liverpool.ac.uk

Visu Makam
Institute for Advanced Study, Princeton, NJ, USA
visu@ias.edu

Rafael Oliveira
University of Waterloo, Canada
rafael@uwaterloo.ca

Michael Walter
Korteweg-de Vries Institute for Mathematics, Institute for Theoretical Physics,
Institute for Logic, Language & Computation,
University of Amsterdam, The Netherlands
m.walter@uva.nl

Avi Wigderson
Institute for Advanced Study, Princeton, NJ, US
avi@ias.edu

Abstract

We consider the problem of computing succinct encodings of lists of generators for invariant rings for group actions. Mulmuley conjectured that there are always polynomial sized such encodings for invariant rings of $\mathrm{SL}_n(\mathbb{C})$-representations. We provide simple examples that disprove this conjecture (under standard complexity assumptions).

We develop a general framework, denoted *algebraic circuit search problems*, that captures many important problems in algebraic complexity and computational invariant theory. This framework encompasses various proof systems in proof complexity and some of the central problems in invariant theory as exposed by the Geometric Complexity Theory (GCT) program, including the aforementioned problem of computing succinct encodings for generators for invariant rings.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases generators for invariant rings, succinct encodings

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.12

Related Version `https://arxiv.org/abs/1910.01251v1`

Funding *Christian Ikenmeyer*: DFG grant IK 116/2-1.
Visu Makam: NSF grant No. DMS -1638352 and NSF grant No. CCF-1412958.
Michael Walter: NWO Veni grant 680-47-459.
Avi Wigderson: NSF grant No. CCF-1412958 and NSF grant CCF-1900460.

1 Introduction

In complexity theory, one often encounters problems that ask for an efficiently computable collection of functions/polynomials satisfying a certain property. Once we are faced with such problems, two natural questions are: how do we represent the property? And how do

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 12; pp. 12:1–12:17

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

we encode the required functions? The answer to these will depend on the context and use. We will first define the informal notion of an algebraic circuit search problem, and then give illustrative examples.

▶ **Definition 1** (Algebraic circuit search problems). *Given an input (of size n), construct an algebraic circuit in a complexity class $\mathcal{C}$ with $k(n)$-inputs and $m(n)$-outputs such that the polynomials they compute satisfy a desirable property $\mathcal{P}$.*

Let us illustrate this definition in the context of algebraic proof complexity: in Nullstellensatz-based proof systems, one is given a set of multivariate polynomials $g_1, \ldots, g_r$ over an algebraically closed field $\mathbb{F}$ and in variables $\mathbf{x} = (x_1, \ldots, x_n)$, and one wants to decide whether the system $g_1(\mathbf{x}) = g_2(\mathbf{x}) = \ldots = g_r(\mathbf{x}) = 0$ has a solution over $\mathbb{F}$. A fundamental result of Hilbert tells us that the system has no solution if and only if there is a set of polynomials $\{f_i\}_{i=1}^r$ such that $\sum_i f_i g_i = 1$. This brings us to the Ideal Proof System [22]:

- **Ideal Proof System (IPS)**: Given a collection of polynomials $g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_r(\mathbf{x})$, we ask to construct a polynomial sized circuit C with $n+r$ inputs. The desirable property $\mathcal{P}$ is that $C(x_1, \ldots, x_n, g_1(\mathbf{x}), \ldots, g_r(\mathbf{x})) = 1$ and that $C(x_1, \ldots, x_n, 0, \ldots, 0) = 0$. It is not so hard to see these conditions will give us a linear combination of the form $\sum_i f_i g_i = 1$ as required.

In [22] the authors show that super-polynomial lower bounds in this proof system imply algebraic circuit lower bounds (i.e., VP $\neq$ VNP), which remains a long standing open problem in complexity theory. Another important point to make is that an instance of 3-SAT, say ϕ, can be encoded as a collection of polynomials $\{g_i\}$ such that ϕ is satisfiable if and only if the $\{g_i\}$ have a common solution. In other words, ϕ is unsatisfiable if and only if $\exists$ polynomials f_i such that $\sum_i f_i g_i = 1$. This converts a co-NP complete problem (unsatisfiability of 3-SAT, called co-3-SAT) into an algebraic circuit search problem of the IPS form described above. The existence of a polynomial sized circuit as demanded by the IPS proof system would mean the existence of f_i with small circuits. But that would mean that co-3-SAT is in NP, thus proving NP = co-NP.

Other important examples of algebraic search problems in proof complexity (with different desirable properties) are the original Nullstellensatz proof system, Polynomial Calculus, and the Positivstellensatz[1] for sum of squares (SOS) proofs. For more on these systems we refer the reader to [29, Chapter 6].

▶ Remark 2. An analogous notion of a "boolean circuit search problem" can also be introduced in the boolean setting. Also here, important problems such as the construction of pseudorandom generators and the construction of extractors can be captured as boolean circuit search problems.

1.1 Geometric Complexity Theory

The GCT program was proposed by Mulmuley and Sohoni (see [32, 33]) as an approach (via representation theory and algebraic geometry) to the VP vs. VNP problem. While there have been some negative results[2] in recent years regarding the techniques one can use towards this program, these results do not disrupt the core framework of the GCT program. Instead, these results indicate the difficulty of the problem from the viewpoint of algebraic combinatorics,

[1] In this case our field is the real numbers, which is *not* algebraically closed.
[2] The results of Bürgisser, Ikenmeyer and Panova, which show that occurrence obstructions cannot give a super-polynomial lower bound on the determinantal complexity of the permanent polynomial (see [9]).

and have identified new directions of research in asymptotic algebraic combinatorics. In [31], Mulmuley views the VP vs. VNP problem through the lens of computational invariant theory, and identifies important and interesting problems in computational invariant theory that form a path towards resolving the VP vs. VNP problem. These include several conjectures, some of which fit into the framework of algebraic circuit search problems, and have important connections and consequences to problems in optimization, algebraic complexity, non-commutative computation, functional analysis and quantum information theory (see [19, 20, 6]). We therefore believe that a better understanding of algebraic circuit search problems will likely result in fundamental advances in the aforementioned areas. Some evidence for these conjectures has emerged over the past few years as they have been established for special cases (see, for example, [31, 19, 27, 17, 12, 13, 11]).

Let us briefly mention an important algebraic circuit search problem and one that will be central to this paper: given a group action, describe a set of generators for the invariant ring (we will elaborate on invariant theory in a subsequent section). Unfortunately, the number of generators for an invariant ring is usually exponential (in the input size of the description of the action). So, in order to get a computational handle on them, Mulmuley suggests in [31] that we should look for a *succinct encoding* (defined below in Definition 3) using some auxiliary variables. One amazing feature of such a succinct encoding is that it would immediately give efficient randomized algorithms for null cone membership and the orbit closure intersection problems which can then be derandomized in some cases (see, e.g., [17, 27, 13]). We will define these problems in a subsequent section, but here we are content to say that many important algorithmic problems such as graph isomorphism, bipartite matching, (non-commutative) rational identity testing, tensor scaling and a form of quantum entanglement distillation are all specific instances (or arise in the study) of null cone membership and orbit closure intersection problems.

Mulmuley conjectures ([31, Conjecture 5.3][3]) the existence of polynomial sized succinct encodings for generators of invariant rings. The main goal of this paper is to (conditionally) disprove this conjecture. More precisely we give an example of an invariant ring (for a torus action) where the existence of such a circuit would imply a polynomial time algorithm for the 3D-matching problem, which is well known to be NP-hard. We also give another example (where the group is $\mathrm{SL}_n(\mathbb{C})$) where the existence of such a circuit would imply VP $\neq$ VNP. Further, the nature of the latter example makes it clear that no simple modification of this conjecture can hold.

The rest of this section will proceed as follows. We first give a brief introduction to invariant theory. Then, we discuss the algebraic search problems of interest in computational invariant theory, followed by the precise statements of our main results. Finally, we discuss some open problems and future directions.

1.2 Invariant Theory

Invariant theory is the study of symmetries, captured by group actions on vector spaces (more generally, algebraic varieties), by focusing on the functions (usually, polynomials) that are left *invariant* under these actions. It is a rich mathematical field in which computational methods are sought and well developed (see [10, 39]). While significant advances have been made on computational problems involving invariant theory, most algorithms are based on Gröbner bases techniques, and hence still require exponential time (or longer).

[3] In the conjecture, the group is specified to be $\mathrm{SL}_n(\mathbb{C})$, which was done for the purpose of accessibility and brevity, but it is natural to ask this problem for general connected reductive groups. We will discuss this further in a later section.

The basic setting is that of a continuous group[4] G acting (linearly) on a finite-dimensional vector space $V = \mathbb{C}^m$.

An *action* (also called a *representation*) of a group $G \subseteq \mathrm{GL}_n(\mathbb{C})$ on an m-dimensional complex vector space V is a group homomorphism $\pi\colon G \to \mathrm{GL}_m(\mathbb{C})$, that is, an association of an invertible $m \times m$ matrix $\pi(g)$ for every group element $g \in G$, satisfying $\pi(g_1 g_2) = \pi(g_1)\pi(g_2)$ for all $g_1, g_2 \in G$ (and $\pi(e) = \mathrm{I}_m$, where $e \in G$ is the identity element and I_m is the identity matrix). To be precise, a group element $g \in G$ acts on a vector $v \in V$ by the linear transformation $\pi(g)$, and in this paper we will be dealing with algebraic actions, that is, the entries of the matrix $\pi(g)$ will be rational functions in the entries of the matrix g. We will write $g \cdot v = \pi(g)v$. Invariant theory is nicest when the underlying field is $\mathbb{C}$ and the group G is either finite, the general linear group $\mathrm{GL}_n(\mathbb{C})$, the special linear group $\mathrm{SL}_n(\mathbb{C})$, or a direct product of these groups and their diagonal subgroups. We denote by $\mathbb{C}[V]$ the ring of polynomial functions on V.

Invariant Polynomials. Invariant polynomials are precisely those which cannot distinguish between a vector v and a translate of it by an element of the group, i.e., $g \cdot v$. In other words, a polynomial function $f \in \mathbb{C}[V]$ is called invariant if $f(g \cdot v) = f(v)$ for all $v \in V$ and $g \in G$. Equivalently, invariant polynomials are polynomial functions on V which are left invariant by the action of G. More precisely, the action of G on V gives an induced action of G on $\mathbb{C}[V]$, the space of polynomial functions on V. For a polynomial function p on V, the group element $g \in G$ sends it to the function $g \cdot p$ which is defined by the formula $(g \cdot p)(v) = p(g^{-1} \cdot v)$ for $v \in V$. Then, a polynomial function is invariant if and only if $g \cdot p = p$ for all $g \in G$. A set $\{f_i\}_{i \in I}$ of invariant polynomials is called a *generating set* if any other invariant polynomial can be written as a polynomial in the f_i's. Two simple and illustrative examples are

- The symmetric group $G = \mathcal{S}_n$ acts on $V = \mathbb{C}^n$ by permuting the coordinates. In this case, the invariant polynomials are *symmetric* polynomials, and the n elementary symmetric polynomials form a generating set (a result that dates back to Newton).
- The group $G = \mathrm{SL}_n(\mathbb{C}) \times \mathrm{SL}_n(\mathbb{C})$ acts on $V = \mathrm{M}_n(\mathbb{C})$ by a change of bases of the rows and columns, namely left-right multiplication: that is, the action of (A, B) sends X to AXB^T. Here, $\det(X)$ is an invariant polynomial and in fact every invariant polynomial must be a univariate polynomial in $\det(X)$. In other words, $\det(X)$ generates the invariant ring.

The above phenomenon that the ring of invariant of polynomials (denoted by $\mathbb{C}[V]^G$) is generated by a finite number of invariant polynomials is not a coincidence. The *finite generation theorem* due to Hilbert [24, 25] states that, for a large class of groups (including the groups mentioned above), the invariant ring must be finitely generated. These two papers of Hilbert are highly influential and laid the foundations of modern commutative algebra and algebraic geometry. In particular, "finite basis theorem" and "Nullstellansatz" were proved as "lemmas" on the way towards proving the finite generation theorem!

Orbits and Orbit Closures. The *orbit* of a vector $v \in V$, denoted by $\mathcal{O}_v$, is the set of all vectors obtained by the action of G on v. The *orbit closure* of v, denoted by $\overline{\mathcal{O}}_v$, is the closure of the orbit $\mathcal{O}_v$ in the Euclidean topology.[5] For actions of continuous groups, such

[4] In general, the theory works whenever the group is algebraic and reductive. However in this paper, we will deal with groups that are well understood such as a torus and the special linear group.

[5] It turns out mathematically more natural to look at closure under the Zariski topology. However, for the group actions we study, the Euclidean and Zariski closures match, a consequence of Chevalley's theorem on constructible sets.

as $\mathrm{GL}_n(\mathbb{C})$, it is more natural to look at orbit closures. The *null cone* for a group action is the set of all vectors which behave like the 0 vector i.e. the 0 vector lies in their orbit closure. Many fundamental problems in theoretical computer science (and many more across mathematics) can be phrased as questions about orbits and orbit closures. Here are some familiar examples:

- Graph isomorphism problem can be phrased as checking if the orbits of two graphs are the same or not, under the action of the symmetric group permuting the vertices.
- Geometric complexity theory (GCT) [32] formulates a variant of the VP vs. VNP question as checking if the (padded) permanent lies in the orbit closure of the determinant (of an appropriate size), under the action of the general linear group on polynomials induced by its natural linear action on the variables.
- Border rank (a variant of tensor rank) of a 3-tensor can be formulated as the minimum dimension such that the (padded) tensor lies in the orbit closure of the unit tensor, under the natural action of $\mathrm{GL}_r(\mathbb{C}) \times \mathrm{GL}_r(\mathbb{C}) \times \mathrm{GL}_r(\mathbb{C})$. In particular, this captures the complexity of matrix multiplication.

1.3 Computational invariant theory, Mulmuley's problems and conjectures

From its origins in the 19th century, the subject of classical invariant theory has been computational in nature – one of its central goals is explicit descriptions of generators of invariant rings, their relations, etc. With the more recent advent of the theory of computation, it is only natural to ask for the complexity of these descriptions. The influence of complexity theory has taken an important role in invariant theory as a consequence of the connections to fundamental problems such as VP vs. VNP that were uncovered as part of the GCT program by Mulmuley in [31]. In [31], Mulmuley considers the computational complexity of various invariant theoretic problems. Let G be a group acting on V.

1. (**Generators**) Output a list of polynomials that generate the invariant ring $\mathbb{C}[V]^G$.
2. (**NNL**) Output a list of polynomials $f_1, \ldots, f_r$, such that each f_i is a homogeneous polynomial and the invariant ring $\mathbb{C}[V]^G$ is integral over $\mathbb{C}[f_1, \ldots, f_r]$.[6]
3. (**Orbit closure intersection**) Given two elements of the vector space, do their orbit closures intersect?
4. (**Null cone membership**) Given an element of the vector space, does the 0 vector lie in its orbit closure?

Let us point out straight away that Generators and NNL (Noether Normalization Lemma) are both algebraic circuit search problems (we will define Generators as an algebraic circuit search problem more precisely below). Orbit closure intersection and Null cone membership are not algebraic circuit search problems, but are related to Generators and NNL in a way that will become clear in a later discussion. We will not get into the details of how the group is given and how the group action is described. It turns out that even for simple groups and group actions, these problems turn out to be interesting. They have been long studied and many algorithms have been developed in the invariant theory community [10, 39]. Mulmuley [31] introduced these problems to theoretical computer science with the hope of making progress on the polynomial identity testing (PIT) problem. Before describing the main conjectures in Mulmuley's paper, let us see what it even means to output a list of generating

[6] This is equivalent to the condition that the zero locus of $f_1, \ldots, f_r$ is precisely the null cone.

polynomials for an invariant ring. Typically the number of generating polynomials can be exponential in the dimension of the group and the vector space. To get around this issue, Mulmuley introduced the following notion of a *succinct encoding* of the generators of an invariant ring (which in fact applies to any collection of polynomials).

▶ **Definition 3** (Succinct encoding of generators). *Fix an action of a group G on a vector space $V = \mathbb{C}^m$. We say that an arithmetic circuit $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$* succinctly encodes *the generators of the invariant ring if the set of polynomials formed by evaluating the y-variables, $\{\mathcal{C}(x_1, \ldots, x_m, \alpha_1, \ldots, \alpha_r)\}_{\alpha_1,\ldots,\alpha_r \in \mathbb{C}}$, is a generating set for the invariant ring $\mathbb{C}[V]^G$.*

▶ Remark 4. The *size* of a succinct encoding as defined above is given by the size of the circuit $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$, which is measured by the bit complexity of the constants used in the computation of $\mathcal{C}$ as well as the number of gates of the computation graph of $\mathcal{C}$. In particular, this means that all constants used in the computation of $\mathcal{C}$ are rationals.

The above notion of a succinct encoding motivates us to define the following algebraic search problem.

▶ **Problem 5** (Generators). *Let G be a group of dimension n and that acts algebraically on an m-dimensional vector space V by linear transformations. Output a* $\mathrm{poly}(n, m)$ *sized circuit $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$ such that the polynomials $\{\mathcal{C}(x_1, \ldots, x_m, \alpha_1, \ldots, \alpha_r)\}_{\alpha_1,\ldots,\alpha_r \in \mathbb{C}}$ form a generating set for the invariant ring $\mathbb{C}[V]^G$. In other words, the problem asks to output a* $\mathrm{poly}(n, m)$ *sized succinct encoding for the generators of $\mathbb{C}[V]^G$.*

▶ **Conjecture 6** (Mulmuley). *In the case that G is a connected reductive algebraic group*[7]*, Problem 5 has a positive answer. That is, there exists a* $\mathrm{poly}(n, m)$ *sized circuit which succinctly encodes the generators of $\mathbb{C}[V]^G$.*

Mulmuley requires the circuit family (that succinctly encodes the generators) to be uniformly computable by a polynomial time algorithm, but we will see that even this weaker conjecture is false (under standard complexity assumptions).

In [31, Conjecture 5.3], Mulmuley states the above conjecture for actions of the group $\mathrm{SL}_n(\mathbb{C})$. However, it is evident that there is nothing special about $\mathrm{SL}_n(\mathbb{C})$ with regard to the GCT program and it is natural to state the conjecture in the generality of connected reductive groups. Let us also note that it was already evident to Mulmuley that one cannot drop the "connected" assumption on the group, because the permanent appears as an invariant polynomial for a non-connected reductive group that would disprove the conjecture immediately using a similar line of reasoning to the one we use in the next section (see, e.g., [4]).

To understand Mulmuley's motivation for the conjecture, let us see what it means for the problems of orbit closure intersection and null cone membership. By definition, invariant polynomials are constant on the orbits (and thus on orbit closures as well). Thus, if $\overline{\mathcal{O}}_{v_1} \cap \overline{\mathcal{O}}_{v_2} \neq \emptyset$, then $p(v_1) = p(v_2)$ for all invariant polynomials $p \in \mathbb{C}[V]^G$. A remarkable theorem due to Mumford says that the converse is also true for the large class of reductive groups:

▶ **Theorem 7** ([34]). *Fix an action of a reductive group G on a vector space V. Given two vectors $v_1, v_2 \in V$, we have $\overline{\mathcal{O}}_{v_1} \cap \overline{\mathcal{O}}_{v_2} \neq \emptyset$ if and only if $p(v_1) = p(v_2)$ for all $p \in \mathbb{C}[V]^G$.*

[7] We have not defined what a connected reductive algebraic group is. One should think of simple groups like the general linear group $\mathrm{GL}_n(\mathbb{C})$, the special linear group $\mathrm{SL}_n(\mathbb{C})$, or a direct product of these groups and their diagonal subgroups.

Now suppose one had a succinct encoding $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$ for action of a group G on $V = \mathbb{C}^m$. Then because of Mumford's theorem, for two vectors v_1 and v_2, their orbit closures intersect iff the two polynomials $\mathcal{C}(v_1(1), \ldots, v_1(m), y_1, \ldots, y_r)$, $\mathcal{C}(v_2(1), \ldots, v_2(m), y_1, \ldots, y_r)$ are identically the same. These are instances of polynomial identity testing (PIT)! Thus if Conjecture 6 were true (and additionally the succinct encoding circuits were polynomial time computable), it immediately gives randomized polynomial time algorithms for the orbit closure intersection and null cone membership problems. This also gives us a nice family of PIT problems to play with. Perhaps one might hope that solving these PIT instances will result in development of new techniques which might shed a light on the general PIT problem. In fact, for the first few group actions that were studied in this line of work, *simultaneous conjugation* [31, 17] and *left-right action* [19, 27, 12], for which there are polynomial sized succinct encodings of generators, the null cone membership problems correspond to PIT problems for restricted models of computation: *read-once algebraic branching programs* and *non-commutative formulas with division*[8], both of which have been successfully derandomized, see [17, 19, 27].

1.4 Our results

While the truth of Conjecture 6 would have great implications, we prove that it is false under plausible complexity hypotheses. We first state our counterexamples (they are very simple, and probably many others exist), and then discuss how a related conjecture may be true and almost as powerful as the original.

For our first counterexample, we analyze a simple (torus) action on 3-tensors. Here, $\mathrm{ST}_n(\mathbb{C})$ denotes the group of $n \times n$ diagonal matrices with determinant 1.

▶ **Theorem 8.** *Consider the natural action of* $G = \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C})$ *on* $V = \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$. *Then any set of generators for the invariant ring cannot have a polynomial sized (in* n*) succinct encoding, unless* $\mathrm{NP} \subseteq \mathrm{P/poly}$.

▶ **Corollary 9.** *Conjecture 6 is false, unless* $\mathrm{NP} \subseteq \mathrm{P/poly}$.

▶ Remark 10. As mentioned previously, a primary motivation for succinct encodings of generators is that they imply (randomized) polynomial time algorithms for null cone membership problem. For the action in Theorem 8, it is important to note that even though we do not have a succinct encoding for generators, we still have a polynomial time algorithm for null cone membership since once can reduce it to an instance of linear programming. For a general connection between null cone membership and optimization, see [5].

For the above counterexample for the torus action, the notion of a succinct encoding is quite crucial to our argument, and it is natural to wonder if tweaking the notion would get rid of the issue. We give another counterexample where it becomes apparent that the precise form of encoding of the generators is not quite as crucial, as we identify an invariant that is hard to compute and is *essential* to any generating set in a sense that we will make precise in Section 4. Moreover, it is an $\mathrm{SL}_n(\mathbb{C})$-action, which provides a counterexample to the exact formulation of the conjecture in [31].

▶ **Theorem 11.** *Let* $k \geq 2$ *be even. Consider the action of* $G = \mathrm{SL}_{2kn}$ *on* $V = \bigotimes^{2k} \mathbb{C}^{2kn}$. *Then any set of generators for the invariant ring cannot have a polynomial sized (in* n*) succinct encoding, unless* $\mathrm{VP} = \mathrm{VNP}$.

▶ **Corollary 12.** *Conjecture 6 is false, unless* $\mathrm{VP} = \mathrm{VNP}$.

[8] Actually a stronger model concerning inverses of matrices.

1.5 Conclusion, open problems and future directions

We have disproved a conjecture of Mulmuley about the existence of polynomial sized succinct encodings of generators for invariant rings. We want to emphasize that this only serves a first guiding light for Mulmuley's program of understanding the orbit closure intersection problems (and null cone membership problems) and connections to PIT. To solve the orbit closure intersection problems, one does not necessarily need a generating set of generators. This motivates the following definition.

▶ **Definition 13** (Separating set of invariants). *For a group G acting algebraically on a vector space V by linear transformations, a subset $S \subseteq \mathbb{C}[V]^G$ is called a* separating set of invariants *if for all $u, v \in V$ such that $\overline{\mathcal{O}}_u \cap \overline{\mathcal{O}}_v \neq \emptyset$, there exists $f \in S$ such that $f(u) \neq f(v)$.*

This leads to a natural algebraic search problem that corresponds to the algorithmic problem of orbit closure intersection. Mulmuley already suggested that a positive answer to the following search problem would suffice for the purposes of GCT.

▶ **Problem 14** (Separators). *Let G be a group of dimension n and suppose it acts algebraically on an m-dimensional vector space V by linear transformations. Output a $\mathrm{poly}(n, m)$ sized circuit $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$, if one exists, such that the set of polynomials $S = \{\mathcal{C}(x_1, \ldots, x_m, \alpha_1, \ldots, \alpha_r)\}_{\alpha_1,\ldots,\alpha_r \in \mathbb{C}}$ is a separating set of invariants.*

Similarly, we can define a search problem that corresponds to the algorithmic problem of null cone membership.

▶ **Problem 15** (Null cone definers). *Let G be a group of dimension n and suppose G acts algebraically on an m-dimensional vector space V by linear transformations. Output a $\mathrm{poly}(n, m)$ sized circuit $\mathcal{C}(x_1, \ldots, x_m, y_1, \ldots, y_r)$ with the property that the set $S = \{\mathcal{C}(x_1, \ldots, x_m, \alpha_1, \ldots, \alpha_r)\}_{\alpha_1,\ldots,\alpha_r \in \mathbb{C}}$ consists of invariant polynomials whose zero locus is precisely the null cone $\mathcal{N}_G(V) = \{v \in V \mid 0 \in \overline{\mathcal{O}}_v\}$.*

We conclude the introduction with some open open problems:

1. Are there polynomial sized succinct encodings for separating invariants or, even simpler, invariants defining the null cone? In other words, do we have positive answers to Problems 14 and 15 for connected reductive groups? Perhaps the first non-trivial example is the natural action of $G = \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C})$ on $V = \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$. Here a tensor T is in the null cone iff there exists vectors $x, y, z \in \mathbb{R}^n$ s.t. $x_i + y_j + z_k > 0$ for all $(i, j, k) \in \mathrm{supp}(T)$[9] and $\sum_i x_i = \sum_j y_j = \sum_k z_k = 0$ (by the Hilbert-Mumford criterion). Is there a polynomial sized circuit $\mathbb{C}((z_{i,j,k}), y_1, \ldots, y_r)$ s.t. $\mathbb{C}(T, y_1, \ldots, y_r)$ is identically zero (as a polynomial in the y-variables) iff T is in the null cone?
2. For the natural action of $\mathrm{SL}_n(\mathbb{C}) \times \mathrm{SL}_n(\mathbb{C}) \times \mathrm{SL}_n(\mathbb{C})$ on $V = \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$, it is not even clear if there exists one invariant which has a polynomial sized circuit. Either produce such an invariant or prove that all invariants are hard to compute.[10]
3. Are there polynomial time algorithms for the orbit closure intersection and null cone membership problems? The analytic approach pursued in the papers [19, 7, 2, 5] seems the most promising approach towards getting such algorithms.

[9] $\mathrm{supp}(T) = \{(i, j, k) \in [n] \times [n] \times [n] : T_{i,j,k} \neq 0\}$.

[10] This problem is known to experts in the field, but has not been written down explicitly anywhere to the best of our knowledge.

4. More broadly, invariant theory is begging for its own complexity theory and connecting it with ours. This includes finding reductions and completeness results, and characterizations/dichotomies about hard/easy actions. An example of a completeness reduction is the reduction from all quiver actions to the simple left-right action [15, 16, 35, 12, 20]. Also the papers [31, 19, 27, 17, 12, 13, 11, 23, 30, 5], as well as the current paper, are trying to identify easy and hard problems in invariant theory.

2 Preliminaries

In this section we establish notation and we formally state basic facts and definitions which we will need in later sections.

▶ **Definition 16** (3-dimensional matching [28]). *The* 3-dimensional matching *problem is defined as follows:*

- ***Input:*** *a set $U \subseteq [n] \times [n] \times [n]$, representing the edges of a tripartite, 3-uniform hypergraph.*
- ***Output:*** *YES, if there is a set of hyperedges $W \subseteq U$ such that $|W| = n$ and no two elements of W agree in any coordinate (that is, they form a matching in this hypergraph). NO, if there is no such set.*

▶ **Theorem 17** (NP-completeness of 3-dimensional matching [28]). *The 3-dimensional matching problem is NP-complete.*

2.1 Basic facts from algebraic complexity

We now give basic facts that from algebraic complexity which we will use in the next sections.

The next proposition shows that homogeneous components of low degree of an arithmetic circuit can be efficiently computed, with a small blow-up in circuit size and without the use of any extra constants. This proposition was originally proved by Strassen in [38] and its proof can be found in [37, Theorem 2.2]. In the following proposition, given a polynomial $p(\mathbf{x})$, we denote its degree-d homogeneous component by $H_d[p(\mathbf{x})]$.

▶ **Proposition 18** (Efficient computation of homogeneous components). *Given a circuit $\mathcal{C}(\boldsymbol{x})$ of size s, then for every $r \in \mathbb{N}$ there is a homogeneous circuit $\Psi(\boldsymbol{x})$ of size $O(r^2 s)$ computing $H_0[\mathcal{C}(\boldsymbol{x})], H_1[\mathcal{C}(\boldsymbol{x})], \ldots, H_r[\mathcal{C}(\boldsymbol{x})]$. Moreover, the constants used in the computation of the components $H_i[\mathcal{C}(\boldsymbol{x})]$ are a subset of the coefficients used in the computation of $\mathcal{C}(\boldsymbol{x})$.*

The next theorem, proved by [1, Theorem 4.10] gives us a randomized polynomial time algorithm to test whether an algebraic circuit of polynomial size, with rational coefficients, is identically zero. Another randomized algorithm easily follows from [36, Lemma 2], when adapted for polynomials with rational coefficients.

▶ **Theorem 19** (PIT for poly-sized circuits [1]). *Let $P(\boldsymbol{x}) \in \mathbb{Q}[\boldsymbol{x}]$ be a polynomial in the variables $\boldsymbol{x} = (x_1, \ldots, x_n)$, with each variable x_i having degree bounded by d_i, and whose coefficients are rationals with bit complexity bounded by B. If $P(\boldsymbol{x})$ is given as an arithmetic circuit of size s, then there exists a randomized algorithm running in time $\mathrm{poly}(n, s, \log(B), 1/\epsilon)$ and using $O\left(\sum_{i=1}^{n} \log(d_i) + \log(B)\right)$ random bits which tests whether $P(\boldsymbol{x})$ is identically zero. If $P(\boldsymbol{x})$ is the identically zero polynomial then the algorithm always succeeds. Otherwise, it errs with probability at most ϵ.*

3 Hardness of Generators for torus actions

Let $\mathbb{C}^*$ denote the multiplicative group consisting of all non-zero complex numbers. A direct product $\mathrm{T}_n = (\mathbb{C}^*)^n$ is called a torus, and is clearly an abelian group. Tori are important examples of reductive groups – any abelian connected reductive group is a torus! It is often the case that it is easier to understand tori in comparison with more general (non-abelian) reductive groups. This is no different for invariant theory, see for example [10, 40]. We also point to [14, Proposition 3.3] for an elementary linear algebraic description of the invariant ring for torus actions. Conjecture 6 already fails in this well behaved setting. This is the content of our Theorem 8, which we will prove in this section. Recall that $\mathrm{ST}_n(\mathbb{C}) \cong \{\mathbf{z} \in \mathrm{T}_n : z_1 \cdots z_n = 1\}$, which is itself a torus.

▶ **Theorem 20** (Theorem 8, restated). *Consider the natural action of* $G = \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C})$ *on* $V = \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$*, where an element* $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in G$ *acts on a tensor* $u \in V$ *as follows:* $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \cdot u := v$*, such that* $v_{ijk} = a_i b_j c_k u_{ijk}$*. Any set of generators for the invariant ring of this action cannot have a polynomial sized (in* n*) succinct encoding, unless* $\mathrm{NP} \subseteq \mathrm{P/poly}$.

Proof. Suppose that the natural action above has a set of generators with a polynomial sized succinct encoding. Thus, there is an arithmetic circuit $\mathcal{C}(\mathbf{x}, \mathbf{y})$ of size $s = \mathrm{poly}(n)$, where $\mathbf{x} = (x_{ijk})_{i,j,k=1}^n$ is the set of variables corresponding to V and $\mathbf{y} = (y_1, \ldots, y_r)$ is the set of auxiliary variables, with $r = \mathrm{poly}(n)$. Moreover, from the definition of the size of a succinct encoding we also have that the constants used in the computation of $\mathcal{C}(\mathbf{x}, \mathbf{y})$ are rational numbers with bit complexity bounded by $b = \mathrm{poly}(n)$. In particular, $\mathcal{C}(\mathbf{x}, \mathbf{y}) \in \mathbb{Q}[\mathbf{x}, \mathbf{y}]$.

Let us consider the circuit $\mathcal{C}(\mathbf{x}, \mathbf{y})$ as a circuit whose constants are in $\mathbb{Q}[\mathbf{y}]$ and whose variables are only the $\mathbf{x}$ variables, that is, a circuit in $\mathbb{Q}[\mathbf{y}][\mathbf{x}]$. Then, Proposition 18 tells us that there exists a homogeneous circuit $\mathcal{C}_n(\mathbf{x}, \mathbf{y})$, in the $\mathbf{x}$ variables, of degree n and size $O(n^2 s)$ that computes the homogeneous component of $\mathcal{C}(\mathbf{x}, \mathbf{y})$ of degree n as a function of $\mathbf{x}$. Moreover, the constants of this circuit are a subset of the constants used in the circuit $\mathcal{C}(\mathbf{x}, \mathbf{y})$. Since we consider the latter as a circuit in only the $\mathbf{x}$ variables, the constants in this case are given by the elements of $\mathbb{Q}$ used in the computation of $\mathcal{C}$ as well as the auxiliary variables $\mathbf{y}$. In particular, $\mathcal{C}_n(\mathbf{x}, \mathbf{y})$ can be written in the following way:

$$\mathcal{C}_n(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{m} \in \mathcal{N}_n(\mathbf{x})} f_{\mathbf{m}}(\mathbf{y}) \cdot \mathbf{m}, \tag{1}$$

where $\mathcal{N}_n(\mathbf{x})$ is the set of all monomials of degree n in the variables $\mathbf{x}$ and $f_{\mathbf{m}}(\mathbf{y})$ are polynomials in the variables $\mathbf{y}$ of degree at most 2^s, as the circuit $\mathcal{C}$ has size at most s.

In Proposition 21 below, we will show that the invariants of minimum degree of our action are in degree n, and these are spanned by the (maximum) 3-dimensional matching monomials. Thus, if a monomial of degree n is invariant under our action, it must be the case that this monomial corresponds to a 3-dimensional matching. Moreover, the action maps any monomial (invariant or not) to a constant times itself. As $\mathcal{C}_n(\mathbf{x}, \mathbf{y})$ must only compute invariant polynomials, this implies that equation (1) is actually of the following form:

$$\mathcal{C}_n(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{m} \in \mathcal{M}_n(\mathbf{x})} f_{\mathbf{m}}(\mathbf{y}) \cdot \mathbf{m}, \tag{2}$$

where $\mathcal{M}_n(\mathbf{x})$ is the set of all 3-dimensional matching monomials in the variables $\mathbf{x}$. Moverover, since $\mathcal{C}(\mathbf{x}, \mathbf{y})$ succinctly encodes of a set of generators, the span of $\{\mathcal{C}_n(\mathbf{x}, \alpha)\}_{\alpha \in \mathbb{C}^r}$ must necessarily be the same as the span of the 3-dimensional matching polynomials.

We will now show that the existence of the circuit $\mathcal{C}_n(\mathbf{x}, \mathbf{y})$ implies that NP $\subseteq$ P/poly. For that purpose, we will show that given $\mathcal{C}_n(\mathbf{x}, \mathbf{y})$ one can solve the 3-dimensional matching problem in P/poly. Let H be a tripartite 3-uniform hypergraph, whose edges are given by a subset $E \subseteq [n] \times [n] \times [n]$. We can associate to this graph the tensor $v \in V$ where $v_{ijk} = 1$ if hyperedge $(i, j, k) \in E$ and $v_{ijk} = 0$ otherwise. Note that H has a 3-dimensional matching of size n if and only if at least one of the 3-dimensional matching monomials *does not vanish* on our tensor v. This last condition is equivalent to the fact that the circuit $\mathcal{C}_n(v, \mathbf{y})$ does not compute the zero polynomial (as we know that the span of the set $\{\mathcal{C}_n(\mathbf{x}, \alpha)\}_{\alpha \in \mathbb{C}^r}$ is the same as the span of all 3-dimensional matching monomials). Thus, to solve the 3-dimensional matching problem in P/poly it is enough to give a randomized polynomial time algorithm for testing whether $\mathcal{C}_n(v, \mathbf{y})$ is the zero polynomial or not.[11]

Since $\mathcal{C}_n(v, \mathbf{y})$ is a circuit of size poly(n) with rational constants of bit complexity poly(n), it computes a polynomial $P(\mathbf{y})$ with rational coefficients having bit complexity at most exp(poly(n)) and degree at most exp(poly(n)). This is the setting in which Theorem 19 applies, giving us the desired randomized polynomial time algorithm. This concludes our proof modulo Proposition 21, which we will now turn our attention to. ◀

In the following proposition, we denote by $\mathcal{S}_n$ the symmetric group on n letters.

▶ **Proposition 21.** *The maximum 3-dimensional matching monomials $\prod_{i=1}^n x_{i\sigma(i)\tau(i)}$, where $\sigma, \tau \in \mathcal{S}_n$, span the invariants of degree n of the natural action of $G = \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C}) \times \mathrm{ST}_n(\mathbb{C})$ on $V = \mathbb{C}^n \otimes \mathbb{C}^n \otimes \mathbb{C}^n$. Moreover, there are no nonconstant invariants of degree less than n for this action.*

Proof. Since the action maps any monomial to a constant times itself, it is easy to see that the invariant polynomials are generated by invariant monomials. To prove the proposition, it is therefore enough to show that the matching monomials are invariant, that there are no other invariant monomials of degree n, and that there are no invariant monomials of smaller degree.

We first prove that the matching monomials are invariant. Note that the natural action of G on V induces the following action on the variables x_{ijk}: $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \cdot x_{ijk} = (a_i b_j c_k)^{-1} \cdot x_{ijk}$.[12] Additionally, note that $\prod_{\ell=1}^n a_\ell = \prod_{\ell=1}^n b_\ell = \prod_{\ell=1}^n c_\ell = 1$. Given a matching monomial $\prod_{i=1}^n x_{i\sigma(i)\tau(i)}$, we therefore have that

$$\begin{aligned}(\mathbf{a}, \mathbf{b}, \mathbf{c}) \cdot \prod_{i=1}^n x_{i\sigma(i)\tau(i)} &= \prod_{i=1}^n \left((a_i b_{\sigma(i)} c_{\tau(i)})^{-1} \cdot x_{i\sigma(i)\tau(i)}\right) \\ &= \prod_{i=1}^n (a_i b_{\sigma(i)} c_{\tau(i)})^{-1} \cdot \prod_{i=1}^n x_{i\sigma(i)\tau(i)} \\ &= \prod_{i=1}^n x_{i\sigma(i)\tau(i)}\end{aligned}$$

where in the last equality we note that for any permutation $\sigma \in \mathcal{S}_n$ (or τ) we have $1 = \prod_{\ell=1}^n a_\ell = \prod_{\ell=1}^n a_{\sigma(\ell)}$ (and similarly for $\mathbf{b}$ and $\mathbf{c}$). This proves that the matching monomials are invariant monomials of the natural G-action on V.

Now, let us prove that no other monomial of degree n is an invariant for this action. Let $\prod_{m=1}^n x_{i_m j_m k_m}$ be a monomial, where $(i_m, j_m, k_m) \in [n]^3$, that is not a matching monomial. Then there exists some coordinate, w.l.o.g. the first coordinate, for which the set $\{i_m\}_{m=1}^n$ is a

[11] It is enough to give a randomized polynomial time algorithm because we know that BPP/poly = P/poly.
[12] The inverse comes from the general formula $(g \cdot p)(v) := p(g^{-1} \cdot v)$.

strict subset of $[n]$. Equivalently, there is an element $\ell \in [n]$ such that $\ell \notin \{i_m\}_{m=1}^n$. W.l.o.g., we can assume that $\ell = 1$. Thus, the action of $\mathbf{a} = (\alpha^{n-1}, \alpha^{-1}, \ldots, \alpha^{-1}), \mathbf{b} = \mathbf{c} = (1, \ldots, 1)$ on our monomial $\prod_{m=1}^n x_{i_m j_m k_m}$ is as follows:

$$(\mathbf{a}, \mathbf{b}, \mathbf{c}) \cdot \prod_{m=1}^{n} x_{i_m j_m k_m} = \alpha^n \cdot \prod_{m=1}^{n} x_{i_m j_m k_m}$$

which proves that this monomial is not an invariant. This completes the proof that the matching monomials span the invariants of degree n.

Now we are left with proving that there are no nonconstant monomials of degree less than n that are invariant. Note that if we have a monomial with degree less than n, we can represent it as $\prod_{m=1}^{d} x_{i_m j_m k_m}$, where $d < n$ and by the pigeonhole principle, we know that there exists $\ell \in [n]$ such that ℓ does not appear as a first coordinate entry in the set of tuples $\{(i_m, j_m, k_m)\}$. If $d > 0$ then, analogously to the previous paragraph, we know that such monomials cannot be invariants of the natural action of G over V, therefore showing that no nonconstant monomial of degree $< n$ can be an invariant. This completes the proof. ◀

4 Invariant Theory for $\mathrm{SL}_n(\mathbb{C})$ and Mulmuley's conjecture

In this section, we will give another example of a group action on tensors for which any set of generating invariants is hard to compute, i.e., we will prove Theorem 11. Even though the previous section already gives a counterexample, this example illustrates something more. The feature of this group action is that invariants of minimial degree span a 1-dimensional space. In other words, up to scaling, we have a unique invariant of minimal degree. This unique invariant in the minimal degree is called the *hyperpfaffian polynomial* (introduced by Barvinok in 1995 as a natural generalization of the well-known Pfaffian polynomial to higher order tensors). We then study the hyperpfaffian's computational complexity and prove that it is VNP-complete. The importance of this example is that such a unique invariant in the minimal degree is *essential* in any generating set.[13] So, it is not even possible to give a generating set consisting of invariant polynomials that are easy to compute, even if we remove all restrictions on the size of the generating set.[14] Moreover, the group action is by $\mathrm{SL}_n(\mathbb{C})$ rather than a torus. Therefore our counterexample disproves Mulmuley's original conjecture in a strong sense.

4.1 Invariant Rings and Symmetric Tensors

The special linear group $\mathrm{SL}_n(\mathbb{C})$ consists of complex $n \times n$-matrices with unit determinant and acts canonically on $\mathbb{C}^n$ by matrix-vector multiplication. This action extends to any m-th tensor power $\bigotimes^m \mathbb{C}^n$ by

$$g \cdot (v_1 \otimes \cdots \otimes v_m) := (g \cdot v_1) \otimes \cdots \otimes (g \cdot v_m) \tag{3}$$

and linear continuation. We will always use the standard bilinear form on $\bigotimes^m \mathbb{C}^n$ that satisfies $\langle g^T \cdot v, w \rangle = \langle v, g \cdot w \rangle$ for all $v, w \in \bigotimes^m \mathbb{C}^n$, $g \in \mathrm{SL}_n(\mathbb{C})$.

[13] Unique invariants in minimal degree have been studied and used in the context of GCT before in [8], although the problems pursued there are quite different from the one we are considering in this paper.

[14] A very similar argument also works in the action of $\mathrm{SL}_n^{\times 4}$ on $(\mathbb{C}^n)^{\otimes 4}$, in which case, the unique minimal invariant is called Pascal determinant and also known to be VNP–hard. This appeared in an earlier version of this paper, see [21]. However, our current example using the hyperpfaffian has the added advantage of disproving the exact formulation of Mulmuley's conjecture.

Let V be an arbitrary finite-dimensional $\mathrm{SL}_n(\mathbb{C})$-representation (such as $V = \bigotimes^m \mathbb{C}^n$). Then $\mathrm{SL}_n(\mathbb{C})$ also acts on $\mathbb{C}[V]_d$, the vector space of degree-d homogeneous polynomials on V, via the formula

$$(g \cdot p)(v) := p(g^T \cdot v),$$

where $p \in \mathbb{C}[V]_d, g \in G$ and $v \in V$. The formula above is the dual representation of the action on the ring of all polynomial functions $\mathbb{C}[V] = \bigoplus_{d=0}^{\infty} \mathbb{C}[V]_d$ that we explained in in Section 1.2. Using the dual here is only for presentation purposes, as it gives a clearer connection to multilinear algebra as follows. Note that a polynomial $p \in \mathbb{C}[V]$ is invariant if and only if $\forall g \in \mathrm{SL}_n(\mathbb{C})$ we have $g \cdot p = p$.

It is convenient to identify polynomial functions with symmetric tensors. Note that $\mathrm{SL}_n(\mathbb{C})$ acts canonically on any d-th tensor power $\bigotimes^d V$ of V. This action restricts to the d-th symmetric tensor power $\mathrm{Sym}^d V$, i.e., the $\mathcal{S}_d$-invariant subspace of $\bigotimes^d V$. Recall that $\mathcal{S}_d$ is the symmetric group on d letters; it acts on $V^{\otimes d}$ by permuting tensor factors. For any $t \in \mathrm{Sym}^d V$, we can define a homogeneous degree-d polynomial $p \in \mathbb{C}[V]_d$ by $p(v) := \langle t, v^{\otimes d} \rangle$. Here we use the quadratic form on $\mathrm{Sym}^d V$ induced by a non-degenerate bilinear form on V that satisfies $\langle g^t \cdot v, w \rangle = \langle v, gw \rangle$ for all $v, w \in V$, $g \in \mathrm{SL}_n(\mathbb{C})$ as above. Then, p is invariant if and only if the symmetric tensor t is invariant, i.e., if $\forall g \in \mathrm{SL}_n(\mathbb{C})$ we have $g \cdot t = t$. We will tacitly go back and forth between symmetric tensors in $\mathrm{Sym}^d \bigotimes^m \mathbb{C}^n$ and homogeneous polynomials in $\mathbb{C}[\bigotimes^m \mathbb{C}^n]_d$.

Now, we turn to studying hyperpfaffians.

4.2 Hyperpfaffians

The *Pfaffian* is the unique (up to scale) homogeneous $\mathrm{SL}_{2n}(\mathbb{C})$-invariant of degree n on $\mathbb{C}^{2n} \otimes \mathbb{C}^{2n}$. There are no $\mathrm{SL}_{2n}(\mathbb{C})$-invariants in lower degrees. If we identify $\mathbb{C}^{2n} \otimes \mathbb{C}^{2n}$ with the space of complex $2n \times 2n$ matrices A, then the Pfaffian is invariant under the action of $\mathrm{SL}_{2n}(\mathbb{C})$ given by $g \cdot A := gAg^T$. The defining property of the Pfaffian generalizes to tensors of even order as follows (the classical Pfaffian is the special case of $k = 1$):

▶ **Proposition 22.** *For any k and n, there is a unique (up to scale) homogeneous $\mathrm{SL}_{2kn}(\mathbb{C})$-invariant polynomial $\mathrm{Pf}_{k,n}$ of degree n on $\bigotimes^{2k} \mathbb{C}^{2kn}$. $\mathrm{Pf}_{k,n}$ identifies with the symmetric tensor $e_1 \wedge \cdots \wedge e_{2kn} \in \mathrm{Sym}^n \bigotimes^{2k} \mathbb{C}^{2kn}$. There are no nonconstant $\mathrm{SL}_{2kn}(\mathbb{C})$-invariants of lower degree.*

Before proving Proposition 22 we recall some representation theory. The material is well-known, and we refer to standard texts (e.g., [18]) for details. A *partition* λ is a nonincreasing sequence of natural numbers with finite support. We write $\lambda \vdash_n m$ to say that $|\lambda| := \sum_i \lambda_i = m$ and $\lambda_{n+1} = 0$. If $\lambda_{n+1} = 0$, then we say that λ is an n-partition. The irreducible polynomial $\mathrm{GL}_n(\mathbb{C})$-representations are indexed by n-partitions. For a partition λ, let $\{\lambda\}$ denote the irreducible $\mathrm{GL}_n(\mathbb{C})$-representation corresponding to λ. Restricted to $\mathrm{SL}_n(\mathbb{C})$, the representation $\{\lambda\}$ is trivial if and only if $\lambda_1 = \ldots = \lambda_n$; note that this implies that $n \mid m$. The irreducible representations of $\mathcal{S}_m$ are indexed by partitions λ with $|\lambda| = m$. Let $[\lambda]$ denote the irreducible $\mathcal{S}_m$-representation corresponding to λ.

Consider $\bigotimes^m \mathbb{C}^n$. This space has an action of $\mathrm{SL}_n(\mathbb{C})$ by (3), but also an action of $\mathcal{S}_m$ that permutes the tensor factors. Both actions commute, so we have an action of the product group $\mathrm{SL}_n(\mathbb{C}) \times \mathcal{S}_m$. The following well-known result will be crucial for our purposes.

▶ **Theorem 23** (Schur–Weyl duality). *As an* $\mathrm{SL}_n(\mathbb{C}) \times \mathcal{S}_m$*-representation, we have the decomposition:*

$$\bigotimes^m \mathbb{C}^n = \bigoplus_{\lambda \vdash_n m} \{\lambda\} \otimes [\lambda].$$

Using Schur–Weyl duality, one sees immediately that $\bigotimes^m \mathbb{C}^n$ contains nonzero $\mathrm{SL}_n(\mathbb{C})$-invariant vectors if and only if $n \mid m$. This is because a vector is invariant if and only if it spans a trivial irreducible representation – but $\{\lambda\}$ is trivial if and only if $\lambda_1 = \ldots = \lambda_n$, as mentioned above. For $m = n$, there is a unique (up to scale) $\mathrm{SL}_n(\mathbb{C})$-invariant vector. This is because the invariants in $\bigotimes^n \mathbb{C}^n$ correspond to the component $\{1^n\} \otimes [1^n]$, where we write 1^n for the partition $\lambda_1 = \ldots = \lambda_n = 1$. Here, $\{1^n\}$ is the trivial representation of $\mathrm{SL}_n(\mathbb{C})$ and $[1^n]$ is one-dimensional, as it is the sign representation of $\mathcal{S}_n$. Thus the space of invariants is one-dimensional. This unique vector (up to scale) is given by the wedge product $e_1 \wedge e_2 \wedge \cdots \wedge e_n$, where $a \wedge b := \frac{1}{2}(a \otimes b - b \otimes a)$, and higher order wedge products are defined analogously.

Proof of Proposition 22. It suffices to show that $\mathrm{Sym}^d \bigotimes^{2k} \mathbb{C}^{2kn}$ contains no $\mathrm{SL}_{2kn}(\mathbb{C})$-invariant vector if $0 < d < n$ and that it contains a unique such vector if $d = n$. Note that $\mathrm{Sym}^d \bigotimes^{2k} \mathbb{C}^{2kn}$ is a subspace of $\bigotimes^d \bigotimes^{2k} \mathbb{C}^{2kn} \simeq \bigotimes^{2kd} \mathbb{C}^{2kn}$. Thus the first claim holds since $\bigotimes^{2kd} \mathbb{C}^{2kn}$ contains $\mathrm{SL}_{2kn}(\mathbb{C})$-invariant vectors only if $2kn \mid 2kd$. Thus if $0 < d < n$, there are no invariants. For $d = n$, $\bigotimes^d \bigotimes^{2k} \mathbb{C}^{2kn} \simeq \bigotimes^{2kn} \mathbb{C}^{2kn}$ contains the unique $\mathrm{SL}_{2kn}(\mathbb{C})$-invariant vector $v = (e_1 \wedge \cdots \wedge e_{2k}) \wedge \cdots \wedge (e_{2k(n-1)+1} \wedge \cdots \wedge e_{2kn})$. It remains to show that v is symmetric, i.e., an element of $\mathrm{Sym}^d \bigotimes^{2k} \mathbb{C}^{2kn}$, which is a subspace of $\bigotimes^d \bigotimes^{2k} \mathbb{C}^{2kn}$. But this is easy to see since each of the d blocks has even size $2k$ and the wedge product is skew-commutative. This proves the second claim. ◀

The polynomial $\mathrm{Pf}_{k,n}$ was introduced in [3, Def. 3.4] in its monomial presentation, where it is called the *hyperpfaffian*. Note that, for fixed k, $\mathrm{Pf}_k := (\mathrm{Pf}_{k,1}, \mathrm{Pf}_{k,2}, \ldots)$ is a p-family (i.e., both the degree and the number of variables are polynomially bounded), since $\mathrm{Pf}_{k,n}$ has degree n and $(2kn)^{2k}$ variables. The monomial presentation in [3] immediately yields that $\mathrm{Pf}_k \in \mathrm{VNP}$.

▶ **Theorem 24.** *For even* k*,* Pf_k *is* VNP*-complete.*

Proof. We present a projection of $\mathrm{Pf}_{k,d}$ to the $d \times d$ permanent. The same projection yields the determinant if k is odd, which explains why the proof does not work for the classical Pfaffian ($k = 1$). The case $k = 2$ is enough to disprove Mulmuley's conjecture.

By Proposition 22, the Pfaffian $\mathrm{Pf}_{k,d}$ identifies with the symmetric tensor

$$v := e_1 \wedge \cdots \wedge e_{2kd} \in \mathrm{Sym}^d \bigotimes^{2k} \mathbb{C}^{2kd}.$$

Thus, the evaluation $\mathrm{Pf}_{k,d}(p)$ at a tensor $p \in \bigotimes^{2k} \mathbb{C}^{2kd}$ is given by $\langle v, p^{\otimes d} \rangle$ (cf. [26, Sec. 4.2(A)]). We choose

$$p = \sum_{i,j=0}^{d-1} x_{i+1,j+1}(e_{1+2ki} \otimes e_{2+2ki} \otimes \cdots \otimes e_{k+2ki} \otimes e_{k+1+2kj} \otimes e_{k+2+2kj} \otimes \cdots \otimes e_{2k+2kj}),$$

where the $x_{i,j}$ $(1 \le i, j \le d)$ are formal variables.

The point p is parametrized linearly by the $x_{i,j}$, so the evaluation of $\mathrm{Pf}_{k,d}$ at p is a projection of $\mathrm{Pf}_{k,d}$. We verify that the evaluation of $\mathrm{Pf}_{k,n}$ at p gives the $d \times d$ permanent (up to a constant nonzero scalar) as follows.

$$p^{\otimes d} = \sum_{i_1,j_1,\ldots,i_d,j_d=0}^{d-1} x_{i_1+1,j_1+1}\cdots x_{i_d+1,j_d+1} \\ (e_{1+2ki_1}\otimes e_{2+2ki_1}\otimes\cdots\otimes e_{k+2ki_1}\otimes e_{k+1+2kj_1}\otimes e_{k+2+2kj_1}\otimes\cdots\otimes e_{2k+2kj_1}) \\ \otimes\cdots\otimes(e_{1+2ki_d}\otimes e_{2+2ki_d}\otimes\cdots\otimes e_{k+2ki_d}\otimes e_{k+1+2kj_d}\otimes e_{k+2+2kj_d}\otimes\cdots\otimes e_{2k+2kj_d})$$

and by linearity

$$\langle v, p^{\otimes d}\rangle = \sum_{i_1,j_1,\ldots,i_d,j_d=0}^{d-1} x_{i_1+1,j_1+1}\cdots x_{i_d+1,j_d+1} \\ \langle v, (e_{1+2ki_1}\otimes e_{2+2ki_1}\otimes\cdots\otimes e_{k+2ki_1}\otimes e_{k+1+2kj_1}\otimes e_{k+2+2kj_1}\otimes\cdots\otimes e_{2k+2kj_1}) \\ \otimes\cdots\otimes(e_{1+2ki_d}\otimes e_{2+2ki_d}\otimes\cdots\otimes e_{k+2ki_d}\otimes e_{k+1+2kj_d}\otimes e_{k+2+2kj_d}\otimes\cdots\otimes e_{2k+2kj_d})\rangle$$

A crucial property of v is that $\langle v, e_{\pi(1)}\otimes e_{\pi(2)}\otimes\cdots\otimes e_{\pi(n)}\rangle \neq 0$ iff π is a permutation, in which case it is equal to the sign of the permutation. It follows that the nonzero summands in $\langle v, p^{\otimes d}\rangle$ are precisely those for which $i = (i_1,\ldots,i_d)$ and $j = (j_1,\ldots,j_d)$ are permutations of $\{0,\ldots,d-1\}$. For a single summand with i and j permutations we see:

$$x_{i_1+1,j_1+1}\cdots x_{i_d+1,j_d+1} \\ \langle v, (e_{1+2ki_1}\otimes e_{2+2ki_1}\otimes\cdots\otimes e_{k+2ki_1}\otimes e_{k+1+2kj_1}\otimes e_{k+2+2kj_1}\otimes\cdots\otimes e_{2k+2kj_1}) \\ \otimes\cdots\otimes(e_{1+2ki_d}\otimes e_{2+2ki_d}\otimes\cdots\otimes e_{k+2ki_d}\otimes e_{k+1+2kj_d}\otimes e_{k+2+2kj_d}\otimes\cdots\otimes e_{2k+2kj_d})\rangle \\ = \operatorname{sgn}(i)^k\operatorname{sgn}(j)^k x_{i_1+1,j_1+1}\cdots x_{i_d+1,j_d+1}.$$

Hence, for even k we obtain $\langle v, p^{\otimes d}\rangle = d!\operatorname{Per}_d$. ◀

Finally, we put together the preceding results to prove Theorem 11.

▶ **Theorem 25** (Theorem 11, restated). *Let $k \geq 2$ be even. Consider the action of $G = \mathrm{SL}_{2kn}(\mathbb{C})$ on $V = \bigotimes^{2k}\mathbb{C}^{2kn}$. Then any set of generators for the invariant ring cannot have a polynomial sized (in n) succinct encoding, unless* VP = VNP.

Proof. We summarize the results so far. Let $k \geq 2$. Consider the action of $G = \mathrm{SL}_{2kn}(\mathbb{C})$ on $V = \bigotimes^{2k}\mathbb{C}^{2kn}$. Then:

1. There are no homogeneous invariant polynomials of degree $< n$.
2. The space of homogeneous invariant polynomials of degree n is 1-dimensional, and spanned by the hyperpfaffian polynomial $\mathrm{Pf}_{k,n}$.
3. The hyperpfaffian polynomial $\mathrm{Pf}_{k,n}$ is VNP-complete.

The rest of the proof proceeds along the same lines as in the proof of Theorem 8 in Section 3. If we had a poly-sized succinct encoding for the generators of this invariant ring, then one would be able to extract the lowest degree part, which would yield a poly-sized circuit computing $\mathrm{Pf}_{k,n}$. This is not possible unless VP = VNP, since $\mathrm{Pf}_{k,n}$ is VNP-complete. ◀

References

1 Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. *Journal of the ACM (JACM)*, 50(4):429–443, 2003.

2 Zeyuan Allen-Zhu, Ankit Garg, Yuanzhi Li, Rafael Oliveira, and Avi Wigderson. Operator scaling via geodesically convex optimization, invariant theory and polynomial identity testing. *STOC*, 2018.

3 Alexander I. Barvinok. New algorithms for linear k-matroid intersection and matroid k-parity problems. *Mathematical Programming*, 69(1):449–470, 1995.

4 Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1193–1206, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188832.

5 Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Towards a theory of non-commutative optimization: Geodesic 1st and 2nd order methods for moment maps and polytopes. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 845–861. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00055.

6 Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Efficient algorithms for tensor scaling, quantum marginals, and moment polytopes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 883–897. IEEE, 2018.

7 Peter Bürgisser, Ankit Garg, Rafael Oliveira, Michael Walter, and Avi Wigderson. Alternating minimization, scaling algorithms, and the null-cone problem from invariant theory. *Proceedings of Innovations in Theoretical Computer Science (ITCS 2018)*, 2017. arXiv:1711.08039.

8 Peter Bürgisser and Christian Ikenmeyer. Fundamental invariants of orbit closures. *J. Algebra*, 477:390–434, 2017. doi:10.1016/j.jalgebra.2016.12.035.

9 Peter Bürgisser, Christian Ikenmeyer, and Greta Panova. No occurrence obstructions in geometric complexity theory. *J. Amer. Math. Soc.*, 32(1):163–193, 2019. doi:10.1090/jams/908.

10 Harm Derksen and Gregor Kemper. *Computational invariant theory*, volume 130 of *Encyclopaedia of Mathematical Sciences*. Springer, Heidelberg, enlarged edition, 2015. With two appendices by Vladimir L. Popov, and an addendum by Norbert A'Campo and Popov, Invariant Theory and Algebraic Transformation Groups, VIII. doi:10.1007/978-3-662-48422-7.

11 Harm Derksen and Visu Makam. Generating invariant rings of quivers in arbitrary characteristic. *J. Algebra*, 489:435–445, 2017. doi:10.1016/j.jalgebra.2017.06.035.

12 Harm Derksen and Visu Makam. Polynomial degree bounds for matrix semi-invariants. *Adv. Math.*, 310:44–63, 2017. doi:10.1016/j.aim.2017.01.018.

13 Harm Derksen and Visu Makam. Algorithms for orbit closure separation for invariants and semi-invariants of matrices. *arXiv e-prints*, January 2018. arXiv:1801.02043.

14 Harm Derksen and Visu Makam. An exponential lower bound for the degrees of invariants of cubic forms and tensor actions. *Advances in Mathematics*, 368:107136, 2020. doi:10.1016/j.aim.2020.107136.

15 Harm Derksen and Jerzy Weyman. Semi-invariants of quivers and saturation for littlewood-richardson coefficients. *Journal of the American Mathematical Society*, 13(3):467–479, 2000.

16 Mátyás Domokos and Alexander N Zubkov. Semi-invariants of quivers as determinants. *Transformation groups*, 6(1):9–24, 2001.

17 Michael A. Forbes and Amir Shpilka. Explicit Noether normalization for simultaneous conjugation via polynomial identity testing. In *Approximation, randomization, and combinatorial optimization*, volume 8096 of *Lecture Notes in Comput. Sci.*, pages 527–542. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40328-6_37.

18 William Fulton and Joe Harris. *Representation theory*, volume 129 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1991. A first course, Readings in Mathematics. doi:10.1007/978-1-4612-0979-9.

19 Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. A deterministic polynomial time algorithm for non-commutative rational identity testing. In *57th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2016*, pages 109–117. IEEE Computer Soc., Los Alamitos, CA, 2016.

20 Ankit Garg, Leonid Gurvits, Rafael Oliveira, and Avi Wigderson. Algorithmic and optimization aspects of brascamp-lieb inequalities, via operator scaling. *Geometric and Functional Analysis*, 28(1):100–145, 2018.

21 Ankit Garg, Visu Makam, Rafael Oliveira, and Avi Wigderson. Search problems in algebraic complexity, GCT, and hardness of generator for invariant rings. *arXiv e-prints*, October 2019. arXiv:1910.01251v1.

22 Joshua A Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 110–119. IEEE, 2014.

23 Joshua A. Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. *CoRR*, abs/1907.00309, 2019. arXiv:1907.00309.

24 David Hilbert. Ueber die Theorie der algebraischen Formen. *Math. Ann.*, 36(4):473–534, 1890. doi:10.1007/BF01208503.

25 David Hilbert. Ueber die vollen Invariantensysteme. *Math. Ann.*, 42(3):313–373, 1893. doi:10.1007/BF01444162.

26 Christian Ikenmeyer. *Geometric Complexity Theory, Tensor Rank, and Littlewood-Richardson Coefficients*. PhD thesis, Institute of Mathematics, University of Paderborn, 2012. Online available at http://digital.ub.uni-paderborn.de/ubpb/urn/urn:nbn:de:hbz:466:2-10472.

27 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *Comput. Complexity*, 27(4):561–593, 2018. doi:10.1007/s00037-018-0165-7.

28 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

29 Jan Krajíček. *Proof complexity*, volume 170. Cambridge University Press, 2019.

30 Visu Makam and Avi Wigderson. Singular tuples of matrices is not a null cone (and, the symmetries of algebraic varieties). *arXiv e-prints*, September 2019. arXiv:1909.00857.

31 Ketan D. Mulmuley. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017. doi:10.1090/jams/864.

32 Ketan D. Mulmuley and Milind Sohoni. Geometric complexity theory. I. An approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526, 2001. doi:10.1137/S009753970038715X.

33 Ketan D. Mulmuley and Milind Sohoni. Geometric complexity theory. II. Towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.*, 38(3):1175–1206, 2008. doi:10.1137/080718115.

34 David Mumford. *Geometric invariant theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete, Neue Folge, Band 34. Springer-Verlag, Berlin-New York, 1965.

35 Aidan Schofield and Michel Van den Bergh. Semi-invariants of quivers for arbitrary dimension vectors. *Indagationes Mathematicae*, 12(1):125–138, 2001.

36 Jacob T Schwartz. Probabilistic algorithms for verification of polynomial identities. In *International Symposium on Symbolic and Algebraic Manipulation*, pages 200–215. Springer, 1979.

37 Amir Shpilka, Amir Yehudayoff, et al. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010.

38 Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

39 Bernd Sturmfels. *Algorithms in Invariant Theory*. Texts & Monographs in Symbolic Computation. Springer, 2nd edition, 2008.

40 David L. Wehlau. Constructive invariant theory for tori. *Ann. Inst. Fourier (Grenoble)*, 43(4):1055–1066, 1993. URL: http://www.numdam.org/item?id=AIF_1993__43_4_1055_0.

Statistical Physics Approaches to Unique Games

Matthew Coulson
Department of Mathematics, Universitat Politècnica de Catalunya, Barcelona, Spain
Matthew.John.Coulson@upc.edu

Ewan Davies
Department of Computer Science, University of Colorado, Boulder, CO, USA
Ewan.Davies@colorado.edu

Alexandra Kolla
Department of Computer Science, University of Colorado, Boulder, CO, USA
Alexandra.Kolla@colorado.edu

Viresh Patel
Korteweg-de Vries Institute for Mathematics, University of Amsterdam, The Netherlands
V.S.Patel@uva.nl

Guus Regts
Korteweg-de Vries Institute for Mathematics, University of Amsterdam, The Netherlands
G.Regts@uva.nl

Abstract

We show how two techniques from statistical physics can be adapted to solve a variant of the notorious Unique Games problem, potentially opening new avenues towards the Unique Games Conjecture. The variant, which we call Count Unique Games, is a promise problem in which the "yes" case guarantees a certain number of highly satisfiable assignments to the Unique Games instance. In the standard Unique Games problem, the "yes" case only guarantees at least one such assignment. We exhibit efficient algorithms for Count Unique Games based on approximating a suitable partition function for the Unique Games instance via (i) a zero-free region and polynomial interpolation, and (ii) the cluster expansion. We also show that a modest improvement to the parameters for which we give results would be strong negative evidence for the truth of the Unique Games Conjecture.

2012 ACM Subject Classification Mathematics of computing → Approximation algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Algorithm design techniques

Keywords and phrases Unique Games Conjecture, approximation algorithm, Potts model, cluster expansion, polynomial interpolation

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.13

Related Version A full version of the paper is available at `http://arxiv.org/abs/1911.01504`.

Funding Part of this work was done while E. Davies, A. Kolla, and G. Regts were visiting the Simons Institute for the Theory of Computing. Part of this work was done while M. Coulson was visiting V. Patel and G. Regts at the University of Amsterdam.
Matthew Coulson: Visit to the University of Amsterdam funded by a Universitas 21 Travel Grant from the University of Birmingham. Supported by the Spanish Ministerio de Economía y Competitividad project MTM2017–82166–P and an FPI Predoctoral Fellowship from the Spanish Ministerio de Economía y Competitividad with reference PRE2018–083621.
Alexandra Kolla: Partially supported by NSF Career award 1452923.
Viresh Patel: Partially supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

Acknowledgements We would like to thank Tyler Helmuth for insightful discussions, and the anonymous referees for comments.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 13; pp. 13:1–13:27

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

1 Introduction

Over the last two decades, the Unique Games Problem has emerged as an obstacle to the approximability of many combinatorial optimization problems. More precisely, the Unique Games Conjecture (UGC) states that there is no polynomial-time algorithm to solve the Unique Games Problem within a certain performance guarantee. If the UGC is true, the current best-known approximation algorithms for many problems such as Min-2Sat-Deletion [19], Vertex Cover [21], Max-Cut [20] and Non-Uniform Sparsest Cut [9, 22] are in fact optimal. On the other hand, falsification of the conjecture is likely to provide powerful new algorithmic techniques that apply to many important computational problems. The fact that either resolution of the conjecture could be an important advance in the understanding of approximation algorithms and complexity theory is one reason why the UGC has played a key role in recent theoretical computer science research.

Our main contribution is a pair of algorithms, each deeply rooted in ideas from statistical physics, that solve a natural variant of the Unique Games Problem. We give some important definitions before the statement of these results.

▶ **Definition 1.** *In a Unique Games problem we are given a constraint graph $G = (V, E)$, a set of colours $[k] = \{1, \dots, k\}$ which is referred to as the alphabet, a set of variables $\{x_u\}_{u \in V}$, one for each vertex u, and a set of permutations (also known as constraints) $\pi_{uv} : [k] \to [k]$, one for each edge $uv \in E$. We study assignments giving a colour from $[k]$ to each variable x_u, and are interested in the number of* satisfied edges *(or satisfied constraints) of the form*[1] *$\pi_{uv}(x_u) = x_v$. The* value *of the assignment is the fraction of satisfied constraints. The* value *of the Unique Games instance is the maximum fraction of constraints that can be simultaneously satisfied.*

We denote by $UG(k, \varepsilon, \delta)$ the promise problem consisting of a Unique Games instance with alphabet size k and the promise that the instance either has value at least $1 - \varepsilon$, or has value at most δ. To solve the problem is to correctly determine which of the two cases hold.

When the parameters are unimportant or clear from context they are omitted, and we will always be interested in $\varepsilon, \delta \geq 0$ with $1 - \varepsilon > \delta$, otherwise the problem is ill-posed or impossible to solve. The Unique Games Conjecture of Khot [19] can now be stated as follows.

▶ **Conjecture** (UGC). *For any constants $\varepsilon, \delta > 0$ with $1 - \varepsilon > \delta$, there is a positive $k(\varepsilon, \delta)$ such that for any alphabet size $k > k(\varepsilon, \delta)$, the problem $UG(k, \varepsilon, \delta)$ is NP-hard.*

We note that in [11, 12] it was shown that for any $\delta > 0$, for all large enough k the problem $\mathrm{UG}(k, 1/2 - \delta, \delta)$ is NP-hard.

Upon translating a Unique Games problem into a form amenable to methods from statistical physics, which we elaborate upon later, a natural variant of the problem arises.

▶ **Definition 2.** *Count Unique Games, or $CUG(f, k, \varepsilon, \delta)$, is the promise problem consisting of a Unique Games instance with alphabet size k and the promise that the instance either has at least $(fk)^{|V|}$ colourings with value at least $1 - \varepsilon$, or that every colouring has value at most δ. To solve the problem is to correctly determine which of the two cases hold, and when the parameters are clear from context they are omitted.*

---

[1] Formally, let G be an oriented graph so each edge has a direction. Then the edge (or constraint) $u \to v$ is satisfied when $\pi_{uv}(x_u) = x_v$, or equivalently $\pi_{uv}^{-1}(x_v) = x_u$. Without orientations it is unspecified whether to use π_{uv} or π_{uv}^{-1} here. We suppress this detail as in other parts of the paper it is more natural to consider undirected graphs.

Note that with $f = k^{-1}$, CUG$(f, k, \varepsilon, \delta)$ corresponds exactly to UG(k, ε, δ); and with $f = 1$ the problem is easily solvable as the guarantee covers all colourings in both cases. To solve CUG$(1, k, \varepsilon, \delta)$ simply requires checking the value of an arbitrary colouring. So via the parameter f CUG can be smoothly reduced in difficulty from equal to UG to trivial. We can now state our main results.

▶ **Theorem 3.** *For $\varepsilon, \delta > 0$ with $1 - \varepsilon > \delta$ and any fixed integer $k \geq 3$, there exists $\Delta_0(\varepsilon, \delta, k)$ such that for all $\Delta \geq \Delta_0$ the following holds.*

Let G be an n-vertex, Δ-regular CUG$(f, k, \varepsilon, \delta)$ instance with $f \geq k^{\varepsilon+\delta-\frac{1}{2}}$. Then there is a deterministic algorithm that solves CUG for G in time

$$n \exp\left(e^{O(\log^2 k)} \log^2 \Delta\right).$$

▶ **Theorem 4.** *For $\varepsilon, \delta > 0$ with $1 - \varepsilon > \delta$ and any k satisfying $\log k \geq \Delta^{3/2} \log \Delta$, there exists $\Delta_0(\varepsilon, \delta)$ such that for all $\Delta \geq \Delta_0$ the following holds.*

Let G be an n-vertex, Δ-regular CUG$(f, k, \varepsilon, \delta)$ instance with $f \geq k^{2\varepsilon+2\delta-1}$. Then there is a deterministic algorithm that solves CUG for G in time $kn^{O(1)}e^{O(\Delta)}$.

To illustrate how close Theorem 4 gets to an algorithm for usual Unique Games, consider an n-vertex, Δ-regular instance of UG and suppose there exists an assignment of value $1 - \varepsilon$. Let S be an arbitrary set of εn vertices, and consider all assignments obtained by relabelling vertices in S. There are at most $\varepsilon\Delta n$ constraints that could be violated by modifying the labels of vertices in S, hence each such assignment has value at least $1 - 3\varepsilon$. Thus there are at least $k^{\varepsilon n}$ assignments of value $1 - 3\varepsilon$, which corresponds to having parameter $f = k^{\varepsilon-1}$ as a CUG instance. In summary, if we were permitted to take $f = k^{\varepsilon/3-1}$ in Theorem 4, which is only slightly smaller than the stated $f \geq k^{2\varepsilon+2\delta-1}$, then we would be able to refute the Unique Games Conjecture.

The idea at the heart of both Theorem 3 and Theorem 4 is to encode a CUG problem as the problem of approximating the value of a *partition function* $Z(G; w)$ which depends on the instance G, and is a polynomial in w. The partition function is intimately connected to statistical physics, and we use two techniques recently developed for approximating partition functions to prove our two main results. Theorem 3 is proved via *polynomial interpolation*, a method due to Barvinok (see [4] and references therein) and furthered by Patel and Regts [26] who improved the running time in many examples. Theorem 4 is proved via the *cluster expansion* and methods given in [8]. These techniques have recently been developed and applied to the problem of approximating the partition function of the Potts model [4, 6, 8, 24], random cluster model [8], and other models from statistical physics [4, 17, 26]. The main conceptual advances in this paper are demonstrations that these techniques may be adapted to UG instances, cleanly handling the constraints assigned to edges that are not present in the standard Potts model. The main technical advance in this paper is a zero-free region for (a generalization of) the ferromagnetic Potts model partition function, that may be of independent interest. See Theorem 10 below.

▶ Remark 5. We note that in the majority of interesting cases for the Unique Games conjecture, the degree Δ of the constraint graphs is not very large (e.g. the hypercube), so by considering Δ polylogarithmic in n, we get a quasi-polynomial time algorithm from Theorem 4.

1.1 Paper organization

In the following subsection we summarize related work on the UGC. In Section 2 we define our partition function and relate it to solving UG problems. This involves stating our algorithmic results for approximating the partition function, and proving that our main results follow from these algorithms. In Sections 3 and 4 we discuss approximation algorithms for our partition function with the polynomial interpolation and cluster expansion methods respectively. Certain details are deferred to the appendices. Finally, we discuss an important open problem arising from our work and identify plausible barriers to improving our methods in Section 5.

1.2 Related work

An intimate connection between the UGC and semidefinite programming (SDP) can be traced back to a seminal paper by Goemans and Williamson [15] on the Max-Cut problem. An instance of Max-Cut can be seen as a system of linear equations over $\mathbb{Z}_2$, and thus it is a Unique Games instance with alphabet size two. Goemans and Williamson gave an SDP-based algorithm for Max-Cut which, on inputs where the maximal cut is of size $1-\varepsilon$, produces a cut that satisfies at least a fraction $1-(2/\pi)\sqrt{\varepsilon}$ of the constraints. A matching integrality gap was found by [18] and [13], and in [20] it was proven that if the UGC is correct, then the Goemans–Williamson algorithm has the best approximation ratio that a polynomial-time algorithm for Max-Cut can achieve. Raghavendra [29] proved that for every constraint satisfaction problem there is a polynomial time, semidefinite programming-based algorithm which, if the UGC is true, achieves the best possible approximation ratio for the problem. These results cement the central role of the UGC in the theory of approximation algorithms.

There are also spectral algorithms that give good polynomial-time or quasi-polynomial-time approximations algorithms for large classes of Unique Games instances. These include expanders [3, 25], local expanders [2, 30], and more generally, graphs with few large eigenvalues [23]. In [1], the authors gave a general sub-exponential algorithm for Unique Games based on spectral techniques.

In contrast to previous approaches to refuting the UGC, our methods use techniques from statistical physics and naturally lead to the consideration of CUG. The strengthened promise of CUG connects to an active area of research for other computational problems such as Sat. With no assumptions finding a satisfying assignment for a 3CNF-formula is NP-hard, but how fast can we find a satisfying assignment under the assumption that many exist? When a constant fraction of the possible assignments satisfy the formula, simply trying random assignments performs quite well; and it is an intriguing problem to match this performance with a deterministic algorithm. Servedio and Tan [31] gave such an algorithm that uses a deterministic algorithm for approximating the number of satisfying assignments of a formula as a key building block. We note that deterministic approximate counting is at the heart of our methods too.

2 Solving a Unique Games problem with a partition function

In this section we define a partition function $Z(G;w)$ and describe how to solve CUG instances via knowledge of the partition function, leading to proofs of Theorems 3 and 4. We also observe how CUG naturally arises from UG in this context.

A partition function is a mathematical object that encodes as a polynomial some weighted substructures in a graph. The general definition arises in the statistical physics of spin systems, and important examples include the independence polynomial and matching polynomials of a graph, see e.g. [4]. Here we will only describe the partition function we define to study Unique Games instances, which is closely related to the Potts and random cluster models.

▶ **Definition 6.** *Given a Unique Games instance $G = (V, E, \pi)$, the partition function $Z(G; w)$ is a polynomial in a parameter $w \in \mathbb{C}$ given as a sum of terms w^i for each colouring of the graph with the alphabet $[k]$ that has i satisfied constraints (i.e. that has value $i/|E|$). That is,*

$$Z(G; w) := \sum_{\{x_u\}_{u\in V}\in[k]^V} \prod_{\substack{(u,v)\in E,\\ x_v=\pi_{uv}(x_u)}} w,$$

where the sum is over all assignments of colours in $[k]$ to the labels $\{x_u\}_{u\in V}$.

If the permutations π_{uv} are all the identity, then a satisfied constraint corresponds to a *monochromatic edge*: both endpoints of the edge received the same colour. In this case the above $Z(G; w)$ corresponds to the partition function of the Potts model from statistical physics. The relevance to Unique Games problems arises from the fact that the cases of the promise in (C)UG give contrasting bounds on $Z(G; w)$. When $w \geq 1$ is real, in the case that a highly satisfying assignment is guaranteed to exist we have a lower bound, and in the case that no highly satisfying assignments exist we have an upper bound. When the upper bound is less than the lower bound, at most one of the bounds can hold for any given instance, so knowledge of $Z(G; w)$ immediately solves the problem. If the bounds are sufficiently far apart an approximate value of $Z(G; w)$ suffices.

▶ **Lemma 7.** *Consider an instance $G = (V, E)$ of $CUG(f, k, \varepsilon, \delta)$, and let $\alpha > 0$. Then to solve the instance it suffices to know any value ξ satisfying $e^{-\alpha} \leq Z(G; w)/\xi \leq e^{\alpha}$ for any real w such that*

$$\log w > \frac{|V|\log(1/f) + 2\alpha}{(1-\varepsilon-\delta)|E|}.$$

In the case that G has average degree Δ (so $2|E| = \Delta|V|$), and $\alpha = C|V|$ this becomes

$$\log w > \frac{2}{1-\varepsilon-\delta}\frac{\log(1/f) + 2C}{\Delta}.$$

Proof. Consider any real $w \geq 1$. Then if there are $(fk)^{|V|}$ colourings of G with value $1-\varepsilon$ we have

$$e^{\alpha}\xi \geq Z(G; w) \geq (fk)^{|V|}w^{(1-\varepsilon)|E|},$$

and if every colouring has value at most δ we have

$$e^{-\alpha}\xi \leq Z(G; w) \leq k^{|V|}w^{\delta|E|}.$$

Since these bounds go in opposite directions, knowledge of ξ immediately yields a solution to the problem when the implied intervals for ξ are disjoint. This occurs precisely for w as in the statement of the lemma. ◀

We now see that increasing f allows CUG instances to be solved via $Z(G;w)$ for smaller w, and this is how the Count variant of Unique Games naturally arises. We are unable to approximate $Z(G;w)$ with w large enough to permit $f = 1/k$ (i.e. refute UGC), but by slightly increasing f we bring w into a range amenable to our methods. In Section 5 we discuss the problem of how large f can be while CUG(f) is still equivalent to UG, and identify natural barriers to using algorithms for larger w. To obtain Theorems 3 and 4 we need a pair of algorithms and some calculations.

▶ **Theorem 8.** *Let $k \in \mathbb{N}_{\geq 3}$, $\Delta \in \mathbb{N}_{\geq 3}$, and $w^* = 1 + (\log k - 1)/\Delta$. Then there exists a deterministic algorithm which, given α satisfying $0 < \alpha < \frac{2n}{e\Delta} e^{O(\log^2 k)}$, and an n-vertex UG(k) instance G of maximum degree at most Δ, computes a number ξ satisfying $e^{-\alpha} \leq Z(G;w^*)/\xi \leq e^{\alpha}$ in time bounded by*

$$n \exp\left(e^{O(\log^2 k)} \log\left(\frac{n\Delta}{\alpha} e^{O(\log^2 k)}\right) \log(\Delta\sqrt{k})\right).$$

▶ **Theorem 9.** *Let $\Delta \in \mathbb{N}$ and let $\zeta = 8/\sqrt{\Delta}$. For $k \geq \exp\left((18\Delta + 4\Delta \log \Delta)/\zeta\right)$ and $w^* = \exp((2-\zeta)\log(k)/\Delta)$ there exists an deterministic algorithm, which given $\alpha > 0$ and an n-vertex UG(k) instance G of maximum degree at most Δ, computes ξ satisfying $e^{-\alpha} \leq |Z(G;w^*)/\xi| \leq e^{\alpha}$ in time bounded by $kn^{O(1)}(n/\alpha)^{O(\Delta)}$.*

Note that we write UG(k) above as neither algorithm depends on value or the promise on the value of the instance (and hence the values of ε and δ are irrelevant). These parameters feature in the application of these algorithms to prove our main results. The above results are proved in Sections 3 and 4 respectively. In both cases we define a series for $\log Z(G;w)$, show that it converges, and obtain an additive approximation to it by evaluating a truncation of the series. Here we give the calculations that show what CUG problems we can solve with these algorithms.

Proof of Theorem 3. Take w^* as in Theorem 8. As $\Delta \to \infty$, and with approximation error $\alpha = Cn$ for any $C < 2e^{O(\log^2 k)-1}/\Delta$, by Lemma 7 we require

$$\log w^* = (1 - o(1))\frac{\log k}{\Delta} > \frac{2}{1-\varepsilon-\delta}\frac{\log(1/f) + 2C}{\Delta}.$$

For large enough Δ, $C = \log(k)/\Delta$ is valid in Theorem 8 and implies the above for any

$$f \geq k^{\varepsilon+\delta-\frac{1}{2}}.$$ ◀

We remark that a very similar calculation gives a result for k growing with Δ, but we present the special case of constant k here as it is instructive of our methods and permits a concise expression for f and the running time.

Proof of Theorem 4. Take w^* as in Theorem 9. With $\Delta \geq e^{9/2}$, $\zeta = 8/\sqrt{\Delta}$, $k \geq \Delta^{\Delta^{3/2}}$, and $\alpha = Cn$ for some $C > 0$ we choose later, by Lemma 7 we require

$$\log w = (2-\zeta)\frac{\log k}{\Delta} > \frac{2}{1-\varepsilon-\delta}\frac{\log(1/f) + 2C}{\Delta},$$

which holds when

$$f > e^{2C} k^{-\frac{1}{2}(2-\zeta)(1-\varepsilon-\delta)}.$$

With $\Delta \geq \Delta_0(\varepsilon,\delta)$ and $C \leq \frac{1}{4}(\varepsilon+\delta)\log k$, this holds for $f \geq k^{2\varepsilon+2\delta-1}$. For large enough Δ_0 (which makes k sufficiently large) we can take e.g. $C = 1/2$ and obtain a running time of $kn^{O(1)}e^{O(\Delta)}$. ◀

3 Polynomial interpolation

The proof of Theorem 8 proceeds by an influential method known as polynomial interpolation introduced by Barvinok, see e.g. [4]. In our application of this method, for some real $w^* > 0$ we show that there is a region $\mathcal{U} \subset \mathbb{C}$ containing the interval $[1, w^*]$ on which $Z(G; w) \neq 0$ for any UG instance G of maximum degree Δ. The region $\mathcal{U}$ is a *zero-free region*, and it guarantees that a Taylor series for (a suitable modification of) $\log Z$ converges inside $\mathcal{U}$. We then approximate Z by computing the coefficients of a truncation of the Taylor series. We require that $\mathcal{U}$ is independent of the size of the graph G to obtain a good approximation. The analysis yielding the approximation from $\mathcal{U}$ is rather standard, e.g. [4, 5, 26], though we include it in Appendix A for completeness. The main technical work is in the following theorem establishing the region $\mathcal{U}$. We write $\mathcal{N}(S, \eta)$ for an open set in $\mathbb{C}$ containing the open ball of radius η around every point in S.

▶ **Theorem 10.** *Let $k \in \mathbb{N}_{\geq 3}$ and $\Delta \in \mathbb{N}_{\geq 3}$. Then with $w^* = 1 + (\log k - 1)/\Delta$ there exists $\eta = \omega(\frac{1}{\Delta \log k})$ such that for any $w \in \mathcal{N}([1, w^*], \eta)$ and any UG(k) instance G of maximum degree at most Δ, $Z(G; w) \neq 0$.*

Our proof of Theorem 10 is inductive in the style of [6] which gives a zero-free region for the antiferromagnetic Potts model, though here we have a generalization of the *ferromagnetic* Potts model rather than than the *antiferromagnetic* Potts model that was studied in [6]. We give a proof sketch here and defer the full details to Appendix B.

Consider an n-vertex UG(k) instance with $G = (V, E, \pi)$ with maximum degree Δ. In order to prove our results, we will need to work more generally with the partition function with boundary conditions. For $m > 0$ and a list $W = w_1 \dots w_m$ of distinct vertices of V and a list $L = \ell_1 \dots \ell_m$ of pre-assigned colours in $[k]$ for the vertices in W the *restricted partition function* $Z_L^W(G; w)$ is defined by

$$Z_L^W(G; w) := \sum_{\substack{\{x_u\}_{u \in V} \in [k]^V \\ \{x_u\}_{u \in V} \text{ respects } (W, L)}} \prod_{\substack{(u,v) \in E, \\ x_v = \pi_{uv}(x_u)}} w,$$

where we say that a colour assignment $\{x_u\}_{u \in V}$ respects (W, L) if for all $i = 1 \dots, m$ we have $x_{w_i} = \ell_i$. As it does not vary in the steps of the proof, we will omit the parameter w and write $Z_L^W(G)$ for $Z_L^W(G; w)$. We call the vertices $w_1, \dots, w_m$ *fixed* and refer to the remaining vertices in V as *free* vertices. The length of W (resp. L), written $|W|$ (resp. $|L|$) is the length of the list. Given a list of distinct vertices $W' = w_1 \dots w_m$, and a vertex u (distinct from $w_1, \dots, w_m$) we write $W = W'u$ for the concatenated list $W = w_1 \dots w_m u$ and we use similar notation $L'\ell$ for concatenation of lists of colours. We write $\deg(v)$ for the degree of a vertex v and we write $G \setminus uv$ ($G - u$) for the graph obtained from G by removing the edge uv (by removing the vertex u).

To prove Theorem 10 we consider the same statement for restricted partition functions and induct over the number of vertices whose colour is not fixed by the boundary conditions. With a strengthened induction hypothesis we can argue that unfixing the specified colour of a single vertex cannot affect the value of the partition function too much and continue the induction. The main technical difficulties are to bound the change in angle and radius unfixing a vertex can induce in the value of the partition function (as a complex number).

▶ **Lemma 11.** *Let $\Delta \in \mathbb{N}_{\geq 3}$ and let $k \in \mathbb{N}_{\geq 3}$. Let $c = \log k - 1$ and $\alpha = \log k^{1/2} - 1$. Then there exists constants $0 < \varepsilon < \theta < \frac{\pi}{3\Delta}$ with $\varepsilon, \theta = \omega(1/\Delta)$ and $\eta = \omega(1/(\Delta \log k))$ such that for any $w \in \mathcal{N}([1, 1 + c/\Delta], \eta)$ and any UG(k) instance G of maximum degree at most Δ the following hold.*

1. *For all lists W of distinct vertices of G and all lists of pre-assigned colours L of length $|W|$, $Z_L^W(G) \neq 0$.*
2. *For all lists $W = W'u$ of distinct vertices of G such that u is a leaf and any two lists $L'l$, $L'l'$ of length $|W|$, the following hold.*
   a. *If the unique neighbour v of u is free,*
      i. *the angle between vectors $Z_{L'l}^{W'u}(G)$ and $Z_{L'l'}^{W'u}(G)$ is at most θ, and*
      ii. $\dfrac{Z_{L'l}^{W'u}(G)}{Z_{L'l'}^{W'u}(G)} \leq 1 + \dfrac{\alpha}{\Delta}$.
   b. *If the unique neighbour v of u is fixed,*
      i. *the angle between vectors $Z_{L'l}^{W'u}(G)$ and $Z_{L'l'}^{W'u}(G)$ is at most ε, and*
      ii. $\dfrac{|Z_{L'l}^{W'u}(G)|}{|Z_{L'l'}^{W'u}(G)|} \leq 1 + \dfrac{c}{\Delta}$.
3. *For all lists $W = W'u$ of distinct vertices of G, and for all lists of pre-assigned colours L' of length $|W'|$, let d be the number of free neighbours of u, and let $b = \Delta - d$. Then for any pair of colours l, l',*
   a. *the angle between vectors $Z_{L'l}^{W'u}(G)$ and $Z_{L'l'}^{W'u}(G)$ is at most $d\theta + b\varepsilon$, and*
   b. $\dfrac{|Z_{L'l}^{W'u}(G)|}{|Z_{L'l'}^{W'u}(G)|} \leq (1 + \alpha/\Delta)^d (1 + c/\Delta)^{\Delta - d}$.

Note that Statement 1 with $W = L = \emptyset$ is the result we want for Theorem 10, Statement 2 shows that changing the fixed colour of a leaf (degree 1) vertex u affects the angle and length of the restricted partition function by a small amount (depending on whether the neighbour of u is itself free or fixed), and Statement 3 is a similar but weaker version for any vertex. We give the proof of Lemma 11 in Appendix B.

4 Cluster expansion

On the surface our proof of Theorem 9 has a similar flavour to the polynomial interpolation method: we define a series expansion for $\log Z(G; w)$, show that it converges, and approximate $Z(G; w)$ by computing the coefficients of a truncation of the series. Instead of working with a Taylor series and a zero-free region, we work with a different formal power series for $\log Z$ called the *cluster expansion* which expresses $\log Z$ as a sum involving weights given to connected subgraphs of G. This technique was recently applied to approximating the partition functions of the Potts and random cluster models in [8], where the *random cluster model* is a random graph model from statistical physics that generalizes the Ising and Potts models, and the concept of percolation[2]. To obtain the result we adapt a standard reduction to express our partition function $Z(G; w)$ in terms of the random cluster model, and apply the method of [8] which gives an approximation algorithm via the cluster expansion.

4.1 The random cluster model

The random cluster model, instead of counting graph labellings according to satisfied edges, counts connected subgraphs according to some weights. We adapt the standard reduction comparing the Potts model and random cluster model partition functions to our $Z(G; w)$ for Unique Games instances.

[2] Note the distinct uses of the term "cluster" in "cluster expansion" and "random cluster model", though there is a common theme of connected subgraphs in both uses.

We start by rewriting $Z(G;w)$ for a given instance $G=(V,E,\pi)$. Let (V',E') be a connected component of G, so that V' is a nonempty subset of V and $E'\subset E\cap\binom{V'}{2}$. We consider a singleton vertex $\{u\}$ to comprise the connected component $(\{u\},\emptyset)$. Define

$$\mathrm{sat}_\pi(V',E') := \sum_{\{x_u\}_{u\in V'}\in[k]^{V'}} \prod_{(u,v)\in E'} \mathbf{1}\{x_v=\pi_{uv}(x_u)\},$$

where $\mathbf{1}\{P\}=1$ if P is true, and 0 otherwise. In other words $\mathrm{sat}_\pi(V',F')$ counts the number of assignments of value 1 (perfectly satisfying assignments) of the Unique Games instance restricted to the subgraph (V',E'). The definition means that $\mathrm{sat}_\pi(\{u\},\emptyset)=k$ as there are no constraints and the empty product is 1. Since we work with (V',E') connected, we also have

$$0\le \mathrm{sat}_\pi(V',E')\le k, \tag{1}$$

as given any starting colour for an arbitrary first vertex $u\in V'$, there is at most one completion of the colouring to a perfectly satisfying assignment obtained by following the constraints out along the component from u.

We use the notation $\mathcal{C}(V,F)$ for the set of connected components of the graph (V,F), taken as pairs (V',E') with $E'\subset F$. The following lemma gives the reduction from $Z(G;w)$ to the random cluster model partition function. The simple proof is given in Appendix C.

▶ **Lemma 12.** *Let $G=(V,E,\pi)$ be a UG instance and $w\in\mathbb{C}$. Then*

$$Z(G;w)=\sum_{F\subseteq E}(w-1)^{|F|}\prod_{(V',E')\in\mathcal{C}(V,F)}\mathrm{sat}_\pi(V',E').$$

4.2 The cluster expansion

We closely follow the notation and setup of [8, 17]. Given a UG instance $G=(V,E,\pi)$, define a *polymer* γ to be a connected subgraph of G with at least two vertices. A collection of polymers is *compatible* if the polymers contained in it are pairwise vertex disjoint. We define the incompatibility graph H_G on the collection of all polymers as follows: vertices of H_G are the polymers and two polymers are connected by an edge if they are not compatible (that is if they share a vertex). Write $|\gamma|:=|V(\gamma)|$, $\|\gamma\|:=|E(\gamma)|$, and given $w\in\mathbb{C}$, define the *weight* of a polymer γ as

$$w_\gamma := (w-1)^{\|\gamma\|}k^{-|\gamma|}\,\mathrm{sat}_\pi(\gamma),$$

where we write $\mathrm{sat}_\pi(\gamma)$ for the more cumbersome $\mathrm{sat}_\pi(V(\gamma),E(\gamma))$. Then by Lemma 12 and the observation that for a single vertex u we have $\mathrm{sat}_\pi(\{u\},\emptyset)=k$, we have

$$\Xi(G) := \sum_{\Gamma=\{\gamma_1,\dots,\gamma_t\}}\prod_{i=1}^{t} w_{\gamma_i} = k^{-|V|}Z(G;w),$$

where the sum is over all sets Γ of (pairwise) compatible polymers. Note that $\Xi(G)$ is the multivariate independence polynomial of the compatibility graph H_G.

The *cluster expansion* is the following formal power series for $\log\Xi(G)$:

$$\log\Xi(G)=\sum_{\substack{\Gamma\subset V(H_G)\\ H_G[\Gamma]\text{ connected}}}\phi(\Gamma)\prod_{\gamma\in\Gamma}w_\gamma, \tag{2}$$

where $\phi(\Gamma)$ is the *Ursell function* of the graph $H_G[\Gamma] = (\Gamma, F)$, defined as

$$\phi(\Gamma) := \frac{1}{|\Gamma|!} \sum_{\substack{A \subseteq F \\ (\Gamma, A) \text{ connected}}} (-1)^{|A|}.$$

For $\Gamma \subset V(H_G)$, let $\|\Gamma\|$ be given by $\|\Gamma\| := \sum_{\gamma \in \Gamma} \|\gamma\|$, and define the truncated cluster expansion as follows

$$T_m := \sum_{\substack{\Gamma \subset V(H_G),\, \|\Gamma\| < m \\ H_G[\Gamma] \text{connected}}} \phi(\Gamma) \prod_{\gamma \in \Gamma} w_\gamma. \tag{3}$$

With the definitions and a reduction to the right partition function in place, we can now state a result essentially proved in [8] that gives convergence of the cluster expansion and an approximation guarantee.

▶ **Lemma 13** (Borgs et al. [8, Lemma 2.1]). *Suppose that polymers are connected subgraphs containing at least one edge of a graph G of maximum degree Δ on n vertices, that $\|\gamma\|$ is the number of edges of the polymer γ, and that*

$$|w_\gamma| \leq e^{-(7+\log \Delta)\|\gamma\|}. \tag{4}$$

Then the cluster expansion converges absolutely and for any $m \in \mathbb{N}$, $|T_m - \log \Xi(G)| \leq n e^{-3m}$.

To prove Theorem 9 we simply check that these conditions hold, which we state as a lemma below. The details are in Appendix C.

▶ **Lemma 14.** *Let $\Delta \in \mathbb{N}_{\geq 16}$, let $C = e^{-9-2\log \Delta}$, and let $\zeta = 8\sqrt{1/\Delta}$. Then if $k \geq C^{-2\Delta/\zeta}$ and $1 \leq w \leq e^{(2-\zeta)\log(k)/\Delta}$, Lemma 13 holds for UG($k$) instances G of maximum degree Δ.*

We deduce the following runtime guarantees from our setup and the analyses of [17, 26]. The truncated series T_m can be computed in time $e^{O(\Delta m + \log n)}$ given an enumeration of all polymers on fewer than m edges and their weights (see [17]). We can enumerate the polymers in time $O(n^2 m^7 (e\Delta)^{2m})$ as they are connected subgraphs of a graph of maximum degree Δ (see [26]), and compute each weight in time $O(km)$ as all perfectly satisfying assignments on a connected graph are found by following each of the k assignments of an initial vertex and propagating along constraints. To get an approximation of the form $e^{-\alpha} \leq Z(G; w^*)/\xi \leq e^{\alpha}$ we take $m = \log(n/\alpha)/3$ which means the entire computation of ξ can be done in time

$$e^{O(\Delta m + \log n)} + O(k m^8 n^2 (e\Delta)^{2m}) = k n^{O(1)} (n/\alpha)^{O(\Delta)}.$$

We see that the number of colours needed to make the lemma work is of the order $\Delta^{O(\Delta^{3/2})}$. It would be interesting to get a better dependence on Δ.

5 Conclusions

Lemma 7 shows that a hypothetical polynomial-time algorithm for computing $Z(G; w)$ exactly when

$$\log w = \frac{2}{1 - \varepsilon - \delta} \frac{\log k}{\Delta}$$

would refute the UGC. This problem is likely #P hard so we resort to approximation, which we can only do for some range of w. In Theorem 10 we have $\log w = (1 - o(1)) \log(k)/\Delta$ when k is small enough that $\log k = o(\Delta)$, and in Theorem 9 we have $\log w = (2 - o(1)) \log(k)/\Delta$

when k is larger than some $\Delta^{\text{poly}(\Delta)}$. This means we must increase f from k^{-1} to solve any CUG problems, and the size of w in these results is what gives the bound on f in Theorems 3 and 4. It is therefore important to determine the threshold f^* such that CUG(f) is equivalent to UG when $f \leq f^*$. Trivially we have $k^{-1} \leq f^* \leq 1$, but if one could show e.g. that $f^* \geq k^{2\theta-1}$ then Theorem 4 would mean the Unique Games problem is in P for bounded-degree graphs when k is large enough and $\varepsilon + \delta < \theta$.

5.1 Phase transitions

In this subsection we focus on the ferromagnetic Potts model, which is the special case of our partition function $Z(G;w)$ when the constraints on each edge are the identity permutation and we take $w \geq 1$. The behaviour of the Potts model on bounded-degree graphs is strongly related to the *phases* of the model on the infinite Δ-regular tree $\mathbb{T}_\Delta$. We will not define precisely what we mean by a phase or a phase transition here, but as the parameter w varies, the behaviour of the model undergoes certain changes that seem to affect both zeros of the partition function and the dynamics of associated Markov chains. Häggström [16] showed that the *uniqueness phase transition* on $\mathbb{T}_\Delta$ occurs at $w = w_u(k,\Delta)$, the unique value of w for which the polynomial

$$(k-1)x^\Delta + (2-w-k)x^{\Delta-1} + wx - 1 \tag{5}$$

has a double root in $(0,1)$. There is a further *ordered/disordered phase transition* (see [14]) at

$$w_o(k,\Delta) := \frac{k-2}{(k-1)^{1-2/\Delta}-1}.$$

Below we relate the values of w found in Theorems 9 and 10 to these phase transitions.

5.2 Potential barriers to improving Theorem 3

To strengthen Theorem 3 to a result that would refute the UGC, we need roughly a factor two improvement in the leading constant in $\log w^* \sim \log(k)/\Delta$ as $k \to \infty$ (provided $\log k = o(\Delta)$) from Theorem 10, but there are reasons to believe it may be hard to make such an improvement.

The authors of [7] analysed a natural Markov chain known as the *Glauber dynamics* which walks the set of possible colourings of a graph G, and when this mixes rapidly we expect an efficient, randomised approximation algorithm for the Potts model partition function to follow. They showed for the Potts model that Glauber dynamics mixes rapidly on graphs of maximum degree Δ when

$$\log w \leq (1+o(1))\frac{\log k}{\Delta-1},$$

as $k \to \infty$, and that on almost all Δ-regular graphs (for $\Delta \geq 3$), Glauber dynamics mixes slowly when w is just a little larger, satisfying

$$\log w > (1+o(1))\frac{\log k}{\Delta-1-\frac{1}{\Delta-1}}.$$

These bounds sandwich the phase transition point w_u; they also showed that as $k \to \infty$,

$$\log w_u = \frac{\log k}{\Delta-1} + O(1).$$

Thus it appears that w_u is a barrier for approximating $Z(G;w)$ via Glauber dynamics.

Similarly, we expect that w_u is a barrier for the zero-free region. Underpinning Lemma 11 is a complex dynamical system (see [28] for a treatment of the case $k = 2$), and equation (5) appears in the analysis of this system. Essentially, the behaviour of a fixed point in the complex dynamics changes at $w = w_u$ in a way which means it is reasonable to expect zeros of $Z(G; w)$ to accumulate near w_u for some G with maximum degree Δ. Thus we suspect that the method cannot work for $w > w_u$.

5.3 Potential barriers to improving Theorem 4

There are several regimes of interest for the algorithm in Theorem 9 that gives Theorem 4. When we apply Theorem 9 to solve CUG problems, we only need an approximation with $\alpha = Cn$ for constant C in which case the running time is bounded by $kn^{O(1)}e^{O(\Delta)}$. Recall that we also need $k \geq \Delta^{O(\Delta^{3/2})}$), and hence when Δ and k do not grow too fast with n, the running time is sub-exponential in n. For the hypercube with $\Delta = \log n$ and with k as small as the result allows, the algorithm is quasi-polynomial. In the case where Δ is constant, to get an approximation as accurate as α being constant the running time of the algorithm is polynomial in n. It is interesting to compare this with a paper of Galanis et al. [14] who show that it is #BIS-hard to approximate the partition function of the Potts model with k colours on graphs of maximum degree Δ when $w > w_o$. With $\zeta = 8/\sqrt{\Delta}$, if we take $k = k_0 = \exp((18\Delta + 4\Delta \log \Delta)/\zeta)$, then Theorem 9 shows that we can approximate the Potts model partition function on graphs of maximum degree at most Δ with $w = k_0^{(2-\zeta)/\Delta}$. A quick calculation shows that, as $\Delta \to \infty$ (and hence $k \to \infty$), $w_o(k_0, \Delta) \sim k_0^{2/\Delta}$. We conclude that if one assumes that there are no efficient algorithms for approximating #BIS-hard problems, Theorem 9 is very close to optimal in this regime.

References

1 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *FOCS*, pages 563–572. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.59.

2 Sanjeev Arora, Russell Impagliazzo, William Matthews, and David Steurer. Improved algorithms for unique games via divide and conquer. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:41, 2010. URL: https://eccc.weizmann.ac.il/report/2010/041/.

3 Sanjeev Arora, Subhash Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy: extended abstract. In *STOC*, pages 21–28. ACM, 2008. doi:10.1145/1374376.1374380.

4 Alexander Barvinok. Combinatorics and complexity of partition functions. *Algorithms and Combinatorics*, 2016. doi:10.1007/978-3-319-51829-9.

5 Alexander Barvinok. Approximating real-rooted and stable polynomials, with combinatorial applications. *Online Journal of Analytic Combintaorics*, 14, 2019. URL: https://web.math.rochester.edu/misc/ojac/vol14/186.pdf.

6 Ferenc Bencs, Ewan Davies, Viresh Patel, and Guus Regts. On zero-free regions for the anti-ferromagnetic Potts model on bounded-degree graphs. *arXiv preprint*, 2018. arXiv:1812.07532.

7 Magnus Bordewich, Catherine S. Greenhill, and Viresh Patel. Mixing of the Glauber dynamics for the ferromagnetic Potts model. *Random Struct. Algorithms*, 48(1):21–52, 2016. doi:10.1002/RSA.20569.

8 Christian Borgs, Jennifer Chayes, Tyler Helmuth, Will Perkins, and Prasad Tetali. Efficient sampling and counting algorithms for the Potts model on $\mathbb{Z}_d$ at all temperatures. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, 2020. doi:10.1145/3357713.3384271.

9 Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. In *IEEE Conference on Computational Complexity*, pages 144–153. IEEE Computer Society, 2005. doi:10.1109/CCC.2005.20.

10 R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Adv. Comput. Math.*, 5(4):329–359, 1996. doi:10.1007/BF02124750.

11 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. On non-optimally expanding sets in Grassmann graphs. In *STOC*, pages 940–951. ACM, 2018. doi:10.1145/3188745.3188806.

12 Irit Dinur, Subhash Khot, Guy Kindler, Dor Minzer, and Muli Safra. Towards a proof of the 2-to-1 games conjecture? In *STOC*, pages 376–389. ACM, 2018. doi:10.1145/3188745.3188804.

13 Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. *Random Struct. Algorithms*, 20(3):403–440, 2002. doi:10.1002/RSA.10036.

14 Andreas Galanis, Daniel Stefankovic, Eric Vigoda, and Linji Yang. Ferromagnetic Potts model: Refined #BIS-hardness and related results. *SIAM Journal on Computing*, 45(6):2004–2065, 2016. doi:10.1137/140997580.

15 Michel X. Goemans and David P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *STOC*, pages 422–431. ACM, 1994. doi:10.1145/195058.195216.

16 Olle Häggström. The random-cluster model on a homogeneous tree. *Probability Theory and Related Fields*, 104(2):231–253, 1996. doi:10.1007/BF01247839.

17 Tyler Helmuth, Will Perkins, and Guus Regts. Algorithmic Pirogov-Sinai theory. In *STOC*, pages 1009–1020. ACM, 2019. doi:10.1145/3313276.3316305.

18 Howard J. Karloff. How good is the Goemans–Williamson MAX CUT algorithm? In *STOC*, pages 427–434. ACM, 1996. doi:10.1145/237814.237990.

19 Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC*, pages 767–775. ACM, 2002. doi:10.1145/509907.510017.

20 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for Max-Cut and Other 2-Variable CSPs? In *FOCS*, pages 146–154. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.49.

21 Subhash Khot and Oded Regev. Vertex Cover Might be Hard to Approximate to within $2-\varepsilon$. In *IEEE Conference on Computational Complexity*, pages 379–386. IEEE Computer Society, 2003. doi:10.1109/CCC.2003.1214437.

22 Subhash Khot and Nisheeth K. Vishnoi. The Unique Games Conjecture, Integrality Gap for Cut Problems and Embeddability of Negative Type Metrics into ℓ_1. In *FOCS*, pages 53–62. IEEE Computer Society, 2005. doi:10.1109/SFCS.2005.74.

23 Alexandra Kolla. Spectral algorithms for unique games. In *IEEE Conference on Computational Complexity*, pages 122–130. IEEE Computer Society, 2010. doi:10.1109/CCC.2010.20.

24 Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. A deterministic algorithm for counting colorings with 2Δ colors. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, 2019. doi:10.1109/FOCS.2019.00085.

25 Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In *WAOA*, volume 6534 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2010. doi:10.1007/978-3-642-18318-8_17.

26 Viresh Patel and Guus Regts. Deterministic Polynomial-Time Approximation Algorithms for Partition Functions and Graph Polynomials. *SIAM J. Comput.*, 46(6):1893–1919, 2017. doi:10.1137/16M1101003.

27 Viresh Patel and Guus Regts. Computing the number of induced copies of a fixed graph in a bounded degree graph. *Algorithmica*, 81(5):1844–1858, 2019. doi:10.1007/S00453-018-0511-9.

28 Han Peters and Guus Regts. Location of zeros for the partition function of the Ising model on bounded degree graphs. *Journal of the London Mathematical Society*, page jlms.12286, November 2019. doi:10.1112/jlms.12286.

29 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC*, pages 245–254. ACM, 2008. doi:10.1145/1374376.1374414.

30 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *STOC*, pages 755–764. ACM, 2010. doi:10.1145/1806689.1806792.

31 Rocco A. Servedio and Li-Yang Tan. Deterministic search for CNF satisfying assignments in almost polynomial time. In *FOCS*, pages 813–823. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.80.

A Details for the proof of Theorem 8

What follows is a rather precise description of results developed by Barvinok in [4, 5], lightly specialised to our application and notation.

▶ **Definition 15.** *Let $f : \mathbb{C} \to \mathbb{C}$ be any function, and $m \geq 0$. Then we define the* degree-m Taylor polynomial of f about zero, *$T_m(f)$, as the polynomial in z given by*

$$T_m(f)(z) := f(0) + \sum_{k=1}^{m} \frac{f^{(k)}(0)}{k!} z^k.$$

▶ **Lemma 16** (see [4, Lemma 2.2.1] or [5, Lemma 2.1])**.** *Let $g : \hat{\mathbb{C}} \to \hat{\mathbb{C}}$ be a polynomial of degree at most N, and for $\beta > 1$ suppose that $g(z) \neq 0$ for $|z| < \beta$.*

Then given a choice of branch for $f(z) = \log g(z)$ where $|z| < \beta$, we have

$$|f(1) - T_m(f)(1)| \leq \frac{N}{(m+1)\beta^m(\beta - 1)}.$$

▶ **Corollary 17** (cf. [5, Corollary 2.2])**.** *For any $c > 0$ there exists $c' > 0$ such that the following holds. Suppose that the conditions of Lemma 16 hold, and in addition that $\beta \leq 1 + c$.*

Then for any $0 < \alpha < N/e$, and for any

$$m \geq \frac{c'}{\beta - 1} \log\left(\frac{N}{\alpha}\right),$$

we have $|f(1) - T_m(f)(1)| \leq \alpha$.

The only differences from [5, Corollary 2.2] are the relaxation of the assumption $\alpha < 1$ to $\alpha \leq N/e$, the additional assumption that β is close to 1, and a more precise analysis of m. In fact one can take $c' = c/\log(1 + c)$.

Proof. By Lemma 16, we are done if

$$\frac{N}{(m+1)\beta^m(\beta - 1)} \leq \alpha$$

for m as in the statement of the corollary. This holds if and only if

$$(m+1)\beta^{m+1} \geq \frac{N}{\alpha}\frac{\beta}{\beta - 1} \iff (m+1)\log\beta \geq W\left(\frac{N}{\alpha}\frac{\beta\log\beta}{\beta - 1}\right),$$

where W is the upper real branch of the Lambert W-function, see [10]. We take this branch because $\beta > 1$ so $(m+1)\log\beta > 0$. Since W is increasing and $\frac{\log\beta}{\beta - 1} < 1$, this is implied by

$$m + 1 \geq \frac{W(N\beta/\alpha)}{\log\beta}.$$

Now we have $\log x \geq W(x)$ for all $x \geq e$, so since $\beta > 1$ and $N/\alpha \geq e$ this is implied by

$$m + 1 \geq \frac{\log(N\beta/\alpha)}{\log \beta} = \frac{\log(N/\alpha)}{\log \beta} + 1.$$

Then it suffices to take $m \geq \log(N/\alpha)/\log \beta$. But since $\log \beta \sim \beta - 1$ as $\beta \to 1$ and $\beta < 1 + c$, we can find c' depending only on c such that $c' \log \beta \geq \beta - 1$. Then it suffices to take

$$m \geq \frac{c'}{\beta - 1} \log\left(\frac{N}{\alpha}\right) \qquad \blacktriangleleft$$

Corollary 17 tells us how many terms of a Taylor expansion of $\log g$ we need to get an additive error of at most α, under the condition that g has no roots in the disc $\{z \in \mathbb{C} : |z| < \beta\}$. We want to work with a zero-free region of the form $\mathcal{N}([0,1], \eta)$, the open set containing a ball of radius η around each point in $[0,1]$. Barvinok [4, 5] also gives constructions that perform well for this situation which we reproduce below.

▶ **Lemma 18** ([4, Lemma 2.2.3]). *For $0 < \rho < 1$ there is a polynomial p of degree*

$$N_\rho := \left\lfloor \left(1 + \frac{1}{\rho}\right) e^{1+\frac{1}{\rho}} \right\rfloor \geq 14$$

such that $p(0) = 0$, $p(1) = 1$, and

$$-\rho \leq \Re\, p(z) \leq 1 + 2\rho \quad \text{and} \quad |\Im\, p(z)| \leq 2\rho$$

for all z such that $|z| \leq \beta(\rho)$ where

$$\beta_\rho := \frac{1 - e^{-1-\frac{1}{\rho}}}{1 - e^{-\frac{1}{\rho}}} > 1.$$

▶ **Corollary 19** (cf. [5, Theorem 1.6]). *Suppose that $0 < \eta < 1$, and g is a polynomial of degree N such that $g(z) \neq 0$ for all $z \in \mathcal{N}([0,1], \eta)$. Then given any $0 < \alpha < Ne^{6/\eta - 1}$, it suffices to compute the first*

$$m = e^{6/\eta} \log\left(\frac{Ne^{6/\eta}}{\alpha}\right)$$

coefficients of g to obtain a number ξ satisfying $|\log g(1) - \xi| \leq \varepsilon$.

Proof. Let $\rho = \eta/\sqrt{8}$ and β_ρ, N_ρ be given by Lemma 18. Then since $\eta < 1$, we note that

$$N_\rho \leq e^{6/\eta} \quad \text{and} \quad \beta_\rho \geq 1 + \frac{1}{2e^{\frac{1}{\rho}}} \geq 1 + e^{-4/\eta}.$$

Now the polynomial p as in Lemma 18 maps $\{z \in \mathbb{C} : |z| \leq \beta_\rho\}$ into $\mathcal{N}([0,1], \eta)$, so the polynomial $g \circ p(z)$ is a degree $NN_\rho \leq Ne^{6/\eta}$ polynomial which is nonzero for all $z \in \mathbb{C}$ such that $|z| \leq 1 + e^{-4/\eta}$.

We now apply Corollary 17 to $g \circ p$. With $f(z) = \log(g \circ p(z))$ we have $f(1) = \log g(1)$ since $p(1) = 1$, and so the Taylor polynomial $T_m(1)$ (as defined in Definition 15) for this f is the quantity we want for ξ. More precisely, as described in [4, Section 2.2.2], to compute $T_m(\log g \circ p)$ at $z = 0$ it suffices to compute $T_m(g \circ p)$. In turn, to compute $T_m(g \circ p)$ it suffices to compute $T_m(g)$ and the truncation p_m of p obtained by deleting all monomials of degree higher than m, and then the composition of polynomials $T_m(g) \circ p_m$. The final step is to truncate $T_m(g) \circ p_m$ to be degree m, and to obtain ξ by evaluating this polynomial at $z = 1$. This can be done in time $O(m)$, and by Corollary 17 applied to $g \circ p$, when we have

$$m \geq e^{6/\eta} \log\left(\frac{Ne^{6/\eta}}{\alpha}\right),$$

and $\alpha < Ne^{6/\eta - 1}$, we have the desired $|\log g(1) - \xi| \leq \alpha$. $\blacktriangleleft$

We know now how many coefficients are needed for an approximation to a polynomial. For the complexity of computing these coefficients we refer to Patel and Regts [26, 27]. Our partition function $Z(G;w)$ is an edge-coloured BIGCP in the sense of Patel and Regts' definition in [27]. $Z(G;w)$ has degree at most $\Delta n/2$ with BIGCP parameters $\alpha = 2$ and $\beta_i = O(k^i)$ according to [26, Section 6]. Then by [27, Theorem 2.1] there is a deterministic algorithm to compute the first m coefficients of Z in time

$$\tilde{O}\big(n(4e\Delta\sqrt{k})^{2m}\big),$$

where $\tilde{O}$ means that we omit factors polynomial in m.

To prove Theorem 8 we want an approximation for $\log Z(G;w)$ with error $\alpha = Cn$, and we have a zero-free region surrounding $[1, w^*]$ with $w^* = 1 + (\log k - 1)\Delta$ at distance $\eta = \omega(1/(\Delta \log k))$. Then we can transform Z into a polynomial with zero-free region around $[0,1]$ of distance $\eta/(w^* - 1) = \omega(1/(\log k)^2)$, so we need

$$C \leq \frac{2}{e\Delta} e^{O(\log^2 k)},$$

and

$$m = e^{O(\log^2 k)} \log\left(\frac{\Delta}{2C} e^{O(\log^2 k)}\right),$$

according to Corollary 19.

Then we have a running time of

$$\tilde{O}\big(n(4e\Delta\sqrt{k})^{2m}\big) = n \exp\left(e^{O(\log^2 k)} \log\left(\frac{\Delta}{C} e^{O(\log^2 k)}\right) \log(\Delta\sqrt{k})\right).$$

B Details for the proof of Theorem 10

Lemma 11 directly implies Theorem 10, and we give the proof in this section.

B.1 Preliminaries

First, we state a lemma of Barvinok which is useful for evaluating sums of restricted partition functions.

▶ **Lemma 20** (Barvinok [4, Lemma 3.6.3]). *Let $u_1, \ldots, u_n \in \mathbb{R}^2$ be non-zero vectors such that the angle between any two vectors u_i and u_j is at most α for some $\alpha \in [0, 2\pi/3)$. Then the u_i all lie in a cone of angle at most α and*

$$\left|\sum_{i=1}^{n} u_i\right| \geq \cos(\alpha/2) \sum_{i=1}^{n} |u_i|.$$

Furthermore the following simple corollary of of the cosine rule will come in handy.

▶ **Lemma 21.** *Let z, z' be two complex numbers at an angle of at most $\pi/3$, then $|z - z'| \leq \max\{|z|, |z'|\}$.*

Proof. Recall the cosine rule, for a triangle with sides a, b and c; and angles A, B and C where side a is not adjacent to angle A, then

$$|a|^2 = |b|^2 + |c|^2 - 2|b||c|\cos(A),$$

where $|a|$ is the length of side a. Now consider the triangle with vertices in $\mathbb{C}$ at the origin, z and z'. The sides have length $|z|, |z'|$ and $|z - z'|$ and the angle at the origin is the angle $\theta \leq \pi/3$ between z and z'. As $\cos(x) \geq 1/2$ for $x \leq \pi/3$,

$$|z - z'|^2 \leq |z|^2 + |z'|^2 - |z||z'| \leq \max\{|z|^2, |z'|^2\}. \quad \blacktriangleleft$$

To prove Lemma 11 we need some definitions and an auxiliary lemma. We define rational functions (which depend on k and w) in two variables z_0, z and respectively $k-1$ variables $z_0, \ldots, z_{k-2}$ by

$$R(z_0, z; w, k) := \frac{wz_0 + (k-2)z + 1}{z_0 + (k-2)z + w},$$
$$R_k(z_0, z_1, \ldots, z_{k-2}; w) := \frac{wz_0 + z_1 + \ldots + z_{k-2} + 1}{z_0 + z_1 + \ldots + z_{z-2} + w}.$$

Consider the cone

$$C(\theta) := \{z = re^{i\vartheta} \mid r \geq 0 \text{ and } |\vartheta| \leq \theta\},$$

and define for $d = 0, \ldots, \Delta$ and $c, \alpha > 0$, the region

$$K(\theta, d, c, \alpha, \varepsilon) := C(d\theta + \Delta - \varepsilon) \cap \left\{z : \left(1 + \frac{c}{\Delta}\right)^{d-\Delta} \left(1 + \frac{\alpha}{\Delta}\right)^{d} \leq |z| \leq \left(1 + \frac{c}{\Delta}\right)^{\Delta-d} \left(1 + \frac{\alpha}{\Delta}\right)^{d}\right\}.$$

▶ **Lemma 22.** *Let $\Delta \in \mathbb{N}_{\geq 3}$ and let $k \in \mathbb{N}_{\geq 3}$. Define $c = \log k - 1$ and $\alpha = \log k^{1/2} - 1$. Then there exists $0 < \varepsilon < \theta < \pi/(3\Delta)$ and $\eta = \omega(\frac{1}{\Delta})$ such that for each $d = 0, \ldots, \Delta$, and any $z_0, \ldots, z_{k-2} \in K_d := K(\theta, d, c, \alpha, \varepsilon)$ such that for each i, j, $z_i/z_j \in K_d$ and any $w \in \mathcal{N}([1, 1 + c/\Delta], \eta)$ the ratio $R = R_k(z_0, z_1, \ldots, z_{k-2}; w)$ satisfies*

$$(1 + \alpha/\Delta)^{-1} < |R| < 1 + \alpha/\Delta \quad \text{and} \quad |\arg(R)| < \theta. \tag{6}$$

In particular the following values suffice,

$$\theta = \frac{1}{5\Delta}, \qquad \varepsilon = \frac{\theta}{100 \log k}, \qquad \eta = \min\left\{\frac{\Delta c}{800(\Delta + \alpha)^2}, \frac{1}{2400(\Delta + \alpha)}, \frac{c}{800\Delta}\right\}.$$

We will prove this lemma in the next subsection, but we first utilize it to prove Lemma 11, which we restate here for convenience.

▶ **Lemma 11.** *Let $\Delta \in \mathbb{N}_{\geq 3}$ and let $k \in \mathbb{N}_{\geq 3}$. Let $c = \log k - 1$ and $\alpha = \log k^{1/2} - 1$. Then there exists constants $0 < \varepsilon < \theta < \frac{\pi}{3\Delta}$ with $\varepsilon, \theta = \omega(1/\Delta)$ and $\eta = \omega(1/(\Delta \log k))$ such that for any $w \in \mathcal{N}([1, 1 + c/\Delta], \eta)$ and any $UG(k)$ instance G of maximum degree at most Δ the following hold.*

1. *For all lists W of distinct vertices of G and all lists of pre-assigned colours L of length $|W|$, $Z_L^W(G) \neq 0$.*
2. *For all lists $W = W'u$ of distinct vertices of G such that u is a leaf and any two lists $L'l$, $L'l'$ of length $|W|$, the following hold.*
   - **a.** *If the unique neighbour v of u is free,*
     - **i.** *the angle between vectors $Z_{L'l}^{W'u}(G)$ and $Z_{L'l'}^{W'u}(G)$ is at most θ, and*
     - **ii.** $\dfrac{Z_{L'l}^{W'u}(G)}{Z_{L'l'}^{W'u}(G)} \leq 1 + \dfrac{\alpha}{\Delta}$.
   - **b.** *If the unique neighbour v of u is fixed,*

   i. *the angle between vectors* $Z^{W'u}_{L'l}(G)$ *and* $Z^{W'u}_{L'l'}(G)$ *is at most* ε*, and*
   ii. $\dfrac{|Z^{W'u}_{L'l}(G)|}{|Z^{W'u}_{L'l'}(G)|} \leq 1 + \dfrac{c}{\Delta}$.

3. *For all lists* $W = W'u$ *of distinct vertices of* G*, and for all lists of pre-assigned colours* L' *of length* $|W'|$*, let* d *be the number of free neighbours of* u*, and let* $b = \Delta - d$*. Then for any pair of colours* l, l'*,*
   a. *the angle between vectors* $Z^{W'u}_{L'l}(G)$ *and* $Z^{W'u}_{L'l'}(G)$ *is at most* $d\theta + b\varepsilon$*, and*
   b. $\dfrac{|Z^{W'u}_{L'l}(G)|}{|Z^{W'u}_{L'l'}(G)|} \leq (1+\alpha/\Delta)^d(1+c/\Delta)^{\Delta-d}$.

Proof. The choice of constants is the same as in Lemma 22 except that we need to choose η small enough so that each $w \in \mathcal{N}([1, 1+c/\Delta], \eta)$ has argument at most ε. It thus suffices to take $\eta = \omega(\frac{1}{\Delta \log(k)})$.

We prove the lemma by induction on the number of free vertices of G. For the base case, we have no free vertices and so every vertex is fixed. Therefore $Z^W_L(G)$ is a product of non-zero terms, hence is non-zero, proving 1. Statement 22a is vacuous as there are no free vertices. Statement 22b follows as the products $Z^{W'u}_{L'l}(G)$ and $Z^{W'u}_{L'l'}(G)$ differ in at most one term. Thus their ratio is either 1, w or w^{-1}. Similarly we deduce Statement 3 (in which d must be zero) from the fact that the products $Z^{W'u}_{L'l}(G)$ and $Z^{W'u}_{L'l'}(G)$ differ in at most Δ terms.

Now, we assume that Statements 1, 2, and 3 hold for graphs with $r \geq 0$ free vertices. We prove the statements for $r+1$ free vertices. First, we shall prove 1.

Suppose that u is a free vertex. Note that $Z^W_L(G) = \sum_{j=1}^k Z^{Wu}_{Lj}(G)$. As each term in the sum on the right hand side of this expression has one fewer free vertex, we may apply induction to deduce that all of these terms are non-zero by 1. Furthermore, by 3 each pair has angle at most $d\theta + (\Delta - d)\varepsilon$ where d is the number of free neighbours of u. Lemma 20 tells us that the Z^{Wu}_{Lj} all lie in a cone of angle at most $d\theta + (\Delta - d)\varepsilon$ and

$$|Z^W_L(G)| = \left|\sum_{j=1}^k Z^{Wu}_{Lj}(G)\right| \geq \cos(d\theta/2 + (\Delta - d)\varepsilon/2) \sum_{j=1}^k |Z^{Wu}_{Lj}(G)| \neq 0.$$

Next, we shall prove 22a so consider the ratios,

$$R_{j,l}(G) = \frac{Z^{W'u}_{L'j}(G)}{Z^{W'u}_{L'\ell}(G)}, \qquad R^v_{j,\ell}(G) = \frac{Z^{W'v}_{L'j}(G-u)}{Z^{W'v}_{L'\ell}(G-u)}.$$

As v is the unique neighbour of u and is free, we may write, denoting j^* for $\pi_{uv}(j)$ and ℓ^* for $\pi_{uv}(\ell)$,

$$R_{j,l}(G) = \frac{\sum_i Z^{Wuv}_{Lji}(G)}{\sum_i Z^{Wuv}_{L\ell i}(G)} = \frac{wZ^{Wv}_{Lj^*}(G-u)\sum_{i\notin\{j^*,\ell^*\}} Z^{Wv}_{Li}(G-u) + Z^{Wv}_{L\ell^*}(G-u)}{Z^{Wv}_{Lj^*}(G-u) + \sum_{i\notin\{j^*,\ell^*\}} Z^{Wv}_{Li}(G-u) + wZ^{Wv}_{L\ell^*}(G-u)}.$$

Dividing both the numerator and denominator by $Z^{Wv}_{L\ell^*}(G-u)$ (which by induction is nonzero) we obtain,

$$\frac{wR^v_{j^*,\ell^*}(G) + \sum_{i\neq j^*,\ell^*} R^v_{i,\ell^*}(G) + 1}{R^v_{j^*,\ell^*}(G) + \sum_{i\neq j^*,\ell^*} R^v_{i,l}(G) + w} = R_k(R^v_{j^*,\ell^*}(G), R^v_{1,\ell^*}(G), \ldots, R^v_{k,\ell^*}(G); w). \tag{7}$$

Where the function R_k in (7) takes as arguments all $R^v_{i,\ell^*}(G)$ for $i \neq \ell^*$ precisely once (and so takes precisely $k-1$ arguments as expected.)

Suppose that v has d free neighbours that are not u. Since $G-u$ has one fewer free vertex than G, we may apply the inductive hypothesis. By 3 we find that for any $i \neq \ell^*$, we have $R^v_{i,\ell}(G) \in K_d$. However, we also have that for any $i,j \neq \ell^*$, that

$$\frac{R^v_{i,\ell^*}(G)}{R^v_{j,\ell^*}(G)} = \frac{Z^{W'v}_{L'i}(G-u)}{Z^{W'v}_{L'j}(G-u)} = R^v_{i,j}(G) \in K_d.$$

To prove 22a2(a)i, observe that the angle between $Z^{W'u}_{L'j}$ and $Z^{W'u}_{L'l}$ is precisely the angle of $R_{j,l}(G)$ from the real axis in $\mathbb{C}$ and so is bounded by the absolute value of the argument of $R_{j,l}(G)$, which by Lemma 22 bounded by θ as desired. Statement 22a2(a)ii also follows immediately from Lemma 22.

For the proof of 22b, we note that as v is fixed, then

$$Z^{W'u}_{L'j}(G) \in \{w^{-1}Z^{W'u}_{L'l}(G), Z^{W'u}_{L'l}(G), wZ^{W'u}_{L'l}(G)\}$$

from which both 2(b)i and 2(b)ii follow.

Finally, we prove 3. To do so we consider the graph $G \star u$ which is formed as follows. Let $v_1,\dots,v_r$ be the neighbours of u ordered arbitrarily. Let $u_1,\dots,u_r$ be r new vertices which will be copies of u. Then $G \star u$ is the graph obtained by deleting u and its incident edges, adding the vertices $u_1,\dots,u_r$ and edges $u_1v_1,\dots,u_rv_r$. Furthermore, $G \star u$ inherits any colouring of G and if u is coloured, all of the new vertices inherit this colour. Note that if u is coloured, then the graph $G \star u$ has the same partition function as G. Also, in this case $G \star u$ has the same number of free vertices as G. This allows us to prove 3 from 2 by changing the colour of one copy of u at a time. That is,

$$\frac{Z^{W'u}_{L'j}(G)}{Z^{W'u}_{L'l}(G)} = \frac{Z^{W'u_1\dots u_r}_{L'j\dots j}(G\star u)}{Z^{W'u_1\dots u_r}_{L'l\dots l}(G\star u)} = \prod_{i=1}^{r} \frac{Z^{W'u_1\dots u_{i-1}u_i\dots u_r}_{L'j\dots jl\dots l}(G\star u)}{Z^{W'u_1\dots u_iu_{i+1}\dots u_r}_{L'j\dots jl\dots l}(G\star u)} \tag{8}$$

By 2 each of the terms in the product in (8) has angle at most θ and absolute value at most $1+\alpha/\Delta$ (if u_i is free) or angle at most ε and absolute value at most $1+c/\Delta$ (if u_i is fixed). As u has d free neighbours and at most $\Delta-d$ fixed neighbours, this allows us to conclude 33a and 33b, completing the induction. ◀

B.2 Proof of Lemma 22

We will require a technical lemma which concerns the real and imaginary parts of the ratios $R(z_1,z_2;w,k)$.

▶ **Lemma 23.** *Let $z_1,z_2 \in \mathbb{C}$ be defined as $z_1 = xe^{i\theta_x}$, $z_2 = ye^{i\theta_y}$ with $x,y \in \mathbb{R}^+$ and $\theta_x,\theta_y \in [0,2\pi)$ and suppose $w \in [1,1+\frac{c}{\Delta}]$ is real. Then, the real and imaginary parts of $R(z_1,z_2;w,k)$ are as follows where N is a nonzero constant,*

$$\begin{aligned}\Re(R(z_1,z_2;w,k)) &= N(wx^2 + (w+1)(k-2)xy\cos(\theta_x-\theta_y) + (k-2)^2y^2 \\ &\quad + (w^2+1)x\cos(\theta_x) + (w+1)(k-2)y\cos(\theta_y) + w),\end{aligned} \tag{9}$$

$$\begin{aligned}\Im(R(z_1,z_2;w,k)) &= N(w-1)((k-2)xy\sin(\theta_x-\theta_y) \\ &\quad + (1+w)x\sin(\theta_x) + (k-2)y\sin(\theta_y)).\end{aligned} \tag{10}$$

▶ Remark 24. Set $\theta = \max(|\theta_x|,|\theta_y|,|\theta_x-\theta_y|)$ and assume $|\theta| \le 1$. Then as $|\sin t| \le |t|$ and $\cos t \ge 1-t^2/2$ for all t, and using $w \ge 1$ we obtain the following bounds:

$$\begin{aligned}\Re(R(z_1,z_2;w,k)) &\ge N(1-\theta^2/2)(wx^2 + (w+1)(k-2)xy + (k-2)^2y^2 \\ &\quad + (w^2+1)x + (w+1)(k-2)y + w) \\ &\ge N(1-\theta^2/2)(x+(k-2)y+w)(wx+(k-2)y+1);\end{aligned}$$

and

$$\Im(R(z_1,z_2;w,k)) \le N(w-1)((k-2)xy|\theta_x-\theta_y| + (1+w)x|\theta_x| + (k-2)y|\theta_y|).$$

Hence

$$\left|\frac{\Im(R(z_1,z_2;w,k))}{\Re(R(z_1,z_2;w,k))}\right| \leq \frac{(w-1)\left((k-2)xy|\theta_x-\theta_y|+(1+w)x|\theta_x|+(k-2)y|\theta_y|\right)}{(1-\frac{\theta^2}{2})(x+(k-2)y+w)(wx+(k-2)y+1)}. \tag{11}$$

Proof. We may write $z_1 = x\cos(\theta_x) + ix\sin(\theta_x)$ and $z_2 = y\cos(\theta_y) + iy\sin(\theta_y)$. Hence,

$$\begin{aligned} &R(z_1,z_2;w,k) \\ &\quad= \frac{w(x\cos(\theta_x)+ix\sin(\theta_x))+(k-2)(y\cos(\theta_y)+iy\sin(\theta_y))+1}{x\cos(\theta_x)+ix\sin(\theta_x)+(k-2)(y\cos(\theta_y)+iy\sin(\theta_y))+w} \\ &\quad= \frac{wx\cos(\theta_x)+(k-2)y\cos(\theta_y)+1+i(wx\sin(\theta_x)+(k-2)y\sin(\theta_y))}{x\cos(\theta_x)+(k-2)y\cos(\theta_y)+w+i(x\sin(\theta_x)+(k-2)y\sin(\theta_y))} \end{aligned} \tag{12}$$

Rationalising the denominator in (12), we obtain

$$\begin{aligned} &R(z_1,z_2;w,k) \\ &\quad= \frac{1}{N}\left(wx\cos(\theta_x)+(k-2)y\cos(\theta_y)+1+i(wx\sin(\theta_x)+(k-2)y\sin(\theta_y))\right) \\ &\qquad\times\left(x\cos(\theta_x)+(k-2)y\cos(\theta_y)+w-i(x\sin(\theta_x)+(k-2)y\sin(\theta_y))\right) \end{aligned} \tag{13}$$

where $N = |x\cos(\theta_x)+(k-2)y\cos(\theta_y)+w+i(x\sin(\theta_x)+(k-2)y\sin(\theta_y))|^2$. Expanding the expression in (13), the real and imaginary parts are given by the following expressions in which we write c_x for $\cos(\theta_x)$ and similarly define c_y, s_x and s_y to simplify notation.

$$\begin{aligned} \Re(R(z_1,z_2;w,k)) &= N^{-1}(wx^2c_x^2+(w+1)(k-2)xyc_xc_y+(k-2)^2c_y^2 \\ &\quad+wx^2s_x^2+(w+1)(k-2)xys_xs_y+(k-2)^2s_y^2 \\ &\quad+(w^2+1)xc_x+(w+1)(k-2)yc_y+w) \\ \Im(R(z_1,z_2;w,k)) &= N^{-1}((k-2)xy(c_xs_y+ws_xc_y)-(k-2)xy(wc_xs_y+s_xc_y) \\ &\quad+(w^2-1)xs_x+(w-1)(k-2)ys_y) \end{aligned}$$

Combining these expressions with the trigonometric identities

$$\begin{aligned} &\cos^2(\vartheta)+\sin^2(\vartheta)=1 \\ &\sin(\alpha-\beta)=\sin(\alpha)\cos(\beta)-\sin(\beta)\cos(\beta) \\ &\cos(\alpha-\beta)=\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta) \end{aligned}$$

yields the expressions (9) and (10) as claimed.

By an application of the triangle law combined with an applications of the approximations, $|\sin\theta| \leq |\theta|$ and $\cos\theta \geq 1-\theta^2/2$, we obtain

$$|\Im(R(z_1,z_2;w,k))| \leq N^{-1}(w-1)\left((k-2)xy|\theta_x-\theta_y|+(1+w)x|\theta_x|+(k-2)y|\theta_y|\right), \tag{14}$$

$$\begin{aligned} \Re(R(z_1,z_2;w,k)) \geq N^{-1}&\left((wx+(k-2)y+1)(x+(k-2)y+w)\right. \\ &\left.-\left((w+1)(k-2)(x+1)y+(w^2+1)x\right)\theta^2/2\right). \end{aligned} \tag{15}$$

Dividing (14) by (15), noting that for θ small this is maximised when $w = 1+\frac{c}{\Delta}$ and regrouping some terms yields the bound (11). ◀

We can now give a proof of Lemma 22.

Proof of Lemma 22. We first prove a slightly stronger version of the lemma for w' real. That is, we will show that there exist a small constants $\kappa = c/100$ and $\kappa' = 0.02$ such that

$$(1+(\alpha-\kappa)/\Delta)^{-1} < |R| < 1+(\alpha-\kappa)/\Delta \quad \text{and} \quad |\arg(R)| < (1-\kappa')\theta. \tag{16}$$

To do so we start by taking a constant δ small enough so that for all $k \geq 3$, c and α satisfy the strict inequality

$$\frac{ce^c}{cos(\delta)(e^c+k-1)} < \alpha - \kappa; \tag{17}$$

for example $\delta = 1/2$ is sufficient.

Fix $d \in \{0,\dots,\Delta\}$. First we observe that we may assume that $|R| \geq 1$. Indeed, if $|R| < 1$, then

$$1/R = \frac{z_0 + \sum_{i=1}^{k-2} z_i + w}{wz_0 + \sum_{i=1}^{k-2} z_i + 1} = \frac{1 + \sum_{i=1}^{k-2} z_i/z_0 + w/z_0}{w + \sum_{i=1}^{k-2} z_i/z_0 + 1/z_0}$$

and $|1/R| > 1$. Since for each $i,j \geq 0$, the pairs z_i/z_0 and z_j/z_0 also satisfy our assumptions this shows our claim. We start by showing that $|R|$ is bounded by $1+(\alpha-\kappa)/\Delta$.

We observe that (setting $z = (z_1+\cdots+z_{k-2})/k$)

$$|R| = |R(z_0,z;w,k)| = \left|1 + \frac{(w-1)z_0 + (1-w)}{z_0 + \sum_{i=1}^{k-2} z_i + w}\right| \leq 1 + \frac{\frac{c}{\Delta}|z_0 - 1|}{|z_0 + \sum_{i=1}^{k-2} z_i + w|}. \tag{18}$$

Lower bounding the denominator of (18) may be done with an application of Barvinok's lemma. For the numerator we apply Lemma 21 as the angle between z_0 and 1 is certainly less than $\pi/3$. This allows us to deduce that

$$|R(z_0,z;w,k)| \leq 1 + \frac{\frac{c}{\Delta}\max\{|z_0|,1\}}{\cos(d\theta/2 + (\Delta-d)\varepsilon/2))(|z_0| + \sum_{i=1}^{k-2}|z_i| + 1)}.$$

We next observe that by symmetry we may assume that $|z_0| \leq 1$; otherwise we divide the numerator and the denominator by z_0. To maximize the above quantity clearly one should take each $|z_i|$ as small as possible. So we take $|z_i| = (1+c/\Delta)^{d-\Delta}(1+\alpha/\Delta)^{-d} \geq (1+c/\Delta)^{-\Delta} \geq e^{-c}$ and noting that $\theta \leq \delta/\Delta$, we rearrange to deduce that

$$|R(z_0,z;w,k)| < 1 + \frac{c/\Delta}{\cos(\delta)((k-1)e^{-c}+1)} < 1 + (\alpha-\kappa)/\Delta$$

by (17). This proves the first bound in (16).

For the other bound in (16), recall that $z = \frac{1}{k-2}\sum_{i=1}^{k-2} z_j$ so that $R_k(z_0,z_1,\dots,z_{k-2};w) = R(z_0,z;w,k)$. Note that $z \in C(d\theta + (\Delta-d)\varepsilon)$ by convexity of the cone and so by Lemma 20 we have

$$\cos(d\theta/2+(\Delta-d)\varepsilon/2)(1+c/\Delta)^{d-\Delta}(1+\alpha/\Delta)^{-d} \leq |z| \leq (1+c/\Delta)^{\Delta-d}(1+\alpha/\Delta)^d.$$

To prove the bound on the argument of $R(z_0,z;w,k)$ we use the inequality, $|\beta| \leq |\tan(\beta)|$. It therefore suffices to bound the ratio $\frac{|\Im R(z_0,z;w,k)|}{|\Re R(z_0,z;w,k)|} = \tan(\arg(R(z_0,z;w,k)))$, which by Lemma 23 and Remark 24 is bounded by

$$\frac{(w-1)\left((k-2)|z_0 z||\theta_0-\theta_z| + (1+w)|z_0\theta_0| + (k-2)|z\theta_z|\right)}{(1-\frac{\theta^2}{2})(|z_0|+(k-2)|z|+w)(w|z_0|+(k-2)|z|+1)}. \tag{19}$$

Now suppose we can prove that

$$\frac{((k-2)|z_0 z||\theta_0-\theta_z|+(1+w)|z_0\theta_0|+(k-2)|z\theta_z|)}{(|z_0|+(k-2)|z|+w)(w|z_0|+(k-2)|z|+1)} < \frac{\Delta\tau\theta}{c} \tag{20}$$

where $\tau = 7/e^2 < 0.96$. Then by choosing $\theta \leq 0.2$ and $\kappa' < 0.02$, we have that $\Delta\tau\theta/c < (1-\frac{\theta^2}{2})(w-1)^{-1}(1-\kappa')\theta$ (using $w < 1 + c/\Delta$). This together with (20) proves that (19) is at most $\theta(1-\kappa')$ and hence $|\arg(R(z_0,z;w,k))| < \theta(1-\kappa')$, as desired.

We will now show that (20) holds. So, first note that

$$\begin{aligned}&\frac{((k-2)|z_0 z||\theta_0-\theta_z|+(1+w)|z_0\theta_0|+(k-2)|z\theta_z|)}{(|z_0|+(k-2)|z|+w)(w|z_0|+(k-2)|z|+1)}\\ &\leq \frac{((k-2)|z_0 z||\theta_0-\theta_z|+2|z_0\theta_0|+(k-2)|z\theta_z|)}{(|z_0|+(k-2)|z|+1)^2},\end{aligned} \tag{21}$$

which can be observed by computing the derivative of the left hand side of (20) with respect to w and noting it is strictly negative for $w \geq 1$. Now, we maximize (21), so first we show that there is a maximum point where exactly two of $|\theta_0-\theta_z|, |\theta_0|, |\theta_z|$ are as large as possible and one is zero. To see this, first note that clearly at least one of $|\theta_0-\theta_z|, |\theta_0|, |\theta_z|$ must be as large as possible i.e. equal to $d\theta + (\Delta-d)\varepsilon$. In fact exactly two of these must be maximised as the maximization with respect to the θ terms only is of the form $f(\theta_0,\theta_z) = a|\theta_0-\theta_z| + b|\theta_0| + c|\theta_z|$ for constants $a,b,c>0$. So if $|\theta_0-\theta_z| = d\theta+(\Delta-d)\varepsilon$ for example, then if $b \geq c$ we may set $\theta_0 = d\theta+(\Delta-d)\varepsilon$, $\theta_z = 0$ increasing $f(\theta_0,\theta_z)$. Similar logic allows one to conclude that two of $|\theta_0-\theta_z|, |\theta_0|, |\theta_z|$ are equal to $d\theta+(\Delta-d)\varepsilon$ and one is 0 in every other case.

This leaves us with three maximization problems over $R_d \subseteq \mathbb{R}^2$ defined by

$$\begin{aligned}R_d = \{(x,y)|&(1+c/\Delta)^{d-\Delta}(1+\alpha/\Delta)^{-d} \leq x \leq (1+c/\Delta)^{\Delta-d}(1+\alpha/\Delta)^d,\\ \cos(d\theta/2+(\Delta-d)\varepsilon/2)&(1+c/\Delta)^{d-\Delta}(1+\alpha/\Delta)^{-d} \leq y \leq (1+c/\Delta)^{\Delta-d}(1+\alpha/\Delta)^d,\\ \cos(d\theta/2+(\Delta-d)\varepsilon/2)&(1+c/\Delta)^{d-\Delta}(1+\alpha/\Delta)^{-d} \leq y/x \leq (1+c/\Delta)^{\Delta-d}(1+\alpha/\Delta)^d\}.\end{aligned}$$

We enlarge the region slightly obtaining the region $\widetilde{R}_d \subseteq \mathbb{R}^2$ defined by

$$\begin{aligned}\widetilde{R}_d &= \left\{(x,y)\middle| \frac{\cos(\delta)}{\exp\left(\frac{d}{\Delta}\alpha+(1-\frac{d}{\Delta})c\right)} \leq x,y,y/x \leq \frac{\exp\left(\frac{d}{\Delta}\alpha+(1-\frac{d}{\Delta})c\right)}{\cos(\delta)}\right\}\\ &= \left\{(x,y)\middle| \frac{e\cos(\delta)}{k^{1-\frac{d}{2\Delta}}} \leq x,y,y/x \leq \frac{k^{1-\frac{d}{2\Delta}}}{e\cos(\delta)}\right\}\end{aligned}$$

The functions to maximise are

$$f_1(x,y) = \frac{(k-2)(xy+y)}{(x+(k-2)y+1)^2}, \qquad f_2(x,y) = \frac{(k-2)xy+2x}{(x+(k-2)y+1)^2},$$
$$f_3(x,y) = \frac{2x+(k-2)y}{(x+(k-2)y+1)^2}.$$

First we look at f_1, it has critical points along the line $x+1 = (k-2)y$ where it attains its maximum value of $1/4$. However, note that due to our choice of c and α, this line does not lie inside of $\widetilde{R_d}$, hence the maximum must be attained at a boundary point. Furthermore both f_2 and f_3 have no critical points strictly inside the first quadrant, so again their maxima must be attained at a boundary point. This allows us to reduce the problem to eighteen univariate maximization problems, each of which has maximum at most $3e^{-1}k^{-\frac{d}{2\Delta}}$ over $\widetilde{R}_d$ (see Appendix B.3 for details).

Thus (21) is upper bounded by $(d\theta + (\Delta - d)\varepsilon)3e^{-1}k^{-\frac{d}{2\Delta}}$. As a function of d, this is maximised when $d = (\frac{2}{\log k} - \frac{\varepsilon}{\theta-\varepsilon})\Delta$, which (if $d \geq 1$) gives an upper bound to (21) of

$$\frac{6e^{-2}\Delta(\theta-\varepsilon)}{\log k}\exp\left(\frac{\varepsilon}{2(\theta-\varepsilon)}\log k\right).$$

Thus (20) is satisfied provided $\frac{6}{7}(\theta-\varepsilon)\exp(\frac{1}{2}\log k\frac{\varepsilon}{\theta-\varepsilon}) < \theta$. By taking $\varepsilon = \theta x/\log k$ and assuming $\log k \geq 1$, the left hand side is bounded above by $\frac{6}{7}(1-x)\exp(x/2(1-x))\theta$ and this is at most θ (as required) by taking $x = 1/100$ as assumed in the statement of the lemma. If $d = 0$, then as f_1, f_2 and f_3 are all bounded above by 1, provided $\varepsilon < \frac{\theta}{100\log(k)}$, the left hand side of (20) at most $\varepsilon\Delta < \tau\theta\Delta/c$. This completes the proof of (20) and hence of (16).

We finally extend the proof to the case that $w \in \mathcal{N}([1, c/\Delta], \eta)$ for $\eta = 1/[800(\Delta+\alpha)^2]$ using continuity. First observe that $R_k(w) := R_k(z_0, \dots, z_{k-2}; w)$ satisfies

$$R_k(w) = z_0 + \frac{(z_0 + (k-2)z + 1)(1-z_0)}{z_0 + (k-2)z + w}.$$

Then

$$|R_k(w+\eta) - R_k(w)| = \left|\frac{[z_0 + (k-2)z + 1](1-z_0)}{(z_0 + (k-2)z + w + \eta)(z_0 + (k-2)z + w)}\eta\right|.$$

The numerator is upper bounded by $[|z_0| + (k-2)|z| + 1](1 + |z_0|)|\eta|$, while the denominator is lower bounded by

$$\left[(|z_0| + (k-2)|z| + |w| - |\eta|\cos^{-1}(\Delta\theta/2))(|z_0| + (k-2)|z| + |w|)\cos(\Delta\theta/2)\right]^2$$

where we use the fact that the angle between any two of w, z_0, z is at most $\Delta\theta$ and so we can apply Barvinok's lemma. In the statement of the lemma, we assume $\Delta\theta \leq \pi/3$ so $\cos(\Delta\theta) \geq 1/2$. Then using that $(x+a)/(x+b) \leq a/b$ for $x \geq 0$ and $a \geq b$ and using that $|w| \geq 1$ and that $\eta < 1/4$ (so that $|w| - |\eta|\cos^{-1}(\Delta\theta/2) > 1/2$), we have

$$\frac{|z_0| + (k-2)|z| + 1}{(|z_0| + (k-2)|z| + |w| - |\eta|\cos^{-1}(\Delta\theta))\cos(\Delta\theta)} \leq 4$$

and

$$\frac{|z_0| + 1}{(|z_0| + (k-2)|z| + |w|)\cos(\Delta\theta)} \leq 2.$$

Combining the above inequalities we obtain $|R_k(w+\eta) - R_k(w)| \leq 8\eta$. Recall $\eta \leq \min\{\Delta c/[800(\Delta+\alpha)^2], 1/[2400(\Delta+\alpha)], c/[800\Delta]\}$. Then for $w \in \mathcal{N}([1, c/\Delta], \eta)$, we can write $w = w' + \eta$ with $w' \in [1, \frac{c}{\Delta}]$ real. Writing $R = R(w)$, we have

$$\left(1 + \frac{\alpha}{\Delta}\right)^{-1} \leq \left(1 + \frac{\alpha-\kappa}{\Delta}\right)^{-1} - 8\eta \leq |R(w')| - 8\eta < |R|$$
$$< |R(w)| + 8\eta \leq \left(1 + \frac{\alpha-\kappa}{\Delta}\right) + 8\eta \leq 1 + \frac{\alpha}{\Delta},$$

where the first and last inequalities follow by our choice of η.

It follows from simple geometry that if $|z_1 - z_2| \leq \mu$ for $z_1, z_2 \in \mathbb{C}$ and $\mu \in \mathbb{R}^+$ with $\mu < |z_1|$, then $|\arg(z_1) - \arg(z_2)| < \arcsin(\mu/|z_1|)$. Using this, and since $|R(w')| > (1+\frac{\alpha}{\Delta})^{-1}$, we see that

$$\arg(R) < \arg(R(w')) + \arcsin\left(8\eta\left(1+\frac{\alpha}{\Delta}\right)\right) = \theta(1-\kappa') + \arcsin\left(8\eta\left(1+\frac{\alpha}{\Delta}\right)\right) < \theta;$$

in order to check the last inequality holds, it is sufficient to check that $8\eta(1+\frac{\alpha}{\Delta}) < \sin(\kappa'\theta)$. Noting that $\sin x > x - x^3/6 > 5x/6$ for $x \in (0,1)$, it is sufficient that $8\eta(1+\frac{\alpha}{\Delta}) < 5\kappa'\theta/6$ (since $\kappa'\theta < 1$ by our choice of κ' and θ) and this holds by our choice of η. This completes the proof of the lemma. ◀

B.3 Maximization problems

We look at the maximization problems coming from Appendix B.2 and claim that each has an upper bound of at most $3k^{-\frac{d}{2\Delta}}/e$. We find eighteen of them, one for each of the three functions with either x, y, or y/x fixed to one of the two corresponding boundary values. This allows us to reduce to the univariate maximization problems detailed below. To simplify the expressions we will let $r = k-2$, $s = \cos(\delta)ek^{\frac{d}{2\Delta}-1}$ and $t = k^{1-\frac{d}{2\Delta}}(e\cos(\delta))^{-1}$.

	f_1	f_2	f_3
$x = s$	$p_1(y) = \frac{ry(1+s)}{(s+ry+1)^2}$	$p_2(y) = \frac{rsy+2s}{(s+ry+1)^2}$	$p_3(y) = \frac{2s+ry}{(s+ry+1)^2}$
$x = t$	$p_4(y) = \frac{ry(1+t)}{(t+ry+1)^2}$	$p_5(y) = \frac{rty+2t}{(t+ry+1)^2}$	$p_6(y) = \frac{2t+ry}{(t+ry+1)^2}$
$y = s$	$p_7(x) = \frac{rs(1+x)}{(x+rs+1)^2}$	$p_8(x) = \frac{rxs+2x}{(x+rs+1)^2}$	$p_9(x) = \frac{2x+rs}{(x+rs+1)^2}$
$y = t$	$p_{10}(x) = \frac{rt(1+x)}{(x+rt+1)^2}$	$p_{11}(x) = \frac{rxt+2x}{(x+rt+1)^2}$	$p_{12}(x) = \frac{2x+rt}{(x+rt+1)^2}$
$y/x = s$	$p_{13}(x) = \frac{rs(1+x^{-1})}{(x^{-1}+rs+1)^2}$	$p_{14}(x) = \frac{rs+2x^{-1}}{(x^{-1}+rs+1)^2}$	$p_{15}(x) = \frac{2x^{-1}+rx^{-1}s}{(x^{-1}+rs+1)^2}$
$y/x = t$	$p_{16}(x) = \frac{rt(1+x^{-1})}{(x^{-1}+rt+1)^2}$	$p_{17}(x) = \frac{rt+2x^{-1}}{(x^{-1}+rt+1)^2}$	$p_{18}(x) = \frac{2x^{-1}+rx^{-1}t}{(x^{-1}+rt+1)^2}$

To begin the maximization, first observe that under the map $x \mapsto x^{-1}$, each of the functions $p_j(x)$ is the same as some function $p_l(x)$ for some $13 \leq j \leq 18$ and $7 \leq l \leq 12$. Furthermore, $y = s$ yields the bounds $s \leq x \leq 1$ and $y/x = s$ gives $1 \leq x \leq t$. Similarly we may compare $y = t$ and $y/x = t$. Thus the ranges for x are identical after inverting x. Hence we may ignore p_{13} through p_{18} leaving us with 12 problems.

Next, consider p_{10}, p_{11} and p_{12}, each of which can be bounded above by

$$\frac{2rtx}{(x+rt+1)^2} \leq \frac{2rtx}{r^2t^2} \leq \frac{2}{r},$$

where the final inequality follows as $x \leq t$.

Similarly, we can bound p_4, p_5 and p_6. As it must be the case that $y \geq 1$, the numerator of each is bounded above by $2try$. Thus an upper bound for all three is $2t/ry$. Furthermore, $r \geq \frac{2k}{3\cos(\delta)}$ provided $k \geq 7$ and δ small enough. So we are left with an upper bound of $3k^{-\frac{d}{2\Delta}}/e$.

The remaining problems are similar. The numerators may all be bounded above by $rs(1+x) \leq 2rs$ (or for p_1, p_2 and p_3 by $2ry$.) The denominators are all bounded from below by r^2s^2 and r^2y^2 respectively. Thus all six of these are upper bounded by $2/rs$ which is at most $3k^{-\frac{d}{2\Delta}}/e$.

Hence an upper bound on all of the problems p_1 through p_{18} is $3k^{-\frac{d}{2\Delta}}/e$ as claimed.

B.4 Improvements for small k

When k is small, then the parameter $c = \log(k) - 1$ is also very small. In fact we do not obtain a better constant than what is known for the Ising model until $k \geq 21$. However it is possible to do better, we can choose different values for α and c which work better in these cases. In this appendix we will show how to derive such values.

First, we note that we may do the the analysis in an identical way until we find ourselves with the maximization problems f_1, f_2 and f_3. Now we maximise these more carefully than in appendix B.3. First, for f_1 we apply the AM-GM inequality to the denominator to deduce that $f_1(x, y) \leq \frac{1}{4}$ for any x, y. This allows us to take any $c < 4$ and as k is small this is all we need and so we may ignore this constraint. This leaves us to maximise f_2 and f_3. A similar argument to the one in the proof of Lemma 22 allows us to deduce that the maxima are on the boundary of R_d and hence we need only consider the boundary of $\widetilde{R_d}$.

Now, we proceed as in Appendix B.3 with different choices of s and t where this time we will take $t = e^{d/\Delta\alpha + (1-d/\Delta)c}$ and $s = t^{-1}$. We start with 12 maximization problems which we reduce to 8 by symmetry as before. Furthermore, $f_2 > f_3$ if and only if $x > 1$ which allows us to halve the number of problems left to consider leaving us with 4 problems. More precisely, we are left with p_3, p_5, p_9 and p_{11}. All of these are of the form $f(x) = (a_1 x + a_2)(x + a_3)^{-2}$ which has a maximum at $x = a_3 - 2a_2/a_1$. See the following table for the maximization of these 4 functions.

Function	a_1	a_2	a_3	x^*	$f(x^*)$
p_3	$\frac{1}{k-2}$	$\frac{2s}{(k-2)^2}$	$\frac{s+1}{k-2}$	$\frac{1-3s}{k-2}$	$\frac{1}{4(1-s)}$
p_5	$\frac{t}{k-2}$	$\frac{2t}{(k-2)^2}$	$\frac{t+1}{k-2}$	$\frac{t-3}{k-2} \leq 1$	$\frac{kt}{(t+k-1)^2}$
p_9	2	$(k-2)s$	$1+(k-2)s$	1	$\frac{1}{(2+(k-2)s)}$
p_{11}	$2+(k-2)t$	0	$1+(k-2)t$	$1+(k-2)t > t$	$\frac{(k-2)t^2+2t}{((k-1)t+1)^2}$

Note that in the cases of p_5 and p_{11} the maximum value x_* is outside the domain which we are maximising over and thus we maximise at the endpoints of the domain instead.

Now, recall that the maximum values obtained above must also satisfy (17). Also, when $s = e^{-\alpha}$ it must be the case that $(2 + (k-2)e^{-\alpha})^{-1} < c^{-1}$ (from p_9). Combining these after rearrangement yields the inequity

$$\frac{ce^c}{e^c + k - 1} \leq \alpha \leq \log\left(\frac{k-2}{c-2}\right) \tag{22}$$

We may solve this inequality computationally for c, and deduce that there is a choice of α, c which satisfies (22) provided that $c \leq c_k$ for some c_k which can be found in the following table. The corresponding value of α_k is also provided. We give both c_k and α_k rounded to three decimal places.

k	3	4	5	6	7	8	9	10	11	12
α_k	1.767	1.803	1.849	1.896	1.944	1.990	2.034	2.076	2.116	2.154
c_k	2.171	2.330	2.472	2.600	2.716	2.820	2.916	3.003	3.084	3.160

Now, we check that these are indeed the maximum values. To do this, we first note that we have $p_3 \leq 1/4$ and applying AM-GM to the denominator of the maximum for p_5 yields a result which is smaller than the values from we obtained for the maximum of p_9. Finally, for p_{11}, the denominator is at least $(k-1)(k-2)t^2 + 2t(k-1)$. Thus, after cancellations we are left with $p_{11} \leq 1/(k-1)$ which suffices for $k \geq 4$. For $k = 3$ we can easily check that $(t^2 + 2t)(2t+1)^{-2}$ is maximised when $t = 1$ and hence is certainly at most $1/3 < 1/2.17$.

Recall when computing the maximum of p_9, we took s as large as possible where one would expect that we should do the opposite to maximise p_9. We now justify this choice. So recall that we must ensure $d\theta p_9(x) \leq \Delta\theta/c$. Furthermore, s may be considered as a function of d and as such is equal to $\exp(-d/\Delta\alpha - (1-d/\Delta)c)$. Thus we must ensure that

$$g(d) = \frac{dc/\Delta}{2+(k-2)s} \leq 1.$$

Writing λ for d/Δ gives the following function with domain $[0,1]$

$$G(\lambda) = \frac{\lambda c}{2+(k-2)e^{-\lambda\alpha-(1-\lambda)c}}.$$

Differentiating this with respect to λ, we see that either $c-\alpha<1$ and G is increasing on $[0,1]$ or there is a maximum with $\lambda>1$ which is not inside the domain. Thus, we maximise G at one of its boundary points and it is easy to see that $\lambda=1$ is the maximum point rather than $\lambda=0$ where $G(\lambda)=0$.

C Details for the proof of Theorem 9

We collect the proofs of results required to give Theorem 9 here.

First, we show how to transform the partition function in Definition 6 to the partition function of the random cluster model. Here is the statement again for convenience.

▶ **Lemma 12.** *Let $G=(V,E,\pi)$ be a UG instance and $w\in\mathbb{C}$. Then*

$$Z(G;w) = \sum_{F\subseteq E}(w-1)^{|F|}\prod_{(V',E')\in\mathcal{C}(V,F)}\mathrm{sat}_\pi(V',E').$$

Proof. This follows by writing $w=1+(w-1)$ and expanding the partition function:

$$\begin{aligned}
Z(G;w) &= \sum_{\{x_u\}_{u\in V}\in[k]^V}\prod_{\substack{(u,v)\in E,\\ x_v=\pi_{uv}(x_u)}}(1+(w-1))\\
&= \sum_{\{x_u\}_{u\in V}\in[k]^V}\prod_{(u,v)\in E}\Big(1+(w-1)\mathbf{1}\{x_v=\pi_{uv}(x_u)\}\Big)\\
&= \sum_{\{x_u\}_{u\in V}\in[k]^V}\sum_{F\subset E}\prod_{(u,v)\in F}(w-1)\mathbf{1}\{x_v=\pi_{uv}(x_u)\}\\
&= \sum_{F\subset E}(w-1)^{|F|}\sum_{\{x_u\}_{u\in V}\in[k]^V}\prod_{(u,v)\in F}\mathbf{1}\{x_v=\pi_{uv}(x_u)\},
\end{aligned}$$

where the second line follows from writing the product over all edges instead of just satisfied edges, the third line follows by expanding the product, writing F for the edges for which the term $(w-1)\mathbf{1}\{x_v=\pi_{uv}(x_u)\}$ is taken, and the final line follows by interchanging the order of summation. Now if we break the final sum over colour assignments and product over satisfied edges into a sum and product for each component (V',E') of (V,F), and recall the definition of $\mathrm{sat}_\pi(V',E')$, we obtain

$$Z(G;w) = \sum_{F\subset E}(w-1)^{|F|}\prod_{(V',E')\in\mathcal{C}(V,F)}\mathrm{sat}_\pi(V',E').$$ ◀

The final task is to prove Lemma 14 which, via Lemma 13, shows that the cluster expansion converges. We restate these lemmas below.

▶ **Lemma 13** (Borgs et al. [8, Lemma 2.1]). *Suppose that polymers are connected subgraphs containing at least one edge of a graph G of maximum degree Δ on n vertices, that $\|\gamma\|$ is the number of edges of the polymer γ, and that*

$$|w_\gamma| \leq e^{-(7+\log\Delta)\|\gamma\|}. \tag{4}$$

Then the cluster expansion converges absolutely and for any $m \in \mathbb{N}$, $|T_m - \log \Xi(G)| \leq ne^{-3m}$.

▶ **Lemma 14.** *Let $\Delta \in \mathbb{N}_{\geq 16}$, let $C = e^{-9-2\log\Delta}$, and let $\zeta = 8\sqrt{1/\Delta}$. Then if $k \geq C^{-2\Delta/\zeta}$ and $1 \leq w \leq e^{(2-\zeta)\log(k)/\Delta}$, Lemma 13 holds for $UG(k)$ instances G of maximum degree Δ.*

Proof of Lemma 14. We proceed in a manner inspired by [8, Theorem 2.4]. First observe that our polymers are connected subgraphs of G containing at least one edge. Recalling that $C = e^{-9-2\log\Delta}$, we now show that the conditions $k \geq C^{-2\Delta/\zeta}$ and $1 \leq w \leq e^{(2-\zeta)\log(k)/\Delta}$ give

$$|w_\gamma| \leq C^{\|\gamma\|}. \tag{23}$$

We verify (23) in three cases according to the value of $s = \|\gamma\|$. We also recall the bound (1).

Case 1: $s > 2\Delta/\zeta$. We use that $|\gamma| \geq 2\|\gamma\|/\Delta$. Then

$$\begin{aligned}|w_\gamma| &= k^{-|\gamma|}(w-1)^{\|\gamma\|}\operatorname{sat}_\pi(\gamma) \leq k^{-|\gamma|}(w-1)^{\|\gamma\|}k \leq k^{-2\|\gamma\|/\Delta}(w-1)^{\|\gamma\|}k \\ &\leq k^{-2s/\Delta}\cdot k^{s(2-\zeta)/\Delta}\cdot k \leq k^{1-s\zeta/\Delta} \leq k^{-s\zeta/(2\Delta)},\end{aligned}$$

which is bounded above by C^{-s} since $k \geq C^{-2\Delta/\zeta}$.

Case 2: $\Delta < s \leq 2\Delta/\zeta$. We use that fact that $\|\gamma\| \leq \binom{|\gamma|}{2}$ and thus $\sqrt{2s} < |\gamma|$. Then

$$|w_\gamma| \leq kk^{(2-\zeta)s/\Delta}k^{-\sqrt{2s}} = k^{1+(2-\zeta)s/\Delta-\sqrt{2s}}.$$

Looking at the exponent of k we see by our assumptions on s that

$$1 + (2-\zeta)s/\Delta - \sqrt{2s} \leq 1 + 4/\zeta - \sqrt{2\Delta} \leq 1 - \sqrt{\Delta}/2 \leq -1$$

for Δ large enough (i.e $\Delta \geq 16$ suffices). So since $k \geq C^{-2\Delta/\zeta}$ we are in business.

Case 3: $1 \leq s \leq \Delta$. If $|\gamma| = 2$ we have $s = 1$ and therefore

$$|w_\gamma| \leq k^{-1}k^{(2-\zeta)/\Delta} \leq k^{-1/2}$$

provided $\Delta \geq 4$. If $|\gamma| \geq 3$ we have

$$|w_\gamma| \leq k^{-2}k^{(2-\zeta)} = k^{-\zeta}.$$

So since $k \geq C^{-2\Delta/\zeta}$ the required bound holds.

This finishes the proof. ◀

Schur Polynomials Do Not Have Small Formulas If the Determinant Doesn't

Prasad Chaugule
Department of Computer Science & Engineering, IIT Bombay, India
prasad@cse.iitb.ac.in

Mrinal Kumar
Department of Computer Science & Engineering, IIT Bombay, India
mrinal@cse.iitb.ac.in

Nutan Limaye
Department of Computer Science & Engineering, IIT Bombay, India
nutan@cse.iitb.ac.in

Chandra Kanta Mohapatra
Department of Computer Science & Engineering, IIT Bombay, India
ckm@cse.iitb.ac.in

Adrian She
Department of Computer Science, University of Toronto, Canada
ashe@cs.toronto.edu

Srikanth Srinivasan
Department of Mathematics, IIT Bombay, India
srikanth@math.iitb.ac.in

Abstract

Schur Polynomials are families of symmetric polynomials that have been classically studied in Combinatorics and Algebra alike. They play a central role in the study of Symmetric functions, in Representation theory [39], in Schubert calculus [26] as well as in Enumerative combinatorics [14, 38, 39]. In recent years, they have also shown up in various incarnations in Computer Science, e.g, Quantum computation [17, 31] and Geometric complexity theory [21].

However, unlike some other families of symmetric polynomials like the Elementary Symmetric polynomials, the Power Symmetric polynomials and the Complete Homogeneous Symmetric polynomials, the computational complexity of syntactically computing Schur polynomials has not been studied much. In particular, it is not known whether Schur polynomials can be computed efficiently by algebraic formulas. In this work, we address this question, and show that unless *every* polynomial with a small algebraic branching program (ABP) has a small algebraic formula, there are Schur polynomials that cannot be computed by algebraic formula of polynomial size. In other words, unless the algebraic complexity class VBP is equal to the complexity class VF, there exist Schur polynomials which do not have polynomial size algebraic formulas.

As a consequence of our proof, we also show that computing the determinant of certain *generalized* Vandermonde matrices is essentially as hard as computing the general symbolic determinant. To the best of our knowledge, these are one of the first hardness results of this kind for families of polynomials which are not *multilinear*. A key ingredient of our proof is the study of composition of *well behaved* algebraically independent polynomials with a homogeneous polynomial, and might be of independent interest.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Schur polynomial, Jacobian, Algebraic independence, Generalized Vandermonde determinant, Taylor expansion, Formula complexity, Lower bound

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.14

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 14; pp. 14:1–14:27

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

Related Version https://eccc.weizmann.ac.il/author/1282/, https://arxiv.org/abs/1911.12520.

Funding *Mrinal Kumar*: A part of the work of the author was done during a postdoctoral stay at the University of Toronto.
Nutan Limaye: Supported by MATRICS grant MTR/2017/000909 awarded by SERB, Government of India.
Srikanth Srinivasan: Supported by MATRICS grant MTR/2017/000958 awarded by SERB, Government of India.

Acknowledgements Mrinal thanks Ramprasad Saptharishi, Amir Shpilka and Noam Solomon and Srikanth thanks Murali K. Srinivasan and Krishnan Sivasubramanian for many insightful discussions. The authors thank Amir for allowing us to include Question 42 and related discussion in this draft.

1 Introduction

In this paper, we explore a theme at the intersection of *Algebraic Complexity Theory*, which studies the computational complexity of computing multivariate polynomials using algebraic operations, and *Algebraic Combinatorics*, which studies, among other things, algebraic identities among polynomials associated to various combinatorial objects.

Specifically, the questions we study are related to the computational complexity of *Symmetric Polynomials*, which are polynomials in $\mathbb{C}[x_1, \ldots, x_n]$ that are invariant under permutations of the underlying variable set $x_1, \ldots, x_n$.[1] Examples of such polynomials include

- The *Elementary Symmetric polynomials* $e_0, e_1, \ldots, e_n$ where $e_d = \sum_{|S|=d} \prod_{j \in S} x_j$ is the sum of all multilinear monomials of degree exactly d,
- The *Complete Homogeneous Symmetric polynomials* $h_0, h_1, \ldots$ where h_d is the sum of all monomials (multilinear or otherwise) of degree exactly d, and
- The *Power Symmetric polynomials* $p_0, p_1, \ldots$ where $p_d = \sum_{i=1}^{n} x_i^d$.

It is a standard fact that the above three families generate *all* symmetric polynomials in a well-defined sense. More precisely, the Fundamental Theorem of Symmetric Polynomials states that every symmetric polynomial f can be written *uniquely* as a polynomial in $\{e_1, \ldots, e_n\}$, and similarly in $\{h_1, \ldots, h_n\}$ and $\{p_1, \ldots, p_n\}$, each of which is thus an algebraically independent set of polynomials. In particular, for $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_\ell)$ a non-increasing sequence of positive integers, if we define $e_\lambda = \prod_{i \in [\ell]} e_{\lambda_i}$, then the $\{e_\lambda\}_\lambda$ are linearly independent, and moreover the set $E_d := \{e_\lambda \mid \sum_i \lambda_i = d\}$ forms a basis for the vector space Λ_d of homogeneous symmetric polynomials of degree d; the same is true also of h_λ and p_λ (defined analogously), yielding bases H_d and P_d respectively for Λ_d.

Symmetric Polynomials in Mathematics

The study of Symmetric Polynomials is a classical topic in Mathematics, with close connections to combinatorics and representation theory, among other fields (see, e.g., [28, 33]). In representation theory , it is known that the entries of the change-of-basis matrices between different bases for the space Λ_d yield important numerical invariants of various representations

[1] In Combinatorics literature, these are more commonly known as *Symmetric Functions*. One can also consider symmetric functions over fields other than the complex numbers, but throughout this paper, we will stick to $\mathbb{C}$.

of the symmetric group S_d. In algebraic and enumerative combinatorics, the study of symmetric polynomials leads to formulas and generating functions for interesting combinatorial quantities such as *Plane Partitions* (see, e.g. [39, Chapter 7]). These studies have in turn given rise to many interesting algebraic identities and generating functions for various families of symmetric polynomials. Some of these, as we note below, have already had consequences for computational complexity.

Algebraic Complexity of Symmetric Polynomials

Symmetric polynomials have also been intensively investigated by researchers in Algebraic complexity [30, 36, 35, 19, 13], with several interesting consequences. The famous "Ben-Or trick" in algebraic complexity (also known simply as "interpolation") was discovered by Ben-Or [36] in the context of using a standard generating function for the Elementary Symmetric Polynomials to obtain small depth-3 formulas for $e_1, \ldots, e_n$.[2] The same idea also yields small constant-depth formulas for the complete homogeneous symmetric polynomials. Symmetric polynomials have also been used to prove lower bounds for several interesting models of computation including homogeneous and inhomogeneous $\Sigma\Pi\Sigma$ formulas [30, 36, 35], homogeneous multilinear formulas [19] and homogenous $\Sigma\Pi\Sigma\Pi$ formulas [13]. Further, via reductions, the elementary and power symmetric polynomials have been used to define restricted models of algebraic computation known as the symmetric circuit model [35] and the $\Sigma \wedge \Sigma$ model (or Waring rank), which in turn have been significantly investigated (see, e.g. [34, 25, 32]).

Schur polynomials

In this paper, we study the complexity of an important family of symmetric polynomials called the *Schur Polynomials*, which we now define.

▶ **Definition 1.** *Let $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ be a non-increasing sequence of positive integers with $\sum_i \lambda_i = d$. We define the Schur polynomial $s_\lambda(x_1, \ldots, x_n)$ of degree d as follows.*

$$s_\lambda = \frac{\det\left((x_i^{\lambda_j+n-j})_{i,j\in[n]}\right)}{\det\left((x_i^{n-j})_{i,j\in[n]}\right)}$$

(Here, if $\lambda = (\lambda_1, \ldots, \lambda_\ell)$, then we define $\lambda_j = 0$ for $j > \ell$.)

The Schur polynomials are known to generalize the elementary symmetric polynomials as well as homogeneous symmetric polynomials. It is also known that the Schur polynomials of degree d form a basis for Λ_d, which is the vector space of all homogeneous symmetric polynomials of degree d.

The Schur polynomials occupy a central place in the study of symmetric polynomials. Their importance in representation theory can be seen for instance by the fact that, they describe the characters of representations of the general linear and symmetric groups. In particular, consider the general linear group $\mathrm{GL}(V)$ over a complex vector space V of dimension n. If ρ is an irreducible representation of $\mathrm{GL}(V)$ that is polynomial, meaning that the eigenvalues of $\rho(A)$ can be expressed as a polynomial in the eigenvalues of A, then the character $\mathrm{Tr}(\rho(A))$ is a Schur polynomial $s_\lambda(x_1, \ldots, x_n)$ evaluated at the eigenvalues

[2] The generating function referred to is $\prod_{i=1}^{n}(t - x_i) = \sum_{j=0}^{n}(-1)^{n-j}e_{n-j}t^j$.

$x_1, \ldots, x_n$ of A, where λ is a partition with at most n non-zero parts. Furthermore, the entries of the change-of-basis matrix (for the vector space Λ_d) from the power symmetric polynomials to the Schur polynomials are exactly the values of the irreducible characters of the symmetric group S_d. Specifically, *the Murnaghan-Nakayama rule* states that when expanded into the basis of power symmetric polynomials we have

$$s_\lambda = \sum_{\mu=(\mu_1,\ldots,\mu_l)\vdash n} \chi^\lambda(\mu) \prod_{i=1}^{l} p_{\mu_i}$$

where $\chi^\lambda(\mu)$ is an irreducible character of the symmetric group S_n evaluated at a permutation of cycle type μ. See [39, Chapter 7] for further details about these uses in representation theory.

Beyond representation theory, Schur polynomials are used in algebraic geometry in the Schubert calculus [26], which is used to calculate the number of ways in which Schubert subvarieties in the Grassmannian (the set of all k dimensional linear subspaces in an n-dimensional space) may intersect. Schur polynomials are also used in enumerative combinatorics, as they provide generating functions for counting various combinatorial objects, including plane partitions, tableaux [39, Chapter 7], reduced decompositions of permutations [38], and graph colourings [14].

Being one of the most well-studied objects in the theory of symmetric functions, Schur polynomials appear in many different avatars in the literature. The following classical definition is also known to capture Schur polynomials. The definition uses combinatorial structures called Ferrers diagrams. A Ferrers diagram (or a Young diagram or simply a diagram) of shape λ, is a left-aligned two-dimensional array of boxes with the ith row containing λ_i many boxes. (See, e.g. Stanley [39], for more about Ferrers diagrams.)

▶ **Definition 2.** *Consider a Ferrers diagram of shape λ. For any non-decreasing sequence $\mu = (\mu_1, \ldots, \mu_m)$ with $\sum_j \mu_j = d$, we define the* Kostka number $K_{\lambda\mu}$ *to be the number of ways of filling the boxes of the Ferrers diagram with numbers from $1, \ldots, m$ such that each row is non-decreasing, each column is strictly increasing, and the number of i's equals μ_i for each $i \in [m]$.*

The Schur polynomial $s_\lambda(x_1, \ldots, x_n) \in \Lambda_d$ is defined so that the coefficient of $x_1^{\mu_1} \cdots x_m^{\mu_m}$ is the Kostka number $K_{\lambda\mu}$ (the coefficients of other monomials are defined by symmetry).

In particular, $s_\lambda = 0$ if $n < \ell$. So we assume that $n \geq \ell$ throughout.

From this definition it is easy to see that Schur polynomials generalize both elementary symmetric polynomials (when $\ell = d$ and $\lambda_1 = \lambda_2 \cdots = \lambda_d = 1$ in the definition above) and homogeneous symmetric polynomials (when $\ell = 1$ and $\lambda_1 = d$ in the definition above).

The Kostka numbers used in the definition above have been investigated extensively both from combinatorial and computational perspectives. (See for instance [39, 29].)

Algebraic Complexity of Schur Polynomials

In this work we focus on the algebraic complexity of Schur polynomials. As stated in Definition 1, which is also known as the *bialternant formula* of Jacobi, the Schur polynomial s_λ can be expressed as the ratio of two determinants. In particular, this implies that the Schur polynomials have algebraic circuits of size $\mathrm{poly}(n, d)$. In fact, it also implies that these polynomials belong to the smaller algebraic complexity class VBP,[3] for which the Determinant is the complete polynomial.

[3] The class of polynomial families which can be efficiently computed by algebraic branching programs.

However, this upper bound is quite a bit weaker than what is known for other well-studied symmetric polynomials mentioned above, all of which have *constant-depth* formulas of polynomial size. We consider the question of whether the Schur polynomials have constant-depth formulas of polynomial size or even general (arbitrary depth) formulas of polynomial size.

Our main result is that under reasonable complexity assumptions, the answer to the above question is negative for many different λ. (Note that since the elementary and complete homogeneous symmetric polynomials are particular examples of Schur polynomials, there are *some* Schur polynomials that have formulas of polynomial size.)

▶ **Theorem 3** (Main Theorem). *Assume that $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ is such that $\lambda_i \geq \lambda_{i+1} + (\ell - 1)$ for all $i \in [\ell - 1]$, also $\lambda_\ell \geq \ell$ and let $d = \sum_i \lambda_i$. Then, for $n \geq \lambda_1 + \ell$, if $s_\lambda(x_1, \ldots, x_n)$ has an algebraic formula of size s and depth Δ, then the $\ell \times \ell$ determinant $(\det_\ell)$ has an algebraic formula of size poly(s) and depth $\Delta + O(1)$.*

For suitable choices of ℓ, d, n above, we can ensure that these parameters are all polynomially related. The theorem then implies that the Schur polynomials do not have algebraic formulas of polynomial size unless the entire complexity class VBP collapses to the complexity class VF which consists of polynomials with small formulas. Moreover, the Schur polynomials do not have *constant-depth* formulas of subexponential size unless the determinant does, a result that would greatly improve the state-of-the-art in this direction [16].

The above theorem and its proof have several interesting aspects that we now elaborate on.

Newton iteration and formula complexity

Theorem 3 is motivated in part by a recent result of Bläser and Jindal [1] who prove the following interesting result about symmetric polynomials. As mentioned earlier, it is known that any symmetric polynomial $f_{sym} \in \mathbb{C}[x_1, \ldots, x_n]$ can be written *uniquely* as a polynomial in (say) the elementary symmetric polynomials $e_1, \ldots, e_n$. I.e., there exists a unique $f_E \in \mathbb{C}[x_1, \ldots, x_n]$ such that

$$f_{sym}(x_1, \ldots, x_n) = f_E(e_1, \ldots, e_n).$$

Motivated by a question of Lipton and Regan [27], Bläser and Jindal studied the computational complexity of f_{sym} vis-a-vis that of f_E. It is clear that if f_E has algebraic circuits of polynomial size (resp. formulas) then so does f_{sym}, since the elementary symmetric polynomials have algebraic formulas of polynomial size. Interestingly, Bläser and Jindal showed a converse to this statement: they showed that if f_{sym} has small algebraic circuits, then so does f_E. [4]

At first sight, this looks highly relevant to our theorem, since by the classical *Jacobi-Trudi* identity (see, e.g. Theorem 24 in this paper or [39, Theorem 7.16.1]), when f_{sym} is a Schur polynomial of the type assumed in Theorem 3, then f_E is in fact the determinant (on a subset of its variables). We could hope to use the theorem of Bläser and Jindal to prove that if the Schur polynomial has a small formula, then so does the determinant. However, this doesn't quite work, since the proof of [1] only yields small *circuits* for the polynomial f_E, even if we assume that the polynomial f_{sym} has small *formulas.*

[4] Bläser and Jindal work throughout with the elementary symmetric polynomials. However, using algebraic identities that link various symmetric polynomials with each other, we observe in this paper that their result also holds for the complete homogeneous and power symmetric polynomials.

We briefly outline the reason for this, noting that this hurdle occurs quite often in trying to adapt results in algebraic circuit complexity to algebraic formulas. As mentioned above, the polynomials $e_1, \ldots, e_n$ are algebraically independent. A standard proof of this (see, e.g., [35]) goes via showing that the map

$$\overline{e} : \mathbb{C}^n \to \mathbb{C}^n, \text{ defined by } \overline{a} = (a_1, \ldots, a_n) \mapsto (e_1(\overline{a}), \ldots, e_n(\overline{a}))$$

is surjective. Hence, for each $\overline{b} \in \mathbb{C}^n$, there exists an $\overline{a} \in \mathbb{C}^n$ such that $\overline{e}(\overline{a}) = \overline{b}$. The reason this is relevant to the result of [1] is that if we have an efficient algorithm for "inverting" $\overline{e}$ in this way and we additionally have an efficient algorithm for computing f_{sym}, then we immediately obtain an efficient algorithm for computing f_E on any given input $\overline{b} \in \mathbb{C}^n$ by first inverting the map $\overline{e}$ to obtain an $\overline{a}$ as above, and then applying the algorithm for f_{sym} to obtain $f(\overline{a}) = f_E(\overline{b})$. The main technical result in Bläser and Jindal's work is to show how to invert the map $\overline{e}$ as above using an algebraic circuit. The inversion is done by carefully applying a standard algebraic version of *Newton iteration*, which can be performed by an efficient algebraic circuit. Having done this, we plug the output of this circuit into the circuit for f_{sym} to obtain the circuit for f_E.

The reason the above proof does not work in the setting of algebraic formulas is the use of Newton iteration, which is not known to be doable with small formulas (or even within the seemingly larger class VBP). Indeed, this is the main bottleneck in translating several results in algebraic complexity on polynomial factorization [23, 8, 6] and hardness-randomness tradeoffs [22, 9, 5] that are known in the context of algebraic circuits to the setting of algebraic formulas.

In the proof of the main theorem, we show how to get around the use of Newton iteration in this setting and use it to prove a (slightly weaker) version of the result of Bläser and Jindal for algebraic formulas. We hope that the ideas we use here can be adapted and extended to circumvent the use of Newton iteration in some of the other settings mentioned above as well. Our main technical lemma is the following.

▶ **Lemma 4** (Main Technical Lemma (informal)). *Let $g_1, \ldots, g_n \in \mathbb{C}[x_1, \ldots, x_n]$ be "well-behaved" algebraically independent polynomials. Then, for any homogeneous polynomial $\tilde{f}$, if $f = \tilde{f}(g_1, \ldots, g_n)$ has a formula of size s and depth Δ, the polynomial $\tilde{f}$ has a formula of size poly(s) and depth $\Delta + O(1)$.*

For a formal definition of what we mean by "well-behaved" and for a formal statement of this lemma, we refer the reader to Definition 26 and Lemma 27 respectively. This lemma, and some of the ideas in its (very simple) proof might be of independent interest and may have other applications, e.g. in Subsection 4.4 we discuss an application of this lemma to some special cases of a question of Amir Shpilka on proving lower bounds on the partial derivative complexity of a product of algebraically independent polynomials.

Generalized Vandermonde determinants

The Vandermonde matrix $(x_i^{n-j})_{i,j\in[n]}$ and its determinant are ubiquitous in computation because of their relation to polynomial interpolation. More precisely, the problem of finding a degree-$(n-1)$ univariate polynomial that takes prescribed values at a specified set of n distinct points involves solving a linear system of equations where the underlying matrix is precisely the Vandermonde matrix. It is, therefore, an important fact that the Vandermonde determinant is computationally much easier than the general determinant: in fact, it has the following standard depth-2 formula

$$\det_n \left((x_i^{n-j})_{i,j\in[n]} \right) = \prod_{i,j\in[n]:i<j} (x_i - x_j).$$

However, it is unclear whether such small expressions continue to exist if we allow the exponents of the variables to vary more generally. For integers $\mu_1 > \mu_2 > \cdots > \mu_n \geq 0$, consider the *generalized Vandermonde matrix* $(x_i^{\mu_j})_{i,j\in[n]}$. Similar to the Vandermonde matrix, the determinant of this matrix is related to the problem of *sparse polynomial interpolation*, where we are trying to interpolate a polynomial only involving the monomials of degree $\mu_1, \ldots, \mu_n$ through the given points.

Can we expect that computing any generalized Vandermonde determinant is much easier than computing the determinant itself? It follows from Theorem 3 and the bialternant formula from Definition 1 above that the answer to this question is negative: for certain (polynomially large) exponents, the generalized Vandermonde determinant is not much easier than the determinant.

Discussion on Schur Polynomials and Generating functions

In algebraic and enumerative combinatorics, we often study a family of related combinatorial objects by considering a *generating function* that combines them, in the hope that the generating function yields a nice closed-form expression which can further be used to estimate or otherwise understand these objects better. (See e.g. [41, 10] for much more about this.) For instance, we know that the generating functions for the elementary and complete homogeneous symmetric polynomials

$$E(t) = \sum_{i=0}^{n} t^i e_i(\mathbf{x}) \qquad \text{and} \qquad H(t) = \sum_{i=0}^{n} t^i h_i(\mathbf{x})$$

have small expressions given by

$$E(t) = \prod_{i\in[n]} (1 + tx_i) \qquad \text{and} \qquad H(t) = \frac{1}{\prod_{i\in[n]}(1 - tx_i)}.$$

Furthermore, as such expressions are algebraic formulas using additions, multiplications and divisions, we can use these formulas along with division elimination and interpolation to construct small algebraic formulas for the e_is and h_js themselves.

Recall that both the elementary and complete homogeneous symmetric polynomials are special cases of Schur polynomials. It therefore is natural to ask if generating functions can be obtained for other simple sequences of Schur polynomials. Our results imply that the generating function for certain sequences of Schur polynomials do *not* have small closed-form expressions with small formulas unless the determinant has small formulas. This seems like an interesting statement in algebraic combinatorics, conditioned upon a well-known conjecture in Computational Complexity theory.

For concreteness, here is one such "hard" generating function made up of Schur polynomials. For any $\ell \geq 0$, let $\lambda_\ell = (\ell^2, \ell^2 - \ell, \ell^2 - 2\ell, \cdots, \ell)$. Define

$$S(t) = \sum_{\ell\geq 0} t^\ell s_{\lambda_\ell}.$$

Note that this is a finite sum for any fixed n as $s_{\lambda_\ell} = 0$ if $\ell > n$. In algebraic combinatorics, it is common to consider symmetric polynomials in an *infinite* number of variables in which case the above is truly an infinite sum. A simple expression in the infinite case typically leads to a simple expression in the finite case by simply setting all variables other than $x_1, \ldots, x_n$ to 0 in the expression.

Proving hardness of non-multilinear polynomial families

The most natural and widely studied notion of completeness in the algebraic setting is the notion of *projections*. A polynomial $P \in \mathbb{C}[x_1, \ldots, x_n]$ is said to be a projection of a polynomial $Q \in \mathbb{C}[y_1, \ldots, y_m]$ if there is a setting σ of $y_1, \ldots, y_m$ to either field constants or to variables from the set $\{x_1, \ldots, x_n\}$, such that the polynomial $Q(\sigma(y_1), \sigma(y_2), \ldots, \sigma(y_m))$ equals P. While this notion of reductions is very natural and intuitive and in particular, it is clear that *easiness* of Q (with respect to having a small algebraic circuit or formula, for instance) immediately implies the *easiness* of P, there is an inherent difficulty in using this notion of reductions for proving the hardness of families of non-multilinear polynomials. To see this, observe that if Q is non-multilinear in each of its variables, and P is a multilinear polynomial which depends on at least one variable, then P cannot be expressed as a projection of Q. In particular, this notion of reductions cannot be used to prove the hardness of non-multilinear Schur polynomials or the hardness of generalized Vandemonde determinant, assuming the hardness of determinant for algebraic formulas.[5] We avoid this issue by showing that there is a *c-reduction* from the Determinant to the non-multilinear polynomials we study.[6] More precisely, we show that a small formula for any of our hard non-multilinear polynomials can be used to construct a small formula for the Determinant polynomial with only a small blow-up in size and depth.

Other related work

The algebraic complexity of Schur polynomials has been studied in various restricted models of computation. Koev [24], Chan et al. [3] and Fomin et al. [11] consider the complexity of computing Schur polynomials in *the subtraction-free* algebraic circuit model. An algebraic circuit is *subtraction-free* if it uses only addition, multiplication and division operators.[7] They showed that $s_\lambda(x_1, \ldots, x_n)$ has subtraction-free circuits of size polynomial in n and λ_1. In Fomin et al. [11], the authors also proved polynomial bounds on the size of the subtraction-free circuits computing other interesting variants of Schur polynomials such as *double Schur polynomials* and *skew Schur polynomials*. All the algorithms presented in [24, 3, 11] for computing Schur polynomials used division in non-trivial ways.

Demmel et al. [7] and Fomin et al. [12] studied the monotone complexity of Schur polynomials. In the monotone setting, only addition and multiplication operators are used. (Both division and subtraction operators are not allowed.) They proved exponential upper bounds on the monotone complexity of Schur polynomials and conjectured an exponential lower bound. The exact complexity of Schur polynomials is not resolved in the monotone setting. However, Grigoriev et al. [15] proved an exponential monotone circuit lower bound for a related family of symmetric polynomials, called the *monomial symmetric polynomials*.

---

[5] There is a more general notion of projections which involves substituting $y_1, \ldots, y_m$ with affine linear functions in $x_1, \ldots, x_n$. While such projections can be used to reduce multilinear polynomials to non-multilinear ones, the reduction must rely on clever cancellations in order to achieve this. Designing such reductions is in general challenging. We do not know of such a reduction in our setting.

[6] A c-reduction in algebraic complexity is similar in spirit to Turing reductions in standard Computational Complexity. See for example [2, 20].

[7] For example, consider the polynomial $x^2 - xy + y^2$. It is computed by the following subtraction-free circuit: $(x^3 + y^3)/(x + y)$.

Organization of the paper

The rest of the paper is organized as follows. We start with a brief discussion of some of the preliminaries in Section 2 and a brief introduction to Symmetric polynomials and Schur polynomials in Section 3. We formally state and prove Lemma 4 in Subsection 4.1, followed by its application to the proof of Theorem 3 in Subsection 4.2. We discuss further applications of some of these ideas to extending the result of Bläser and Jindal's [1] in Subsection 4.3 and to the question of proving lower bounds on the partial derivative complexity of a product of algebraically independent polynomials in Subsection 4.4. We conclude with some open questions in Section 5.

2 Notations and Preliminaries

Throughout this paper, we assume that we are working over the field $\mathbb{C}$. It is not very hard to see that the results we present can be made to work for fields of characteristic zero or fields of sufficiently large characteristic. Boldface letters are used for tuples of variables e.g. $\mathbf{x}$ for $(x_1, x_2, \ldots, x_n)$. For $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{N}^n$ and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we use $\mathbf{x}^{\mathbf{b}}$ to denote $\prod_{i=1}^{n} x_i^{b_i}$. We use $|\mathbf{b}|_1$ to denote $\sum_{i\in[n]} b_i$.

2.1 Models of computation

We start by defining some of the standard models of algebraic computation that we work with in the rest of the paper.

▶ **Definition 5** (Algebraic circuit). *An algebraic circuit C is a directed acyclic graph with a unique sink node, called the root node. The source nodes are called leaves and are labelled with field constants or variables. All the other nodes are labelled with $+$ or $\times$. Each $+$ node computes the addition of the polynomials computed by its children and similarly, each $\times$ node computes the multiplication of the polynomials computed by its children. The circuit is said to compute the polynomial computed by the root node.*

▶ **Definition 6** (Algebraic formula). *If the underlying graph of an algebraic circuit is a directed tree (which is a special type of a directed acyclic graph) then the circuit is called a formula.*

▶ **Definition 7** (Algebraic branching program). *An algebraic branching program (ABP) is a layered directed acyclic graph with a unique source vertex denoted s and a unique sink vertex denoted t. Each edge is labelled by a linear polynomial. The weight of a path p is the product of the labels of the edges in p. The polynomial that the ABP computes is the sum of all the weights of paths from s to t.*

2.2 Interpolation and Division elimination

We now state two fairly standard facts about algebraic formulas. The first of these relates the formula complexity of a polynomial to the formula complexity of each of its homogeneous components.

▶ **Lemma 8.** *Let $P(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$ be a polynomial which can be computed by a formula of size at most s and depth Δ. Then, for every d, the homogeneous component of P of degree d can be computed by a formula of size at most $O(s^2)$ and depth $\Delta + O(1)$.*

The proof of this lemma is via a standard interpolation argument, where we consider the polynomial $Q(t) = P(x_1t, x_2t, \ldots, x_nt) \in \mathbb{C}(\mathbf{x})[t]$ as a univariate in t. The point to note is that the homogeneous components of P are coefficients of various powers of t in this new polynomial, and hence can be computed as a linear combination of sufficiently many evaluations of $Q(t)$ for distinct values of t in the base field (which we assume to be large enough). For every $\alpha \in \mathbb{C}$, the formula size of $Q(\alpha)$ is upper bounded by the formula size of P. Similarly the depth of the formula of $Q(\alpha)$ is bounded by the depth of P. The number of such distinct evaluations needed is upper bounded by one more than the degree of Q, which is one more than the degree of P itself. The final observation needed for proving the size upper bound is that a polynomial which can be computed by a formula of size s has degree upper bounded by s. Thus, we need to take an appropriately weighted linear combination of $s+1$ distinct substitutions of t in Q, each of which has a formula of size at most s; thereby giving us an upper bound of $O(s^2)$. Taking linear combinations of such substitutions can be done in depth $O(1)$, which gives the overall depth bound of $\Delta + O(1)$.

The next statement we need is about the formula complexity of a polynomial which can be written as quotient of two polynomials with small formulas.

▶ **Lemma 9.** *Let P and R be polynomials in $\mathbb{C}[\mathbf{x}]$ of formula (ABP/circuit) size at most s and depth at most Δ such that R divides P. Then, the polynomial $Q = \frac{P}{R}$ can be computed by a formula (an ABP/circuit resp.) of size at most poly(s) and depth at most $\Delta + O(1)$.*

The proof of this lemma goes via the standard division elimination argument of Strassen and that of Lemma 8. We refer the reader to the excellent survey of Shpilka and Yehudayoff [37] for formal details on division elimination.

2.3 Algebraic independence and the Jacobian

The notion of algebraic independence that we now define plays a crucial role in the proofs in the paper. We start with a formal definition.

▶ **Definition 10.** *Polynomials $q_1, q_2, \ldots, q_k \in \mathbb{C}[\mathbf{x}]$ are said to be algebraically independent over $\mathbb{C}$ if there is no non-zero polynomial $g(y_1, y_2, \ldots, y_k) \in \mathbb{C}[\mathbf{y}]$ such that $g(q_1, q_2, \ldots, q_k)$ is identically zero.*

This definition generalizes the notion of *linear* independence, which is the special case when there is no non-zero polynomial g in k variables and degree 1 such that $g(q_1, q_2, \ldots, q_k)$ is identically zero. As we shall see next, over fields of characteristic zero (or sufficiently large characteristic), the notion of algebraic independence is characterized by the rank of the Jacobian matrix defined below.

▶ **Definition 11.** *The Jacobian matrix of a tuple $(q_1, q_2, \ldots, q_k)$ of n-variate polynomials in $\mathbb{C}[\mathbf{x}]$, denoted by $\mathcal{J}(q_1, q_2, \ldots, q_k)$ is a $k \times n$ matrix with entries from $\mathbb{C}[\mathbf{x}]$ whose $(i,j)^{th}$ entry equals $\frac{\partial q_i}{\partial x_j}$.*

Thus, the row corresponding to q_i in $\mathcal{J}(q_1, q_2, \ldots, q_k)$ contains all of the n first order partial derivatives of q_i. In other words, the i^{th} row of the Jacobian gives us the gradient of q_i. The connection between algebraic independence and the Jacobian stems from the following (almost folklore) theorem.

▶ **Theorem 12** (Jacobian and Algebraic Independence)**.** *Let $(q_1, q_2, \ldots, q_k)$ be a k tuple of n-variate polynomials in $\mathbb{C}[\mathbf{x}]$ of degree at most d. Then, $q_1, q_2, \ldots, q_k$ are algebraically independent over $\mathbb{C}$ if and only if the the rank of the Jacobian matrix $\mathcal{J}(q_1, q_2, \ldots, q_k)$ over the field $\mathbb{C}(\mathbf{x})$ is equal to k.*

A proof of this theorem can be found in the survey of Chen, Kayal and Wigderson [4, Chapter 3].

2.4 Taylor's expansion

For our proof, we need the following well-known form of Taylor's expansion.

▶ **Theorem 13.** *Let $P \in \mathbb{C}[\mathbf{x}]$ be an n-variate polynomial of degree at most d, and let $\mathbf{a} \in \mathbb{C}^n$ be a point. Then, for an n-tuple of variables $\mathbf{z}$*

$$P(\mathbf{a}+\mathbf{z}) = \sum_{i=0}^{d} \left(\sum_{\mathbf{u}\in\mathbb{N}^n, |\mathbf{u}|_1 = i} \frac{\mathbf{z}^{\mathbf{u}}}{\mathbf{u}!} \cdot \frac{\partial P}{\partial \mathbf{x}^{\mathbf{u}}}(\mathbf{a}) \right)$$

where, for $\mathbf{u} = (u_1, u_2, \ldots, u_n)$, $\mathbf{u}! = u_1! \cdot u_2! \cdots u_n!$.

Note that for $i = 0$, the summand $\left(\sum_{\mathbf{u}\in\mathbb{N}^n, |\mathbf{u}|_1 = i} \frac{\mathbf{z}^{\mathbf{u}}}{\mathbf{u}!} \cdot \frac{\partial P}{\partial \mathbf{x}^{\mathbf{u}}}(\mathbf{a})\right)$ is just equal to $P(\mathbf{a})$, and for every positive integer i at most d, this summand is a homogeneous polynomial of degree equal to i in $\mathbf{z}$. Of particular utility to us is the following easy corollary of Theorem 13.

▶ **Corollary 14.** *Let $P \in \mathbb{C}[\mathbf{x}]$ be an n-variate polynomial of degree $d \geq 1$, and let $\mathbf{a} \in \mathbb{C}^n$ be a point. Then, for an n-tuple of variables $\mathbf{z}$*

$$P(\mathbf{a}+\mathbf{z}) = P(\mathbf{a}) + \sum_{j=1}^{n} z_j \cdot \frac{\partial P}{\partial x_j}(\mathbf{a}) \mod \langle \mathbf{z} \rangle^2 .$$

2.5 Two useful lemmas

We use the following (well known) lemma in our arguments. While the lemma is essentially folklore, we sketch a proof for completeness.

▶ **Lemma 15.** *Let $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$ and $P(\mathbf{x}, y) \in \mathbb{C}[\mathbf{x}, y]$ be polynomials such that $P(\mathbf{x}, f(\mathbf{x}))$ is identically zero. Then, there exists a polynomial $Q(\mathbf{x}, y) \in \mathbb{C}[\mathbf{x}, y]$ such that $P(\mathbf{x}, y) = (y - f(\mathbf{x})) \cdot Q(\mathbf{x}, y)$.*

Proof. Let d be the degree of P in y and let $C_0(\mathbf{x}), C_1(\mathbf{x}), \ldots, C_d(\mathbf{x})$ be polynomials in $\mathbb{C}[\mathbf{x}]$ such that

$$P(\mathbf{x}, y) = \sum_{i=0}^{d} C_i(\mathbf{x}) \cdot y^i .$$

Therefore, $P(\mathbf{x}, f(\mathbf{x}))$ can be written as

$$P(\mathbf{x}, f(\mathbf{x})) = \sum_{i=0}^{d} C_i(\mathbf{x}) \cdot f(\mathbf{x})^i .$$

Subtracting the two expressions above, we get

$$P(\mathbf{x}, y) - P(\mathbf{x}, f(\mathbf{x})) = \sum_{i=0}^{d} C_i(\mathbf{x}) \cdot y^i - \sum_{i=0}^{d} C_i(\mathbf{x}) \cdot f(\mathbf{x})^i .$$

Now, on the right hand side, the term for $i = 0$ cancels out and on further simplification, we get

$$P(\mathbf{x}, y) - P(\mathbf{x}, f(\mathbf{x})) = \sum_{i=1}^{d} C_i(\mathbf{x}) \cdot \left(y^i - f(\mathbf{x})^i\right) .$$

Note that for every natural number $i \geq 1$, $y^i - f(\mathbf{x})^i$ is divisible by $(y - f(\mathbf{x}))$. Therefore, every summand on the right hand side has $(y - f(\mathbf{x}))$ as a factor, and thus there is a polynomial $Q(\mathbf{x}, y)$ such that

$$P(\mathbf{x}, y) - P(\mathbf{x}, f(\mathbf{x})) = (y - f(\mathbf{x})) \cdot Q(\mathbf{x}, y)\,.$$

Moreover, since $P(\mathbf{x}, f(\mathbf{x}))$ is identically zero, we have that $P(\mathbf{x}, y) = (y - f(\mathbf{x})) \cdot Q(\mathbf{x}, y)$, thereby completing the proof of the lemma. ◀

In particular, if $|S| \geq d+1$, then there exists some $\mathbf{a} \in S^n$ satisfying $P(\mathbf{a}) \neq 0$. This gives us a brute force deterministic algorithm, running in time $(d+1)^n$, to test if an arithmetic circuit computing a polynomial of degree at most d in n variables is identically zero.

3 Symmetric polynomials

A polynomial is said to be symmetric if it is invariant under a permutation of variables. We now define some of the families of symmetric polynomials that are discussed in this paper and briefly discuss some of their properties. For a more detailed introduction on symmetric polynomials, we refer the reader to the book [28]. We start with the definitions.

▶ **Definition 16** (Elementary symmetric polynomials)**.** *The elementary symmetric polynomial of degree k on n variables denoted by $e_k(\mathbf{x})$ is defined as follows:*

$$e_k(\mathbf{x}) = \sum_{S \subseteq [n]} \prod_{i \in S} x_i\,.$$

The following fact states a property of the elementary symmetric polynomials which will be useful for our proofs in the later sections.

▶ **Fact 17.** *For all $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{C}$, if $c_1, c_2, \ldots, c_n$ are field elements such that*

$$\prod_{i=1}^{n}(z - \alpha_i) = z^n - c_1 \cdot z^{n-1} + c_2 \cdot z^{n-2} - \cdots + (-1)^n c_n\,,$$

then, for every $j \in [n]$, $c_j = e_j(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

▶ **Definition 18** (Homogeneous (Complete) symmetric polynomials)**.** *The homogeneous symmetric polynomial of degree k on n variables denoted by $h_k(\mathbf{x})$ is defined as follows:*

$$h_k(\mathbf{x}) = \sum_{\mathbf{b} \in \mathbb{N}^n : |\mathbf{b}|_1 = k} \mathbf{x}^{\mathbf{b}}.$$

▶ **Definition 19** (Homogeneous (Complete) symmetric polynomials)**.** *The homogeneous symmetric polynomial of degree k on n variables denoted by $h_k(\mathbf{x})$ is defined as follows:*

$$h_k(\mathbf{x}) = \sum_{\mathbf{b} \in \mathbb{N}^n : |\mathbf{b}|_1 = k} \mathbf{x}^{\mathbf{b}}.$$

▶ **Definition 20** (Power symmetric polynomials)**.** *The power symmetric polynomial of degree k on n variables denoted by $p_k(\mathbf{x})$ is defined as follows: $p_k = \sum_{i=1}^{n} x_i^k$.*

These sets of polynomials are algebraically independent. The following fact states this formally.

▶ **Fact 21.** *Let* $\mathbf{x}$ *be an* n *tuple of variables. Then, elementary symmetric polynomials* $e_1(\mathbf{x}), \ldots, e_n(\mathbf{x})$ *are algebraically independent over* $\mathbb{C}$*. Similarly, homogeneous symmetric polynomials* $h_1(\mathbf{x}), \ldots, h_n(\mathbf{x})$ *and power symmetric polynomials* $p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_n(\mathbf{x})$ *are also algebraically independent over* $\mathbb{C}$*.*

We now formally state the fundamental theorem of symmetric polynomials, which essentially says that over field of characteristic zero, every symmetric polynomial can be written as a unique polynomial combinations of the elementary symmetric polynomials (similarly, for power symmetric polynomials or homogeneous symmetric polynomials).

▶ **Theorem 22** (The fundamental theorem of symmetric polynomials)**.** *For a symmetric polynomial* $f_{sym} \in \mathbb{C}[\mathbf{x}]$ *there exists a unique polynomial* $f \in \mathbb{C}[\mathbf{x}]$ *s.t.* $f_{sym} = f_E(e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots e_n(\mathbf{x}))$ *where* $e_i(\mathbf{x})$ *is the elementary symmetric polynomial of degree* i*.*

Similarly, there exists a unique polynomial $f_H \in \mathbb{C}[\mathbf{x}]$ *such that* $f_{sym} = f_H(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_n(\mathbf{x}))$ *and a unique polynomial* $f_P \in \mathbb{C}[\mathbf{x}]$ *such that* $f_{sym} = f_P(p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots p_n(\mathbf{x}))$*, where* $h_i(\mathbf{x})$ *is the homogeneous symmetric polynomial of degree* i *and* $p_i(\mathbf{x})$ *is the power symmetric polynomial of degree* i*.*

3.1 Schur polynomials

A partition of a natural number d is any sequence $\lambda = (\lambda_1, \lambda_2 \ldots, \lambda_\ell)$ of non-negative integers in a non-increasing order $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_\ell \geq 0$ such that $\sum_{i=1}^{\ell} \lambda_i = d$.[8] The number of non-zero parts of λ is called the length of λ and is denoted by $l(\lambda)$. The weight of λ, denoted by $|\lambda|$ is defined to be the sum of each individual component, i.e. $|\lambda| = \lambda_1 + \lambda_2 + \cdots + \lambda_{l(\lambda)}$. If $|\lambda| = d$, then we say that λ is a partition of the number d or alternatively a partition of *degree* d.

Let λ be a partition of the number d. A Ferrers diagram (or simply a diagram) of shape λ is is a left-aligned two-dimensional array of boxes with the ith row containing λ_i many boxes. The conjugate of λ, denoted by λ', is the diagram obtained by switching the rows and columns of the diagram of λ.

▶ **Definition 23** (Schur polynomials)**.** *Let* λ *be a partition of degree* d *and let* $l(\lambda) \leq n$*. Then the Schur polynomial* $s_\lambda(\mathbf{x})$ *is defined as*

$$s_\lambda(\mathbf{x}) = \frac{a_{\lambda+\delta}(\mathbf{x})}{a_\delta(\mathbf{x})}$$

where,

$$\delta = (n-1, n-2, \ldots\ldots 2, 1, 0)$$

$$a_{\lambda+\delta}(\mathbf{x}) = \det(x_i^{\lambda_j+n-j})_{1\leq i,j\leq n}$$

$$a_\delta(\mathbf{x}) = \det(x_i^{n-j})_{1\leq i,j\leq n} = \prod_{1\leq i<j\leq n} (x_i - x_j)$$

[8] Usually the λ_is are assumed to be positive integers as defined earlier. For the sake of notational convenience we allow trailing zeroes in the definition of λ.

We first observe that $s_\lambda(\mathbf{x})$ is a symmetric polynomial. To see this, note that if $x_i = x_j$ for any $i \neq j$ then $a_{\lambda+\delta}(\mathbf{x})$ is 0. Thus, by Lemma 15 and the fact that $x_i - x_j$ and $x_{i'} - x_{j'}$ do not share a common factor unless $\{i, j\} = \{i', j'\}$, $\prod_{i<j}(x_i - x_j)$ is factor of $a_{\lambda+\delta}(\mathbf{x})$ i.e., $a_\delta(\mathbf{x})$ is a factor of $a_{\lambda+\delta}(\mathbf{x})$. Therefore, $s_\lambda(\mathbf{x})$ is a polynomial. Moreover, for any permutation of variables, the sign changes in the numerator and the denominator are the same, and thus their ratio does not see a change in sign. This implies that $s_\lambda(\mathbf{x})$ is a symmetric polynomial.

We now state the classical Jacobi-Trudi identities which relates Schur polynomials to the elementary symmetric and homogeneous symmetric polynomials.

▶ **Theorem 24** (Jacobi-Trudi identities)**.**
(1) $s_\lambda(\mathbf{x}) = \det(h_{\lambda_i - i+j}(\mathbf{x}))_{1\leq i,j\leq \ell}$, *where* $\lambda = (\lambda_1, \ldots, \lambda_\ell)$.
(2) $s_\lambda(\mathbf{x}) = \det(e_{\lambda'_i - i+j}(\mathbf{x}))_{1\leq i,j\leq m}$, *where* λ' *is the conjugate of* λ *and* $m = l(\lambda')$.

In particular,

$$s_\lambda = \begin{vmatrix} h_{\lambda_1} & h_{\lambda_1+1} & \cdots & h_{\lambda_1+\ell-1} \\ h_{\lambda_2-1} & h_{\lambda_2} & \cdots & h_{\lambda_2+\ell-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\lambda_\ell-\ell+1} & h_{\lambda_\ell-\ell+2} & \cdots & h_{\lambda_\ell} \end{vmatrix}_{\ell\times\ell}$$

$$= \begin{vmatrix} e_{\lambda'_1} & e_{\lambda'_1+1} & \cdots & e_{\lambda'_1+m-1} \\ e_{\lambda'_2-1} & e_{\lambda'_2} & \cdots & e_{\lambda'_2+m-2} \\ \vdots & \vdots & \ddots & \vdots \\ e_{\lambda'_m-m+1} & e_{\lambda'_m-m+2} & \cdots & e_{\lambda'_m} \end{vmatrix}_{m\times m}$$

Note that the identity depends only on the properties of λ *(i.e.* ℓ *or* m*) and does not depend on the number of variables* n*.*

▶ **Theorem 25.** *For any* λ*,* $s_\lambda(\mathbf{x})$ *can be computed using a small ABP, hence by a small algebraic circuit.*

Proof. For polynomials $P, Q \in \mathbb{C}[x_1, x_2 \ldots x_n]$ such that both P and Q have small ABPs, then by Lemma 9; $R = \frac{P}{Q}$ also has a small ABP. Homogenization or interpolation can be used to extract the required polynomial without much blow up. Here both $a_{\lambda+\delta}(\mathbf{x})$, $a_\delta(\mathbf{x})$ have small ABPs (as they are small determinants), thus $s_\lambda(\mathbf{x})$ has an ABP of polynomial size which also implies that is has an algebraic circuit of polynomial size. ◀

It is well-known that $a_\delta(\mathbf{x})$, also known as the Principal Vandermonde Determinant, has a small algebraic formula. However much less is known about the complexity of $a_{\lambda+\delta}(\mathbf{x})$. These polynomials are also known as Generalized Vandermonde determinants and are well studied (see for instance [18]). To the best of our knowledge, before this work it was not known whether for all λ, $a_{\lambda+\delta}(\mathbf{x})$s have small formulas. Suppose they did, then by Lemma 9, we get that $s_\lambda(\mathbf{x})$ also have small formulas for all λ. In this paper we show that there exists some λ for which $s_\lambda(\mathbf{x})$ does not have a small formula unless the Determinant has a small formula. This in particular implies that there exist λ such that $a_{\lambda+\delta}(\mathbf{x})$ cannot be computed using small formulas (unless the Determinant can be computed by a small formula).

4 Proofs of main results

4.1 Proof of Lemma 4

We start with the following definition, which is crucial for our proofs.

▶ **Definition 26** (Property S). *A set of n-variate polynomials $\{q_1, q_2, \ldots, q_k\} \subseteq \mathbb{C}[\mathbf{x}]$ is said to satisfy Property S, if there exists an $\mathbf{a} \in \mathbb{C}^n$ such that*

- *For all $i \in [k]$, $q_i(\mathbf{a}) = 0$, and,*
- *The rank of the Jacobian matrix of $q_1, q_2, \ldots, q_k$ when evaluated at $\mathbf{a}$ is equal to its symbolic rank, i.e.* $\operatorname{rank}_{\mathbb{C}}\left(\mathcal{J}(q_1, q_2, \ldots, q_k)(\mathbf{a})\right) = \operatorname{rank}_{\mathbb{C}(\mathbf{x})}\left(\mathcal{J}(q_1, q_2, \ldots, q_k)\right)$.

Property S gives us a concrete way to capture an appropriate notion of *niceness* of a set of algebraically independent polynomials. The following lemma which uses this notion is a key technical ingredient of our proofs.

▶ **Lemma 27.** *Let $\{q_1, q_2, \ldots, q_k\} \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ be a set of algebraically independent polynomials which satisfy Property S. Let $g \in \mathbb{C}[z_1, z_2, \ldots, z_k]$ be a homogeneous k-variate polynomial of degree equal to d such that the composed polynomial $g(q_1, q_2, \ldots, q_k) \in \mathbb{C}[\mathbf{x}]$ has an algebraic formula of size s and depth Δ. Then, $g(z_1, z_2, \ldots, z_k)$ can be computed by an algebraic formula of size $O(s^2 n)$ and depth $\Delta + O(1)$.*

Proof. Let Φ be the formula of size s which computes the polynomial $g\left(q_1(\mathbf{x}), q_2(\mathbf{x}), \ldots, q_k(\mathbf{x})\right)$. Since $q_1, q_2, \ldots, q_k$ satisfy Property S, there is an $\mathbf{a} \in \mathbb{C}^n$ such that $q_1(\mathbf{a}) = q_2(\mathbf{a}) = \cdots = q_k(\mathbf{a}) = 0$ and $\operatorname{rank}_{\mathbb{C}}\left(\mathcal{J}(q_1, q_2, \ldots, q_k)(\mathbf{a})\right) = \operatorname{rank}_{\mathbb{C}(\mathbf{x})}\left(\mathcal{J}(q_1, q_2, \ldots, q_k)\right)$. Moreover, since they are algebraically independent, the rank of $\left(\mathcal{J}(q_1, q_2, \ldots, q_k)(\mathbf{a})\right)$ is equal to k. Thus,

$$\operatorname{rank}_{\mathbb{C}}\left(\mathcal{J}(q_1, \ldots, q_k)(\mathbf{a})\right) = k\,.$$

Applying Corollary 14 to each $q_i(\mathbf{x})$ around this point $\mathbf{a} \in \mathbb{C}^n$, we get

$$q_i(\mathbf{a} + \mathbf{x}) = \sum_{j=1}^{n} x_j \cdot \frac{\partial q_i}{\partial x_j}(\mathbf{a}) \mod \langle \mathbf{x} \rangle^2\,.$$

Observe that i^{th} row of the matrix $\left(\mathcal{J}(q_1, q_2, \ldots, q_k)(\mathbf{a})\right)$ is the vector $\left(\frac{\partial q_i}{\partial x_1}(\mathbf{a}), \frac{\partial q_i}{\partial x_2}(\mathbf{a}), \cdots, \frac{\partial q_i}{\partial x_n}(\mathbf{a})\right)$ and by the choice of $\mathbf{a}$, these vectors are linearly independent. Thus, the homogeneous linear forms $u_1(\mathbf{x}), u_2(\mathbf{x}), \ldots, u_k(\mathbf{x})$ are linearly independent, where $u_i(\mathbf{x})$ is defined as

$$u_i(\mathbf{x}) = \sum_{j=1}^{n} x_j \cdot \frac{\partial q_i}{\partial x_j}(\mathbf{a})\,.$$

The rest of the proof follows immediately from the following two claims. We state the claims and use them to complete the proof of this lemma, and then move on to prove the claims.

▷ Claim 28. Let d be the degree of $g(\mathbf{x})$. Then, the homogeneous component of degree d of the polynomial $g(q_1, q_2, \ldots, q_k)$ is equal to $g(u_1, u_2, \ldots, u_k)$.

▷ Claim 29. If $g(u_1, u_2, \ldots, u_k)$ has a formula of size s' and depth Δ, then $g(\mathbf{z})$ has a formula of size at most $s'n$ and depth $\Delta + O(1)$.

To complete the proof of the lemma, observe that given the formula Φ of size at most s and depth at most Δ which computes $g(q_1, q_2, \ldots, q_k)$, we know from Lemma 8 that the homogeneous component of degree d of $g(q_1, q_2, \ldots, q_k)$ can be computed by a formula Φ_1

of size at most $O(s^2)$ and depth at most $\Delta + O(1)$. Moreover, from the homogeneity of g and Claim 28, we also know that Φ_1 computes the polynomial $g(u_1, u_2, \ldots, u_k)$, where $u_1, u_2, \ldots, u_k$ are linearly independent linear forms. Thus, from Claim 29, this implies that $g(z_1, z_2, \ldots, z_k)$ has a formula of size at most $O(s^2 n)$ and depth $\Delta + O(1)$. This completes the proof of the lemma, modulo the two claims which we prove next. ◀

Proof of Claim 28. Let $f_1, f_2, \ldots, f_k \in \mathbb{C}[\mathbf{x}]$ be polynomials which are zero modulo $\langle \mathbf{x} \rangle^2$ (i.e. every monomial in $f_1, f_2, \ldots, f_k$ has degree at least 2) such that for every i, $q_i(\mathbf{a} + \mathbf{x}) = u_i(\mathbf{x}) + f_i(\mathbf{x})$. Since g is a homogeneous polynomial of degree d, it can be expressed as

$$g(\mathbf{z}) = \sum_{\mathbf{b} \in \mathbb{N}^k, |\mathbf{b}|_1 = d} \alpha_{\mathbf{b}} \mathbf{z}^{\mathbf{b}},$$

for field constants $\alpha_{\mathbf{b}}$. Let $\mathbf{b} \in \mathbb{N}^k$ be any vector such that $|\mathbf{b}|_1 = d$. Observe that the homogeneous component of degree d of the polynomial

$$\prod_{j=1}^{k} q_j(\mathbf{a} + \mathbf{x})^{b_j} = \prod_{j=1}^{k} (u_j(\mathbf{x}) + f_j(\mathbf{x}))^{b_j}$$

is equal to $\prod_{j=1}^{k} u_j^{b_j}$. Thus, by linearity, the homogeneous component of degree d of

$$g\left(q_1(\mathbf{a} + \mathbf{x}), q_2(\mathbf{a} + \mathbf{x}), \ldots, q_k(\mathbf{a} + \mathbf{x})\right) = \sum_{\mathbf{b} \in \mathbb{N}^k, |\mathbf{b}|_1 = d} \alpha_{\mathbf{b}} \cdot \prod_{j=1}^{k} (q_j(\mathbf{a} + \mathbf{x}))^{\mathbf{b}_j}$$

equals

$$\sum_{\mathbf{b} \in \mathbb{N}^k, |\mathbf{b}|_1 = d} \alpha_{\mathbf{b}} \cdot \prod_{j=1}^{k} (u_j)^{\mathbf{b}_j},$$

which, in turn is the equal to the polynomial $g(u_1, u_2, \ldots, u_k)$. ◁

Proof of Claim 29. The idea for the proof of this claim is to show that each variable x_j can be replaced by a homogeneous linear form $\ell_j(\mathbf{z})$ in the variables $\mathbf{z}$ such that for every $i \in [k]$, the linear form u_i satisfies $u_i(\ell_1(\mathbf{z}), \ell_2(\mathbf{z}), \ldots, \ell_n(\mathbf{z})) = z_i$. Thus, under this linear transformation, the composed polynomial $g(u_1(\mathbf{x}), u_2(\mathbf{x}), \ldots, u_k(\mathbf{x})) \in \mathbb{C}[\mathbf{x}]$ gets mapped to the polynomial $g(z_1, z_2, \ldots, z_k)$. Once we can show an existence of these linear forms $\ell_1, \ell_2, \ldots, \ell_n$, the bounds on the formula size and depth follow immediately since all we need to do to obtain a formula for $g(\mathbf{z})$ is to replace every occurrence of a variable in $\mathbf{x}$, e.g x_j by the linear form $\ell_j(\mathbf{z})$. Since every such linear form has a formula of size at most k and depth $O(1)$, this process blows up the formula size by a multiplicative factor of at most $O(k)$ and the depth by an additive factor of $O(1)$.

Intuitively, to obtain these linear forms, we just *solve* the system of linear equations $U \cdot \mathbf{x}^T = \mathbf{z}^T$, where U is the $k \times n$ matrix whose i^{th} row is u_i. Since the rank of U is equal to k, let U' be an invertible $k \times k$ submatrix of U, and let V be the inverse of U' and let $v_1, v_2, \ldots, v_k$ be the rows of V. Moreover, for brevity, let us assume that U' consists of the first k columns of U. Observe that $(V \cdot U)$ is a $k \times n$ matrix such that its leftmost $k \times k$ sub-matrix is the identity matrix. We are now ready to define the linear forms $\{\ell_j : j \in [n]\}$. For $j \in [k]$, let $\ell_j(\mathbf{z})$ be equal to $v_j(\mathbf{z})$ and for $j > k$, $\ell_j(\mathbf{z})$ is defined to be zero. It is straightforward to check that this definition satisfies the desired property and we skip the details. ◁

4.2 Formula complexity of Schur polynomials

We are now ready to prove Theorem 3. Our first stop, which we reach in the next two lemmas, is to show that which shows that the elementary symmetric polynomials of degree at most $n-1$ on n variables are *well behaved*. We start by establishing a sufficient condition for their Jacobian matrix to have full rank at a point.

▶ **Lemma 30.** *Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be an n-tuple of variables. Let $J(\mathbf{x})$ be the Jacobian of $e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_{n-1}(\mathbf{x})$, and let $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{C}^n$ be such that for all $i \neq j$, $b_i \neq b_j$. Then,*

$$\mathrm{rank}_{\mathbb{C}(\mathbf{x})}\left(J(\mathbf{x})\right) = \mathrm{rank}_{\mathbb{C}}(J(\mathbf{b})) = n-1\,.$$

Proof. Using Fact 21, we can observe that the n-variate polynomials $e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_{n-1}(\mathbf{x})$ are algebraically independent. Therefore, from Theorem 12 we know that $\mathrm{rank}_{\mathbb{C}(\mathbf{x})}\left(J(\mathbf{x})\right)$ is equal to $n-1$.

Let $J'(\mathbf{x})$ be any $n-1 \times n-1$ submatrix of $J(\mathbf{x})$ of rank equal to $n-1$ and by symmetry we can assume that the columns in $J'(\mathbf{x})$ come from the first $n-1$ columns of $J(\mathbf{x})$. Thus, for $i, j \in [n-1]$, the (i,j) entry of $J'(\mathbf{x})$ is equal to $\frac{\partial e_i}{\partial x_j}$. We now show that the rank of $J'(\mathbf{b})$ over $\mathbb{C}$ is equal to $n-1$ and this would complete the proof. To this end, we observe that the determinant of $J'(\mathbf{x})$ is a non-zero scalar multiple of $\prod_{i,i' \in [n-1], i \neq i'}(x_i - x_{i'})$. Since the coordinates of $\mathbf{b}$ are all distinct, this determinant remains non-zero on $\mathbf{b}$.

From the definition of $J'(\mathbf{x})$, we know that the entries in its i^{th} row are homogeneous polynomials of degree equal to $i-1$. Thus, $\det(J'(\mathbf{x}))$ is a homogeneous polynomial in $\mathbf{x}$ of degree equal to

$$0 + 1 + 2 + \cdots + n - 2 = \frac{(n-2)(n-1)}{2}\,.$$

Recall that the (i,j) entry of $J'(\mathbf{x})$, is equal to $\frac{\partial e_i}{\partial x_j}$, which is equal to $\sum_{S \subseteq [n]/\{i\}} \prod_{k \in S} x_k$. Thus, if we replace every occurrence of the variable x_i by the variable $x_{i'}$ for $i \neq i' \in [n-1]$, then columns i, i' in $J'(\mathbf{x})$ become identical, and hence $\det(J'(\mathbf{x}))$ is identically zero. Thus, by Lemma 15, $(x_i - x_{i'})$ is a factor of $\det(J'(\mathbf{x}))$. Also, for any two distinct sets $\{i_1, i_1'\}$ and $\{i_2, i_2'\}$ where $i_1 \neq i_1'$ and $i_2 \neq i_2'$, the polynomials $(x_{i_1} - x_{i_1'})$ and $(x_{i_2} - x_{i_2'})$ do not share a non-trivial divisor. Thus, the determinant of $J'(\mathbf{x})$ must be divisible by the polynomial $\prod_{i,i' \in [n-1], i \neq i'}(x_i - x_{i'})$. Moreover, we observed that the degree of determinant of $J'(\mathbf{x})$ is equal to $\frac{(n-2)(n-1)}{2}$, which is also equal to the degree of $\prod_{i,i' \in [n-1], i \neq i'}(x_i - x_{i'})$. Thus, they must be non-zero scalar multiples of each other. This observation, together with the fact that the coordinates of $\mathbf{b}$ are all distinct, shows that $\det(J'(\mathbf{x}))$ is non-zero at $\mathbf{b}$ and hence, $\mathrm{rank}(J(\mathbf{b}))$ equals $n-1$ over $\mathbb{C}$. ◀

We now use Lemma 27 to show that $e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_{n-1}(\mathbf{x})$ satisfy Property S, where n is the number of variables.

▶ **Lemma 31.** *Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be an n-tuple of variables. Then, the set of elementary symmetric polynomials of degree at most $n-1$ on $\mathbf{x}$, i.e. the set $\{e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_{n-1}(\mathbf{x})\}$ of polynomials satisfies Property S.*

Proof. Let $\mathbf{a} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$, where ω is the primitive n^{th} root of unity. So, we have the following identity

$$z^n - 1 = \prod_{i=1}^{n}(z - \omega^{i-1})\,.$$

However, from Fact 17, we also know that $z^n - 1$, which equals $\prod_{i=1}^{n}(z - a_i)$ can be expressed as

$$\prod_{i=1}^{n}(z - a_i) = z^n - c_1 \cdot z^{n-1} + c_2 \cdot z^{n-2} - \cdots + (-1)^n c_n \,,$$

where $c_i = e_i(\mathbf{a})$. Comparing these two expressions for $z^n - 1$, we get that for all $1 \leq i < n$, $c_i = e_i(\mathbf{a}) = 0$. Thus, $\mathbf{a} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$ is a common zero of $e_1, e_2, \ldots, e_{n-1}$, which satisfies the first item in Definition 26. Moreover, since ω is a primitive n^{th} root of one, we also know that for all pairs $i \neq j$, $a_i \neq a_j$. Therefore, by Lemma 30, $\mathbf{a}$ satisfies the second condition in Definition 26. Thus, the n-variate polynomials $e_1(\mathbf{x}), e_2(\mathbf{x}), \ldots, e_{n-1}(\mathbf{x})$ satisfy Property S. ◀

We now observe that analogous statements are also true for homogeneous symmetric polynomials and power symmetric polynomials as well.

▶ **Lemma 32.** *Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be an n-tuple variables. Then the set of complete symmetric polynomials of degree at most $n-1$ on $\mathbf{x}$, i.e. the set $\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_{n-1}(\mathbf{x})\}$ of polynomials satisfies Property S. Similarly, the set $\{p_1(\mathbf{x}), p_2(\mathbf{x}), \ldots, p_{n-1}(\mathbf{x})\}$ of power symmetric polynomials of degree at most $n-1$ also satisfies Property S.*

Proof sketch. The proof goes via generating functions of e_i, h_i, p_i and the relations among them. We denote the generating function of e_i, h_i, p_i by $E(t), H(t), P(t)$ respectively. The following relations are known between these polynomials. (See for instance [28].)

$$E(t) = \prod_{i \geq 1}(1 + x_i t) = \sum_{k \geq 0} e_k t^k$$

$$H(t) = \prod_{i \geq 1} \frac{1}{1 - x_i t} = \sum_{k \geq 0} h_k t^k$$

$$P(t) = \prod_{i \geq 1} \frac{x_i}{1 - x_i t} = \sum_{k \geq 1} p_k t^{k-1}$$

It is easy to see that $E(-t)H(t) = 1$. Therefore we get,

$$H(t) = \frac{1}{E(-t)} = \frac{1}{1 - e_1 t + e_2 t^2 \ldots + (-1)^n e_n t^n} \tag{1}$$

From Lemma 31, we know there exists a point $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, where $e_1(\mathbf{a}), e_2(\mathbf{a}), \ldots, e_{n-1}(\mathbf{a})$ vanish and $e_n(\mathbf{a})$ is non-zero. We evaluate the above equation at the same $\mathbf{a}$. We denote the above evaluation by $H(t)|_{\mathbf{x}=\mathbf{a}}$.

$$H(t)|_{\mathbf{x}=\mathbf{a}} = 1 - (-1)^n e_n(\mathbf{a}) t^n + ((-1)^n e_n(\mathbf{a}) t^n)^2 \ldots$$

Observing the equation for $H(t)|_{\mathbf{x}=\mathbf{a}}$, it follows that at point $\mathbf{a}$, $h_1(\mathbf{a}), h_2(\mathbf{a}), \ldots, h_{n-1}(\mathbf{a})$ are zero and $h_n(\mathbf{a})$ is non-zero. An analogous relation between $P(t)$ and $E(t)$, given by

$$P(t) = \frac{E'(-t)}{E(-t)} \,,$$

where $E'(-t)$ is the first derivative of $E(-t)$ with respect to t, can be used to show that the power symmetric polynomials $p_1(\mathbf{a}), \ldots, p_{n-1}(\mathbf{a})$ are zero and $p_n(\mathbf{a})$ is non-zero.

Thus, we are halfway towards showing that the set of power symmetric polynomials of degree at most $n-1$ and homogeneous symmetric polynomials of degree at most $n-1$ also satisfy Property S. It remains to be argued that the rank of the Jacobian of these polynomials at $\mathbf{a}$ is equal to $n-1$. The proof is analogous to the argument for the elementary symmetric polynomials, as in the proof of Lemma 31. We skip the remaining details for brevity. ◀

From Definition 26 of Property S and Theorem 12, it follows that is a set A of polynomials satisfies Property S, then all the non-empty subsets of A also satisfy Property S. Thus, we have the following corollary of Lemma 31 and Lemma 32.

▶ **Corollary 33.** *Let $i_1 < \cdots < i_k < n$ be positive integers. Consider the sets $E = \{e_{i_1}(\mathbf{x}), \ldots, e_{i_k}(\mathbf{x})\}$ and $H = \{h_{i_1}(\mathbf{x}), \ldots, h_{i_k}(\mathbf{x})\}$ of elementary and homogeneous symmetric polynomials respectively in $n > i_k$ variables. Then both E and H satisfy property S.*

We are now ready to prove the main theorem.

▶ **Theorem 34** (Main theorem). *Let $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ be a partition of d such that $\lambda_i \geq \lambda_{i+1} + (\ell - 1)$ for all $i \in [\ell - 1]$, and $\lambda_\ell \geq \ell$. Then, for all n, such that $n \geq \lambda_1 + \ell$, if $s_\lambda(x_1, \ldots, x_n)$ has an algebraic formula of size s and depth Δ, then the $\ell \times \ell$ determinant $(\det_\ell)$ has an algebraic formula of size poly(s) and depth $\Delta + O(1)$.*

Proof. We use Jacobi-Trudi Identity from Theorem 24, which expresses $s_\lambda(\mathbf{x})$ in the form of homogeneous symmetric determinant.

$$s_\lambda = \begin{vmatrix} h_{\lambda_1} & h_{\lambda_1+1} & \ldots & h_{\lambda_1+\ell-1} \\ h_{\lambda_2-1} & h_{\lambda_2} & \ldots & h_{\lambda_2+\ell-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\lambda_\ell-\ell+1} & h_{\lambda_\ell-\ell+2} & \ldots & h_{\lambda_\ell} \end{vmatrix}$$

Let M_λ denote the matrix on the right hand side in the above equality. By our choice of λ, observe that the highest degree entry of M_λ is $h_{\lambda_1+\ell-1}$, which has degree at most $n-1$, and the lowest degree entry of M_λ is $h_{\lambda_\ell-\ell+1}$ which for $\lambda_\ell \geq \ell$ has degree at least 1. Moreover, from the choice of λ, it also follows that all the entries of M_λ are distinct.

From Lemma 32, we know that there exists a point $\mathbf{a}$ for which the n-variate polynomials $\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_{n-1}(\mathbf{x})\}$ satisfy the Property S, where n can be taken to be strictly greater than ℓ^2. Thus, now if we take the ℓ^2-variate homogeneous polynomial g to be the symbolic determinant of an $\ell \times \ell$ matrix, then s_λ is obtained by a composition of g with a subset of polynomials $h_1, h_2, \ldots, h_{n-1}$.

Thus, by Lemma 27, we get that if $g(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_{\ell^2}(\mathbf{x}))$ (i.e s_λ) has an algebraic formula of size s and depth Δ, then $g(x_1, x_2, \ldots x_{\ell^2})$ also has a small formula of size poly(s) and depth $\Delta + O(1)$. ◀

▶ Remark 35 (Contrasting hard and easy λs). Let $\lambda = (\ell^2, \ell^2 - \ell, \ldots, 2\ell, \ell, 0, 0, \ldots, 0, 0, 0)$ and let $\tilde{\lambda} = (n\ell, (n-1)\ell, \ldots, \ell^2, \ell^2 - \ell, \ldots, 3\ell, 2\ell, \ell)$, where we will take $n \geq \ell^2 + \ell$ and $\ell > 0$. It is easy to see that $\tilde{\lambda}$ forms an arithmetic progression in which the difference between the successive terms is ℓ. Whereas λ is a truncated arithmetic progression, in which $n - \ell$-many elements are zeroes and the non-zero elements form an arithmetic progression.

Here the structure of λ and $\tilde{\lambda}$ is quite similar, but the algebraic complexities of s_λ and $s_{\tilde{\lambda}}$ are different. In particular, Theorem 34 is applicable to λ. Therefore we can conclude that s_λ does not have a small algebraic formula unless the determinant has a small algebraic formula.

On the other hand, there are small formulas for $s_{\tilde{\lambda}}$. To see this, observe that $\tilde{\lambda}+\delta = ((n-1)(\ell+1)+\ell, (n-2)(\ell+1)+\ell, \ldots, (\ell+1)+\ell, \ell)$, which is also an arithmetic progression in which the difference between successive terms is $\ell+1$. Therefore, we have

$$a_{\tilde{\lambda}+\delta} = \begin{vmatrix} x_1^{\ell} & x_2^{\ell} & \cdots & x_n^{\ell} \\ x_1^{(\ell+1)+\ell} & x_2^{(\ell+1)+\ell} & \cdots & x_n^{(\ell+1)+\ell} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n-1)(\ell+1)+\ell} & x_2^{(n-1)(\ell+1)+\ell} & \cdots & x_n^{(n-1)(\ell+1)+\ell} \end{vmatrix}_{n\times n}$$

$$a_{\tilde{\lambda}+\delta} = \left(\prod_{i=1}^{n} x_i^{\ell}\right) \begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1^{\ell+1} & x_2^{\ell+1} & \cdots & x_n^{\ell+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n-1)(\ell+1)} & x_2^{(n-1)(\ell+1)} & \cdots & x_n^{(n-1)(\ell+1)} \end{vmatrix}_{n\times n}$$

$$s_{\tilde{\lambda}} = \frac{\left(\prod_{i=1}^{n} x_i^{\ell}\right) \prod_{i<j} (x_j^{\ell+1} - x_i^{\ell+1})}{\prod_{i<j} (x_j - x_i)}$$

In the above expression the numerator and denominator have small formulas and therefore using Lemma 9 we get that $s_{\tilde{\lambda}}$ has a small formula of size $\mathrm{poly}(\ell, n)$.

4.2.1 Generalization to Skew Schur Polynomials

A straightforward generalization of the previous result is to prove that a class of skew Schur polynomials(defined in [28]) also hard for formulas. They can be defined via a Jacobi-Trudi like formula.

▶ **Theorem 36.** *Let μ and λ be partitions with $\mu_i \leq \lambda_i$ for every part i. Then, the skew Schur polynomial $s_{\lambda/\mu}$ satisfies*

(1) $s_{\lambda/\mu} = \det(h_{\lambda_i-\mu_j-i+j})_{1\leq i,j\leq k}$ *where $k = l(\lambda)$.*

(2) $s_{\lambda/\mu} = \det(e_{\lambda'_i-\mu'_j-i+j})_{1\leq i,j\leq k}$ *where $k = l(\lambda')$.*

From these definitions of skew Schur polynomials, we can see that they also have ABPs of polynomial size, like Schur polynomials. However, skew Schur polynomials are in general linear combinations of Schur polynomials, by the Littlewood-Richardson rule

$$s_{\lambda/\mu} = \sum_{\nu\vdash n-m} c^{\lambda}_{\mu,\nu} s_{\nu}.$$

The Littlewood-Richardson coefficients $c^{\lambda}_{\mu,\nu}$ count tableau whose Young diagram is of shape λ/μ and whose content satisfy certain technical conditions. They are also important in representation theory as they describe how Schur polynomials multiply, or equivalently how a tensor product of polynomial GL(n) representations decomposes into irreducible representations. They are also known to be #P-hard to compute [29].

Hardness of skew Schur polynomials assuming hardness of determinant follows from the following lemma, Corollary 33 and Lemma 27.

▶ **Lemma 37.** *Let $l \geq 2$ and $\mu_1 \geq 1$ be positive integers. Let λ, μ be partitions with l parts with $\lambda_i = (l-(i-1))l + \mu_1$ and $\mu_i = \mu_1$ for all $i < l$ and $\mu_l = \mu_1 - 1$. Then all entries of the homogeneous Jacobi-Trudi determinant $s_{\lambda/\mu}$ are distinct.*

Proof. For simplicity, we call the integer k the label of the homogeneous symmetric polynomial h_k. Here, the (i,j) entry of the matrix is $h_{(l-(i-1))l-i+j}$ for $1 \leq j < l$ and $h_{(l-(i-1))l+1-i+j}$ for $j = l$. Hence, for a fixed row i, all entries are distinct and the labels increase from left to right. Furthermore,

$$(l-i)l + 1 - (i+1) + l = (l-i)(l-1) + 1 - i < l(l-(i-1)) - i + 1$$

so the label of the last entry of row $(i+1)$ is strictly less than the label of the first entry of row i. Hence, all entries of the Jacobi-Trudi determinant are distinct. ◀

4.3 Extensions of the results of Bläser and Jindal [1]

Shifted variants for formulas

A fairly direct consequence of our techniques is the following theorem which can be considered a partial generalization of the result of Bläser-Jindal [1] for algebraic formulas.

▶ **Theorem 38.** *There exist field constants $a_1, a_2, \ldots, a_n$ such that the following is true: if for an n-variate homogeneous polynomial g over $\mathbb{C}$, the composition $g(e_1 - a_1, \ldots, e_n - a_n)$ has a formula of size at most s, then the polynomial $g(\mathbf{y})$ has a formula of size at most $O(s^2 n)$.*

Proof. We will primarily use Lemma 27 to prove the above theorem. Note that the set $\{e_1 - a_1, \ldots, e_n - a_n\}$ continue to be algebraically independent as the Jacobian matrix $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n) = \mathcal{J}(e_1, \ldots, e_n)$. Recall that $(e_1 \ldots e_n)$ are algebraically independent over n variables and thus has a full rank Jacobian matrix, which also implies the algebraic independence of $(e_1 - a_1, \ldots, e_n - a_n)$ using Theorem 12. The next step would be to find a point $\mathbf{c}$ where $e_i(c) - a_i$ is zero for all i. We also need to make sure that at point $\mathbf{c}$ the $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)$ matrix has full rank. It is easy to verify that the determinant of $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)$ is a polynomial of degree $\binom{n}{2}$. As per our assumption, the field we use is quite large, in fact much larger than n^2. Thus, from the Polynomial Identity lemma, we know that over every large enough set $S \subseteq \mathbb{C}$, there exists a point $\mathbf{c} \in S^n$ at which the determinant of $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)$ is non-zero and hence, the matrix $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)(\mathbf{c})$ is full rank. By setting $a_i = e_i(\mathbf{c})$ for all i, $\{e_1 - a_1, \ldots, e_n - a_n\}$ satisfy the Property S mentioned in Definition 26 for point $\mathbf{c}$. The proof of this theorem is now an immediate corollary of Lemma 27. ◀

The proof above gives a slightly stronger statement than what is stated in Theorem 38. Moreover, the statement of Theorem 38 holds for many such $a_1, a_2, \ldots, a_n$. To see this, note that the only property that the proof above uses about $a_1, a_2, \ldots, a_n$ is that $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ is a point in the image of the polynomial map σ from $\mathbb{C}^n$ to $\mathbb{C}^n$ which is given by mapping $\mathbf{y} \in \mathbb{C}^n$ to $(e_1(\mathbf{y}), e_2(\mathbf{y}), \ldots, e_n(\mathbf{y}))$ such that there is a point $\mathbf{c}$ in the pre-image of $\mathbf{a}$ where the Jacobian $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)(\mathbf{b})$ (which is equal to $\mathcal{J}(e_1, \ldots, e_n)(\mathbf{b})$) is full rank. Now, observe that the map σ is invertible. To see this, note that σ can be thought of as mapping n roots $b_1, b_2, \ldots, b_n$ of the univariate polynomial $\prod_{i=1}^{n}(z - b_i)$ to its coefficients, and hence its inverse is the map which maps the n coefficients of a monic degree n polynomial to its roots.

Thus, if we take $\mathbf{b}$ to be a random point from a large enough grid, then the Jacobian $\mathcal{J}(e_1 - a_1, \ldots, e_n - a_n)$ has rank n with high probability. Moreover, whenever this event happens, Theorem 38 holds with $\mathbf{a}$ being set to be the image of $\mathbf{b}$ under σ.

Generalizing the results in [1] to other bases

We know that the fundamental theorem of symmetric polynomials also holds for other bases and not just for elementary symmetric basis. For any given n-variate symmetric polynomial f_{sym}, Bläser-Jindal efficiently finds f such that $f_{sym} = f(e_1, e_2 \dots e_n)$. The degree of f is also given apriori. We generalize the Bläser-Jindal work to other bases such as homogeneous symmetric base and power symmetric base efficiently. In order to prove that, we need to show there exists an efficient transformation which can represent any elementary symmetric polynomial in the form of homogeneous symmetric or power symmetric polynomial. The following lemma illustrates that.

▶ **Lemma 39.** *Any n-variate elementary symmetric polynomial of degree k can be written as an algebraic combination of homogeneous symmetric polynomials (or power symmetric polynomials) using a small formula.*

Proof. It is well known that these transformations are doable using ABP of polynomial size as the transformation uses small determinants [28] . We prove the same for formula. Recall that $E(t)H(-t) = 1$

$$E(t) = \frac{1}{H(-t)} = \frac{1}{1 - h_1 t + h_2 t^2 \dots + (-1)^n h_n t^n} = \frac{1}{1-z} = \sum_{i \geq 0} z^i$$

where $z = h_1 t - h_2 t^2 \dots + (-1)^{n-1} h_n t^n$.

Consider the truncated polynomial $A(t) = E(t) \mod \langle z \rangle^{k+1}$, where $\langle z \rangle$ is the ideal generated by z. Now we use interpolation to find the coefficient of t^k in the polynomial $A(t)$, which is precisely e_k. $A(t)$ can have degree at most nk, which implies a formula size of $O(nk)$ for $A(t)$ (assuming the field to be algebraically closed). But, $k \leq n$, hence the trivial formula complexity for expressing e_k in the form of h_i's is $O(n^4)$ using Lemma 8.

From the definition of the generating functions $P(t)$ and $E(t)$, it is easy to verify that

$$E(t) = e^{\int P(-t)dt} = e^{\int (\sum_{m \geq 1} p_m t^{m-1})dt} = e^{(\sum_{m \geq 1} \frac{p_m t^m}{m!})} = 1 + q + \frac{q^2}{2!} \cdots ,$$

where $q = \sum_{m \geq 1} \frac{p_m t^m}{m!}$.

Now we consider the truncated polynomial containing degree up to q^k and interpolate this polynomial to get the coefficient of t^k. The formula complexity for expressing e_k as power symmetric polynomials is $O(n^4)$. Also, the proof outline is very similar to the homogeneous symmetric case. ◀

▶ **Theorem 40** (Bläser-Jindal for other bases)**.** *For any n-variate symmetric polynomial $f_{sym} \in \mathbb{C}[\mathbf{x}]$, we can efficiently compute the polynomials $f_E, f_H, f_P \in \mathbb{C}[\mathbf{x}]$ that are n-variate and unique s.t $f_{sym} = f_E(e_1, e_2 \dots e_n) = f_H(h_1, h_2 \dots h_n) = f_P(p_1, p_2 \dots p_n)$.*

Proof. Bläser-Jindal proves that f_E can be efficiently computed using an algebraic circuit. They prove that the circuit size for computing f_E is bounded by $O(d^2 S(f_{sym}) + \text{poly}(n, d))$ where d is the degree of f_E, $S(f_{sym})$ denotes the circuit size of f_{sym} and n is the number of working variables. We extend their work for computing f_H and f_P. To prove that, we shall use Bläser-Jindal method as a black box. We use Bläser-Jindal technique and get the circuit for f_E. We denote this circuit by $\mathcal{C}_{f_E}$. $\mathcal{C}_{f_E}$ can be visualized as a circuit having elementary symmetric polynomials(e_i's) as its input. But, from Lemma 39, we know e_i's can be uniquely expressed in the form of h_i's and p_i's using a formula of polynomial size. We

denote the modified circuit as $\mathcal{C}_{f_H}$ after transforming e_i's to h_i's in $\mathcal{C}_{f_E}$. The output of $\mathcal{C}_{f_H}$ is f_H because h_i's are algebraically independent and also satisfy fundamental theorem of symmetric polynomials. The proof for f_P is similar. Thus f_H and f_P can still be computed using a circuit of size $O(d^2S(f_{sym})+\text{poly}(n,d))$ as the transformation just adds some poly factor to the previously calculated size. ◀

Bläser-Jindal method works for the circuit and we do not know whether it works for ABPs or formula. If it works for ABP, then the basis transformation is trivial using a small determinant (i.e a small ABP). If it works for formula, then Theorem 40 would be useful to extend the notion for other bases while still staying in the formula regime.

4.4 Partial derivatives of a product of algebraically independent polynomials

We digress a little in this section to discuss another application of the ideas used in the proof of Lemma 27 to a question of Amir Shpilka on the partial derivative complexity of a product of algebraically independent polynomials. We start with the definition of the partial derivative complexity.

▶ **Definition 41.** *The partial derivative complexity of a polynomial $P \in \mathbb{C}[\mathbf{x}]$ is the dimension of the linear space of polynomials over $\mathbb{C}$ spanned by all the partial derivatives of P.*

The following question was asked by Amir Shpilka and our techniques provide a partial answer. As far as we are aware, the general question remains open.

▶ **Question 42** (Shpilka)**.** *Let $g_1, g_2, \ldots, g_k \in \mathbb{C}[\mathbf{x}]$ be algebraically independent polynomials. Then, prove (or disprove) that the partial derivative complexity of the the product $\prod_{i=1}^{k} g_i(\mathbf{x})$ is at least $\exp(\Omega(k))$.*

A canonical example of polynomials satisfying the hypothesis is when $g_i(\mathbf{x}) = x_i$. Thus, the product polynomial $\prod_{i=1}^{k} g_i(\mathbf{x})$ is equal to the monomial $x_1 \cdot x_2 \cdots x_k$, and indeed the partial derivative complexity of this monomial is at least 2^k, since for every $S \subseteq [k]$, the monomial $\prod_{i \in S} x_i$ is a partial derivative and these monomials are all linearly independent over $\mathbb{C}$. Thus, in general, we cannot hope for a better lower bound on the partial derivative complexity of such polynomials. Using our techniques, we observe the following two statements which answer special cases of this question.

▶ **Theorem 43.** *Let $g_1, g_2, \ldots, g_k \in \mathbb{C}[\mathbf{x}]$ be algebraically independent polynomials which satisfy Property S. Then, the partial derivative complexity of $\prod_{i=1}^{k} g_i(\mathbf{x})$ is at least 2^k.*

▶ **Theorem 44.** *Let $g_1, g_2, \ldots, g_k \in \mathbb{C}[\mathbf{x}]$ be algebraically independent polynomials. Then, there are field constants $a_1, a_2, \ldots, a_n$ such that the partial derivative complexity of $\prod_{i=1}^{k} (g_i(\mathbf{x}) + a_i)$ is at least 2^k.*

In fact, the lower bound holds for almost all choices of $a_1, a_2, \ldots, a_k$.

The following observation essentially follows from the definition of Property S (Definition 26) and Theorem 13. The proof is also implicit in the proof of Lemma 27.

▶ **Observation 45.** *Let $q_1(\mathbf{x}), q_2(\mathbf{x}), \ldots, q_k(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$ be algebraically independent polynomials which satisfy Property S. Then, there is an $\mathbf{a} \in \mathbb{C}^n$ such that the degree zero homogeneous component of the the polynomials $q_1(\mathbf{x}+\mathbf{a}), q_2(\mathbf{x}+\mathbf{a}), \ldots, q_k(\mathbf{x}+\mathbf{a})$ are all zero, and their homogeneous components of degree one are all linearly independent.*

Theorem 43 and Theorem 44 are essentially immediate consequences of Observation 45 and some standard properties of partial derivatives, which we now discuss.

▶ **Lemma 46.** *Let $P \in \mathbb{C}[\mathbf{x}]$ be a polynomial. Then,*

1. *For every $\mathbf{a} \in \mathbb{C}^n$, the partial derivative complexity of P is equal to the partial derivative complexity of $P(\mathbf{x}+\mathbf{a})$. More generally, the partial derivative complexity is invariant under invertible linear transformation of variables.*
2. *Let i be the degree of the lowest degree homogeneous component of P which is non-zero and let P_i denote this homogeneous component. Then, the partial derivative complexity of P is at least as large as the partial derivative complexity of P_i.*
3. *The partial derivative complexity of a product of k linearly independent homogeneous linear forms is at least 2^k.*

We briefly sketch the main ideas in the proof.

Proof Sketch. For the first item, we prove the statement for first order partial derivatives. The argument easily extends to higher order derivatives as well. By the chain rule, $\frac{\partial P(\mathbf{x}+\mathbf{a})}{\partial x_i}$ equals $\frac{\partial P(\mathbf{x}+\mathbf{a})}{\partial (x_i+a_i)} \cdot \frac{\partial (x_i+a_i)}{\partial x_i}$. Observe that this is equal to the polynomial obtained by taking the partial derivative $\frac{\partial P(\mathbf{x})}{\partial x_i}$ of the original polynomial and then shifting the variables by $\mathbf{a}$, i.e. replacing x_j by $x_j + a_j$ for every j. Thus, the linear space spanned by the first order partial derivatives of $P(\mathbf{x}+\mathbf{a})$ is equal to the linear space obtained by taking the space of first order partial derivatives of $P(\mathbf{x})$ and shifting the variables by $\mathbf{a}$. It is not hard to see that this preserves the dimension of the space. The proof of the *moreover* part needs a bit more care, but follows similarly.

For the second item, observe that for any set of polynomials $\{Q_1, Q_2, \ldots, Q_t\}$, the dimension of the linear span of $\{Q_1, Q_2, \ldots, Q_t\}$ is at least as large as the dimension of the linear span of the lowest degree non-zero homogeneous components of $Q_1, Q_2, \ldots, Q_t$. Also, observe that for any monomial $\mathbf{x}^{\mathbf{b}}$, if the partial derivative $\frac{\partial P_i}{\partial \mathbf{x}^{\mathbf{b}}}$ is non-zero, then the lowest degree non-zero homogeneous component of $\frac{\partial P}{\partial \mathbf{x}^{\mathbf{b}}}$ equals $\frac{\partial P_i}{\partial \mathbf{x}^{\mathbf{b}}}$. Now, let S be the set of monomials such that the space of partial derivatives of P_i with respect to monomials in S is a basis for the linear space of all partial derivatives of P_i. From the two earlier observations in this paragraph, it follows that the derivatives of P with respect to monomials in the set S are all linearly independent, thereby implying the desired lower bound.

The third item is an immediate consequence of the observation that the partial derivative complexity of the monomial $\prod_{i=1}^{k} x_i$ is equal to 2^k and the "moreover" part of the first item of this lemma, which says that partial derivative complexity is invariant under invertible linear transformations. ◀

We now sketch the main ideas in the proof of Theorem 43.

Proof of Theorem 43. The goal is to prove a lower bound on the partial derivative complexity of the polynomial $\prod_{i=1}^{k} q_i(\mathbf{x})$. Let $\mathbf{a} \in \mathbb{C}^n$ be the point guaranteed by Observation 45. Thus, $q_1(\mathbf{x}+\mathbf{a}), q_2(\mathbf{x}+\mathbf{a}), \ldots, q_k(\mathbf{x}+\mathbf{a})$ are all zero, and their homogeneous components of degree one are all linearly independent. From the first item of Lemma 46, we know that it suffices to prove a lower bound on the partial derivative complexity of $\prod_{i=1}^{k} q_i(\mathbf{x}+\mathbf{a})$.

The claim now is that the lowest degree homogeneous component of $\prod_{i=1}^{k} q_i(\mathbf{x}+\mathbf{a})$ of which is non-zero is the homogeneous component of degree equal to k, and this is equal to the product of the homogeneous components of degree one of $q_1(\mathbf{x}+\mathbf{a}), q_2(\mathbf{x}+\mathbf{a}), \ldots, q_k(\mathbf{x}+\mathbf{a})$. This immediately follows from Claim 28 in the proof of Lemma 27. But once we have this claim, the theorem follows from the third item of Lemma 46. We skip rest of the details. ◀

Theorem 44 follows from observing that we can pick $a_1, a_2, \ldots, a_n$ so that $q_1 + a_1, q_2 + a_2, \ldots, q_k + a_k$ satisfy Property S. This follows from a similar observation in the proof of Theorem 38. Once we have this observation, we are back in the setting of Theorem 43.

Before we conclude this section, we note that in order to generalize Theorem 43 and Theorem 44 to completely answer Question 42 in affirmative, it would suffice to prove the following conjecture.

▶ **Conjecture 47.** *For all constants $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{C}$ and linearly independent homogeneous linear forms $\ell_1, \ell_2, \ldots, \ell_k$ the following is true : if $q_1, q_2, \ldots, q_k$ are polynomials such that their minimum degree non-zero monomial has degree at least two, then the partial derivative complexity of the polynomial $\prod_{i=1}^{k}(\alpha_i + \ell_i + q_i)$ is at least 2^k.*

If the conjecture is false, a counterexample to the conjecture may be instructive towards understanding how the partial derivative complexity behaves over taking a product of polynomials.

5 Open problems

We conclude with some open problems.

(i) Perhaps the most natural question would be to characterize the formula complexity of *all* Schur polynomials. As discussed in Remark 35, there exist partitions λ for which the corresponding s_λ have small (polynomial sized) algebraic formulas. On the other hand, as shown in this work, there exist families of λs which do not have polynomial sized formulas unless the determinant does. Due to classical results such as [40], we know that the latter class of λs in fact have formulas of size $n^{O(\log n)}$. It would be of great interest to extend these two results and get a complete characterization of the formula complexity of s_λ, as a function of the partition λ.

(ii) Bläser and Jindal gave a computationally efficient version of the fundamental theorem for symmetric polynomials. In particular, they showed that if f_{sym} is a symmetric polynomial computed by a polynomial-sized algebraic circuit then the unique polynomial f_E such that $f_{sym} = f_E(e_1, \ldots, e_n)$, also has a polynomial-sized algebraic circuit.
A natural question one can ask is: if f_{sym} has a polynomial-sized algebraic formula (or ABP) then does f_E also have a polynomial-sized algebraic formula (ABP resp.)? In Theorem 38 we take a step towards proving this statement. We show that there exists $\mathbf{a} \in \mathbb{C}^n$ such that if f_{sym} can be expressed as $f_E(e_1 - a_1, \ldots, e_n - a_n)$ for a homogeneous f_E then f_E has a small algebraic formula if f_{sym} does. To get the exact Bläser-Jindal-like statement in the formula setting, we would have to improve our result in two ways. We would have to prove it for general f_E rather than for homogeneous f_E and we would have to prove it for $\mathbf{a} = 0^n$. We believe that both of these are interesting directions to pursue.

(iii) Another interesting extension of the results here would be to show that there are families of Generalized Vandermonde matrices such that circuit complexity of computing their permanent is essentially as large as the circuit complexity of the Permanent. This would be a VNP analogue of Theorem 3.

(iv) Yet another interesting direction would be to extend Theorem 43 to answer Question 42 completely.

References

1 Markus Bläser and Gorav Jindal. On the Complexity of Symmetric Polynomials. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2019.47.

2 Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Algorithms and Computation in Mathematics. Springer, 2000. URL: https://www.springer.com/gp/book/9783540667520.

3 Cy Chan, Vesselin Drensky, Alan Edelman, Raymond Kan, and Plamen Koev. On computing schur functions and series thereof. *Journal of Algebraic Combinatorics*, 50(2):127–141, September 2019. doi:10.1007/s10801-018-0846-y.

4 Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity. *Foundations and Trends in Theoretical Computer Science*, 2011.

5 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Some closure results for polynomial factorization and applications. *CoRR*, abs/1803.05933, 2018. arXiv:1803.05933.

6 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure of VP under taking factors: a short and simple proof. *CoRR*, abs/1903.02366, 2019. arXiv:1903.02366.

7 James Demmel and Plamen Koev. Accurate and efficient evaluation of schur and jack functions. *Mathematics of Computation*, 75(253):223–239, 2006. URL: http://www.jstor.org/stable/4100151.

8 Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the roots: Uniform closure results for algebraic classes under factoring. In *STOC*, pages 1152–1165, 2018. doi:10.1145/3188745.3188760.

9 Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2010. Preliminary version in STOC'08. doi:10.1137/080735850.

10 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2014.

11 Sergey Fomin, Dima Grigoriev, and Gleb Koshevoy. Subtraction-free complexity, cluster transformations, and spanning trees. *Found. Comput. Math.*, 16(1):1–31, February 2016. doi:10.1007/s10208-014-9231-y.

12 Sergey Fomin, Dima Grigoriev, Dorian Nogneng, and Éric Schost. On semiring complexity of schur polynomials. *Comput. Complex.*, 27(4):595–616, December 2018. doi:10.1007/s00037-018-0169-3.

13 Hervé Fournier, Nutan Limaye, Meena Mahajan, and Srikanth Srinivasan. The shifted partial derivative complexity of elementary symmetric polynomials. *Theory of Computing*, 13(1):1–34, 2017. doi:10.4086/toc.2017.v013a009.

14 Vesselin Gasharov. Incomparability graphs of (3+ 1)-free posets are s-positive. *Discrete Mathematics*, 157(1-3):193–197, 1996.

15 Dima Grigoriev and Gleb Koshevoy. Complexity of tropical schur polynomials. *Journal of Symbolic Computation*, 74:46–54, 2016. doi:10.1016/j.jsc.2015.05.005.

16 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

17 Sean Hallgren, Alexander Russell, and Amnon Ta-Shma. Normal subgroup reconstruction and quantum computation using group representations. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 627–635. ACM, 2000.

18 E. R. Heineman. Generalized vandermonde determinants. *Transactions of the American Mathematical Society*, 31(3):464–476, 1929. URL: http://www.jstor.org/stable/1989528.

19 Pavel Hrubes and Amir Yehudayoff. Homogeneous formulas and symmetric polynomials. *Computational Complexity*, 20(3):559–578, 2011. doi:10.1007/s00037-011-0007-3.

20 Christian Ikenmeyer and Stefan Mengel. On the relative power of reduction notions in arithmetic circuit complexity. *Information Processing Letters*, 130:7–10, 2018.

21 Christian Ikenmeyer and Greta Panova. Rectangular kronecker coefficients and plethysms in geometric complexity theory. *Advances in Mathematics*, 319:40–66, 2017.

22 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*. doi:10.1007/s00037-004-0182-6.

23 Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989.

24 Plamen. Koev. Accurate computations with totally nonnegative matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(3):731–751, 2007. doi:10.1137/04061903X.

25 J. M. Landsberg and Zach Teitler. On the ranks and border ranks of symmetric tensors. *Foundations of Computational Mathematics*, 10(3):339–366, June 2010. doi:10.1007/s10208-009-9055-3.

26 Veerle Ledoux and Simon JA Malham. Introductory schubert calculus, 2010.

27 Dick Lipton and Ken Regan. Arithmetic complexity and symmetry, 2009. URL: https://rjlipton.wordpress.com/2009/07/10/arithmetic-complexity-and-symmetry/.

28 Ian G. Macdonald. *Symmetric functions and Hall polynomials*. Oxford University Press, 1979.

29 Hariharan Narayanan. On the complexity of computing kostka numbers and littlewood-richardson coefficients. *Journal of Algebraic Combinatorics*, 24(3):347–354, 2006.

30 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. Available on citeseer:10.1.1.90.2644. doi:10.1007/BF01294256.

31 Ryan O'Donnell and John Wright. Quantum spectrum testing, 2015. arXiv:1501.05028.

32 Luke Oeding. Border ranks of monomials, 2016. arXiv:1608.02530.

33 Bruce Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Springer, 2001.

34 Nitin Saxena. Diagonal Circuit Identity Testing and Lower Bounds. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, pages 60–71, 2008. doi:10.1007/978-3-540-70575-8_6.

35 Amir Shpilka. Affine projections of symmetric polynomials. In *In Proc. 16th Annual IEEE Conference on Computational Complexity*, pages 160–171, 2001.

36 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. Preliminary version in the *14th Annual IEEE Conference on Computational Complexity (CCC 1999)*. doi:10.1007/PL00001609.

37 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010. doi:10.1561/0400000039.

38 Richard P Stanley. On the number of reduced decompositions of elements of coxeter groups. *European Journal of Combinatorics*, 5(4):359–372, 1984.

39 Richard P Stanley. Enumerative combinatorics. vol. 2, volume 62 of. *Cambridge Studies in Advanced Mathematics*, 1999.

40 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983. Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*. doi:10.1137/0212043.

41 Herbert S. Wilf. *Generatingfunctionology*. A. K. Peters, Ltd., third edition, 2006.

Algorithms and Lower Bounds for De Morgan Formulas of Low-Communication Leaf Gates

Valentine Kabanets
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
https://www.cs.sfu.ca/~kabanets/
kabanets@cs.sfu.ca

Sajin Koroth
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
http://www.sfu.ca/~skoroth/
sajin_koroth@sfu.ca

Zhenjian Lu
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
Department of Computer Science, University of Warwick, UK
zhenjian_lu@sfu.ca

Dimitrios Myrisiotis
Department of Computing, Imperial College London, UK
d.myrisiotis17@ic.ac.uk

Igor C. Oliveira
Department of Computer Science, University of Warwick, UK
https://www.dcs.warwick.ac.uk/~igorcarb/
igor.oliveira@warwick.ac.uk

Abstract

The class $\mathsf{FORMULA}[s] \circ \mathcal{G}$ consists of Boolean functions computable by size-s de Morgan formulas whose leaves are any Boolean functions from a class $\mathcal{G}$. We give *lower bounds* and (SAT, Learning, and PRG) *algorithms* for $\mathsf{FORMULA}[n^{1.99}] \circ \mathcal{G}$, for classes $\mathcal{G}$ of functions with *low communication complexity*. Let $R^{(k)}(\mathcal{G})$ be the maximum k-party number-on-forehead randomized communication complexity of a function in $\mathcal{G}$. Among other results, we show that:

- The Generalized Inner Product function GIP_n^k cannot be computed in $\mathsf{FORMULA}[s] \circ \mathcal{G}$ on more than $1/2 + \varepsilon$ fraction of inputs for
$$s = o\left(\frac{n^2}{\left(k \cdot 4^k \cdot R^{(k)}(\mathcal{G}) \cdot \log(n/\varepsilon) \cdot \log(1/\varepsilon)\right)^2}\right).$$
This significantly extends the lower bounds against bipartite formulas obtained by [61]. As a corollary, we get an average-case lower bound for GIP_n^k against $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{PTF}^{k-1}$, i.e., sub-quadratic-size de Morgan formulas with degree-$(k-1)$ PTF (polynomial threshold function) gates at the bottom.
- There is a PRG of seed length $n/2 + O\left(\sqrt{s} \cdot R^{(2)}(\mathcal{G}) \cdot \log(s/\varepsilon) \cdot \log(1/\varepsilon)\right)$ that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$. For the special case of $\mathsf{FORMULA}[s] \circ \mathsf{LTF}$, i.e., size-$s$ formulas with LTF (linear threshold function) gates at the bottom, we get the better seed length $O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\right)$. In particular, this provides the first non-trivial PRG (with seed length $o(n)$) for intersections of n half-spaces in the regime where $\varepsilon \leq 1/n$, complementing a recent result of [44].
- There exists a randomized 2^{n-t}-time #SAT algorithm for $\mathsf{FORMULA}[s] \circ \mathcal{G}$, where
$$t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log^2(s) \cdot R^{(2)}(\mathcal{G})}\right)^{1/2}.$$
In particular, this implies a nontrivial #SAT algorithm for $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{LTF}$.
- The Minimum Circuit Size Problem is not in $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$; thereby making progress on hardness magnification, in connection with results from [45, 12]. On the algorithmic side, we show that the concept class $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ can be PAC-learned in time $2^{O(n/\log n)}$.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 15; pp. 15:1–15:41

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

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Circuit complexity; Theory of computation → Communication complexity; Theory of computation → Complexity classes

Keywords and phrases de Morgan formulas, circuit lower bounds, satisfiability (SAT), pseudorandom generators (PRGs), learning, communication complexity, polynomial threshold functions (PTFs), parities

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.15

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2020/018/`.

Funding *Igor C. Oliveira*: This work was funded in part by a Royal Society University Research Fellowship (URF\R1\191059).

Acknowledgements We would like to thank Rocco Servedio for bringing to our attention the work by Kalai, Mansour, and Verbin [36], which is a central ingredient in the proof of Theorem 7. We also thank Mahdi Cheraghchi for several discussions on the analysis of Boolean circuits with a bottom layer of parity gates.

1 Introduction

A (de Morgan) Boolean formula over $\{0,1\}$-valued input variables $x_1, \ldots, x_n$ is a binary tree whose internal nodes are labelled by AND or OR gates, and whose leaves are marked with a variable or its negation. The power of Boolean formulas has been intensively investigated since the early years of complexity theory (see, e.g., [57, 42, 38, 5, 47, 30, 27, 58, 19]). The techniques underlying these complexity-theoretic results have also enabled algorithmic developments. These include learning algorithms [52, 55], satisfiability algorithms (cf. [59]), compression algorithms [14], and the construction of pseudorandom generators [29] for Boolean formulas of different sizes. But despite many decades of research, the current non-trivial algorithms and lower bounds apply only to formulas of less than cubic size, and understanding larger formulas remains a major open problem in circuit complexity.

In many scenarios, however, understanding smaller formulas whose leaves are replaced by certain functions would also be very useful. Motivated by several recent works, we initiate a systematic study of the $\mathsf{FORMULA} \circ \mathcal{G}$ model, i.e., Boolean formulas whose leaves are labelled by an arbitrary function from a fixed class $\mathcal{G}$. This model unifies and generalizes a variety of models that have been previously studied in the literature:

- Oliveira, Pich, and Santhanam [45] show that proving certain lower bounds against formulas of size $n^{1+\varepsilon}$ over parity (XOR) gates would have significant consequences in complexity theory. Note that de Morgan formulas of size $n^{3+\varepsilon}$ can simulate such devices. Therefore, a better understanding of the $\mathsf{FORMULA} \circ \mathcal{G}$ model even when $\mathcal{G} = \mathsf{XOR}$ is *necessary* before we are able to analyze super-cubic size formulas.[1]
- Tal [61] obtains almost quadratic lower bounds for the model of bipartite formulas, where there is a fixed partition of the input variables into $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, and a formula leaf can compute an *arbitrary* function over either $\vec{x}$ or $\vec{y}$. This model was originally investigated by Pudlák, Rödl, and Savický [49], where it was referred to as graph complexity. The model is also equivalent to PSPACE-protocols in communication complexity (cf. [23]).

[1] We remark that even a single layer of XOR gates can compute powerful primitives, such as error-correcting codes and hash functions.

- Abboud and Bringmann [1] consider formulas where the leaves are threshold gates whose input wires can be arbitrary functions applied to either the first or the second half of the input. This extension of bipartite formulas is denoted by $\mathcal{F}_2$ in [1]. Their work establishes connections between faster $\mathcal{F}_2$-SAT algorithms, the complexity of problems in P such as Longest Common Subsequence and the Fréchet Distance Problem, and circuit lower bounds.
- Polytopes (i.e. intersection of half-spaces), which corresponds to $\mathcal{G}$ being the family of linear-threshold functions, and the formula contains only AND gates as internal gates. The constructing of PRGs for this model has received significant attention in the literature (see [44] and references therein).

We obtain in a unified way several new results for the $\mathsf{FORMULA} \circ \mathcal{G}$ model, for natural classes $\mathcal{G}$ of functions which include parities, linear (and polynomial) threshold functions, and indeed many other functions of interest. In particular, we show that this perspective leads to stronger lower bounds, general satisfiability algorithms, and better pseudorandom generators for a broad class of functions.

1.1 Results

We now describe in detail our main results and how they contrast to previous works. Our techniques will be discussed in Section 1.2, while a few open problems are mentioned in Section 1.3.

We let $\mathsf{FORMULA}[s] \circ \mathcal{G}$ denote the set of Boolean functions computed by formulas containing at most s leaves, where each leaf computes according to some function in $\mathcal{G}$. The set of parity functions and their negations will be denoted by XOR.

We use the following notation for communication complexity. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we let $D(f)$ be the two-party deterministic communication complexity of f, where each party is given an input of $n/2$ bits. Similarly, for a Boolean function $g\colon \{0,1\}^n \to \{0,1\}$, we denote by $R_\delta^{(k)}(g)$ the communication cost of the best k-party *number-on-forehead* (NOF) communication protocol that computes g with probability at least $1-\delta$ on every input, where the probability is taken over the random choices of the protocol. For simplicity, we might omit the superscript (k) from $R_\delta^{(k)}(g)$ when $k=2$. One of our results will also consider k-party *number-in-hand* (NIH) protocols, and this will be clearly indicated in order to avoid confusion. We always assume a canonical partition of the input coordinates in all statements involving k-party communication complexity, unless stated otherwise. We generalize these definitions for a class of functions $\mathcal{G}$ in the natural way. For instance, we let $R_\delta^{(k)}(\mathcal{G}) = \max_{g\in\mathcal{G}} R_\delta^{(k)}(g)$.

Our results refer to standard notions in the literature, but in order to fix notation, Section 2 formally defines communication protocols, Boolean formulas, and other notions relevant in this work. We refer to the textbooks [39] and [32] for more information about communication complexity and Boolean formulas, respectively. To put our results into context, here we only briefly review a few known upper bounds on the communication complexity of certain classes $\mathcal{G}$.

Parities (XOR) and Bipartite Formulas. Clearly, the deterministic two-party communication complexity of any parity function is at most 2, since to agree on the output it is enough for the players to exchange the parity of their relevant input bits. Moreover, note that the bipartite formula model discussed above precisely corresponds to formulas whose leaves are computed by a two-party protocol of communication cost at most 1.

Halfspaces and Polynomial Threshold Functions (PTFs). Recall that a halfspace, also known as a Linear Threshold Function (LTF), is a Boolean function of the form $\mathsf{sign}(\sum_i^n a_i \cdot x_i - b)$, where each $a_i, b \in \mathbb{R}$ and $x \in \{0,1\}^n$, and that a degree-d PTF is its natural generalization where degree-d monomials are allowed. It is known that if $g(x_1, \ldots, x_n)$ is a halfspace, then its randomized two-party communication complexity, namely $R_\delta^{(2)}(g)$, satisfies $R_\delta^{(2)}(g) = O(\log(n) + \log(1/\delta))$ [43]. On the other hand, if $g(x_1, \ldots, x_n)$ is a degree-d PTF, then $R_\delta^{(d+1)}(g) = O\big((d \log d)(d \log n + \log(1/\delta))\big)$ [43, 64].

Degree-d Polynomials over $\mathsf{GF}(2)$. It is well known that a degree-d $\mathsf{GF}(2)$-polynomial admits a $(d+1)$-party deterministic protocol of communication cost $d+1$ under *any* variable partition, since in the number-on-forehead model each monomial is entirely seen by some player. In particular, the Inner Product function $\mathsf{IP}_n(x, y) = \sum_i x_i \cdot y_i \pmod 2$ satisfies $R_{1/3}^{(3)}(\mathsf{IP}_n) = O(1)$.

1.1.1 Lower bounds

Prior to this work, the only known lower bound against $\mathsf{FORMULA} \circ \mathsf{XOR}$ or bipartite formulas was the recent result of [61] showing that IP_n is hard (even on average) against nearly sub-quadratic formulas. In contrast, we obtain a significantly stronger result and establish lower bounds for different Boolean functions. We define such functions next.

GIP_n^k. The Generalized Inner Product function $\mathsf{GIP}_n^k \colon \{0,1\}^n \to \{0,1\}$ is defined as

$$\mathsf{GIP}_n^k\left(x^{(1)}, x^{(2)}, \ldots, x^{(k)}\right) = \sum_{j=1}^{n/k} \bigwedge_{i=1}^{k} x_j^{(i)} \pmod 2,$$

where $x^{(i)} \in \{0,1\}^{n/k}$ for each $i \in [k]$.

MKtP. In the Minimum Kt Problem, where Kt refers to Levin's time-bounded Kolmogorov complexity[2], we are given a string $x \in \{0,1\}^n$ and a string 1^ℓ. We accept $(x, 1^\ell)$ if and only if $\mathsf{Kt}(x) \leq \ell$.

MCSP. In the Minimum Circuit Size Problem, we are given as input the description of a Boolean function $f \colon \{0,1\}^{\log n} \to \{0,1\}$ (represented as an n-bit string), and a string 1^ℓ. We accept $(f, 1^\ell)$ if and only the circuit complexity of f is at most ℓ.

▶ **Theorem 1** (Lower bounds). *The following unconditional lower bounds hold:*

1. *If GIP_n^k is $(1/2+\varepsilon)$-close under the uniform distribution to a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$, then*

$$s \;=\; \Omega\left(\frac{n^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$

[2] For a string $x \in \{0,1\}^*$, $\mathsf{Kt}(x)$ denotes the minimum value $|M| + \log t$ taken over M and t, where M is a machine that prints x when it computes for t steps, and $|M|$ is the description length of M according to a fixed universal machine U.

2. *If* $\mathsf{MKtP} \in \mathit{FORMULA}[s] \circ \mathcal{G}$*, then*

$$s = \widetilde{\Omega}\left(\frac{n^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G})}\right).$$

3. *If* $\mathsf{MCSP} \in \mathit{FORMULA}[s] \circ \mathsf{XOR}$*, then* $s = \widetilde{\Omega}(n^2)$*, where* $\widetilde{\Omega}$ *hides inverse* $\mathsf{polylog}(n)$ *factors.*

Observe that, while [61] showed that the Inner Product function IP_n is hard against sub-quadratic bipartite formulas, Theorem 1 Item 1 yields lower bounds against formulas whose leaves can compute bounded-degree PTFs and $\mathsf{GF}(2)$-polynomials, including IP_n. Previously, only sub-linear lower bounds were known [43, 64] for circuits with PTF gates of similar degree.

Let us now comment on the relevance of Items 2 and 3. Both MCSP and MKtP are believed to be computationally much harder than GIP^k_n. However, it is more difficult to analyze these problems compared to GIP^k_n because the latter is mathematically "structured," while the former problems do not seem to be susceptible to typical algebraic, combinatorial, and analytic techniques.

More interestingly, MCSP and MKtP play an important role in the theory of hardness magnification (see [45, 12]). In particular, if one could show that MCSP restricted to an input parameter $\ell \leq n^{o(1)}$ is not in $\mathsf{FORMULA}[n^{1+\varepsilon}] \circ \mathsf{XOR}$ for some $\varepsilon > 0$, then it would follow that NP cannot be computed by Boolean formulas of size n^c, where $c \in \mathbb{N}$ is arbitrary. Theorem 1 makes partial progress on this direction by establishing the first lower bounds for these problems in the $\mathsf{FORMULA} \circ \mathcal{G}$ model. (We note that the proof of Theorem 1 Item 3 requires instances where the parameter ℓ is $n^{\Omega(1)}$.)

1.1.2 Pseudorandom generators

We also get pseudorandom generators (PRGs) against $\mathsf{FORMULA} \circ \mathcal{G}$ for various classes of functions $\mathcal{G}$. Recall that a PRG against a class of functions $\mathfrak{C}$ is a function G mapping short Boolean strings (seeds) to longer Boolean strings, so that every function in $\mathfrak{C}$ accepts G's output on a uniformly random seed with about the same probability as that for an actual uniformly random string. More formally, $G\colon \{0,1\}^\ell \to \{0,1\}^n$ is a PRG that ε-fools $\mathfrak{C}$ if for every Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ in $\mathfrak{C}$, we have

$$\left| \mathop{\mathbf{Pr}}_{z \in \{0,1\}^\ell}[h(G(z)) = 1] - \mathop{\mathbf{Pr}}_{x \in \{0,1\}^n}[h(x) = 1] \right| \leq \varepsilon.$$

Furthermore, we require G to run in deterministic time $\mathsf{poly}(n)$ on an input string $z \in \{0,1\}^\ell$. The parameter $\ell = \ell(n)$ is called the seed length of the PRG and is the main quantity to be minimized when constructing PRGs.

There exists a PRG that fools formulas of size s and that has a seed of length $s^{1/3+o(1)}$ [29]. In particular, there are non-trivial PRGs for n-variate formulas of size nearly n^3. Unfortunately, such PRGs cannot be used to fool even linear size formulas over parity functions, since the naive simulation of these enhanced formulas by standard Boolean formulas requires size n^3. Moreover, it is not hard to see that this simulation is optimal: Andreev's function, which is hard against formulas of nearly cubic size (cf. [27]), can be easily computed in $\mathsf{FORMULA}[O(n)] \circ \mathsf{XOR}$. Given that a crucial idea in the construction of the PRG in [29] (shrinkage under restrictions) comes from this lower bound proof, new techniques are needed in order to approach the problem in the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

More generally, extending a computational model for which strong PRGs are known to allow parities at the bottom layer can cause significant difficulties. A well-known example is AC^0 circuits and their extension to AC^0-XOR. While the former class admits PRGs of poly-logarithmic seed length (see e.g. [56]), the most efficient PRG construction for the latter has seed length $(1-o(1))\cdot n$ [21]. Consequently, designing PRGs of seed length $\leq (1-\Omega(1))\cdot n$ can already be a challenge. We are not aware of previous results on PRGs for $\mathsf{FORMULA}\circ\mathcal{G}$ for any non-trivial class $\mathcal{G}$.

By combining ideas from circuit complexity and communication complexity, we construct PRGs of various seed lengths for $\mathsf{FORMULA}\circ\mathcal{G}$, where $\mathcal{G}$ ranges from the class of parity functions to the much larger class of functions of bounded randomized k-party communication complexity.

▶ **Theorem 2** (Pseudorandom generators). *Let $\mathcal{G}$ be a class of n-bits functions. Then,*

1. *In the context of parity functions, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s]\circ\mathsf{XOR}$ of seed length*

$$\ell \;=\; O\big(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)+\log(n)\big)\,.$$

2. *In the context of two-party randomized communication complexity, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s]\circ\mathcal{G}$ of seed length*

$$\ell \;=\; n/2+O\Big(\sqrt{s}\cdot\Big(R^{(2)}_{\varepsilon/(6s)}(\mathcal{G})+\log(s)\Big)\cdot\log(1/\varepsilon)\Big)\,.$$

More generally, for every $k(n)\geq 2$, let $\mathcal{G}$ be the class of functions that have k-party number-in-hand (NIH) *$(\varepsilon/6s)$-error randomized communication protocols of cost at most $R^{(k\text{-NIH})}_{\varepsilon/(6s)}$. There exists a* PRG *that ε-fools $\mathsf{FORMULA}[s]\circ\mathcal{G}$ with seed length*

$$\ell \;=\; n/k+O\Big(\sqrt{s}\cdot\Big(R^{(k\text{-NIH})}_{\varepsilon/(6s)}+\log(s)\Big)\cdot\log(1/\varepsilon)+\log(k)\Big)\cdot\log(k).$$

3. *In the setting of k-party* NOF *randomized communication complexity, there is a* PRG *that ε-fools $\mathsf{FORMULA}[s]\circ\mathcal{G}$ of seed length*

$$\ell \;=\; n-\frac{n}{O\Big(\sqrt{s}\cdot k\cdot 4^k\cdot\Big(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})+\log(n)\Big)\cdot\log(n/\varepsilon)\Big)}.$$

A few comments are in order. Under a standard connection between PRGs and lower bounds (see e.g. [33]), improving the dependence on s in the seed length for $\mathsf{FORMULA}[s]\circ\mathsf{XOR}$ (Theorem 2 Item 1) would require the proof of super-quadratic lower bounds against $\mathsf{FORMULA}\circ\mathsf{XOR}$. We discuss this problem in more detail in Section 1.3. Note that the additive term $n/2$ is necessary in Theorem 2 Item 2, since the model computes in particular every Boolean function on the first $n/2$ input variables (i.e. a protocol of communication cost 1). Similarly, $\ell\geq(1-1/k)\cdot n$ in Theorem 2 Item 3. Removing the exponential dependence on k would also require advances in state-of-the-art lower bounds for multiparty communication complexity.

Theorem 2 Item 2 has an interesting implication for fooling a well-studied class of functions: *intersections of halfspaces*.[3] Note that an intersection of halfspaces is precisely a polytope, or equivalently, the set of solutions of a 0-1 integer linear program. Such objects have found applications in many fields, including optimization and high-dimensional geometry.

[3] Clearly, the intersection of s functions can be computed by an enhanced formula of size $s+1$.

After a long sequence of works on the construction of PRGs for bounded-weight halfspaces, (unrestricted) halfspaces, and generalizations of these classes,[4] the following results are known for the intersection of m halfspaces over n input variables. Gopalan, O'Donnell, Wu, and Zuckerman [24] gave a PRG for this class for error ε with seed length

$$O\big(m \cdot \log(m/\varepsilon) + \log n) \cdot \log(m/\varepsilon)\big).$$

Note that the seed length of their PRG becomes trivial if the number of halfspaces is linear in n. More recently, O'Donnell, Servedio and Tan [44] constructed a PRG with seed length

$$\mathsf{poly}(\log(m), 1/\varepsilon) \cdot \log(n).$$

Their PRG has a much better dependence on m, but it cannot be used in the small error regime. For example, the seed length becomes trivial if $\varepsilon = 1/n$. In particular, before this work it was open to construct a non-trivial PRG for the following natural setting of parameters (cf. [44, Section 1.2]): intersection of n halfspaces with error $\varepsilon = 1/n$.

We obtain the following consequence of Theorem 2 Item 2, which follows from a result of Viola [64] on the k-party *number-in-hand* randomized communication complexity of a halfspace.

▶ **Corollary 3** (Fooling intersections of halfspaces in the low-error regime). *For every $n, m \in \mathbb{N}$ and $\varepsilon > 0$, there is a pseudorandom generator with seed length*

$$O\Big(n^{1/2} \cdot m^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\Big).$$

that ε-fools the class of intersections of m halfspaces over $\{0,1\}^n$.

We note that the PRG from Theorem 2 Item 3 can fool, even in the exponentially small error regime, not only intersections of halfspaces, but also small formulas over bounded-degree PTFs.

Finally, Theorem 2 Item 2 yields the first non-trivial PRG for formulas over symmetric functions. Let SYM denote the class of symmetric Boolean functions on any number of input variables.

▶ **Corollary 4** (Fooling sub-quadratic formulas over symmetric gates). *For every $n, s \in \mathbb{N}$ and $\varepsilon > 0$, there is a pseudorandom generator with seed length*

$$O\Big(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(1/\varepsilon)\Big).$$

that ε-fools n-variate Boolean functions in $\mathsf{FORMULA}[s] \circ \mathsf{SYM}$.

Prior to this work, Chen and Wang [13] proved that the number of satisfying assignments of an n-variate formula of size s over symmetric gates can be approximately counted to an additive error term $\leq \varepsilon \cdot 2^n$ in deterministic time $\exp(n^{1/2} \cdot s^{1/4+o(1)}\sqrt{(\log(n) + \log(s))})$, where $\varepsilon > 0$ is an arbitrary constant. While their upper bound is achieved by a white-box algorithm, Corollary 4 provides a (black-box) PRG for the same task.

[4] We refer to the recent reference [44] for an extensive review of the literature in this area.

1.1.3 Satisfiability algorithms

In the #SAT problem for a computational model $\mathcal{C}$, we are given as input the description of a computational device $D(x_1, \ldots, x_n)$ from $\mathcal{C}$, and the goal is to count the number of satisfying assignments for D. This generalizes the SAT problem for $\mathcal{C}$, where it is sufficient to decide whether D is satisfiable by some assignment.

In this section, we show that #SAT algorithms can be designed for a broad class of functions. We consider the $\mathsf{FORMULA} \circ \mathcal{G}$ model for classes $\mathcal{G}$ that admit two-party communication protocols of bounded cost. We establish a general result in this context which can be used to obtain algorithms for previously studied classes of Boolean circuits.

To put our #SAT algorithms for $\mathsf{FORMULA} \circ \mathcal{G}$ into context, we first mention relevant related work on the satisfiability of Boolean formulas. Recall that in the very restricted setting of CNF formulas, known algorithms run (in the worst-case) in time $2^{n-o(n)}$ when the input formulas can have a super-linear number of clauses (cf. [18]). On the other hand, for the class of general formulas, there is a better-than-brute-force algorithm for formulas of size almost n^3. In more detail, for any $\varepsilon > 0$, there is a deterministic #SAT algorithm for $\mathsf{FORMULA}[n^{3-\varepsilon}]$ that runs in time $2^{n-n^{\Omega(\varepsilon)}}$ [59]. No results are known for formulas of cubic size and beyond, and for the reasons explained in Section 1.1.2, the algorithm from [59] cannot even be applied to $\mathsf{FORMULA} \circ \mathsf{XOR}$.

Before stating our results, we discuss the input encoding in the #SAT problem for $\mathsf{FORMULA} \circ \mathcal{G}$. The top formula F is represented in some canonical way, while for each leaf ℓ of F, the input string contains the description of a protocol Π_ℓ computing a function in $\mathcal{G}$. Our results are robust to the encoding employed for Π_ℓ. Recall that a protocol for a two-party function is specified by a protocol tree and a sequence of functions, where each function is associated with some internal node of the tree and depends on $n/2$ input bits. Since a protocol of communication cost $o(n)$ has a protocol tree containing at most $2^{o(n)}$ nodes, it can be specified by a string of length $2^{n/2+o(n)}$. Our algorithms will run in time closer to 2^n, and using a fully explicit input representation for the protocols is not an issue. Another possibility for the input representation is to use "computational efficient" protocols. Informally, the next bit messages of such protocols can be computed in polynomial time from the current transcript of the protocol and a player input. An advantage of this representation is that an input to our #SAT problem can be succinctly represented. We observe that these input representations can be generalized to randomized two-party protocols in natural ways. We refer to Section 2 for a formal presentation.

We obtain non-trivial satisfiability algorithms assuming upper bounds on the two-party deterministic and randomized communication complexities of functions in $\mathcal{G}$.

▶ **Theorem 5** (Satisfiability algorithms). *The following results hold.*

1. *There is a deterministic* #SAT *algorithm for* $\mathsf{FORMULA}[s] \circ \mathcal{G}$ *that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log(s) \cdot (\log(s) + D(\mathcal{G}))}\right).$$

2. *There is a randomized* #SAT *algorithm for* $\mathsf{FORMULA}[s] \circ \mathcal{G}$ *that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log^2(s) \cdot R_{1/3}(\mathcal{G})}\right)^{1/2}.$$

Theorem 5 readily provides algorithms for many circuit classes. For instance, since one can effectively describe a randomized communication protocol for linear threshold functions [43, 64], the algorithm from Theorem 5 Item 2 can be used to count the number of satisfying assignments of Boolean devices from $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{LTF}$.

▶ **Corollary 6** (#SAT algorithm for formulas of linear threshold functions). *There is a randomized* #SAT *algorithm for* FORMULA$[s] \circ$ LTF *that runs in time*

$$2^{n-t}, \text{ where } t = \Omega\left(\frac{n}{\sqrt{s} \cdot \log^2(s) \cdot \log(n)}\right)^{1/2}.$$

In connection with Corollary 6, prior to this work essentially two lines of research have been pursued. #SAT and/or SAT algorithms were known for bounded-depth circuits of almost-linear size whose gates can compute LTFs or (low-degree) PTFs (see [15, 34, 8]), and for sub-exponential size ACC^0 circuits with two layers of LTFs at the bottom, assuming a sub-quadratic number of them in the layer next to the input variables (see [2] for this result and further related work). Corollary 6 seems to provide the first non-trivial SAT algorithm that operates with unbounded-depth Boolean devices containing a layer with a sub-quadratic number of LTFs.

Theorem 5 can be seen as a generalization of several approaches to designing SAT algorithms appearing in the literature, which often employ ad-hoc constructions to convert bottlenecks in the computation of devices from a class $\mathcal{C}$ into non-trivial SAT algorithms for $\mathcal{C}$. We observe that, before this work, [48] had made a connection between faster SAT algorithms for CNFs and the 3-party communication complexity of a specific function. Their setting is different though: it seems to work only for CNFs, and they rely on conjectured upper bounds on the communication complexity of a particular problem. More recently, [13] employed quantum communication protocols to design *approximate counting* algorithms for several problems.[5] In comparison to previous works, to our knowledge Theorem 5 is the first unconditional result that yields faster #SAT algorithms via communication complexity in a generic way.[6]

1.1.4 Learning algorithms

We describe a learning algorithm for the FORMULA$\circ$XOR class in Leslie Valiant's challenging PAC-learning model [63]. Recall that a (PAC) learning algorithm for a class of functions $\mathcal{C}$ has access to labelled examples $(x, f(x))$ from an unknown function $f \in \mathcal{C}$, where x is sampled according to some (also unknown) distribution $\mathcal{D}$. The goal of the learner is to output, with high probability over its internal randomness and over the choice of random examples (measured by a confidence parameter δ), a hypothesis h that is close to f under $\mathcal{D}$ (measured by an error parameter ε). We refer to [37] for more information about this learning model, and to Section 2 for its standard formalization.

It is known that formulas of size s can be PAC-learned in time $2^{\widetilde{O}(\sqrt{s})}$ [52]. Therefore, formulas of almost quadratic size can be non-trivially learned from random samples of an arbitrary distribution. A bit more formally, we say that a learning algorithm is *non-trivial* if it runs in time $2^n/n^{\omega(1)}$, i.e., noticeably faster than the trivial brute-force algorithm that takes time $2^n \cdot \mathsf{poly}(n)$. Obtaining non-trivial learning algorithms for various circuit classes is closely connected to the problem of proving explicit lower bounds against the class [46] (see also [55] for a systematic investigation of such algorithms). We are not aware of the existence of non-trivial learning algorithms for super-quadratic size formulas. However, it

[5] Recall that approximately counting satisfying assignments is substantially easier than solving #SAT, for which the fastest known algorithms run in time $2^{(1-o(1))n}$.

[6] It has been brought to our attention that Avishay Tal has independently discovered a SAT algorithm for bipartite formulas of sub-quadratic size (see the discussion in [1, Page 7]), which corresponds to a particular case of Theorem 5.

seems likely that such algorithms exist at least for formulas of near cubic size. As explained in Section 1.1.2, this would still be insufficient for the learnability of classes such as (linear size) $\mathsf{FORMULA} \circ \mathsf{XOR}$.

We explore structural properties of $\mathsf{FORMULA} \circ \mathsf{XOR}$ employed in previous results and boosting techniques from learning theory to show that sub-quadratic size devices from this class can be PAC-learned in time $2^{O(n/\log n)}$.

▶ **Theorem 7** ($\mathsf{FORMULA} \circ \mathsf{XOR}$ in sub-exponential time)**.** *For every constant $\gamma > 0$, there is an algorithm that* PAC *learns the class of n-variate Boolean functions $\mathit{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ to accuracy ε and with confidence δ in time $\mathsf{poly}\big(2^{n/\log n}, 1/\varepsilon, \log(1/\delta)\big)$.*

Note that a sub-exponential running time cannot be achieved for $\mathsf{FORMULA} \circ \mathcal{G}$ when we consider the communication complexity of $\mathcal{G}$. Again, the class is too large, for the same reason discussed in Section 1.1.2. It might still be possible to design a non-trivial learning algorithm in this case, but this would possibly require the introduction of new lower bound techniques for $\mathsf{FORMULA} \circ \mathsf{XOR}$.

In contrast to the algorithm mentioned above that learns (standard) formulas of size $s \leq n^{2-o(1)}$ in time $2^{\widetilde{O}(\sqrt{s})}$, the algorithm from Theorem 7 does not learn smaller formulas over parities in time faster than $2^{O(n/\log n)}$. We discuss this in more detail in Sections 1.2 and 1.3.

Finally, we mention a connection to cryptography that provides a conditional upper bound on the size of $\mathsf{FORMULA} \circ \mathsf{XOR}$ circuits that can be learned in time $2^{o(n)}$. It is well known that if a circuit class $\mathcal{C}$ can compute pseudorandom functions (or some variants of this notion), then it cannot be learned in various learning models (see e.g. [37]). It has been recently conjectured that depth-two $\mathsf{MOD}_3 \circ \mathsf{XOR}$ circuits of linear size can compute weak pseudorandom functions of exponential security [10, Conjecture 3.7]. If this conjecture holds, then such circuits cannot be learned in time $2^{o(n)}$. Since MOD_3 gates over a linear number of input wires can be simulated by formulas of size at most $O(n^{2.8})$ [54], under this cryptographic assumption it is not possible to learn $\mathsf{FORMULA}[n^{2.8}] \circ \mathsf{XOR}$ in time $2^{o(n)}$, even if the learner only needs to succeed under the uniform distribution.

1.2 Techniques

In order to explain our techniques, we focus for the most part on the design of PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when $\mathcal{G}$ is of bounded two-party randomized communication complexity (a particular case of Theorem 2 Item 2). This proof makes use of various ingredients employed in other results. After sketching this argument, we say a few words about our strongest lower bound (Theorem 1 Item 1) and the satisfiability and learning algorithms (Theorems 5 and 7, respectively).

We build on a powerful result showing that any small de Morgan formula can be approximated pointwise by a low-degree polynomial:

(A) For every formula $F(y_1, \ldots, y_m)$ of size s, there is a polynomial $p(y_1, \ldots, y_m) \in \mathbb{R}[y_1, \ldots, y_m]$ of degree $O(\sqrt{s})$ such that $|F(a) - p(a)| \leq 1/10$ on every $a \in \{0,1\}^m$.

The only known proof of this result [52] relies on a sequence of works [9, 40, 28, 20, 50, 4, 53] on quantum query complexity, generalizing Grover's search algorithm for the OR predicate [25] to arbitrary formulas. The starting point of many of our results is a consequence of **(A)** which is implicit in the work of Tal [61].

(B) Let $\mathcal{D}$ be a distribution over $\{0,1\}^m$, and $F \in \mathsf{FORMULA}[s] \circ \mathcal{G}$. Then, for every function f,

$$\text{if} \quad \Pr_{x\sim\mathcal{D}}[F(x) = f(x)] \geq 1/2 + \varepsilon, \quad \text{then} \quad \Pr_{x\sim\mathcal{D}}[h(x) = f(x)] \geq 1/2 + \exp(-t)$$

for some function h which is the XOR of at most t functions in $\mathcal{G}$, where $t = \widetilde{\Theta}(\sqrt{s} \cdot \log(1/\varepsilon))$.

Intuitively, if we could understand well enough the XOR of any small collection of functions in $\mathcal{G}$, then we can translate this into results for $\mathsf{FORMULA}[s] \circ \mathcal{G}$, as long as $s \ll n^2$. We adapt the techniques behind **(B)** to provide a general approach to constructing PRGs against $\mathsf{FORMULA} \circ \mathcal{G}$:

Main PRG Lemma. In order for a distribution $\mathcal{D}$ to ε-fool the class $\mathsf{FORMULA}[s] \circ \mathcal{G}$, it is enough for it to $\exp(-t)$-fool the class $\mathsf{XOR}_t \circ \mathcal{G}$, where $t = \widetilde{\Theta}(\sqrt{s} \cdot \log(1/\varepsilon))$.

Recall that, in Theorem 2 Item 2, we consider a class $\mathcal{G}$ of functions that admit two-party randomized protocols of cost $R = R^{(2)}_{\varepsilon/6s}(\mathcal{G})$. It is easy to see that the XOR of any t functions from $\mathcal{G}$ is a function that can be computed by a protocol of cost at most $t \cdot R$. Thus the lemma above shows that it is sufficient to fool, to exponentially small error, a class of functions of bounded two-party randomized communication complexity. Moreover, since a randomized protocol can be written as a convex combination of deterministic protocols, it is possible to prove that fooling functions of bounded deterministic communication complexity is enough.

Pseudorandom generators in the two-party communication model have been known since [31]. Their construction exploits that the Boolean matrix associated with a function of small communication cost can be partitioned into a not too large number of monochromatic rectangles. We provide in Appendix A.2 a slightly modified and self-contained construction based on explicit extractors. It achieves the following parameters: There is an explicit PRG that δ-fools any n-bit function of two-party communication cost D and that has seed length $n/2 + O(D + \log(1/\delta))$. This PRG has non-trivial seed length even when the error is exponentially small, as required by our techniques. One issue here is that the INW PRG was only shown to fool functions with low *deterministic* communication complexity. To obtain our PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when $\mathcal{G}$ admits low-cost *randomized* protocols, we first extend the analysis of the INW PRG to show that it also fools functions with low *randomized* communication complexity. Combining this construction with the aforementioned discussion completes the proof of Theorem 2 Item 2.

The argument just sketched reduces the construction of PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ when functions in $\mathcal{G}$ admit low-cost *randomized* protocols to the analysis of PRGs for functions that admit relatively low-cost *deterministic* protocols. Our lower bound proof for GIP^k_n in Theorem 1 Item 1 proceeds in a similar fashion. We combine statement **(B)** described above with other ideas to show:

Transfer Lemma (Informal). If a function correlates with some small formula whose leaf gates have low-cost *randomized* k-party protocols, then it also non-trivially correlates with some function that has relatively low-cost *deterministic* k-party protocols.

Given this result, we are able to rely on a strong average-case lower bound for GIP^k_n against k-party deterministic protocols from [7] to conclude that GIP^k_n is hard for $\mathsf{FORMULA} \circ \mathcal{G}$.

Our #SAT algorithms combine the polynomial representation of the top formula provided by **(A)**, for which we show that such a polynomial can be obtained *explicitly*, with a decomposition of the Boolean matrix at each leaf that is induced by a corresponding low-cost randomized or deterministic two-party protocol. A careful combination of these two

representations allows us to adapt a standard technique employed in the design of non-trivial SAT algorithms (fast rectangular matrix multiplication) to obtain non-trivial savings in the running time.

Finally, our learning algorithm for $\mathsf{FORMULA} \circ \mathsf{XOR}$ is a consequence of statement **(B)** above coupled with standard tools from learning theory. In a bit more detail, since a parity of parities is just another parity function, **(B)** implies that, under any distribution, every function in $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ is weakly correlated with some parity function. Using the agnostic learning algorithm for parity functions of [36], it is possible to weakly learn $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ in time $2^{O(n/\log n)}$. This weak learner can then be transformed into a (strong) PAC learner using standard boosting techniques [22], with only a polynomial blow-up over its running time.

1.3 Concluding remarks

The main message of our results is that the *computational power* of a subquadratic-size top formula is *not* significantly enhanced by leaf gates of *low communication complexity*. We believe that the idea of decomposing a Boolean device into a computational part and a layer of communication protocols will find further applications in lower bound proofs and algorithm design.

One of our main open problems is to discover a method that can analyze $\mathsf{FORMULA}[s] \circ \mathcal{G}$ when $s \gg n^2$. For instance, is it possible to adapt existing techniques to show an explicit lower bound against $\mathsf{FORMULA}[n^{2.01}] \circ \mathcal{G}$, or achieving this is just as hard as breaking the cubic barrier for formula lower bounds? Results in this direction would be interesting even for $\mathcal{G} = \mathsf{XOR}$.

Finally, we would like to mention a few questions connected to our results and their applications. Is it possible to combine the techniques behind Corollary 3 and [44] to design a PRG of seed length $n^{o(1)}$ and error $\varepsilon = 1/n$ for the intersection of n halfspaces? Can we design a satisfiability algorithm for formulas over k-party number-on-forehead communication protocols? Is it possible to learn $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ in time $2^{\widetilde{O}(\sqrt{s})}$? (The learning algorithm for formulas from [52] relies on techniques from [35], and it is unclear how to extend them to the case of $\mathsf{FORMULA} \circ \mathsf{XOR}$.)

1.4 Organization

Theorem 1 Item 1 is proved in Section 3, while Items 2 and 3 rely on our PRG constructions and are deferred to Section 4. The latter describes a general approach to constructing PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$. It includes the proof of Theorem 2 and other applications. Our satisfiability algorithms (Theorem 5) appear in Section 5. Finally, Section 6 discusses learning results for $\mathsf{FORMULA} \circ \mathsf{XOR}$ and contains a proof of Theorem 7.

2 Preliminaries

2.1 Notation

Let $n \in \mathbb{N}$; we denote $\{1, \ldots, n\}$ by $[n]$, and denote by U_n the uniform distribution over $\{0,1\}^n$. We use $\widetilde{O}(\cdot)$ (and $\widetilde{\Omega}(\cdot)$) to hide polylogarithmic factors. That is, for any $f\colon \mathbb{N} \to \mathbb{N}$, we have that $\widetilde{O}(f(n)) = O(f(n) \cdot \mathsf{polylog}(f(n)))$.

In this paper, we will mainly use $\{-1, 1\}$ as the Boolean basis. In some parts of this paper, we will use the $\{0, 1\}$ basis for the simplicity of the presentation. This will be specified in corresponding sections.

2.2 De Morgan formulas and extensions

▶ **Definition 8.** *An n-variate* de Morgan formula *is a directed rooted tree; its non-leaf vertices (henceforth,* internal gates*) take labels from* $\{\text{AND}, \text{OR}, \text{NOT}\} = \{\wedge, \vee, \neg\}$ *and its leaves (henceforth,* variable gates*) take labels from the set of variables* $\{x_1, \ldots, x_n\}$*. Each internal gate has bounded in-degree (henceforth,* fan-in*); the* NOT *gate in particular has fan-in* 1 *and every variable gate has fan-in* 0*. The* size *of a de Morgan formula is the number of its leaf gates.*

In this work, we denote by $\mathsf{FORMULA}[s]$ the class of Boolean functions computable by size-s de Morgan formulas. Let $\mathcal{G}$ denote some class of Boolean functions; then, we denote by $\mathsf{FORMULA}[s] \circ \mathcal{G}$ the class of functions computable by some size-s de Morgan formula where its leaves are labelled by functions in $\mathcal{G}$.

2.3 Approximating polynomials

▶ **Definition 9** (Point-wise approximation)**.** *For a Boolean function* $f\colon \{-1,1\}^n \to \{-1,1\}$*, we say that the function* $\tilde{f}\colon \{-1,1\}^n \to \mathbb{R}$ *ε-approximates f if for every* $z \in \{-1,1\}^n$*,*

$$\left|f(z) - \tilde{f}(z)\right| \leq \varepsilon.$$

We will need the following powerful result for the approximating degree of de Morgan formulas.

▶ **Theorem 10** ([52], see also [11])**.** *Let $s > 0$ be an integer and $0 < \varepsilon < 1$. Any de Morgan formula* $F\colon \{-1,1\}^n \to \{-1,1\}$ *of size s has a ε-approximating polynomial of degree* $d = O(\sqrt{s} \cdot \log(1/\varepsilon))$*. That is, there exists a degree-d polynomial* $p\colon \{-1,1\}^n \to \mathbb{R}$ *over the reals such that for every* $z \in \{-1,1\}^n$*,*

$$|p(z) - F(z)| \leq \varepsilon.$$

Note that Theorem 10 still holds if we use $\{0,1\}$ as the Boolean basis.

2.4 Communication complexity

We use standard definitions from communication complexity. In this paper we consider the standard two party model of Yao and its generalizations to multiparty setting. We denote deterministic communication complexity of a Boolean function by $D(f)$ in the two party setting. We refer to [39] for standard definitions from communication complexity.

▶ **Definition 11.** *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a Boolean function. The* communication matrix of f*, namely M_f, is a* $2^{n/2} \times 2^{n/2}$ *matrix defined by* $(M_f)_{x,y} := f(x,y)$*.*

▶ **Definition 12.** *A* rectangle *is a set of the form $A \times B$, for* $A, B \subseteq \{0,1\}^n$*. A* monochromatic rectangle *is a rectangle S such that for all pairs* $(x,y) \in S$ *the value* $f(x,y)$ *is the same.*

▶ **Lemma 13.** *Let* Π *be a protocol that computes* $f\colon \{0,1\}^n \to \{0,1\}$ *with at most D bits of communication. Then,* Π *induces a partition of M_f into at most* 2^D *monochromatic rectangles.*

Given a protocol, its *transcript* is the sequence of bits communicated.

▶ **Lemma 14.** *For every transcript z of some communication protocol, the set of inputs* (x,y) *that generate z is a rectangle.*

Below, we recount the definitions of two multiparty communication models used in this work, namely the number-on-forehead and the number-in-hand models.

▶ **Definition 15** ("Number-on-forehead" communication model; informal). *In the k-party "number-on-forehead" communication model, there are k players and k strings $x_1, \ldots, x_k \in \{0,1\}^{n/k}$ and player i gets all the strings except for x_i. The players are interested in computing a value $f(x_1, \ldots, x_k)$, where $f\colon \{0,1\}^n \to \{0,1\}$ is some fixed function. We denote by $D^{(k)}(f)$ the number of bits that must be exchanged by the best possible number on forehead protocol solving f.*

We also use the following weaker communication model.

▶ **Definition 16** ("Number-in-hand" communication model; informal). *In the k-party "number-in-hand" communication model, there are k players and k strings $x_1, \ldots, x_k \in \{0,1\}^{n/k}$ and player i gets only x_i. The players are interested in computing a value $f(x_1, \ldots, x_k)$, where $f\colon \{0,1\}^n \to \{0,1\}$ is some fixed function. We denote by $D^{(k\text{-NIH})}(f)$ the number of bits that must be exchanged by the best possible communication protocol.*

Note that $D^{(k\text{-NIH})}(f) \leq (1 - 1/k) \cdot n + 1$, for any n-variate Boolean function f, as if $k-1$ players write on the blackboard their string, then the player that did not reveal her input may compute $f(x_1, \ldots, x_k)$ on her own and then publish it.

For the communication models mentioned above, there are also bounded-error randomized versions, denoted by R_δ, $R_\delta^{(k)}$, and $R_\delta^{(k\text{-NIH})}$, respectively, where $0 < \delta < 1$ is an upper bound on the error probability of the protocol. In this setting, the players have access to some shared random string, say r, and the aforementioned error probability of the protocol is considered over the possible choices of r. Moreover, we require the error to be at most δ on each fixed choice of inputs.

We can extend the definitions of the communication complexity measures, defined above, to classes of Boolean functions, in a natural way. That is, for any communication complexity measure $M \in \left\{D, D^{(k)}, D^{(k\text{-NIH})}, R_\delta, R_\delta^{(k)}, R_\delta^{(k\text{-NIH})}\right\}$ and for any class of Boolean functions $\mathcal{G}$, we may define

$$M(\mathcal{G}) := \max_{g \in \mathcal{G}} M(g)\,.$$

We note that throughout this paper, we denote by n the number of input bits for the function regardless the communication models. In the k-party communication setting (either NOF or NIH), we assume without loss of generality that n is divisible by k.

2.5 Pseudorandomness

A PRG against a class of functions $\mathfrak{C}$ is a deterministic procedure G mapping short Boolean strings (seeds) to longer Boolean strings, so that G's output "looks random" to every function in $\mathfrak{C}$.

▶ **Definition 17** (Pseudorandom generators). *Let $G\colon \{-1,1\}^\ell \to \{-1,1\}^n$ be a function, $\mathfrak{C}$ be a class of Boolean functions, and $0 < \varepsilon < 1$. We say that G is a* pseudorandom generator of seed length ℓ that ε-fools $\mathfrak{C}$ *if, for every function $f \in \mathfrak{C}$, it is the case that*

$$\left| \mathop{\boldsymbol{E}}_{z \sim \{-1,1\}^\ell}[f(G(z))] - \mathop{\boldsymbol{E}}_{x \sim \{-1,1\}^n}[f(x)] \right| \leq \varepsilon.$$

A PRG G outputting n bits is called *explicit* if G can be computed in $\mathsf{poly}(n)$ time. All PRGs stated in this paper are explicit.

2.6 Learning

For a function $f : \{0,1\}^n \to \{0,1\}$ and a distribution $\mathcal{D}$ supported over $\{0,1\}^n$, we denote by $\mathrm{EX}(f,\mathcal{D})$ a randomized oracle that outputs independent identically distributed labelled examples of the form $(x, f(x))$, where $x \sim \mathcal{D}$.

▶ **Definition 18** (PAC learning model [63]). *Let $\mathcal{C}$ be a class of Boolean functions. We say that a randomized algorithm A* learns *$\mathcal{C}$ if, when A is given oracle access to $\mathrm{EX}(f,\mathcal{D})$ and inputs 1^n, ε, and δ, the following holds. For every n-variate function $f \in \mathcal{C}$, distribution $\mathcal{D}$ supported over $\{0,1\}^n$, and real-valued parameters $\varepsilon > 0$ and $\delta > 0$, $A^{\mathrm{EX}(f,\mathcal{D})}(1^n,\varepsilon,\delta)$ outputs with probability at least $1-\delta$ over its internal randomness and the randomness of the example oracle $\mathrm{EX}(f,\mathcal{D})$ a description of a hypothesis $h : \{0,1\}^n \to \{0,1\}$ such that*

$$\Pr_{x\sim\mathcal{D}}[f(x) = h(x)] \;\geq\; 1-\varepsilon.$$

The sample complexity *of a learning algorithm is the maximum number of random examples from $\mathrm{EX}(f,\mathcal{D})$ requested during its execution.*

3 Lower bounds

In this section, we prove an average-case lower bound for the generalized inner product function against $\mathsf{FORMULA}\circ\mathcal{G}$, where $\mathcal{G}$ is the set of functions that have low-cost randomized communication protocols in the number-on-forehead setting. This corresponds to Item 1 of Theorem 1. Items 2 and 3 rely on our PRG constructions, and the proofs are deferred to Section 4.

▶ **Theorem 19.** *For any integer $k \geq 2$, $s > 0$ and any class of functions $\mathcal{G}$, let $C\colon \{-1,1\}^n \to \{-1,1\}$ be a function in $\mathsf{FORMULA}[s]\circ\mathcal{G}$ such that*

$$\Pr_{x\sim\{-1,1\}^n}\left[C(x) = \mathsf{GIP}^k_n(x)\right] \geq 1/2+\varepsilon.$$

Then

$$s = \Omega\left(\frac{n^2}{k^2\cdot 16^k\cdot\left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G})+\log n\right)^2\cdot\log^2(1/\varepsilon)}\right).$$

We need a couple useful lemmas from [60], whose proofs are presented in Appendix A.1 (Lemma 50 and Lemma 51) for completeness.

▶ **Lemma 20** ([60]). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$, and let $f, C\colon \{-1,1\}^n \to \{-1,1\}$ be such that*

$$\Pr_{x\sim\mathcal{D}}[C(x) = f(x)] \geq 1/2+\varepsilon.$$

Let $\tilde{C}\colon \{-1,1\}^n \to \mathbb{R}$ be a ε-approximating function of C, i.e., for every $x \in \{-1,1\}^n$, $|C(x)-\tilde{C}(x)| \leq \varepsilon$. Then,

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)\cdot f(x)] \geq \varepsilon.$$

▶ **Lemma 21** ([60]). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$ and let $\mathcal{G}$ be a class of functions. For $f\colon \{-1,1\}^n \to \{-1,1\}$, suppose that $D\colon \{-1,1\}^n \to \{-1,1\} \in \mathit{FORMULA}[s] \circ \mathcal{G}$ is such that*

$$\underset{x\sim\mathcal{D}}{\boldsymbol{Pr}}[D(x) = f(x)] \geq 1/2 + \varepsilon_0.$$

Then there exists some $h\colon \{-1,1\}^n \to \{-1,1\} \in \mathit{XOR}_{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)} \circ \mathcal{G}$ such that

$$\underset{x\sim\mathcal{D}}{\boldsymbol{E}}[h(x)\cdot f(x)] \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

We also need the following communication-complexity lower bound for GIP.

▶ **Theorem 22** ([7, Theorem 2]). *For any $k \geq 2$, any function that computes GIP_n^k on more than $1/2+\delta$ fraction of the inputs (over uniformly random inputs) must have k-party deterministic communication complexity at least $\Omega\big(n/(k\cdot 4^k) - \log(1/\delta)\big)$.*

We first show that if a function correlates with some small formula, whose leaves are functions with low *randomized* communication complexity, then it also correlates non-trivially with some function of relatively low *deterministic* communication complexity.

▶ **Lemma 23.** *For any distribution $\mathcal{D}$ over $\{-1,1\}^n$, and any class of functions $\mathcal{G}$, let $f\colon \{-1,1\}^n \to \{-1,1\}$ and $C\colon \{-1,1\}^n \to \{-1,1\} \in \mathit{FORMULA}[s] \circ \mathcal{G}$ be such that*

$$\underset{x\sim\mathcal{D}}{\boldsymbol{Pr}}[C(x) = f(x)] \geq 1/2 + \varepsilon.$$

Then there exists a function h, with k-party deterministic communication complexity at most

$$O\left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})\cdot\sqrt{s}\cdot\log(1/\varepsilon)\right),$$

such that

$$\underset{x\sim\mathcal{D}}{\boldsymbol{Pr}}[h(x) = f(x)] \geq 1/2 + 1/s^{O(\sqrt{s}\cdot\log(1/\varepsilon))}.$$

Proof. Let $C = F(g_1, g_2 \ldots, g_s)$ be the function in $\mathsf{FORMULA}[s]\circ\mathcal{G}$, where F is a formula and $g_1, g_2, \ldots, g_s$ are leaf functions from the class $\mathcal{G}$. For each g_i, consider a k-party randomized protocol Π_i of cost at most $R = R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})$ that has an error $\varepsilon/(2s)$. Now consider the following function

$$\tilde{C}(x) := \underset{\Pi_1,\Pi_2,\ldots,\Pi_s}{\mathbf{E}}[D(x)],$$

where

$$D(x) := F(\Pi_1(x), \Pi_2(x), \ldots, \Pi_s(x)).$$

Note that for any fixed choice of $(\Pi_1, \Pi_2, \ldots, \Pi_s)$, D is a formula whose leaves are functions with *deterministic* communication complexity at most R. Next, we show the following.

▷ Claim 24. The function $\tilde{C}$ ε-approximates C.

Proof. First note that since each Π_i is a $(\varepsilon/(2s))$-error randomized protocol, by taking the union bound over the s leaf functions, we have that for every input $x \in \{-1,1\}^n$,

$$\underset{\Pi_1,\Pi_2,\ldots,\Pi_s}{\mathbf{Pr}}[\Pi_1(x) = g_1(x) \wedge \Pi_2(x) = g_2(x) \wedge \cdots \wedge \Pi_s(x) = g_s(x)] \geq 1 - \varepsilon/2.$$

Denote by $\mathcal{E}$ the event $\Pi_1(x) = g_1(x) \wedge \Pi_2(x) = g_2(x) \wedge \cdots \wedge \Pi_s(x) = g_s(x)$. We have for every $x \in \{-1,1\}^n$,

$$\begin{aligned}\tilde{C}(x) &= \mathop{\mathbf{E}}_{\Pi_1,\Pi_2,\ldots,\Pi_s}[D(x)] \\ &= \mathbf{E}[D(x) \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[D(x) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}] \\ &= C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[D(x) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}].\end{aligned}$$

On the one hand, we have

$$\tilde{C}(x) = C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[D(x) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}] \leq C(x) + \varepsilon/2.$$

On the other hand, we get

$$\tilde{C}(x) = C(x) \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[D(x) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}] \geq C(x) \cdot (1 - \varepsilon/2) + (-1) \cdot (\varepsilon/2) \geq C(x) - \varepsilon.$$

This completes the proof of the claim. ◁

Now by Claim 24 and Lemma 20, we have

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x) \cdot f(x)] \geq \varepsilon. \tag{1}$$

By the definition of $\tilde{C}$, Equation (1) implies that there exists some D, which is a formula whose leaves are functions with *deterministic* communication complexity at most R, such that

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[D(x) \cdot f(x)] \geq \varepsilon,$$

which implies

$$\mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[D(x) = f(x)] \geq 1/2 + \varepsilon/2.$$

Then by Lemma 21, there exists a function h, which can be expressed as the XOR of at most $O(\sqrt{s} \cdot \log(1/\varepsilon))$ leaf functions in D, such that

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[h(x) \cdot f(x)] \geq \frac{1}{s^{O(\sqrt{s}\cdot\log(1/\varepsilon))}},$$

which again implies

$$\mathop{\mathbf{Pr}}_{x\sim\mathcal{D}}[h(x) = f(x)] \geq \frac{1}{2} + \frac{1}{s^{O(\sqrt{s}\cdot\log(1/\varepsilon))}}.$$

Finally, note that the k-party deterministic communication complexity of h is at most

$$O(R \cdot \sqrt{s} \cdot \log(1/\varepsilon)),$$

where $R = R^{(k)}_{\varepsilon/(2s)}(\mathcal{G})$. ◀

We are now ready to show Theorem 19.

Proof of Theorem 19. Consider Lemma 23 with f being GIP^k_n and $\mathcal{D}$ being the uniform distribution. Consider Theorem 22 with $\delta = 1/s^{O(\sqrt{s}\cdot\log(1/\varepsilon))}$. We have

$$O\left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G}) \cdot \sqrt{s} \cdot \log(1/\varepsilon)\right) \geq n/(k4^k) - O\left(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)\right),$$

which implies

$$s \geq \Omega\left(\frac{n^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$ ◀

4 Pseudorandom generators

Some of our PRGs are obtained from a general framework that allows us to reduce the task of fooling $\mathsf{FORMULA} \circ \mathcal{G}$ to the task of fooling the class of functions which are the parity or conjunction of few functions from $\mathcal{G}$.

4.1 The general framework

We show that in order to get a PRG for the class of subquadratic-size formulas with leaf gates in $\mathcal{G}$, it suffices to get a PRG for very simple sublinear-size formulas: either $\mathsf{XOR} \circ \mathcal{G}$ or $\mathsf{AND} \circ \mathcal{G}$.

▶ **Theorem 25** (PRG for $\mathsf{FORMULA} \circ \mathcal{G}$ from PRG for $\mathsf{XOR} \circ \mathcal{G}$ or $\mathsf{AND} \circ \mathcal{G}$)**.** *Let $\mathcal{G}$ be a class of gates on n bits. For any integer $s > 0$ and any $0 < \varepsilon < 1$, there exists a constant $c > 0$ such that the following holds. If a distribution $\mathcal{D}$ over $\{-1,1\}^n$ $\left(2^{-c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools the XOR (parity) or the AND (conjunction) of $c \cdot \sqrt{s} \cdot \log(1/\varepsilon)$ arbitrary functions from $\mathcal{G}$, then $\mathcal{D}$ also ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$.*

Proof. We first show the case where $\mathcal{D}$ fools the parity of a few functions from $\mathcal{G}$. The proof can be easily adapted to the case of conjunction.

Let $C = F(g_1, g_2 \ldots, g_s)$ be a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$, where F is a formula, and $g_1, g_2, \ldots, g_s$ are functions from the class $\mathcal{G}$. Let U be the uniform distribution over $\{-1,1\}^n$. We need to show

$$\mathbf{E}[C(\mathcal{D})] \overset{\varepsilon}{\approx} \mathbf{E}[C(U)]. \tag{2}$$

Let p be a $(\varepsilon/3)$-approximating polynomial for F given by Theorem 10. Note that the degree of p is

$$d = O(\sqrt{s} \cdot \log(1/\varepsilon)).$$

Let us replace F, the formula part of C, with p and let

$$\tilde{C} := p(g_1, g_2 \ldots, g_s).$$

Since $\tilde{C}$ point-wisely approximates C, we have

$$\mathbf{E}[\tilde{C}(U)] \overset{\varepsilon/3}{\approx} \mathbf{E}[C(U)],$$

and

$$\mathbf{E}[\tilde{C}(\mathcal{D})] \overset{\varepsilon/3}{\approx} \mathbf{E}[C(\mathcal{D})].$$

Then to show Equation (2), it suffices to show

$$\mathbf{E}[\tilde{C}(\mathcal{D})] \overset{\varepsilon/3}{\approx} \mathbf{E}[\tilde{C}(U)].$$

We have

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}}[\tilde{C}(x)] = \mathop{\mathbf{E}}_{x \sim D}\left[\sum_{\substack{S \subseteq [s]: \\ |S| \leq d}} \hat{p}(S) \cdot \prod_{i \in S} g_i(x)\right]$$

$$= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i(x)\right]. \tag{3}$$

Now note that for each $S \subseteq [s]$, $\prod_{i\in S} g_i(x)$ computes the XOR of at most d functions from $\mathcal{G}$. Using the fact the distribution $\mathcal{D}$ $\left(\delta = 1/2^{c\cdot\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)}\right)$-fools the XOR of any d functions from $\mathcal{G}$, we get

$$\begin{aligned}
\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\tilde{C}(x)] &= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x\sim D}\left[\prod_{i\in S} g_i(x)\right] \\
&= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S) \cdot \left(\mathop{\mathbf{E}}_{x\sim U}\left[\prod_{i\in S} g_i(x)\right] + \delta_S\right) && \text{(where } |\delta_S| \leq \delta\text{)} \\
&= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \left(\hat{p}(S) \cdot \mathop{\mathbf{E}}_{x\sim U}\left[\prod_{i\in S} g_i(x)\right] + \hat{p}(S)\cdot\delta_S\right) \\
&= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S) \cdot \mathop{\mathbf{E}}_{x\sim U}\left[\prod_{i\in S} g_i(x)\right] + \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\delta_S \\
&= \mathop{\mathbf{E}}_{x\sim U}[\tilde{C}(x)] + \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\delta_S.
\end{aligned}$$

It remains to show

$$\left|\sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\delta_S\right| \leq \varepsilon/3.$$

Note that because $p(z) \in [1-\varepsilon/3, 1+\varepsilon/3]$ for every $z \in \{-1,1\}^s$, we have

$$|\hat{p}(S)| = \left|\mathop{\mathbf{E}}_{z\sim\{-1,1\}^s}\left[p(z)\cdot\prod_{i\in S} z_i\right]\right| \leq 1+\varepsilon/3 < 2.$$

Then,

$$\left|\sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\delta_S\right| \leq \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} |\hat{p}(S)|\cdot|\delta_S| \leq \delta\cdot\sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} |\hat{p}(S)| \leq \delta\cdot s^{O(\sqrt{s}\cdot\log(1/\varepsilon))} \leq \varepsilon/3,$$

where the last inequality holds for some sufficiently large constant c.

To show the case of conjunction, we can write the approximating polynomial as the sum of all degree-d monomials, each of which is the AND of at most d variables. One way to do this is to use the domain $\{0,1\}$ instead of $\{-1,1\}$ in the above argument. We need to show that the coefficients in this case still have small magnitude.

▷ Claim 26. Let $p\colon \{-1,1\}^n \to \mathbb{R}$ be a degree-d polynomial of the form

$$p(x) = \sum_{\substack{S\subseteq[n]:\\ |S|\le d}} \hat{p}(S)\cdot\prod_{i\in S} x_i,$$

and let $q\colon \{0,1\}^n \to \mathbb{R}$ be the corresponding polynomial of p over the domain $\{0,1\}^n$, of the form

$$q(y) = \sum_{\substack{T\subseteq[n]:\\ |T|\le d}} \hat{q}(T)\cdot\prod_{i\in T} y_i.$$

Then,

$$|q|_1 = \sum_{\substack{T\subseteq[n]:\\ |T|\le d}} |\hat{q}(T)| \le n^{O(d)}\cdot \max_{\substack{S\subseteq[n]:\\ |S|\le d}} |\hat{p}(S)|.$$

Proof. We have

$$\begin{aligned} q(y_1,y_2,\ldots,y_n) &= p(1-2y_1,1-2y_2,\ldots,1-2y_n)\\ &= \sum_{\substack{S\subseteq[n]:\\ |S|\le d}} \hat{p}(S)\cdot\prod_{i\in S}(1-2y_i)\\ &= \sum_{\substack{S\subseteq[n]:\\ |S|\le d}} \hat{p}(S)\cdot\left(\sum_{\ell\in\{0,1\}^{|S|}}\prod_{\substack{j\in S:\\ \ell_j=1}} -2y_j\right)\\ &= \sum_{\substack{S\subseteq[n]:\\ |S|\le d}}\sum_{\ell\in\{0,1\}^{|S|}} \hat{p}(S)\cdot(-2)^{|\ell|}\cdot\prod_{\substack{j\in S:\\ \ell_j=1}} y_j. \qquad (\text{where } |\ell| = \textstyle\sum_{i=1}^{|S|}\ell_i) \end{aligned}$$

For a pair (S,ℓ) where $S\subseteq[n]$, $|S|\le d$ and $\ell\in\{0,1\}^{|S|}$, let us define the polynomial $q_{(S,\ell)}$ as

$$q_{(S,\ell)}(y) = \hat{p}(S)\cdot(-2)^{|\ell|}\cdot\prod_{\substack{j\in S:\\ \ell_j=1}} y_j.$$

Note that there are at most $n^d\cdot 2^d$ many pairs of such (S,ℓ)'s and for each (S,ℓ), we have

$$|q_{(S,\ell)}|_1 = \left|\hat{p}(S)\cdot(-2)^{|\ell|}\right| \le 2^d\cdot|\hat{p}(S)|.$$

Finally we have

$$|q|_1 = \left|\sum_{(S,\ell)} q_{(S,\ell)}\right|_1 \le \sum_{(S,\ell)} |q_{(S,\ell)}|_1 \le n^d\cdot 2^d\cdot 2^d\cdot\max_{\substack{S\subseteq[n]:\\ |S|\le d}}|\hat{p}(S)|,$$

as desired. ◁

This completes the proof of Theorem 25. ◀

4.2 Formulas of low-communication functions in the number-in-hand setting

In this subsection, we will use $\{0,1\}$ as the Boolean basis.

▶ **Theorem 27.** *For any integers $k \geq 2$, $s > 0$ and any $0 < \varepsilon < 1$, let $\mathcal{G}$ be the class of functions that have k-party number-in-hand $(\varepsilon/6s)$-error randomized communication protocols of cost at most R. There exists a PRG that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ with seed length*

$$n/k + O\left(\sqrt{s} \cdot (R + \log(s)) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k).$$

We need the following PRG that fools single functions with low communication complexity in the number-in-hand model. The proof is presented in Appendix A.2 (Theorem 52) for completeness.

▶ **Theorem 28** ([6, 31])**.** *For any $k \geq 2$, there exists a PRG that δ-fools any n-bits functions with k-party number-in-hand deterministic communication complexity of at most D', with seed length*

$$n/k + O\left(D' + \log(1/\delta) + \log(k)\right) \cdot \log(k).$$

Next, we show a PRG for $\mathsf{FORMULA} \circ \mathcal{G}$, where $\mathcal{G}$ is the class of functions with low-cost communication protocols in the number-in-hand setting. We first show for the case of deterministic protocols.

▶ **Theorem 29.** *For any integers $k \geq 2$ and $s > 0$, let $\mathcal{G}$ be the class of functions whose k-party number-in-hand deterministic communication complexity are at most D. There is a PRG that ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$ of size s with seed length*

$$n/k + O\left(\sqrt{s} \cdot \log(1/\varepsilon) \cdot (D + \log(s)) + \log(k)\right) \cdot \log(k).$$

Proof. By Theorem 25, it suffices to show a PRG that $\left(\delta = 1/2^{c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools every function that is the XOR of $t = c \cdot \sqrt{s} \cdot \log(1/\varepsilon)$ arbitrary functions from $\mathcal{G}$. Note that such a function has deterministic communication complexity at most $D' = t \cdot D$. Then Theorem 29 follows from Theorem 28. ◀

We now establish the randomized case.

Proof of Theorem 27. Let C be a function in $\mathsf{FORMULA}[s] \circ \mathcal{G}$. For each of the leaf functions in C, consider a k-party number-in-hand randomized protocol of cost at most R that has an error at most $\varepsilon/(6s)$. By taking a union bound over the s leaf functions and by viewing a randomized protocol as a distribution of deterministic protocols (as shown in the proof of Claim 24), we get the following which is a (point-wisely) $(\varepsilon/3)$-approximating function for C:

$$\tilde{C}(x) := \sum_i p_i \cdot D_i(x),$$

where each $p_i \in [0,1]$ is some probability density value (so $\sum_i p_i = 1$), and each D_i is a formula whose leaves are functions with *deterministic* communication complexity at most R. Then to ε-fool C, it suffices to $(\varepsilon/3)$-fool its $(\varepsilon/3)$-approximating function $\tilde{C}$. Also, since $\tilde{C}$ is a convex combination of the D_i's, it suffices to $(\varepsilon/3)$-fools all the D_i's. We will do this using the PRG form Theorem 29. We get that there exists a PRG that $(\varepsilon/3)$-fools each D_i with seed length

$$n/k + O\left(\sqrt{s} \cdot (R + \log(s)) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k),$$

as desired. ◀

4.3 Applications: Fooling formulas of SYMs, LTFs, XORs, and AC^0 circuits

4.3.1 FORMULA ∘ SYM and FORMULA ∘ LTF

Here, we show how the PRG in Theorem 27 implies PRGs for FORMULA∘LTF and FORMULA∘SYM.

▶ **Theorem 30.** *For any size $s > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA$[s] \circ$ LTF with seed length*

$$O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)\right).$$

For FORMULA$[s] \circ$ SYM, the seed length is

$$O\left(n^{1/2} \cdot s^{1/4} \cdot \log(n) \cdot \log(1/\varepsilon)\right).$$

We need the fact that the class of LTF has low communication complexity in the number-in-hand model. Consider the following k-party SUM-GREATER$_m$ problem where the i-th party holds a m-bit number z_i in hand and they want to determine whether $\sum_{i=1}^{k} z_i > \theta$, where θ is a fixed number known to all the parties. Nisan [43] gave an efficient randomized protocol (with public randomness) for this problem.[7]

▶ **Theorem 31** ([43])**.** *Let $m > 0$ be an integer. For any integer $2 \le k \le m^{O(1)}$, and any $0 < \delta < 1$, there exists a δ-error randomized protocol of cost $O(k \cdot \log(m) \cdot \log(m/\delta))$ for the k-party* SUM-GREATER$_m$ *problem.*

By Theorem 31 and the fact that every linear threshold function on n bits has a representation such that the weights are $O(n \log(n))$ integers [41], we get the following.

▶ **Corollary 32.** *For every $k \ge 2$ and $0 < \delta < 1$, the k-party number-in-hand δ-error randomized communication complexity of LTF is $O(k \cdot \log(n) \cdot \log(n/\delta))$.*

Proof of Theorem 30. By Corollary 32 and Theorem 27, for *every* $k \ge 2$ we get a PRG for FORMULA ∘ LTF of seed length

$$n/k + O\left(\sqrt{s} \cdot k \cdot \log(n) \cdot \log(ns/\varepsilon) \cdot \log(1/\varepsilon) + \log(k)\right) \cdot \log(k).$$

By choosing

$$k = \frac{n^{1/2}}{s^{1/4} \cdot \log(n) \cdot \log(n/\varepsilon)},$$

the claimed seed length follows from a simple calculation.

For FORMULA ∘ SYM, note that every n-bit symmetric function has a deterministic k-party number-in-hand communication protocol of cost at most $k \cdot \log(n)$. Then the rest can be shown using a similar argument as above (by choosing $k = n^{1/2}/\left(s^{1/4} \cdot \log(n)\right)$). ◀

[7] Viola [64] gave a δ-error randomized protocol for the k-party SUM-GREATER$_m$ problem of cost $O(k \cdot \log(k) \cdot \log(m/\delta))$, which is better than Nisan's protocol when $k = m^{o(1)}$.

4.3.2 FORMULA ∘ XOR

For the case of FORMULA ∘ XOR, we get a PRG with better seed length.

▶ **Theorem 33.** *For any size $s > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA$[s] \circ$ XOR with seed length*

$$O\left(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon) + \log(n)\right).$$

Proof. By Theorem 25, to fool FORMULA$[s] \circ \mathcal{G}$, it suffices to $\left(\delta = 1/2^{O\left(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)\right)}\right)$-fool the XOR of a few functions from $\mathcal{G}$, where $\mathcal{G}$ in this case is the set of all XOR functions. Note that the XOR of any set of XOR functions simply computes some XOR function. Therefore, we can use small-bias distribution, which fools every XOR function, to fool FORMULA$[s] \circ$ XOR. Finally, note that there are known constructions for δ-bias distributions that use $O(\log(n/\delta))$ random bits (see e.g. [3]). ◀

Using the "locality" of this PRG for FORMULA ∘ XOR, we get a lower bound for MCSP against subquadratic-size formulas of XORs.

▶ **Theorem 34.** *For every integer $s > 0$, if* MCSP *on N-bit can be computed by some function in FORMULA$[s] \circ$ XOR, then $s = \tilde{\Omega}(N^2)$.*

Proof sketch. There is a standard construction of δ-bias distributions that is local (see e.g. [3, Construction 3] and [16, Fact 7]) in the following sense: There exists a circuit of size at most $\tilde{O}(\log(n/\delta) \cdot \log(n))$ such that given a seed of length $O(\log(n/\delta))$ and a index $j \in [n]$, outputs the j-th bit of the distribution. Local PRGs imply MCSP lower bounds (see [16, Section 3]). ◀

4.3.3 FORMULA ∘ AC^0

Another application of Theorem 25 is to take $\mathcal{G}$ to be the set all functions that can be computed by small constant-depth circuits (AC^0). Note the state-of-the-art PRG against size-M depth-d AC^0 has a seed length of $\log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)$ [56]. Below, let $\mathsf{AC}^0_{d,M}$ denote the class of depth-d circuits of size at most M.

▶ **Theorem 35.** *For any size $s, m > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools FORMULA $\circ \mathsf{AC}^0_{d,M}$ of size s with seed length*

$$\log^{d+O(1)}(Mn) \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon).$$

Moreover, by inspecting the construction of PRG in [56], it is not difficult to see that the PRG is also local; there exists a circuit of size at most $\lambda = \log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)$ such that given a seed of length $O\left(\log^{d+O(1)}(Mn) \cdot \log(1/\varepsilon)\right)$ and a index $j \in [n]$, outputs the j-th bit of the PRG. As a result, we get MCSP lower bounds from the this PRG.

▶ **Theorem 36.** *For every $s, d, M \in \mathbb{N}$, if* MCSP *on N-bit can be computed by some function in FORMULA$[s] \circ \mathsf{AC}^0_{d,M}$, then*

$$s \geq N^2 / \log^{2d+O(1)}(Mn).$$

4.4 Formulas of low number-on-forehead communication leaf gates

In this section, we show a PRG with mild seed length for formulas of functions with low *multi-party number-on-forehead* communication complexity.

▶ **Theorem 37.** *Let $\mathcal{G}$ be a class of n-bits functions. For any size $s > 0$, there exists a PRG that ε-fools $\mathit{FORMULA}[s] \circ \mathcal{G}$, with seed length*

$$n - \frac{n}{O\left(\sqrt{s} \cdot k \cdot 4^k \cdot \left(R^{(k)}_{\varepsilon/(2s)}(\mathcal{G}) + \log(n)\right) \cdot \log(n/\varepsilon)\right)}.$$

The PRG is constructed using the hardness vs. randomness paradigm.

4.4.1 Hardness based PRGs

We show how to construct the PRG using the average-case hardness result for formulas of functions with low multi-party communication complexity (Theorem 19). We start with some notations. For $x \in \{-1,1\}^m$ and an integer k such that k divides m, we consider a partition of x into k equal-sized consecutive blocks and write $x = x^{(1)}, x^{(2)}, \ldots, x^{(k)}$, where $x^{(i)} \in \{-1,1\}^{m/k}$ for each $i \in [k]$.

▶ **Lemma 38.** *For any integers $m, t, k > 0$ such that k divides m, t, let $\mathcal{G}$ be a class of functions on $mt + t$ bits, and let $G\colon \{-1,1\}^{m \times t} \to \{-1,1\}^{mt+t}$ be*

$$G(x_1, x_2, \ldots, x_t) = \Big(x_1^{(i)}, x_2^{(i)}, \ldots, x_t^{(i)}, \mathit{GIP}^k_m\left(x_{(i-1)\cdot(t/k)+1}\right), \\ \mathit{GIP}^k_m\left(x_{(i-1)\cdot(t/k)+2}\right), \ldots, \mathit{GIP}^k_m\left(x_{i\cdot(t/k)+1}\right)\Big)_{i\in[k]},$$

where $x_1, x_2, \ldots, x_t \in \{-1,1\}^m$. Then G is a PRG that $(t \cdot \varepsilon)$-fools $\mathit{FORMULA} \circ \mathcal{G}$ of size

$$s = \Omega\left(\frac{m^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2m^2)}(\mathcal{G}) + \log m\right)^2 \cdot \log^2(1/\varepsilon)}\right).$$

Proof. The high level idea is as follows. We argue that if there is a $\mathsf{FORMULA} \circ \mathcal{G}$ of the claimed size that breaks the PRG, then there is a $\mathsf{FORMULA} \circ \mathcal{G}'$ of the same size that computes GIP on m bits, where $\mathcal{G}'$ has a k-party communication complexity that is at most that of $\mathcal{G}$ *with respect to the m-bit input*, and hence contradicts the $\mathsf{FORMULA} \circ \mathcal{G}'$ complexity of the generalized inner product function. The resulting formula is obtained by fixing some input bits of the original $\mathsf{FORMULA} \circ \mathcal{G}$ which breaks the PRG.

We use a hybrid argument. First consider the distribution given by G, where we replace each $\mathsf{GIP}(x_j)$ ($j \in [t]$) with a uniformly random bit; let us denote those random bits as U_j for $j \in [t]$ (note that this is just the uniform distribution). Then for each $j \in [t]$, define H_j to be the distribution that we substitute back $\mathsf{GIP}(x_1), \mathsf{GIP}(x_2), \ldots, \mathsf{GIP}(x_j)$ for the corresponding uniform bits in the previous distribution.

For the sake of contradiction, suppose there exists a $\mathsf{FORMULA} \circ \mathcal{G}$ C of size s such that

$$|\mathbf{Pr}[C(H_t) = 1] - \mathbf{Pr}[F(H_0) = 1]| > t \cdot \varepsilon.$$

By the triangle inequality, there exists a $1 \le j \le k$ such that

$$|\mathbf{Pr}[C(H_j) = 1] - \mathbf{Pr}[C(H_{j-1}) = 1]| > \varepsilon.$$

Then by averaging, there exist some fixings of $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_t$ and $U_{j+1}, \ldots, U_t$ to C such that the above inequality still holds. Let us denote by C' the circuit obtained by C after such fixings and assume without loss of generality $(k-1)t/k \leq j \leq t$. Then we have

$$\left|\mathbf{Pr}\left[C'\left(x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(k)}, \mathsf{GIP}(x_j)\right) = 1\right] - \mathbf{Pr}\left[C'\left(x_j^{(1)}, x_j^{(2)}, \ldots, x_j^{(k)}, U_j\right) = 1\right]\right| > \varepsilon. \quad (4)$$

By a standard "unpredictability implies pseudorandomness" argument [66], we can show that there is some circuit C'', obtained from C' by fixing some value for the last bit, that computes the generalized inner product function on m bits with probability greater than $1/2 + \varepsilon$ over uniformly random inputs. Note that the size of C'' is the same as C' (hence also C) , and also C'' can be computed by some $\mathsf{FORMULA} \circ \mathcal{G}'$, where $R_\delta^{(k)}(\mathcal{G}') \leq R_\delta^{(k)}(\mathcal{G})$ for every δ. This contradicts the hardness of GIP for such circuits (Theorem 19). ◀

We are now ready to prove Theorem 37.

Proof of Theorem 37. Consider Lemma 38. Let $n = mt + t$, and we have $m = \left(\frac{n}{t} - 1\right)$. Then Lemma 38 gives a PRG that ε-fools $\mathsf{FORMULA} \circ \mathcal{G}$ of size

$$\begin{aligned} s &= \Omega\left(\frac{m^2}{k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2m^2)}(\mathcal{G}) + \log m\right)^2 \cdot \log^2(t/\varepsilon)}\right) \\ &\geq \Omega\left(\left(\frac{n}{t}\right)^2 / \left(k^2 \cdot 16^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right)^2 \cdot \log^2(n/\varepsilon)\right)\right), \end{aligned}$$

which yields

$$t \geq \Omega\left(\frac{n}{\sqrt{s} \cdot k \cdot 4^k \cdot \left(R^{(k)}_{\varepsilon/(2n^2)}(\mathcal{G}) + \log n\right) \cdot \log(n/\varepsilon)}\right).$$

Note that the seed length in this case is $n - t$. ◀

4.4.2 MKtP lower bounds

The PRG in Theorem 37 is sufficient to give an MKtP lower bound for formulas of functions with low multi-party communication complexity.

▶ **Theorem 39.** *For any integer $s > 0$ and any class of N-bit function $\mathcal{G}$, if* MKtP *on N-bit can be computed by some function $\mathsf{FORMULA}[s] \circ \mathcal{G}$, then*

$$s = \frac{N^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G}) \cdot \mathsf{polylog}(N)}.$$

Proof. Let C be a function in $\mathsf{FORMULA} \circ \mathcal{G}$ of size less than

$$\frac{N^2}{k^2 \cdot 16^k \cdot R^{(k)}_{1/3}(\mathcal{G}) \cdot \log^c(N)}$$

where $c > 0$ is some sufficiently large constant. By Theorem 37, we have that there is a PRG that $(1/3)$-fools C and its seed length is

$$N - \mathsf{polylog}(N).$$

Also, since the PRG is polynomial-time computable, we get that for every seed, the output of the PRG has Kt complexity at most $\theta = N - \mathsf{polylog}(N)$. However, consider the MKtP function with a threshold parameter θ; this function is not fooled by such a PRG, since it accepts every output of the PRG and rejects a uniformly random string with high probability. ◀

5 Satisfiability algorithms

In this section, we will use $\{0,1\}$ as the Boolean basis.

5.1 Computational efficient communication protocols

▶ **Definition 40** (Computational efficient communication protocols). *Let $t\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$. We say that a two-party communication protocol is* t-efficient *if for each of the parties, given an input x and some previously sent messages $\pi\in\{0,1\}^*$, the next message to send can be computed in time $t(|x|,|\pi|)$ ($\perp$ is being output if there is no next message). We say that such a protocol is* explicit *if $t(|x|,|\pi|) = 2^{o(|x|+|\pi|)}$.*

▶ **Lemma 41.** *Let $f\colon \{0,1\}^n\to 1$ and let Π be a t-efficient communication protocol for f with communication cost at most D. Then the protocol tree of Π can be output in time $O\left(D\cdot t(n/2,D)\cdot 2^{n/2}\cdot 2^D\right)$. That is, there exists an algorithm that outputs a list of all (partial and full) transcripts of length at most D and the rectangles associated with each of the transcripts.*

Proof. It suffices to show that, given an input $x\in\{0,1\}^{n/2}$ and a transcript $\ell\in\{0,1\}^{\le D}$, we can decide whether x belongs to the rectangle indexed by ℓ in time $D\cdot t(n/2,D)$. Suppose x is the input for Alice (resp. Bob), and we want to decide whether x belongs to the rectangle indexed by π. We can carry out the communication task by simulating the behavior of Alice (resp. Bob) using the protocol Π and simulating Bob's (resp. Alice's) behavior using the transcript π, and check whether the messages sent by Alice (resp. Bob) is consistent with the transcript π. This takes time at most $D\cdot t(n/2,D)$. To construct the tree, we do the above for every (partial and full) transcript $\pi\in\{0,1\}^{\le D}$ and every input $x\in\{0,1\}^{n/2}$ for Alice (resp. Bob). The total running time is $O\left(D\cdot t(n/2,D)\cdot 2^{n/2}\cdot 2^D\right)$. ◀

For a protocol Π, we denote by $\mathrm{Leaves}(\Pi)$ the set of full transcripts of Π.

▶ Remark. We note that, in the *white-box* context of the satisfiability problem, there is no need to assume a canonical partition of the input variables among the players. For instance, a helpful partition can either be given as part of the input, or computed by the algorithm. As a consequence, in instantiations of Theorem 5 for a particular circuit class $\mathcal{C}$, it is sufficient to be able to convert the input circuit from $\mathcal{C}$ into some device from $\mathsf{FORMULA}\circ\mathcal{G}$ for which protocols of bounded communication cost can be described.

5.2 Explicit approximating polynomials for formulas

From Theorem 10, we know that every size-s formula has a degree-$O(\sqrt{s})$ polynomial that point-wisely approximates it. In our SAT algorithms, we will need to *explicitly construct* such an approximating polynomial given a formula. One way to do this is to use an *efficient* quantum query algorithm for formulas. It is known that a quantum query algorithm for a function f using at most T queries implies an approximating polynomial for f of degree at most $2T$ [9], and by classically simulating such an quantum algorithm, one can show that

the approximating polynomial can be obtained in time that is polynomial in the number of its monomials, in addition to the time for the classical simulation. For our task, we can use the result of Reichardt [51] which showed an *efficient* quantum algorithm for evaluating size-s formulas with $O\left(\sqrt{s}\cdot\log s\right)$ queries.[8] Here, we present an alternate way to construct approximating polynomials for de Morgan formulas which rely only on the *existence* of such polynomials, without requiring an efficient quantum query algorithm. This "black-box" approach was suggested to us by an anonymous reviewer.

We first need the following structural lemma for formulas.

▶ **Lemma 42** ([29, 58]). *For every integer $s>0$, there exists an algorithm such that given a size-s de Morgan formula F, runs in* $\mathsf{poly}(s)$ *time and outputs a top formula F' with $O(\sqrt{s})$ leaves and each leaf of F' is a sub-formula with $O(\sqrt{s})$ input leaves.*

▶ **Lemma 43.** *For any integer $s>0$ and any $0<\varepsilon<1$, there exists an algorithm of running time $s^{O\left(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)\right)}$ such that given a de Morgan formula F of size s, outputs an ε-approximating polynomial of degree $O(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon))$ for F. That is, the algorithm outputs a multi-linear polynomial (as sum of monomials) over the reals such that for every $x\in\{0,1\}^n$,*

$$|p(x)-F(x)|\leq\varepsilon.$$

Proof. We first note that it suffices to construct a $(1/3)$-approximating polynomial for F with degree $D=O(\sqrt{s}\cdot\log(s))$. This is because given a $(1/3)$-approximating polynomial one can obtain explicitly an ε-approximating polynomial of degree $D\cdot O(\log(1/\varepsilon))$, by feeding $O(1/\varepsilon)$ copies of the $(1/3)$-approximating polynomial to the polynomial computing MAJORITY on $O(1/\varepsilon)$ bits [11] (see also [58, Appendix B]).

We first invoke Lemma 42 on F to obtain a top formula F' with $t=O(\sqrt{s})$ leaves, each of which is a sub-formula of size $O(\sqrt{s})$. We construct a $(1/20)$-approximating (multi-linear) polynomial P for the top formula F', which has degree $d_1=O(s^{1/4})$ by Theorem 10. Note that P can be constructed in time $2^{O(\sqrt{s})}$ because F' has at most $O(\sqrt{s})$ leaves. Next, for each of the t sub-formulas, denoted as $F_1,F_2,\ldots,F_t$, we construct a $(1/(20t))$-approximating polynomial. Note that these polynomials have degree $d_1=O(s^{1/4}\cdot\log(s))$ and can be constructed in time $2^{O(\sqrt{s})}$. Let's denote these t polynomials as $Q_1,Q_2,\ldots,Q_t$. Now for each Q_i ($i\in[t]$), we define

$$q_i(x)=\frac{Q_i(x)+1/(20t)}{1+1/(10t)}.$$

The final approximating polynomial for F is given as

$$p(x)=P\left(q_1(x),q_2(x),\ldots,q_t(x)\right).$$

Note that p has degree $d_1\cdot d_2=O(\sqrt{s}\cdot\log(s))$ and can be constructed (as sum of monomials) in time $s^{O(\sqrt{s}\cdot\log(s))}$. It remains to show that p $(1/3)$-approximates F.

[8] It is also known that there exists a quantum query algorithm for evaluating size-s formulas with $O\left(\sqrt{s}\right)$ queries [52], which implies the existence of an approximating polynomial for size-s formulas of degree $O\left(\sqrt{s}\right)$ (see Theorem 10). However, because this algorithm is not known to be efficient, it is unclear whether such an approximating polynomial can be constructed efficiently with respect to the number of monomials.

For $0 \leq q \leq 1$, let N_q be the distribution over $\{0,1\}$ such that $\mathbf{Pr}_{y\sim N_q}[y=1] = q$. Then for an fixed input $x \in \{0,1\}^s$, we have

$$p(x) = \mathop{\mathbf{E}}_{y_i \sim N_{q_i(x)}} [P(y_1, y_2, \ldots, y_t)]. \tag{5}$$

Let $\mathcal{E}$ be the event that $y_i = F_i(x)$ for all $i \in [t]$. Note that

$$\delta := \mathop{\mathbf{Pr}}_{y_i \sim N_{q_i(x)}} [\neg\mathcal{E}] \leq 1/10. \tag{6}$$

To see Equation (6), note that for every $i \in [t]$, if $F_i(x) = 0$, then $0 \leq q_i(x) \leq 1/(10t)$, which implies

$$\mathop{\mathbf{Pr}}_{y_i \sim N_{q_i(x)}} [y_i \neq F_i(x)] \leq 1/(10t).$$

Similar for the case when $F_i(x) = 1$ (which implies $1 - 1/(10t) < q_i(x) \leq 1$). Then Equation (6) follows from a union bound. Now we can re-write Equation (5) as

$$\begin{aligned} p(x) &= \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}] \\ &= (F'(F_1(x), F_2(x), \ldots, F_t(x)) \pm 1/20) \cdot (1-\delta) + \mathbf{E}[P(y_1, y_2, \ldots, y_t) \mid \neg\mathcal{E}] \cdot \delta. \end{aligned}$$

Note that $P(y) \in [-1/(20t), 1 + 1/(20t)]$ for every $y \in \{0,1\}^t$, and that $\delta \leq 1/10$. A simple calculation shows that

$$p(x) = F'(F_1(x), F_2(x), \ldots, F_t(x)) \pm \frac{1}{3},$$

as desired. ◀

5.3 The #SAT algorithm

In this subsection, we present our #SAT algorithm.

▶ **Theorem 44.** *For any integer $s > 0$, there exists a deterministic #SAT algorithm for $\mathsf{FORMULA}[s] \circ \mathcal{G}$, where $\mathcal{G}$ is the class of functions with explicit two-party deterministic protocols of communication cost at most D, that runs in time*

$$2^{n - \frac{n}{\sqrt{s}\cdot\log(s)\cdot(\log(s)+D)}}.$$

In the case $\mathcal{G}$ is the class of functions with explicit randomized protocols of communication cost at most R, there exists an analogous randomized algorithm with a running time

$$2^{n - \left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot R}\right)^{1/2}}.$$

The algorithm is based on the framework for designing satisfiability algorithms developed by Williams [65]. The idea is to transform a given circuit into a "sparse polynomial" and solve satisfiability by evaluating the polynomial on all points in a faster-than-brute-force manner.

We first need the following fast matrix multiplication algorithm for "narrow" matrices.

▶ **Theorem 45** ([17])**.** *Multiplication of an $N \times N^{.172}$ matrix with an $N^{.172} \times N$ matrix can be done in $O(N^2 \log^2 N)$ arithmetic operations over any field.*

For an even number $n > 0$, and $x \in \{0,1\}^n$, we denote by x^{L} (resp. x^{R}) the first half of x and $x^{\mathrm{R}} \in \{0,1\}^{n/2}$ the second half. We now prove Theorem 44.

Proof of Theorem 44. We first prove the deterministic case.

Let $C = F(g^1, g^2 \ldots, g^s)$ be a device in $\mathsf{FORMULA} \circ \mathcal{G}$ where F is a formula and $g^1, g^2, \ldots, g^s$ are functions that have a explicit communication protocol of cost at most D. The first step is to output the protocol tree for each g^i ($i \in [s]$). Since each g^i has explicit protocol of cost at most D, by Lemma 41, these protocol trees can be output in time $s \cdot 2^{n/2+D+o(n)} \leq 2^{n/1.9}$ (here we assume $D = o(n)$ otherwise the theorem holds trivially).

Let n' be an integer whose value is determined later. Let T be a set of n' variables such that T contains $n'/2$ variables from the first half of the n variables and the rest are from the second half. For a partial assignment $z \in \{0,1\}^{n'}$ to T, denote by C_z the restricted function of C where the variables in T are fixed according to z. To count the number of satisfying assignments of C, we need to compute the following quantity:

$$\sum_{x \in \{0,1\}^{n-n'}} \sum_{z \in \{0,1\}^{n'}} C_z(x). \tag{7}$$

Now consider

$$Q(x) = \sum_{z \in \{0,1\}^{n'}} C_z(x).$$

We will try to obtain the value of $Q(x)$ for every $x \in \{0,1\}^{n-n'}$, in time about $2^{n-n'}$, which will allow us to compute the quantity in Equation (7) in time $O(2^{n-n'})$ by summing $Q(x)$ over all the x's. We do this by first transforming Q into an *approximating* polynomial with not-too-many monomials, and each monomial is a product of *functions that only rely on either the first or the second half of* x. With such a polynomial, we can perform fast multipoint evaluation using the fast matrix multiplication algorithm in Theorem 45.

For each $z \in \{0,1\}^{n'}$, we view the formula C_z as $F(g_z^1, g_z^2, \ldots, g_z^s)$, where F is the de Morgan formula part of C_z and $g_z^1, g_z^2, \ldots, g_z^s$ are the leaf gates. Let us now replace F by a ε-approximating polynomial p, where $\varepsilon = 1/\left(3 \cdot 2^{n'}\right)$, using Lemma 43. Note that the degree of p is at most

$$d \leq O(\sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)) \leq O(\sqrt{s} \cdot \log(s) \cdot n').$$

Now consider the following

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} p(g_z^1(x), g_z^2(x), \ldots, g_z^s(x)).$$

First, note that by the value that we've chosen for the approximating error ε, we have that, for every x,

$$|Q'(x) - Q(x)| \leq 2^{n'} \cdot \varepsilon = 1/3.$$

In other words, given $Q'(x)$, we can recover the value of $Q(x)$, which is supposed to be an integer.

Next, we perform fast multipoint evaluation on Q'. First of all, we re-write Q' as follows:

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} \sum_{\substack{S \subseteq [s]: \\ |S| \leq d}} \hat{p}(S) \cdot \prod_{i \in S} g_z^i(x). \tag{8}$$

Now let Π_i be the protocol of g^i, we can re-write g^i_z as follows:

$$g^i_z(x) = \sum_{\pi_i \in \text{Leaves}(\Pi_i)} \alpha^i\left(z^{\text{L}}x^{\text{L}}, \pi_i\right) \cdot \beta^i\left(z^{\text{R}}x^{\text{R}}, \pi_i\right), \tag{9}$$

where $\alpha^i\left(z^{\text{L}}x^{\text{L}}, \pi_i\right)$ (resp. $\beta^i\left(z^{\text{R}}x^{\text{R}}, \pi_i\right)$) is 1 if and only if $\left(z^{\text{L}}x^{\text{L}}\right)$ (resp. $\left(z^{\text{R}}x^{\text{R}}\right)$) belongs to the rectangle indexed by π_i and the function value of that rectangle is 1. Note that for each $i \in [s]$, given the pre-computed protocol tree of the Π_i, α^i and β^i can be computed in polynomial time (for example, using binary search). After plugging Equation (9) into Equation (8) for every $i \in [s]$ and rearranging, we get

$$Q'(x) = \sum_{z \in \{0,1\}^{n'}} \sum_{\substack{S \subseteq [s]: \\ |S| \leq d}} \sum_{\substack{\vec{\pi} = (\pi_i)_{i \in S}: \\ \pi_i \in \text{Leaves}(\Pi_i)}} \hat{p}(S) \cdot \prod_{i \in S} \alpha^i\left(z^{\text{L}}x^{\text{L}}, \pi_i\right) \cdot \prod_{i \in S} \beta^i\left(z^{\text{R}}x^{\text{R}}, \pi_i\right). \tag{10}$$

Note that Q' can be expressed as the sum of at most m terms, where

$$m \leq 2^{n'} \cdot s^{O(\sqrt{s}\cdot\log(s)\cdot n')} \cdot 2^{O(\sqrt{s}\cdot\log(s)\cdot n'\cdot D)} \leq 2^{O(\sqrt{s}\cdot\log(s)\cdot(\log(s)+D)\cdot n')}.$$

Note that given Lemma 43, we can obtain Q' in time

$$2^{O(\sqrt{s}\cdot\log^2(s)\cdot D\cdot n')}. \tag{11}$$

Next, we construct a $2^{(n-n')/2} \times m$ matrix A and a $m \times 2^{(n-n')/2}$ matrix B as follows:

$$A_{x^{\text{L}},(z,S,\vec{\pi})} = \hat{p}(S) \cdot \prod_{i \in S} \alpha^i\left(z^{\text{L}}x^{\text{L}}, \pi_i\right),$$

and

$$B_{(z,S,\vec{\pi}),x^{\text{R}}} = \prod_{i \in S} \beta^i\left(z^{\text{R}}x^{\text{R}}, \pi_i\right).$$

It is easy to see that for each $x \in \{0,1\}^{n-n'}$,

$$Q'(x) = (A \cdot B)_{x^{\text{L}},x^{\text{R}}}.$$

We now want to compute $A \cdot B$. Therefore, we want $m \leq 2^{.172(n-n')/2}$ so that computing $A \cdot B$ can be done in time $\tilde{O}(2^{n-n'})$ using Theorem 45. For this we can set n' to be

$$n' = \frac{n}{c \cdot \sqrt{s} \cdot \log(s) \cdot (\log(s) + D)},$$

where $c > 0$ is some sufficiently large constant. Together with the running time in Equation (11), The total running time of the algorithm is therefore

$$2^{n - \frac{n}{\sqrt{s}\cdot\log(s)\cdot(\log(s)+D)}}.$$

For the randomized case, for each g^i ($i \in [s]$), we consider a randomized protocol Π_i that has error $\varepsilon' \leq 1/(3 \cdot s \cdot 2^{n'})$, and replace g^i with a randomly picked protocol from Π_i, so we can say that for every $x \in n - n'$, the algorithm computes $Q(x)$ (or $Q'(x)$) with probability at least $2/3$ (via a union bound over all the g^i's and a union bound over all the z's in $\{0,1\}^{n'}$). Then we can repeat the above algorithm $\mathsf{poly}(n)$ times and obtain $Q(x)$ for all $x \in \{0,1\}^{n-n'}$ correctly with high probability. Note that the error of any

randomized protocol with communication complexity R can be reduced to ε' by blowing up the communication complexity by a factor of $O(\log(1/\varepsilon'))$. In this case the, (as we are considering longer transcripts) the number of terms in Q' (as in Equation (10)) will be

$$2^{O(\sqrt{s}\cdot\log^2(s)\cdot R\cdot(n')^2)},$$

and we need to set accordingly

$$n' = \Omega\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot R}\right)^{1/2},$$

which gives the claimed running time for the randomized case. ◀

In fact, using the ideas above we can also get a randomized #SAT algorithm for the more expressive class $\mathsf{FORMULA}\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$, where $\mathsf{AC}^0_{d,M}$ is the class of depth-d size-M circuits and $\mathcal{G}$ is the class of functions that have low-communication complexity[9], by combining with the fact that AC^0 circuits have low-degree *probabilistic polynomials over the reals* (a probabilistic polynomial of a function f is a distribution on polynomials such that for every input x, a randomly picked polynomial from the distribution agrees with f on the input x). More specifically, we have the following.

▶ **Theorem 46.** *For any integers $s, d, M > 0$, there exists a randomized #SAT algorithm for $\mathsf{FORMULA}[s]\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$, where $\mathcal{G}$ is the class of functions with explicit two-party deterministic protocols of communication cost at most D, the algorithm outputs the number of satisfying assignments in time*

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot D}\right)^{1/2}}.$$

In the case $\mathcal{G}$ is the class of functions with explicit randomized protocols of communication cost at most R, there exists an analogous randomized algorithm with a running time

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R}\right)^{1/3}}.$$

Proof sketch. We show the case where $\mathcal{G}$ has low randomized communication complexity. Let

- $\varepsilon_1 = 1/\left(3\cdot 2^{n'}\right)$,
- $\varepsilon_2 = 1/\left(6\cdot s\cdot 2^{n'}\right)$ and
- $\varepsilon_3 = 1/\left(6\cdot M\cdot 2^{n'}\right)$.

As in the proof of Theorem 44, we can replace the formula part of $\mathsf{FORMULA}[s]\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$ with a ε_1-approximating polynomial of degree

$$O(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon_1)) = O(\sqrt{s}\cdot\log(s)\cdot n').$$

Then we replace the $\mathsf{AC}^0_{d,M}$ circuit with a randomly picked polynomial from a ε_2-error probabilistic polynomial. By [26], such a probabilistic polynomial is constructive and has degree at most

$$(\log M)^{O(d)}\cdot\log(1/\varepsilon_2) = (\log M)^{O(d)}\cdot(n'+\log(s)).$$

[9] Here we define the size of a $\mathsf{AC}^0_{d,M}$ circuit to be the number of wires. Note that a circuit in $\mathsf{FORMULA}\circ\mathsf{AC}^0_{d,M}\circ\mathcal{G}$ can have M functions from $\mathcal{G}$ at the bottom.

Finally, we replace each of the bottom functions, which is from $\mathcal{G}$, with a randomly picked protocol from a randomized protocol with error ε_3, and hence has cost at most

$$R \cdot O(\log(1/\varepsilon_3)) = O(R \cdot (n' + \log(M))).$$

As a result, we can express Q' as a polynomial with at most

$$2^{O\left(\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R\cdot(n')^3\right)}$$

monomials, whose variables are functions that depend on either the first half or the second half of x. Note that with our choices of ε_2 and ε_3, for every $x \in \{0,1\}^{n-n'}$, the algorithm computes $Q(x)$ correctly that with probability at least $2/3$ (by union bounds). By the same reasoning as in the proof of Theorem 44, we get a randomized #SAT algorithm with running time

$$2^{n-\left(\frac{n}{\sqrt{s}\cdot\log^2(s)\cdot(\log M)^{O(d)}\cdot R}\right)^{1/3}},$$

as desired. ◀

It is worth noting that unlike Theorem 44, the algorithm in Theorem 46 is *randomized* even if $\mathcal{G}$ is the class of functions with low *deterministic* communication complexity, because of the use of probabilistic polynomials for the AC^0 circuits.

6 Learning algorithms

In this section, we prove the following learning result for the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

▶ **Theorem 47.** *For every constant $\gamma > 0$, there is an algorithm that* PAC *learns the class of n-variate Boolean functions $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ to accuracy ε and with confidence δ in time $\mathsf{poly}\left(2^{n/\log n}, 1/\varepsilon, \log(1/\delta)\right)$.*

We first review some useful results that pertain to agnostically learning parities as well as boosting of learning algorithms.

6.1 Agnostically learning parities and boosting

For a parameter $n \geq 1$, let Δ be a distribution on labelled examples (x, y) supported over $\{0,1\}^n \times \{0,1\}$, and assume that for each x there is at most one y such that $(x, y) \in \mathsf{Support}(\Delta)$. For a function $h\colon \{0,1\}^n \to \{0,1\}$, we denote by $\operatorname{err}_\Delta(h)$ the error of h under this distribution:

$$\operatorname*{err}_{\Delta}(h) \;=\; \operatorname*{\mathbf{Pr}}_{(x,y)\sim\Delta}[h(x) \neq y]\,.$$

Similarly, for a class of functions $\mathcal{C}$, we let $\operatorname{opt}_\Delta(\mathcal{C})$ be the error of the best function in the class:

$$\operatorname*{opt}_{\Delta}(\mathcal{C}) \;=\; \min_{h\in\mathcal{C}} \operatorname*{err}_{\Delta}(h)\,.$$

We will need a result established by Kalai, Mansour, and Verbin [36], which gives a non-trivial time agnostic learning algorithm for the class of parities.

▶ **Lemma 48** ([36]). *Let* XOR *be the class of parity functions on n variables. Then, for any constant $\zeta > 0$, there is a randomized learning algorithm W such that, for every parameter $n \geq 1$ and distribution Δ over labelled examples, when W is given access to independent samples from Δ it outputs with high probability a circuit computing a hypothesis $h : \{0,1\}^n \to \{0,1\}$ such that*

$$\operatorname*{err}_{\Delta}(h) \leq \operatorname*{opt}_{\Delta}(\mathsf{XOR}) + 2^{-n^{1-\zeta}}.$$

The sample complexity and running time of W is $2^{O(n/\log n)}$.

Recall that a boosting procedure for learning algorithms transforms a weak learner that outputs a hypothesis that is just weakly correlated with the unknown function into a (strong) PAC learning algorithm for the same class (i.e., a learner in the sense of Definition 18). We refer for instance to [37] for more information about boosting in learning theory. We shall make use of the following boosting result by Freund [22].

▶ **Lemma 49** ([22]). *Let W be a (weak) learner for a class $\mathcal{C}$ that runs in time $t(n)$ and outputs (under any distribution) a hypothesis of error up to $1/2 - \beta$, for some constructive function $\beta(n) > 0$. Then, there exists a* PAC *learning algorithm for $\mathcal{C}$ that runs in time* $\mathsf{poly}(n, t, 1/\varepsilon, 1/\beta, \log(1/\delta))$.

6.2 PAC-learning small formulas of parities

We are ready to show that sub-quadratic size formulas over parity functions can be learned in time $2^{O(n/\log n)}$. First, we argue that Lemma 48 provides a weak learner that works under any distribution $\mathcal{D}$ supported over $\{0,1\}^n$. This will follow from Lemma 21, which shows that any function in $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ is correlated with some parity function with respect to $\mathcal{D}$. We then obtain a standard PAC learner via the boosting procedure from Lemma 49.

Proof of Theorem 47. Let $\mathcal{C} = \mathsf{FORMULA} \circ \mathsf{XOR}$, where $s = n^{2-\gamma}$ for some constant $\gamma > 0$. For any function $f \in \mathsf{FORMULA}[s] \circ \mathsf{XOR}$ and distribution $\mathcal{D}$ supported over $\{0,1\}^n$, Lemma 21 shows that there exists a parity function $\chi = \chi(f, \mathcal{D})$ such that

$$\Pr_{x \sim \mathcal{D}}[f(x) = \chi(x)] \geq \frac{1}{2} + \frac{1}{2^{n^{1-\lambda}}},$$

for some $\lambda = \lambda(\gamma) > 0$ independent of n, under the assumption that n is sufficiently large. Let $\Delta = \Delta(\mathcal{D}, f)$ be the distribution over labelled examples induced by $\mathcal{D}$ and f. Note that $\operatorname{opt}_{\Delta}(\mathsf{XOR}) \leq 1/2 - \exp(n^{1-\lambda})$. Consequently, by invoking Lemma 48 with parameter $\zeta = \lambda$, it follows that $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ can be learned under an arbitrary distribution to error $\beta(n) \leq 1/2 - \exp(n^{1-\Omega(1)})$ in time $t(n) = 2^{O(n/\log n)}$. Consequently, we can obtain a PAC learner algorithm for $\mathsf{FORMULA}[n^{2-\gamma}] \circ \mathsf{XOR}$ via Lemma 49 that runs in time $\mathsf{poly}(n, t(n), 1/\varepsilon, 1/\beta, \log(1/\delta)) = \mathsf{poly}(2^{n/\log n}, 1/\varepsilon, \log(1/\delta))$. ◀

References

1 Amir Abboud and Karl Bringmann. Tighter connections between formula-SAT and shaving logs. In *ICALP*, pages 8:1–8:18, 2018. doi:10.4230/LIPIcs.ICALP.2018.8.

2 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476, 2016. doi:10.1109/FOCS.2016.57.

3 Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992. doi:10.1002/rsa.3240030308.

4 Andris Ambainis, Andrew M. Childs, Ben Reichardt, Robert Spalek, and Shengyu Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM J. Comput.*, 39(6):2513–2530, 2010. doi:10.1137/080712167.

5 Alexander E Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42(1):63–66, 1987.

6 Roy Armoni, Michael E. Saks, Avi Wigderson, and Shiyu Zhou. Discrepancy sets and pseudorandom generators for combinatorial rectangles. In *FOCS*, pages 412–421, 1996. doi:10.1109/SFCS.1996.548500.

7 László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. doi:10.1016/0022-0000(92)90047-M.

8 Swapnam Bajpai, Vaibhav Krishan, Deepanshu Kush, Nutan Limaye, and Srikanth Srinivasan. A #sat algorithm for small constant-depth circuits with PTF gates. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.8.

9 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097.

10 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In *TCC*, pages 699–729, 2018. doi:10.1007/978-3-030-03810-6_25.

11 Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007. doi:10.1007/s00224-006-1313-z.

12 Lijie Chen, Ce Jin, and Ryan Williams. Hardness magnification for all sparse NP languages. In *FOCS*, 2019. ECCC:TR19-118.

13 Lijie Chen and Ruosong Wang. Classical algorithms from quantum and Arthur-Merlin communication protocols. In *ITCS*, pages 23:1–23:20, 2019. doi:10.4230/LIPIcs.ITCS.2019.23.

14 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015. doi:10.1007/s00037-015-0100-0.

15 Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. *Theory of Computing*, 14(1):1–55, 2018. doi:10.4086/toc.2018.v014a009.

16 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. In *ICALP*, pages 39:1–39:14, 2019. ECCC:TR19-022.

17 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.

18 Evgeny Dantsin and Edward A Hirsch. Worst-case upper bounds. *Handbook of Satisfiability*, 185:403–424, 2009.

19 Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.

20 Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum algorithm for the hamiltonian NAND tree. *Theory of Computing*, 4(1):169–190, 2008. doi:10.4086/toc.2008.v004a008.

21 Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013. doi:10.4086/toc.2013.v009a026.

22 Yoav Freund. Boosting a weak learning algorithm by majority. In *COLT*, pages 202–216, 1990. dl.acm.org/citation.cfm?id=92640.

23 Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018. doi:10.1007/s00037-018-0166-6.

24 Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *CCC*, pages 223–234, 2010. arXiv:1001.1593.

25 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219, 1996. doi:10.1145/237814.237866.

26 Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to AC. *Random Struct. Algorithms*, 54(2):289–303, 2019. doi:10.1002/rsa.20786.

27 Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

28 Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In *STOC*, pages 526–535, 2007. doi:10.1145/1250790.1250867.

29 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *FOCS*, pages 111–119, 2012. ECCC:TR12-057.

30 Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993. doi:10.1002/rsa.3240040202.

31 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *STOC*, pages 356–364, 1994. doi:10.1145/195058.195190.

32 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

33 Valentine Kabanets. Derandomization: A brief overview. *Current Trends in Theoretical Computer Science*, 1:165–188, 2002.

34 Valentine Kabanets and Zhenjian Lu. Satisfiability and derandomization for small polynomial threshold circuits. In *APPROX/RANDOM*, pages 46:1–46:19, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.46.

35 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. doi:10.1137/060649057.

36 Adam Tauman Kalai, Yishay Mansour, and Elad Verbin. On agnostic boosting and parity learning. In *STOC*, pages 629–638, 2008. doi:10.1145/1374376.1374466.

37 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. mitpress.mit.edu/books/introduction-computational-learning-theory.

38 Valeriy M Khrapchenko. Method of determining lower bounds for the complexity of π-schemes. *Mathematical Notes*, 10(1):474–479, 1971.

39 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

40 Sophie Laplante, Troy Lee, and Mario Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15(2):163–196, 2006. doi:10.1007/s00037-006-0212-7.

41 Saburo Muroga, Iwao Toda, and Satoru Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271:376–418, 1961. doi:10.1016/0016-0032(61)90702-5.

42 E.I. Nechiporuk. On a Boolean function. *Doklady Akademii Nauk SSSR*, 169(4):765–766, 1966. English translation in Soviet Mathematics Doklady.

43 Noam Nisan. The communication complexity of threshold gates. In *Proceedings of "Combinatorics, Paul Erdos is Eighty"*, pages 301–315, 1994.

44 Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. Fooling polytopes. In *STOC*, pages 614–625, 2019. arXiv:1808.04035.

45 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *CCC*, pages 27:1–27:29, 2019. doi:10.4230/LIPIcs.CCC.2019.27.

46 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *CCC*, pages 18:1–18:49, 2017. ECCC:TR16-197.

47 Mike Paterson and Uri Zwick. Shrinkage of de Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993. doi:10.1002/rsa.3240040203.

48 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, pages 1065–1075, 2010. doi:10.1137/1.9781611973075.86.

49 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Inf.*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

50 Ben Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *FOCS*, pages 544–551, 2009. doi:10.1109/FOCS.2009.55.

51 Ben Reichardt. Faster quantum algorithm for evaluating game trees. In *SODA*, pages 546–559, 2011. arXiv:0907.1623.

52 Ben Reichardt. Reflections for quantum query algorithms. In *SODA*, pages 560–569, 2011. arXiv:1005.1601.

53 Ben Reichardt and Robert Spalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(1):291–319, 2012. doi:10.4086/toc.2012.v008a013.

54 Igor S Sergeev. Upper bounds for the size and the depth of formulae for MOD-functions. *Discrete Mathematics and Applications*, 27(1):15–22, 2017.

55 Rocco A. Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *ITCS*, pages 30:1–30:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.30.

56 Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In *APPROX/RANDOM*, pages 45:1–45:23, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.45.

57 Bella Abramovna Subbotovskaya. Realization of linear functions by formulas using $\vee$, $\&$, $-$. In *Doklady Akademii Nauk*, volume 136, pages 553–555. Russian Academy of Sciences, 1961.

58 Avishay Tal. Shrinkage of de Morgan formulae by spectral techniques. In *FOCS*, pages 551–560, 2014. ECCC: TR14-048.

59 Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:114, 2015. ECCC:TR15-114.

60 Avishay Tal. The bipartite formula complexity of inner-product is quadratic. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:181, 2016. ECCC:TR16-181.

61 Avishay Tal. Formula lower bounds via the quantum method. In *STOC*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

62 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

63 Leslie G. Valiant. A theory of the learnable. In *STOC*, pages 436–445, 1984. doi:10.1145/800057.808710.

64 Emanuele Viola. The communication complexity of addition. *Combinatorica*, 35(6):703–747, 2015. doi:10.1007/s00493-014-3078-3.

65 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1109/10.1145/2559903.

66 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions. In *FOCS*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

A Proofs of useful lemmas

A.1 Useful lemmas for formulas

The proofs in this section are essentially the same as that of [60].

▶ **Lemma 50** ([60], Lemma 20 restated). *Let $\mathcal{D}$ be a distribution over $\{-1,1\}^n$, and let $f, C\colon \{-1,1\}^n \to \{-1,1\}$ be such that*

$$\Pr_{x\sim\mathcal{D}}[C(x) = f(x)] \geq 1/2 + \varepsilon.$$

Let $\tilde{C}\colon \{-1,1\}^n \to \mathbb{R}$ *be a* ε*-approximating function of* C*, i.e., for every* $x \in \{-1,1\}^n$*,* $|C(x) - \tilde{C}(x)| \leq \varepsilon$*. Then,*

$$\underset{x\sim\mathcal{D}}{\boldsymbol{E}}[\tilde{C}(x)\cdot f(x)] \geq \varepsilon.$$

Proof. Note that since $\tilde{C}$ ε-approximate C, we have for every $x \in \{-1,1\}^n$

$$\tilde{C}\cdot C(x) \geq 1-\varepsilon,$$

and

$$\tilde{C}\cdot(1-C(x)) \geq -1-\varepsilon.$$

Then,

$$\begin{aligned}
\underset{x\sim\mathcal{D}}{\mathbf{E}}[\tilde{C}(x)\cdot f(x)] &= \underset{x\sim\mathcal{D}}{\mathbf{E}}[\tilde{C}(x)\cdot f(x) \mid C(x)=f(x)]\cdot \underset{x\sim\mathcal{D}}{\mathbf{Pr}}[C(x)=f(x)] \\
&\quad + \underset{x\sim\mathcal{D}}{\mathbf{E}}[\tilde{C}(x)\cdot f(x) \mid C(x)\neq f(x)]\cdot \underset{x\sim\mathcal{D}}{\mathbf{Pr}}[C(x)\neq f(x)] \\
&\geq (1-\varepsilon)\cdot \underset{x\sim\mathcal{D}}{\mathbf{Pr}}[C(x)=f(x)] + (-1-\varepsilon)\cdot\left(1-\underset{x\sim\mathcal{D}}{\mathbf{Pr}}[C(x)=f(x)]\right) \\
&= 2\cdot \underset{x\sim\mathcal{D}}{\mathbf{Pr}}[C(x)=f(x)] - 1 - \varepsilon \\
&\geq 2\cdot(1/2+\varepsilon) - 1 - \varepsilon \\
&\geq \varepsilon,
\end{aligned}$$

as desired. ◀

▶ **Lemma 51** ([60], Lemma 21 restated)**.** *Let* $\mathcal{D}$ *be a distribution over* $\{-1,1\}^n$ *and let* $\mathcal{G}$ *be a class of functions. For* $f\colon \{-1,1\}^n \to \{-1,1\}$*, suppose that* $D\colon \{-1,1\}^n \to \{-1,1\} \in \mathit{FORMULA}[s]\circ\mathcal{G}$ *is such that*

$$\underset{x\sim\mathcal{D}}{\boldsymbol{Pr}}[D(x)=f(x)] \geq 1/2+\varepsilon_0.$$

Then there exists some $h\colon \{-1,1\}^n \to \{-1,1\} \in \mathit{XOR}_{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}\circ\mathcal{G}$ *such that*

$$\underset{x\sim\mathcal{D}}{\boldsymbol{E}}[h(x)\cdot f(x)] \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

Proof. Let

$$D = F(g_1, g_2 \ldots, g_s)$$

be a device in $\mathsf{FORMULA}\circ\mathcal{G}$ where F is a formula and $g_1, g_2, \ldots, g_s$ are function from $\mathcal{G}$.

Let $p\colon \{-1,1\}^s \to \mathbb{R}$ be a ε_0-approximating polynomial for F of degree $d = O(\sqrt{s}\cdot\log(1/\varepsilon_0))$. Note that we can write

$$p(z) = \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\prod_{i\in S} z_i.$$

Also, for each $S\subseteq[s]$, we have

$$|\hat{p}(S)| = \left|\underset{z\in\{-1,1\}^s}{\mathbf{E}}[p(z)\cdot\prod_{i\in S} z_i]\right| \leq 1+\varepsilon_0.$$

Now let

$$\tilde{D} := p(g_1, g_2 \ldots, g_s).$$

Note that $\tilde{D}$ is a ε_0-approximating function for D. Therefore, by Lemma 50, we have

$$\begin{aligned}
\varepsilon_0 &\leq \mathop{\mathbf{E}}_{x\sim\mathcal{D}}[D(x)\cdot f(x)] \\
&= \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\left(\sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\prod_{i\in S} g_i\right)\cdot f(x)\right] \\
&= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right] \\
&\leq \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} (1+\varepsilon_0)\cdot\left|\mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right]\right|.
\end{aligned}$$

The above equation is the sum of at most $s^{O(d)}$ summands. Therefore, there exists some $S\subseteq[s]$ such that

$$\left|\mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i\cdot f(x)\right]\right| \geq \frac{\varepsilon_0}{(1+\varepsilon_0)\cdot s^{O(d)}} \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}},$$

which implies that there exists some h, such that either $h=\prod_{i\in S} g_i$ or $h=-\prod_{i\in S} g_i$, and

$$\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[h(x)\cdot f(x)] \geq \frac{1}{s^{O\left(\sqrt{s}\cdot\log(1/\varepsilon_0)\right)}}.$$

Finally, note that such h can be expressed as the XOR of at most d functions from $\mathcal{G}$. ◀

A.2 PRG for low-communication functions in the number-in-hand setting

In this subsection, we show how to fool functions with low communication complexity in the number-in-hand model.

▶ **Theorem 52** ([6, 31], Theorem 28 restated). *For any $k\geq 2$, there exists a PRG that δ-fools any n-bits functions with k-party number-in-hand deterministic communication complexity at most D', with seed length*

$$n/k + O\left(D' + \log(1/\delta) + \log(k)\right)\cdot\log(k).$$

The PRG in Theorem 28 is based on the PRG by Impagliazzo, Nisan and Wigderson [31] that is used to derandomize "network algorithms" and space-bounded computation. We will need to use randomness extractors, which we review below.

▶ **Definition 53** (Min-entropy). *Let X be a random variable. The* min-entropy *of X, denoted by $H_\infty(X)$, is the largest real number k such that $\boldsymbol{Pr}[X=x]\leq 2^{-k}$ for every x in the range of X. If X is a distribution over $\{-1,1\}^{\aleph}$ with $H_\infty(X)\geq k$, then X is called a $(\aleph,k)$-*source.

▶ **Definition 54** (Extractors). *A function* $\mathrm{Ext}\colon \{-1,1\}^{\aleph} \times \{-1,1\}^{d} \to \{-1,1\}^{m}$ *is an* (k,ε)-extractor *if, for any* $(\aleph,k)$*-source* X*, and any test* $T\colon \{-1,1\}^{m} \to \{-1,1\}$*, it is the case that*

$$|\boldsymbol{Pr}[T(\mathrm{Ext}(X,U_d)\,X)=1] - \boldsymbol{Pr}[T(U_m)=1]| \leq \varepsilon.$$

▶ **Theorem 55** ([62, Theorem 6.22]). *For any integer* $m,\kappa > 0$ *and* $0 < \delta' < 0$*, there exists an explicit* (κ,δ') *extractor* $\mathrm{Ext}\colon \{0,1\}^{m} \times \{0,1\}^{d} \to \{0,1\}^{m}$ *with* $d = O(m-k+\log(1/\delta'))$.

We are now ready to show Theorem 52.

Proof of Theorem 52. We first describe the construction of the PRG. In fact, we will construct a sequence of PRGs $G_0, G_1, \ldots, G_{\log(k)}$. We begin by specifying the parameters of these PRGs. Let $t = \log(k)$, and let

$$d = O\left(D' + \log(1/\delta) + t\right).$$

For $i = 0, 1, \ldots, t$, let

- $r_0 = n/k$,
- $r_i = r_{i-1} + d$.

Note that we have $r_i = n/k + i \cdot d$. Also, let

$$\mathrm{Ext}_i\colon \{0,1\}^{r_i} \times \{0,1\}^{d} \to \{0,1\}^{r_i}$$

be a (κ_i,δ')-extractor from Theorem 55, where

$$\kappa_i = r_i - D' - 2t - \log(1/\delta)$$

and

$$\delta' = \delta / \left(3^t \cdot 2^{D'}\right).$$

Note that the seed length of the extractors is $d = O\left(D' + \log(1/\delta) + t\right)$. Finally, define $G_i\colon \{0,1\}^{r_i} \to \{0,1\}^{n/2^{t-i}}$ recursively as follows

- $G_0(a) = a$, where $a \in \{0,1\}^{n/k}$.
- $G_i(a,z) = G_{i-1}(a) \circ G_{i-1}(\mathrm{Ext}_{i-1}(a,z))$, where $a \in \{0,1\}^{r_{i-1}}$ and $z \in \{0,1\}^{d}$.

We will show that $G_t\colon \{0,1\}^{r_t = n/k + t\cdot d} \to \{0,1\}^{n}$ fools any functions f with k-party number-in-hand deterministic communication complexity at most D'. First, note that such f can be written as

$$f(x_1,x_2,\ldots,x_k) = \sum_{i=1}^{2^{D'}} h_1^{(i)}(x_1) \cdot h_2^{(i)}(x_2) \cdot \ldots \cdot h_k^{(i)}(x_k),$$

for some $h_j^{(i)}\colon \{0,1\}^{n/k} \to \{0,1\}$ $\left(i \in \left[2^{D'}\right], j \in [k]\right)$. Therefore, to show that the PRG G_t δ-fool f, it suffices to show that G_t $\left(\delta/2^{D'}\right)$-fools every function g of the form

$$g(x_1,x_2,\ldots,x_k) = h_1(x_1) \cdot h_2(x_2) \cdot \ldots \cdot h_k(x_k).$$

More specifically we show the following.

▷ Claim 56. For every $k \geq 2$ and $0 \leq i \leq t$, the generator G_i defined above $\left(3^i \cdot \delta'\right)$-fools every function $g_i \colon \{0,1\}^{n/2^{t-i}} \to \{0,1\}$ of the form

$$g_i(x_1, x_2, \ldots, x_{k/2^{t-i}}) = h_1(x_1) \cdot h_2(x_2) \cdot \ldots \cdot h_{k/2^{t-i}}(x_{k/2^{t-i}}),$$

where $x_1, x_2, \ldots, x_{k/2^{t-i}} \in \{0,1\}^{n/k}$.

Proof. The proof is by induction on i. The base case is $i = 0$, which is trivial given the definition of G_0. Now suppose the claim holds for $i-1$, we show the case for i. This is done using a hybrid argument. Consider the following four distributions

- $\mathcal{D}_1 = U_{n/2^{t-i}}$,
- $\mathcal{D}_2 = U_{n/2^{t-i+1}} \circ G_{i-1}(U_{r_{i-1}})$,
- $\mathcal{D}_3 = G_{i-1}(U_{r_{i-1}}) \circ G_{i-1}(U'_{r_{i-1}})$ (U and U' are two independent uniform distributions),
- $\mathcal{D}_4 = G_i(U_{r_i})$.

We want show show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq 3^i \cdot \delta'.$$

By the triangle inequality, it suffices to show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_2)]| + |\mathbf{E}[g_i(\mathcal{D}_2)] - \mathbf{E}[g_i(\mathcal{D}_3)]| + |\mathbf{E}[g_i(\mathcal{D}_3)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq 3^i \cdot \delta'. \quad (12)$$

We show Equation (12) by upper bounding each of the three summands.

First summand. We show that

$$|\mathbf{E}[g_i(\mathcal{D}_1)] - \mathbf{E}[g_i(\mathcal{D}_2)]| \leq 3^{i-1} \cdot \delta'. \quad (13)$$

Let us re-write g_i as

$$g_i(x_1, x_2, \ldots, x_{k/2^{t-i}}) = h^{\mathrm{L}}(x_1, x_2, \ldots, x_{k/2^{t-i+1}}) \cdot h^{\mathrm{R}}(x_{k/2^{t-i+1}+1}, x_{k/2^{t-i+1}+2}, \ldots, x_{k/2^{t-i}}),$$

where

$$h^{\mathrm{L}}(y) := \prod_{j=1}^{k/2^{t-i+1}} h_i(y) \quad \text{and} \quad h^{\mathrm{R}}(y) := \prod_{j=k/2^{t-i+1}}^{k/2^{t-1}} h_i(y).$$

Then,

$$\begin{aligned}
\mathbf{E}[g_i(\mathcal{D}_2)] &= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}}) \cdot h^{\mathrm{R}}(G_{i-1}(U_{r_{i-1}}))\right] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \mathbf{E}\left[h^{\mathrm{R}}(G_{i-1}(U_{r_{i-1}}))\right] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \left(\mathbf{E}\left[h^{\mathrm{R}}(U_{n/2^{t-i+1}})\right] \pm 3^{i-1} \cdot \delta'\right) \\
&\qquad \text{(by the induction hypothesis)} \\
&= \mathbf{E}\left[h^{\mathrm{L}}(U_{n/2^{t-i+1}})\right] \cdot \mathbf{E}\left[h^{\mathrm{R}}(U_{n/2^{t-i+1}})\right] \pm 3^{i-1} \cdot \delta' \\
&= \mathbf{E}[g_i(\mathcal{D}_1)] \pm 3^{i-1} \cdot \delta',
\end{aligned}$$

as desired.

Second summand. By a similar argument, it can be shown that

$$|\mathbf{E}[g_i(\mathcal{D}_2)] - \mathbf{E}[g_i(\mathcal{D}_3)]| \leq 3^{i-1} \cdot \delta'. \quad (14)$$

We omit the details here.

Third summand. We show that

$$|\mathbf{E}[g_i(\mathcal{D}_3)] - \mathbf{E}[g_i(\mathcal{D}_4)]| \leq \delta'. \tag{15}$$

We have

$$\begin{aligned}
\mathbf{E}[g_i(\mathcal{D}_4)] &= \mathbf{E}[g_i(G_i(U_{r_i}))] \\
&= \mathbf{E}\left[h^{\mathrm{L}}(G_{i-1}(X)) \cdot h^{\mathrm{R}}(G_{i-1}(\mathrm{Ext}_{i-1}(X,Z)))\right] \\
&\qquad\qquad \text{(where } X \sim \{0,1\}^{r_{i-1}} \text{ and } Z \sim \{0,1\}^{d}\text{)} \\
&= \mathbf{E}[A(X) \cdot B(\mathrm{Ext}_{i-1}(X,Z))] \\
&\qquad\qquad \text{(where } A(\cdot) = h^{\mathrm{L}}(G_{i-1}(\cdot)) \text{ and } B(\cdot) = h^{\mathrm{R}}(G_{i-1}(\cdot))\text{)} \\
&= \mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] \cdot \mathbf{Pr}[A(X)=1].
\end{aligned}$$

Similarly, we get

$$\mathbf{E}[g_i(\mathcal{D}_3)] = \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1] \cdot \mathbf{Pr}[A(X)=1].$$

As a result, we have

$$\begin{aligned}
&|\mathbf{E}[g_i(\mathcal{D}_4)] - \mathbf{E}[g_i(\mathcal{D}_3)]| \\
&= \left|\left(\mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] - \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1]\right) \cdot \mathbf{Pr}[A(X)=1]\right|. \qquad (16)
\end{aligned}$$

On the one hand, if $\mathbf{Pr}[A(X)=1] \leq \delta'$, then Equation (16) is at most δ'. On the other hand, if $\mathbf{Pr}[A(X)=1] > \delta'$, then

$$H_\infty(X \mid A(X)=1) > r_{i-1} - \log(1/\delta') > r_{i-1} - D' - 2t - \log(1/\delta) = \kappa_{i-1}.$$

Then by the fact that Ext_{i-1} is a (κ_{i-1}, δ')-extractor, we have

$$\left|\mathbf{E}[B(\mathrm{Ext}_{i-1}(X,Z)) \mid A(X)=1] - \mathbf{E}[B(U_{r_{i-1}}) \mid A(X)=1]\right| \leq \delta'.$$

Therefore, Equation (16) is at most δ' and this complete the proof of Equation (15). Finally, note that Equation (12) follows from Equation (13), Equation (14) and Equation (15). This completes the proof of Claim 56. ◁

Given Claim 56, Theorem 28 now follows by letting $i = t$. ◀

On the Quantum Complexity of Closest Pair and Related Problems

Scott Aaronson
The University of Texas at Austin, TX, USA
aaronson@cs.utexas.edu

Nai-Hui Chia
The University of Texas at Austin, TX, USA
nai@cs.utexas.edu

Han-Hsuan Lin
The University of Texas at Austin, TX, USA
linhh@cs.utexas.edu

Chunhao Wang
The University of Texas at Austin, TX, USA
chunhao@cs.utexas.edu

Ruizhe Zhang
The University of Texas at Austin, TX, USA
rzzhang@cs.utexas.edu

Abstract

The closest pair problem is a fundamental problem of computational geometry: given a set of n points in a d-dimensional space, find a pair with the smallest distance. A classical algorithm taught in introductory courses solves this problem in $O(n \log n)$ time in constant dimensions (i.e., when $d = O(1)$). This paper asks and answers the question of the problem's quantum time complexity. Specifically, we give an $\widetilde{O}(n^{2/3})$ algorithm in constant dimensions, which is optimal up to a polylogarithmic factor by the lower bound on the quantum query complexity of element distinctness. The key to our algorithm is an efficient history-independent data structure that supports quantum interference.

In polylog(n) dimensions, no known quantum algorithms perform better than brute force search, with a quadratic speedup provided by Grover's algorithm. To give evidence that the quadratic speedup is nearly optimal, we initiate the study of quantum fine-grained complexity and introduce the *Quantum Strong Exponential Time Hypothesis (QSETH)*, which is based on the assumption that Grover's algorithm is optimal for CNF-SAT when the clause width is large. We show that the naïve Grover approach to closest pair in higher dimensions is optimal up to an $n^{o(1)}$ factor unless QSETH is false. We also study the bichromatic closest pair problem and the orthogonal vectors problem, with broadly similar results.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Design and analysis of algorithms; Theory of computation → Quantum complexity theory

Keywords and phrases Closest pair, Quantum computing, Quantum fine grained reduction, Quantum strong exponential time hypothesis, Fine grained complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.16

Funding SA was supported by a Vannevar Bush Fellowship from the US Department of Defense, a Simons Investigator Award, and the Simons "It from Qubit" collaboration. NHC, HHL, and CW were supported by SA's Vannevar Bush Faculty Fellowship. RZ received support from the National Science Foundation (grant CCF-1648712).

Acknowledgements We would like to thank Lijie Chen and Pasin Manurangsi for helpful discussion. We would like to thank anonymous reviewers for their valuable suggestions on this paper.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 16; pp. 16:1–16:43

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

1 Introduction

In the closest pair problem (CP), we are given a list of points in $\mathbb{R}^d$, and asked to find two that are closest. (See Figure 1 for an illustration of this problem.) This is a fundamental problem in computational geometry and has been extensively studied. Indeed, CP is one of the standard examples in textbooks (such as [20] and [32]) to introduce the divide-and-conquer technique. Moreover, CP relates to problems that have critical applications in spatial data analysis and machine learning, such as empirical risk minimization [7], point location [44, 13], time series motif mining [35], spatial matching problems [51], and clustering [36]. Therefore, any improvement on CP may imply new efficient algorithms for related applications.

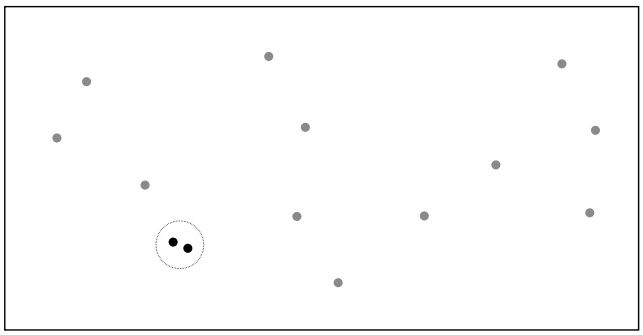


Figure 1 An instance of the CP, where the the closest pair is labeled in the circle.

Like with many other geometric problems, the hardness of CP rises as the dimension d increases. Shamos and Hoey gave the first $O(n \log n)$ deterministic algorithm in $\mathbb{R}^2$ by using Voronoi diagrams [44], improving on the trivial $O(n^2 d)$ upper bound. Then, Bentley and Shamos gave an algorithm with $2^{O(d)} n \log n$ running time via a divide-and-conquer approach [10]. A randomized algorithm by Khuller and Matias [30, 40] takes $2^{O(d)} n$ expected running time. A trivial lower bound for CP is $\Omega(n)$, since one must read all points to find the closest pair in the worst case. Yao showed an $\Omega(n \log n)$ lower bound for CP on the algebraic decision tree model [52].

When we consider CP in polylog(n) dimensions, the running time of all existing algorithms blows up to $\Omega(n^2)$, and thus it is unknown if there exists an algorithm matching the unconditional lower bounds. Nevertheless, under the *Strong Exponential Time Hypothesis (SETH)*, Karthik and Manurangsi [29], and David et al. [22], recently proved a conditional lower bound of $n^{2-o(1)}$ for CP in polylog(n) dimensions. This implies that the brute force approach is nearly optimal in polylog(n) dimensions unless SETH is false. SETH was introduced by Impagliazzo and Paturi [26], and is the assumption that for all $\epsilon > 0$, there exists an integer $k > 2$ such that no algorithm can solve k-SAT in time $O(2^{(1-\epsilon)n})$.

The main idea behind the results of [29, 22] is to prove a "fine-grained" reduction from CNF-SAT to CP in polylog(n) dimensions. Fine-grained reductions are reductions between computational problems that keep track of the exact polynomial exponents. For instance, [29] showed that CNF-SAT with $2^{n(1-o(1))}$ time is reducible to CP in polylog n dimensions with $n^{2-o(1)}$ time, and thus the lower bound for CP in polylog n dimensions is $n^{2-o(1)}$ unless SETH is false.

Surprisingly, to our knowledge, the quantum time complexity of CP was hardly investigated before. The trivial quantum algorithm for CP is to use Grover's search algorithm on all n^2 pairs, which takes $O(nd)$ time. Sadakane et al. [41] sketched a quantum algorithm that runs in $O(n^{1-1/(4\lceil d/2 \rceil)})$ time. Volpato and Moura [47] claimed a quantum algorithm that

uses $O(n^{2/3})$ *queries*, but no analysis was given of the *running time*, and as we will see, the conversion from the query-efficient algorithm to a time-efficient algorithm is nontrivial. As for the lower bound, any quantum algorithm for CP needs $\Omega(n^{2/3})$ time, since Aaronson and Shi [1] proved such a lower bound for element distinctness, and CP contains element distinctness as a special case, where a closest pair has distance 0.

In this work, we resolve the quantum time complexity of CP. In constant dimensions, we observe that by using a quantum walk for element distinctness [5, 33], we can achieve $O(n^{2/3})$ queries for CP. However, to obtain the same time complexity, the algorithm needs some geometric data structure that supports fast updates and checking, and that – crucially – is "history-independent", i.e., the data structure is uniquely represented, disregarding the order of insertion and deletion. History-independence is essential since different representations of the same data would destroy quantum interference between basis states.

We propose a geometric data structure that is history-independent and that supports fast checking and updates. Our data structure works by discretizing $\mathbb{R}^d$ into hypercubes with length $\epsilon/\sqrt{d}$. Then, we use a hash table, skip lists, and a radix tree to maintain the locations of the points and hypercubes. This data structure is history-independent, and we can easily find pairs with distance at most ϵ with it. We then find the closest pair by a binary search. By using our data structure and a quantum walk [5, 33], we achieve quantum time complexity $\widetilde{O}(n^{2/3})$.

For CP in polylog(n) dimensions, one may expect a conditional lower bound under SETH. However, SETH fails when quantum algorithms are considered since a simple application of Grover's search algorithm on all assignments solves CNF-SAT in time $\widetilde{O}(2^{n/2})$. Furthermore, existing fine-grained reductions may require time greater than $O(2^{n/2})$.

In this paper, we introduce the *Quantum Strong Exponential Time Hypothesis (QSETH)* and *quantum fine-grained reductions*. We define QSETH as follows.

▶ **Definition 1** (QSETH). *For all $\epsilon > 0$, there exists some $k \in \mathbb{N}$ such that there is no quantum algorithm solving k-SAT in time $O(2^{(1-\epsilon)\frac{n}{2}})$.*

We then observe that the classical definition of fine-grained reductions cannot capture the features of quantum reductions such as superposed queries and speedups from quantum algorithms. For instance, a fine-grained reduction may reduce problem A to solving many instances of problem B and then output the best solution; in this case, one can use Grover's search algorithm to achieve a quadratic speedup. Therefore, instead of summing the running time over all instances as in Definition 16, we use a quantum algorithm which solves all instances in superposition and outputs the answer. We give a formal definition of quantum fine-grained reductions in Definition 25 and show that under QSETH, any quantum algorithm for CP in polylog(n) dimensions requires $n^{1-o(1)}$ time. This implies that Grover's algorithm is optimal for the problem up to an $n^{o(1)}$ factor.

Intuitively, QSETH is the conjecture that applying Grover's search algorithm over all assignments in superposition is the optimal quantum algorithm for CNF-SAT. This is similar to SETH, which says that a brute force search is optimal for CNF-SAT. A series of works on CNF-SAT [43, 39, 38, 25, 42] shows that for some constant $c \in [1, 2]$, there exist (randomized) algorithms for n-variable k-SAT that run in time $2^{n(1-c/k)}$. As k grows, the running time of these algorithms approach 2^n. When k is small, however, there are algorithms with better running times. For instance, when $k = 3$, Schöning [43] obtained an algorithm with $O(1.334^n)$ running time, which was later improved to $O(1.308^n)$ by Paturi et al. [39]. However, none of the above mentioned algorithms have good running time on larger k's, so SETH remains a plausible conjecture.

When k is small enough, there are also quantum algorithms for k-SAT [4, 21] running in time much less than $O(2^{n/2})$. However, these quantum algorithms mainly use Grover search to speed up the classical algorithms of [43, 39], and thus do not perform well for large k, either. Therefore, we conjecture that for large enough k, no quantum algorithm can do much better than Grover search.

Finally, we study the bichromatic closest pair problem (BCP) and the orthogonal vector problem (OV). Briefly, OV is to find a pair of vectors that are orthogonal given a set of vectors in $\{0,1\}^d \in \mathbb{R}^d$, and BCP is, given two sets A, B (representing two colors) of n points in $\mathbb{R}^d$, to find the pair (a,b) of minimum distance with $a \in A$ and $b \in B$.

We can summarize all of our results as follows.

▶ **Theorem 2** (Informal). *Assuming QSETH, there is no quantum algorithm running in time* $n^{1-o(1)}$ *for* OV, CP, *and* BCP *when* $d = \mathrm{polylog}(n)$.

▶ **Theorem 3** (Informal). *The quantum time complexity of* CP *in* $O(1)$ *dimensions*[I] *is* $\widetilde{\Theta}(n^{2/3})$[II].

▶ **Theorem 4** (Informal). *For any* $\delta > 0$, *there exists a quantum algorithm for* BCP *with* $\widetilde{O}(n^{1-\frac{1}{2d}+\delta})$ *running time. There exists a quantum algorithm which solves* $(1+\xi)$*-approximate* BCP *in time* $\widetilde{O}(\xi^{-d}n^{2/3})$.

▶ **Theorem 5** (Informal). *The quantum time complexity of* OV *in* $O(1)$ *dimensions*[III] *is* $\Theta(n^{1/2})$.

Table 1 also summarizes what is known about upper and lower bounds on the classical and quantum time complexities of all of these problems.

Table 1 A summary of our quantum complexity results and comparison to classical results. The bold entries highlight our contributions in this paper.

	Dimension		Lower Bound	Upper Bound
CP	$\Theta(1)$	Classical	$\widetilde{\Omega}(n)$ [52]	$\widetilde{O}(n)$ [44, 10, 30]
		Quantum	$\mathbf{\Omega(n^{2/3})}$ Theorem 56	$\mathbf{\widetilde{O}(n^{2/3})}$ Corollary 55
	$\mathrm{polylog}\, n$	Classical	$n^{2-o(1)}$ (Under SETH) [29]	$O(n^2)$
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 26	$\mathbf{\widetilde{O}(n)}$ Theorem 15
OV	$\Theta(1)$	Classical	$\Omega(n)$	$O(n)$ [48]
		Quantum	$\mathbf{\Omega(n^{1/2})}$ Theorem 68	$\mathbf{O(n^{1/2})}$ Theorem 68
	$\mathrm{polylog}\, n$	Classical	$n^{2-o(1)}$ (Under SETH) [49]	$n^{2-o(1)}$ [2, 16]
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 26	$\mathbf{\widetilde{O}(n)}$ Theorem 15
BCP	$\Theta(1)$	Classical	$\Omega(n)$	$O\left(n^{2-\frac{2}{\lceil d/2\rceil+1}+\delta}\right)$ [3]
		Quantum	$\mathbf{\Omega(n^{2/3})}$ Theorem 67	$\mathbf{\widetilde{O}(n^{1-\frac{1}{2d}+\delta})}$ for BCP Theorem 66 $\mathbf{\widetilde{O}(\xi^{-d}n^{2/3})}$ for $(1+\xi)$-BCP Theorem 64
	$2^{O(\log^*(n))}$[IV]	Classical	$n^{2-o(1)}$ (Under SETH) [17]	$n^{2-o(1)}$ [2, 16]
		Quantum	$\mathbf{n^{1-o(1)}}$ **(Under QSETH)** Theorem 35	$\mathbf{\widetilde{O}(n)}$ Theorem 15

[I] We actually give a slightly stronger result: the same time complexities still hold when $d = O\left(\frac{\log\log n}{\log\log\log n}\right)$.
[II] The $\widetilde{\Theta}$ notation is Θ with logarithmic factors hidden in both upper and lower bounds.
[III] The same time complexities still hold when $d = O(\log\log n)$.
[IV] $\log^*(n) := \log^*(\log n) + 1$ for $n > 1$ and $\log^*(1) := 0$. Hence, $2^{O(\log^* n)}$ is an extremely slow-growing function.

Related work

A recent independent work by Buhrman, Patro and Speelman [15] also studied quantum strong exponential time hypothesis. They defined (a variant of) QSETH based on the hardness of testing properties on the set of satisfying assignments of a SAT formula, e.g., the parity of the satisfying assignments. Based on these hardness assumptions extended from the original QSETH, they gave conditional quantum lower bounds for OV, the Proofs of Useful Work [8] and the edit distance problem. In comparison, we formally define the *quantum fine-grained reductions* and prove lower bounds for CP, OV, and BCP under the original form of QSETH by showing the existence of quantum fine-grained reductions from CNF-SAT to the these problems.

1.1 Proof overview

For ease of presentation, some notations and descriptions will be informal here. Formal definitions and proofs will be given in subsequent sections.

We give an optimal (up to a polylogarithmic factor) quantum algorithm that solves CP for constant dimensions in time $\widetilde{O}(n^{2/3})$. First note that there exists a Johnson graph corresponding to an instance of CP, where each vertex corresponds to a subset of $n^{2/3}$ points of the input of CP, and two vertices are connected when the intersection of the two subsets (they are corresponding to) has size $n^{2/3}-1$. A vertex is marked if the subset it corresponds to contains a pair with distance at most ϵ. Then, the goal is to find a marked vertex on this Johnson graph and use binary search over ϵ to find the closest pair. Our algorithm for finding a marked vertex is based on the quantum walk search framework by Magniez et al. [33], which can be viewed as the quantum version of the Markov chain search on a graph (in our case, a Johnson graph). The complexity of this quantum walk algorithm is $O(\mathsf{S}+\frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U}+\mathsf{C}))$, where λ is the fraction of marked states in the Johnson graph, δ is its spectral gap, S is the cost for preparing the algorithm's initial state, U is the cost for implementing one step of the quantum walk, and C is the cost for checking the solution. For our Johnson graph, $\lambda = n^{-2/3}$ and $\delta = n^{-2/3}$. If we consider only the query complexity, $\mathsf{S} = n^{2/3}$, $\mathsf{U} = O(1)$, and $\mathsf{C} = 0$. However, the time complexity for C is huge in the straightforward implementation, e.g., storing all points in an array according to the index order, as we need to check all the pairs from the $n^{2/3}$ points, which will kill the quantum speedup. To tackle this, we discretize the space into small hypercubes. With this discretization, it suffices to check $O((\sqrt{d})^d)$ neighbor hypercubes to find a pair with distance at most ϵ. To support the efficient neighborhood search, we need an efficient data structure.

Existing data structures do not meet our need. They either have prohibitive dependence on the dimension, such as $\Omega(n^{\lceil d/2 \rceil})$ time for constructing and storing Voronoi diagrams [31], or do not have unique representation (i.e., they are history-dependent), such as fair-split trees and dynamic trees [13]. Note that the requirement of unique representation is due to the fact that different representations of the same data would destroy the interference that quantum computation relies on. To solve this problem, we propose a uniquely represented data structure that can answer queries about ϵ-close pairs and insert/delete points efficiently. This data structure is based on a hash table, skip lists, and a radix tree. With this data structure, $\mathsf{U} = O(\log n)$ and $\mathsf{C} = O(1)$. Hence, we have the desired time complexity (see Section 4.2). We give another method for solving CP that only uses a radix tree as the data structure. With only a radix tree, the algorithm cannot handle cases with multiple solutions, and we need to subsequently reduce the size of the problem until there is at most one solution (see Section 4.3). These two quantum algorithms have the same time complexity.

Our quantum algorithm for solving approximate BCP follows the same spirit as that for CP, except that we use a finer discritization of the space (see Section 5.1). To solve BCP exactly, we need a history-independent data structure for nearest-neighbor search, but no such data structure is known. Instead, we adapt the nearest-neighbor search data structure by Clarkson [19] to the quantum algorithm proposed by Buhrman et al. [14] for element distinctness, which does not require history-independence of the data structure because in the algorithm of [14], no insertions and deletions are performed once the data structure for a set of points is constructed (see Section 5.2). Sadakane et al. [41] sketched an algorithm for BCP with similar ideas and running time, but we give the first rigorous analysis.

To derive our quantum fine-grained complexity results for OV and CP when $d = \operatorname{polylog} n$ under QSETH, we first define quantum fine-grained reductions. In our definition, we consider problems whose input is given in the quantum query model, and allow the reduction to perform superposed queries and run quantum algorithms, e.g., amplitude amplification. The classical reductions from CNF-SAT to CP [29, 22] and OV [50] are not "quantum fine-grained" under QSETH. These reductions fail because their running time exceeds $2^{n/2(1-\epsilon)}$, which is the conjectured time complexity for CNF-SAT under QSETH. Therefore, we cannot derive from them any non-trivial lower bounds for CP or OV based on QSETH. In the following, we use the advantages of quantum algorithms to make these reductions work.

There are two main obstacles in "quantizing" the fine-grained reductions under QSETH. The first obstacle is that the time cost for preparing the input of the problem we reduce to is already beyond the required running time. For instance, consider the reduction from CNF-SAT to OV. Let φ be a CNF-SAT instance on n variables and m clauses. The classical fine-grained reduction divides all n variables into two sets A and B of size $n/2$, and then maps all assignments for variables in A and B to two sets V_A and V_B of $2^{n/2}$ vectors each. It is obvious that the time for writing down V_A and V_B is already $\Theta(2^{n/2})$. Nevertheless, many quantum algorithms achieve sublinear query complexities by querying the input oracle in superposition. Hence, instead of first constructing the input of OV at once and then running the algorithm, we can simulate it "on-the-fly": whenever the OV's algorithm queries the input oracle with some superposition of indices, we use a quantum subroutine to realize the input oracle by mapping the query indices to the corresponding assignments in CNF-SAT, and then to the corresponding vectors in V_A and V_B. This subroutine takes only $O(n)$ time, and therefore the quantum reduction, which has running time $O(n)$ times the running time of the OV algorithm, is quantum fine-grained.

Another difficulty in quantizing the fine-grained reductions is that some reduction needs to call the oracle multiple times, and the number of calls exceeds the required running time. However, it is possible to achieve quadratic speedup if these oracle calls are non-adaptive. For the reduction from BCP to CP, we can reduce a BCP instance to $n^{1.8+o(1)} \log n$ instances of CP, which is already larger than the conjectured $\Omega(n)$ quantum lower bound of BCP. By further studying the reduction, we find that the solution to BCP is the minimum of the solutions to the the constructed CP instances. Therefore, we can use the quantum minimum-finding algorithm to reduce the total time complexity to $\widetilde{O}(\sqrt{n^{1.8+\epsilon}} \cdot t_{\mathsf{CP}})$, which is enough to show that BCP is quantum fine-grained reducible to CP.

With the above-mentioned techniques, we quantize the classical fine-grained reductions, and show that CNF-SAT, with conjectured lower bound $\Omega(2^{n/2})$, is quantum fine-grained reducible to OV and CP with lower bound $\Omega(n')$[V], when the dimension d is $\operatorname{polylog}(n')$.

[V] n is the input size of CNF-SAT, and n' is the input size of OV and CP.

2 Preliminaries

▶ **Definition 6** (Distance measure). *For any two vectors $a, b \in \mathbb{R}^d$, the distance between them in the ℓ_2-metric is denoted by $\|a-b\| = \left(\sum_{i=1}^{d} |a_i - b_i|^2\right)^{1/2}$. Their distance in the ℓ_0-metric (Hamming distance) is denoted by $\|a-b\|_0 = |\{i \in [d] : a_i \neq b_i\}|$, i.e., the number of coordinates on which a and b differ.*

2.1 Quantum query model

We consider the quantum query model in this work. Let $X := \{x_1, \ldots, x_n\}$ be a set of n input points and $\mathcal{O}_X$ be the corresponding oracle. We can access the i-th data point x_i by making the query

$$|i\rangle |0\rangle \xrightarrow{\mathcal{O}_X} |i\rangle |x_i\rangle , \tag{1}$$

and we can make queries to elements in X in superposition. Note that $\mathcal{O}_X$ is an unitary transformation in the formula above. Hence, a quantum algorithm with access to $\mathcal{O}_X$ can be represented as a sequence of unitary transformations.

Consider a quantum algorithm $\mathcal{A}$ with access to an oracle $\mathcal{O}$ and a initial state $|0\rangle := |0\rangle_Q |0\rangle_A |0\rangle_W$, where the registers Q and A are for the queries and the answers from the oracle, and the register W is the working space which is always hold by $\mathcal{A}$. Then, we can represent the algorithm as

$$U_T \mathcal{O} U_{T-1} \cdots \mathcal{O} U_1 |0\rangle . \tag{2}$$

Let $|\psi\rangle_i = U_i \mathcal{O} \cdots \mathcal{O} U_1 |0\rangle := \sum_{i,z} |i\rangle_Q |0\rangle_A |z\rangle_W$ be the state right before applying the i-th $\mathcal{O}$, then

$$\mathcal{O} |\psi\rangle_i := \sum_{i,z} |i\rangle_Q |x_i\rangle_A |z\rangle_W . \tag{3}$$

2.2 Quantum subroutine for unstructured searching and minimum finding

▶ **Definition 7** (Unstructured search). *Given a set P of n elements in $\{0,1\}$, decide whether there exists a 1 in P.*

▶ **Theorem 8** (Grover's search algorithm [24, 37]). *There is a quantum algorithm for unstructured search with running time $O(\sqrt{n})$.*

By Theorem 8 and BBBV's argument [9], the quantum time complexity of unstructured search is $\Theta(\sqrt{n})$. We can also get a $\widetilde{O}(\sqrt{n})$ quantum algorithm for minimum finding by combining Grover's search algorithm and binary search.

▶ **Theorem 9** (Quantum minimum finding [23]). *There is a quantum algorithm that finds from a set of n elements with values in $\mathbb{R}$, the index of the minimum element of the set, with success probability $\frac{1}{2}$ and run time $\widetilde{O}(\sqrt{n})$*

2.3 Problem definitions

In this subsection, we first formally define OV, CP, and BCP. Then we show the folklore algorithms for CP, BCP, and OV by Grover's algorithm, which run in time $\widetilde{O}(n)$.

▶ **Definition 10** (Orthogonal Vectors, OV). *Given two sets A, B of n vectors in $\{0,1\}^d$ as input, find a pair of vectors $a \in A$, $b \in B$ such that $\langle a, b\rangle = 0$, where the inner product is taken in $\mathbb{Z}$.*[VI]

We denote OV with input length n and dimension d as $\mathsf{OV}_{n,d}$. We will use this notation when we need to specify the parameters in the following sections.

▶ **Definition 11** (Closest Pair Problem, CP). *Given a set P of n points in $\mathbb{R}^d$ and a distance measure Δ, find a pair of distinct points $a, b \in P$ such that $\Delta(a, b)$ is the smallest among all distinct pairs in P.*

Similar to OV, we denote CP with input length n and dimension d as $\mathsf{CP}_{n,d}$. We will use this notation when the parameters in the following sections are required to be specified. Note that in this work, we consider $\Delta(a, b) = \|a - b\|$ as the distance measure for CP and BCP.

▶ **Definition 12** (Bichromatic Closest Pair Problem, BCP). *Given two sets A, B of n points in $\mathbb{R}^d$ and a distance measure Δ, find a pair of points $a \in A$, $b \in B$ such that*

$$\Delta(a, b) = \min_{a \in A, b \in B} \Delta(a, b). \tag{4}$$

We also define an approximate version of BCP as follows.

▶ **Definition 13** ($(1+\xi)$-approximate Bichromatic Closest Pair Problem, $(1+\xi)$-BCP). *Given two sets A, B of n points $\in \mathbb{R}^d$ and a distance measure Δ, find a pair of points $a \in A$, $b \in B$ such that*

$$\Delta(a, b) \leq (1 + \xi) \min_{a \in A, b \in B} \Delta(a, b). \tag{5}$$

Same as CP, we use $\mathsf{BCP}_{n,d}$ and $(1+\xi)$-$\mathsf{BCP}_{n,d}$ to specify the parameters.

▶ **Definition 14** (Element Distinctness Problem, ED). *Let $f : [n] \to [m]$ be a given function. Decide whether there exist distinct $i, j \in [n]$ such that $f(i) = f(j)$.*

For this problem, Ambainis [5] gave a quantum algorithm with time complexity $\widetilde{O}(n^{2/3})$, which matches the lower bound proved by Aaronson and Shi [1] up to a polylogarithmic factor.

▶ **Theorem 15.** *There are $\widetilde{O}(n)$-time quantum algorithms for* CP *and* BCP *when $d = O(\mathrm{poly}\log n)$.*

Proof. We can solve CP and BCP by searching the minimum distance through all pairs by the algorithm of Theorem 9. There are $O(n^2)$ pairs and checking each pair took $O(d)$ time, so the total running time is $O(nd)$. For $d = O(\mathrm{poly}\log n)$, the time complexity equals to $\widetilde{O}(n)$. ◀

2.4 Fine-grained complexity

As we have mentioned earlier in the introduction, a fine-grained reduction from problem P to Q with conjectured lower bounds $p(n)$ and $q(n)$, respectively, has the property that if we can improve the $q(n)$ time for Q, then we can also improve the $p(n)$ time for P. We give the formal definition by Williams [46] in below.

[VI] Our definition is slightly different than some of the literature, for example, [18], which is searching among pairs inside one set. Those two definitions are equivalent up to constant in complexities.

▶ **Definition 16** (Fine-grained reduction, [46]). *Let $p(n)$ and $q(n)$ be non-decreasing functions of n. Problem P is (p,q)-reducible to problem Q, denoted as $(\mathsf{P},p) \leq_{\mathrm{FG}} (\mathsf{Q},q)$, if for every ϵ, there exist $\delta > 0$, an algorithm R for solving P with access to an oracle for Q, a constant d, and an integer $k(n)$, such that for every $n \geq 1$, the algorithm R takes any instance of P of size n and*

- *R runs in at most $d \cdot (p(n))^{1-\delta}$-time,*
- *R produces at most $k(n)$ instances of Q adaptively, that is, the jth instance X_j is a function of $\{(X_i, y_i)\}_{1 \leq i < j}$ where X_i is the ith instance produced and y_i is the answer of the oracle for Q on instance X_i, and*
- *the sizes n_i of the instances X_i for any choice of oracle answers y_i obeys the inequality*

$$\sum_{i=1}^{k(n)} (q(n_i))^{1-\epsilon} \leq d \cdot (p(n))^{1-\delta}. \tag{6}$$

Let $(\mathsf{P},p) \leq_{\mathrm{FG}} (\mathsf{Q},q)$ for some non-decreasing function $p(n)$ and $q(n)$. If for every $\epsilon > 0$, we can solve problem Q in time $q(n)^{1-\epsilon}$ with probability 1 for all input length n, then there exists a $\delta > 0$ such that we can solve the problem P in time $p(n)^{1-\delta}$ by Equation (6).

Here are some known results about fine-grained reductions.

▶ **Theorem 17** ([29, 49]).

$$(\mathsf{CNF\text{-}SAT}_n, 2^n) \leq_{\mathrm{FG}} (\mathsf{OV}_{n_1,d_1}, n_1^2) \leq_{\mathrm{FG}} (\mathsf{BCP}_{n_2,d_2}, n_2^2) \leq_{\mathrm{FG}} (\mathsf{CP}_{n_3,d_3}, n_3^2), \tag{7}$$

where $d_1 = \Theta(\log n_1)$, $d_2 = \Theta(\log n_2)$ and $d_3 = (\log n_3)^{\Omega(1)}$.

▶ Remark 18. The second reduction from OV to BCP has been improved to $d_2 = 2^{O(\log^* n)}$ by Chen [17].

There are several plausible hypotheses in fine-grained complexity, which can imply conditional hardness results for many interesting problems. We first give the definition of the strong exponential time hypothesis (SETH).

▶ **Hypothesis 19** (Strong Exponential Time Hypothesis, SETH). *For every $\epsilon > 0$, there exists a $k = k(\epsilon) \in \mathbb{N}$ such that no algorithm can solve k-SAT (i.e., satisfiability on a CNF of width k) in $O(2^{(1-\epsilon)m})$ time where m is the number of variables. Moreover, this holds even when the number of clauses is at most $c(\epsilon) \cdot m$ where $c(\epsilon)$ denotes a constant that depends only on ϵ.*

Another popular conjecture is the orthogonal vector hypothesis (OVH):

▶ **Definition 20** (Orthogonal Vector Hypothesis, OVH). *For every $\epsilon > 0$, there exists a $c \geq 1$ such that $\mathsf{OV}_{n,d}$ requires $n^{2-\epsilon}$ time when $d = c \log n$.*

▶ Remark 21. Under SETH, we can have the following conclusions from Theorem 17:

- OVH is true.
- For all $\epsilon > 0$, there exists a $c > 0$ such that $\mathsf{BCP}_{n,c\log n}$ cannot be solved by any randomized algorithm in time $O(n^{2-\epsilon})$.
- For all $\epsilon > 0$, there exists a $c > 0$ such that $\mathsf{CP}_{n,(\log n)^c}$ cannot be solved by any randomized algorithm in time $O(n^{2-\epsilon})$.

2.5 The framework for quantum walk search

In this subsection, we review the quantum walk framework for the Markov chain search problem and demonstrate how to use it to solve the element distinctness problem. For simplicity, we use the transition matrix P to refer to a Markov chain, where $P = (p_{xy})_{x,y\in X}$ for X being the state space of P and p_{xy} being the transition probability from x to y. An irreducible and ergodic Markov chain has a unique stationary distribution π, which is also the unique eigenvector of P with eigenvalue 1. Let $M \subseteq X$ be a set of marked elements. In the Markov chain search problem, the objective is to find an $x \in M$. We can perform the following actions: setup, sampling from the π with cost S; update, making a transition with cost U, and checking whether the current state is marked or not with cost C. To solve the search problem classically, we perform a random walk as follows. Start from a point sampled from π and check if it is marked. If not, make a number of transitions on P until it mixes, and then check again. We then repeat this process until a marked state is found. The cost of this random walk algorithm is $O(\mathsf{S} + \frac{1}{\lambda}(\frac{1}{\delta}\mathsf{U} + \mathsf{C}))$, where $\lambda := |M|/|X|$ and δ is the spectral gap of P.

Quantum analogues of random walks, namely, quantum walks, have been developed for solving different problems. In 2003, Ambainis [5] proposed a quantum walk algorithm for solving the element distinctness problem. His algorithm also solves the Markov chain search problem on the Johnson graph with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$. In 2004, Szegedy [45] gave a quantum walk algorithm for more generalized Markov chains with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda\delta}}(\mathsf{U} + \mathsf{C}))$. We can view Szegedy's quantum walk as a quantum counterpart of a random walk, where one checks the state after each transition. Szegedy's quantum walk only detects the presence of a marked state, but cannot find one without extra costs. In 2006, Magniez et al. [33] proposed a quantum walk search framework that unified the advantages of the quantum walks in [5] and [45]. In this quantum walk framework, we can perform the following operations:

- **Setup:** with cost S. preparing the initial state $|\pi\rangle = \frac{1}{\sqrt{|X|}}\sum_x \sqrt{\pi_x}\,|x\rangle$.
- **Update:** with cost U. applying the transformation $|x\rangle\,|0\rangle \mapsto |x\rangle \sum_{y\in X}\sqrt{p_{xy}}\,|y\rangle$.
- **Checking:** with cost C, applying the transformation: $|x\rangle \mapsto \begin{cases} -|x\rangle & \text{if } x \in M \\ |x\rangle & \text{otherwise.} \end{cases}$

The main result of [33] is summarized as follows.

▶ **Lemma 22** ([33]). *Let P be an irreducible and ergodic Markov chain P on X. Let $M \subseteq X$ be a subset of marked elements. Let $\lambda := |M|/|X|$ and δ be the spectral gap of P. Then, there exists a quantum algorithm that with high probability, determines M is empty or finds an $x \in M$ with cost $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C}))$.*

To solve the element distinctness problem, we define a Markov chain, following the work [5, 11, 28]. The state space X is all subsets of $[n]$ with size r. The Markov chain is based on the Johnson graph on X, where an edge is connecting S and S' if and only if $|S \cap S'| = r - 1$. The transition probability on each edge is hence $\frac{1}{r(n-r)}$. A state S is marked when there exist distinct $i, j \in S$ such the i^{th} and the j^{th} items are the same. The Markov chain has spectral gap $\delta \geq 1/r$ (see [28]) and it is easy to verify that $\lambda \geq \binom{n-2}{r-2}/\binom{n}{r} = O(r^2/n^2)$. If we only consider the query complexity, the setup procedure costs r queries, the update procedure costs one query, and the checking procedure does not cost any query. Choosing $r = n^{2/3}$ yields the optimal query complexity $O(n^{2/3})$.

3 Quantum fine-grained complexity

In this section, we give the formal definitions of the quantum fine-grained reduction and quantum strong exponential time hypothesis (QSETH). Moreover, we show that under QSETH, for $d = \text{polylog}(n)$, the lower bounds for $\mathsf{CP}_{n,d}$ and $\mathsf{OV}_{n,d}$ are $n^{1-o(1)}$, which nearly matches the upper bounds given in Theorem 15.

3.1 Quantum fine-grained reduction and QSETH

QSETH is defined based on the assumption that the best quantum algorithm for CNF-SAT is Grover search when the clause width k is large enough.

▶ **Hypothesis 23** (QSETH). *For every $\epsilon > 0$, there exists a $k = k(\epsilon) \in \mathbb{N}$ such that no quantum algorithm can solve k-SAT (i.e., satisfiability on a CNF of width k) in $O(2^{(1/2-\epsilon)n})$ time where n is the number of variables. Moreover, this holds even when the number of clauses is at most $c(\epsilon)n$ where $c(\epsilon)$ denotes a constant that depends only on ϵ.*

Obviously, the Grover search can solve CNF-SAT in $\widetilde{O}(2^{n/2})$. To the best of the our knowledge, there is no quantum algorithm that can do better than $O(2^{n/2})$ for any k.

We recall that in the quantum query model, the input of a problem is given by a quantum oracle. Specifically, let P be a problem, and X be an instance of P in the classical setting. Then, in the quantum query model, X will be given by an oracle $\mathcal{O}_X$. We will denote an algorithm or an oracle $\mathcal{A}$ with access to $\mathcal{O}_X$ by $\mathcal{A}(\mathcal{O}_X)$.

We say $\mathcal{A}_\epsilon$ is an ϵ-oracle for problem P, if for every instance $\mathcal{O}_X$, it holds that

$$\Pr[\mathcal{A}_\epsilon(\mathcal{O}_X) = \mathsf{P}(X)] \geq 1 - \epsilon, \tag{8}$$

and the running time is $O(1)$, where $\mathsf{P}(X)$ is the answer of X for problem P.

▶ **Definition 24** (Quantum oracles). *Let $X := \{x_1, \ldots, x_n\}$ be an instance of some problem and $\mathcal{O}_X$ be the corresponding quantum oracle. To realize $\mathcal{O}_X$, we do not need to write down the whole X; instead, we can just design a quantum circuit to realize the mapping*

$$|i\rangle |0\rangle \xrightarrow{\mathcal{O}_X} |i\rangle |x_i\rangle . \tag{9}$$

▶ **Definition 25** (Quantum fine-grained reduction). *Let $p(n)$ and $q(n)$ be nondecreasing functions of n. Let P and Q be two problems in the quantum query model and $\mathcal{A}_\epsilon$ be an ϵ-oracle for Q with error probability $\epsilon \leq 1/3$. P is quantum (p, q)-reducible to Q, denoted as $(\mathsf{P}, p) \leq_{\text{QFG}} (\mathsf{Q}, q)$, if for every ϵ, there exits a $\delta > 0$, and algorithm R with access to $\mathcal{A}_\epsilon$, a constant d, and an integer $k(n)$, such that for every $n \geq 1$, the algorithm R takes any instance of P of size n and satisfies the following:*

- *R can solve P with success probability at least $2/3$ in time at most $d \cdot p(n)^{1-\delta}$.*
- *R performs at most $k(n)$ quantum queries to A_ϵ. Specifically, in the j^{th} query, let $\mathbf{X}_j := \{X_{1,j}, X_{2,j}, \ldots\}$ be a set instances of Q. Then, R realizes the oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \ldots\}$ in superposition and applies A_ϵ to solve the instances.*
- *The following inequality holds.*

$$\sum_{j=1}^{k(n)} c(\mathbf{X}_j) \cdot q(n_j)^{1-\epsilon} \leq d \cdot p(n)^{1-\delta},$$

where $c(\mathbf{X}_j)$ is the time required for R to realize the oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \ldots\}$ in superposition and $n_j := \max_i |X_{i,j}|$.

In Definition 25, the input of A_ϵ is given as a quantum oracle such that A_ϵ can be a quantum query algorithm with running time strictly less than the input size. Moreover, the quantum reduction R can realize quantum oracles $\{\mathcal{O}_{X_{1,j}}, \mathcal{O}_{X_{2,j}}, \dots\}$ in superposition, and thus the time required is $\max_i c(X_{i,j})$ (where $c(X_{i,j})$ is the time required to realize $\mathcal{O}_{X_{i,j}}$) instead of $\sum_i c(X_{i,j})$. This also allows R to use fast quantum algorithms to process the information of $A'_\epsilon s$ output (e.g., amplitude amplification).

3.2 Lower bounds for CP, OV, and BCP in higher dimensions under QSETH

Here, we give nearly linear lower bounds for OV and CP under QSETH by showing that there exist quantum fine-grained reductions from SAT to these problems.

▶ **Theorem 26.** *Assuming QSETH, for all $\epsilon > 0$, there exists a c such that $\mathsf{OV}_{n,c\log n}$ and $\mathsf{CP}_{n,(\log n)^c}$ cannot be solved by any quantum algorithm in time $O(n^{1-\epsilon})$.*

We prove Theorem 26 by showing that there exist quantum fine-grained reductions from CNF-SAT to OV, OV to BCP, and BCP to CP with desired parameters. We first give the reduction from CNF-SAT to OV as a warm-up.

▶ **Lemma 27.**

$$(\mathsf{CNF\text{-}SAT}_n, 2^{n/2}) \leq_{\mathrm{QFG}} (\mathsf{OV}_{n_1,d_1}, n_1), \tag{10}$$

where $n_1 = 2^{n/2}$ and $d_1 = \Theta(n)$.

Proof. Let ϕ be a CNF formula with n variables and $m = \Theta(n)$ clauses. Let $\mathcal{A}$ be an algorithm for OV. We first recall the classical reduction. Let $\phi := \phi_1 \wedge \cdots \wedge \phi_m$. We divide the n variables into two sets A and B with $|A| = |B| = \frac{n}{2}$. Let $A := \{x_1, \dots, x_{n/2}\}$ and $B := \{x_{n/2+1}, \dots, x_n\}$. We let $S_A := \{a_1, \dots, a_{2^{n/2}}\}$ be all assignments to A and $S_B := \{b_1, \dots, b_{2^{n/2}}\}$ be all assignments to B. We describe two mappings $f_A : S_A \to \{0,1\}^m$ and $f_B : S_B \to \{0,1\}^m$ as follows:

$$f_A(a_i) = [\phi_1(a_i), \dots, \phi_m(a_i)]^T, \text{ and} \tag{11}$$

$$f_B(b_i) = [\phi_1(b_i), \dots, \phi_m(b_i)]^T, \tag{12}$$

where $\phi_j(a_i) = 0$ if a_i is a satisfied assignment for ϕ_j, and $\phi_j(a_i) = 1$ otherwise; we define $\phi_i(b_i)$ in the same way. Let $F_A := \{f_A(a_i) : \ i \in [2^{n/2}]\}$ and $F_B := \{f_B(b_i) : \ i \in [2^{n/2}]\}$. Then, it is obvious that if there exist $v \in F_A$ and $u \in F_B$ such that $\langle v, u\rangle = 0$, then ϕ is satisfiable. However, at first glance, this reduction with $O(2^{n/2})$ running time is not fine-grained since we require the cost of the reduction to be at most $2^{n(1-\delta)/2}$ for some $\delta > 0$ by Definition 25, but writing down elements in F_A and F_B already takes $\Omega(2^{n/2})$.

Nevertheless, as in Definition 24, a quantum fine-grained reduction only needs to realize the functions f_A and f_B, which takes $O(mkn)$ time where k is the width of clauses. This is much less than $O(2^{n(1-\delta)/2})$. More specifically, f_A and f_B are oracles for F_A and F_B, and for any quantum query to elements in F_A or F_B, the reduction can implement oracles f_A and f_B:

$$|e, x\rangle |0\rangle \xrightarrow{f_e} |e, x\rangle |f_e(x)\rangle , \tag{13}$$

where $e \in \{A, B\}$, and the time $c(f_e)$ for the reduction to implement f_e for one quantum query is at most $O(kmn)$. Finally, this reduction only uses one oracle (F_A, F_B). If there is an algorithm for OV which succeeds with probability $2/3$, we can boost the success probability of the reduction by repetition. Therefore, $(\mathsf{CNF\text{-}SAT}, 2^{n/2})$ is quantum reducible to $(\mathsf{OV}_{n_1,d_1}, n_1)$. ◀

Then, to prove $(\mathsf{CNF\text{-}SAT}, 2^{n/2}) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n_3,d_3}, n_3)$, we show that $(\mathsf{BCP}_{n_2,d_2}, n_2) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n_3,d_3}, n_3)$ and $(\mathsf{OV}_{n_1,d_1}, n_1) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n_2,d_2}, n_2)$, where n_2, n_3, d_2, d_3 are some functions of n specified in the following lemmas.

▶ **Lemma 28.** *For $d = \Theta(\log n)$,*

$$(\mathsf{BCP}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{CP}_{n',d'}, n'), \tag{14}$$

where $n' = n^{O(1)}$ and $d' = (\log n)^c$ for some constant c and all points have $\{0,1\}$ entries with the Hamming metric.

▶ Remark 29. The points have coordinate entries in $\{0,1\}$, and the Hamming metric is equivalent to distance in ℓ_2-metric (up to power of 2) in this case. Therefore, in the proof of Lemma 28, we can consider the Hamming distance between points instead of ℓ_2 distance without loss of generality.

We first introduce the classical reductions in [29] and some results we will use to prove Lemma 28.

Classical reduction

We can consider an instance of BCP with two sets of points A and B as a weighted complete bipartite graph $K_{n,n}$, where the vertices are the points in these two sets and edges' weights are equal to the distances between the corresponding points. Then, solving BCP is equivalent to find an edge with the minimum weight in this graph. However, we cannot directly apply the algorithm for CP on this graph since there could be two points in the same set (no edge connecting them) that have a smaller distance than any pairs of points in two sets (connected by an edge). To overcome this difficulty, we can "stretch" the points to make the points in the same set far from each other, which is characterized by the contact dimension of a graph:

▶ **Definition 30** (Contact Dimension). *For any graph $G = (V, E)$, a mapping $\tau : V \to \mathbb{R}^d$ is said to realize G if for some $\beta > 0$, the following holds for every distinct vertices u, v:*

$$\begin{aligned} &\|\tau(u) - \tau(v)\|_2 = \beta \text{ if } \{u, v\} \in E, \\ &\|\tau(u) - \tau(v)\|_2 > \beta \text{ otherwise.} \end{aligned} \tag{15}$$

The contact dimension of G, denoted by $\mathrm{cd}(G)$, is the minimum $d \in \mathbb{N}$ such that there exists $\tau : V \to \mathbb{R}^d$ realizing G.

That is, with the help of τ, we can restrict the optimal solution of CP to be the points connected by an edge in G. But we cannot realize the whole complete bipartite graph since $\mathrm{cd}(K_{n,n}) = \Theta(n)$, which makes the dimension of the CP instance too large. [29] showed that we can realize a subgraph of $K_{n,n}$ and apply permutations to its vertices such that the union of these subgraphs cover $K_{n,n}$. In this way, BCP can be computed by solving CP on each subgraph and outputting the best solution. More specifically, the reduction in [29] relies on the following theorem:

▶ **Theorem 31** (Theorem 4.2 in [29]). *For every $0 < \delta < 1$, there exists a log-dense sequence $(n_i)_{i\in\mathbb{N}}$ such that, for every $i \in \mathbb{N}$, there is a bipartite graph $G_i = (A_i \dot\cup B_i, E_i)$ where $|A_i| = |B_i| = n_i$ and $|E_i| \geq \Omega(n_i^{2-\delta})$, such that $\mathrm{cd}(G_i) = (\log n_i)^{O(1/\delta)}$. Moreover, for all $i \in \mathbb{N}$, a realization $\tau : A_i \dot\cup B_i \to \{0,1\}^{(\log n_i)^{O(1/\delta)}}$ of G_i can be constructed in $n_i^{2+o(1)}$ time.*

The log-dense sequence is defined as follows:

▶ **Definition 32.** *A sequence $(n_i)_{i\in\mathbb{N}}$ of increasing positive integers is log-dense if there exists a constant $c \geq 1$ such that $\log n_{i+1} \leq c \cdot \log n_i$ for all $i \in \mathbb{N}$.*

They also showed that, the permutations for covering the complete bipartite graph can be efficiently found, as shown in the following lemma.

▶ **Lemma 33** (Lemma 3.11 in [29])**.** *For any bipartite graph $G(A\dot\cup B, E_G)$ where $|A| = |B| = n$ and $E_G \neq \emptyset$, there exist side-preserving permutations $\pi_1, \ldots, \pi_k : A \cup B \to A \cup B$ where $k \leq \frac{2n^2 \ln n}{|E_G|} + 1$ such that*

$$\bigcup_{i\in[k]} E_{G_{\pi_i}} = E_{K_{n,n}}. \tag{16}$$

Moreover, such permutations can be found in $O(n^6 \log n)$ time.

Now, we are ready to state the quantum fine-grained reduction by "quantizing" the classical reduction.

Proof of Lemma 28. Let A, B be the two sets of input points of BCP. Suppose for BCP, there is an input oracle $\mathcal{O}_{\mathsf{BCP}}$ which, given an index, returns the corresponding point:

$$|b\rangle|i\rangle|0\rangle \xrightarrow{\mathcal{O}_{\mathsf{BCP}}} \begin{cases} |b\rangle|i\rangle|x_i\rangle & \text{if } b = 0, \\ |b\rangle|i\rangle|y_i\rangle & \text{if } b = 1, \end{cases} \tag{17}$$

where x_i is the i-th point in the set A and y_i is the i-th point in the set B. The sizes of A and B are both equal to n and each point is in $\{0,1\}^{d_1}$, where $d_1 = \Theta(\log n)$ is the dimension of BCP.

For CP, suppose there is a quantum algorithm $\mathcal{A}$ such that for m points in $\{0,1\}^{d_2}$ given by an oracle $\mathcal{M}_{CP}$, $\mathcal{A}^{\mathcal{M}_{CP}}$ returns the closest pair of these n points with probability at least $2/3$.

Then we need to transform $\mathcal{O}_{\mathsf{BCP}}$ to some oracles $\mathcal{M}_i$ for CP, such that by running $\mathcal{A}$ with $\mathcal{M}_i$ as input oracles, we can get the bichromatic closest pair between A and B. The reduction has four steps:

1. Pre-processing

We first follow the classical reduction to pre-process the input points of BCP. For some integer $n' \leq n^{0.1}$, we can partition A and B into n'-size subsets:

$$\begin{aligned} A &= A_1 \dot\cup \cdots \dot\cup A_r, \\ B &= B_1 \dot\cup \cdots \dot\cup B_r, \end{aligned} \tag{18}$$

where $r = \lfloor n/n' \rfloor$. Here, we assume that n is divisible by n'. It follows that

$$\mathsf{BCP}(A, B) = \min_{i,j\in[r]} \mathsf{BCP}(A_i, B_j). \tag{19}$$

Then, we use the algorithm in [29] to construct k mappings $f_1, \ldots, f_k : [2n'] \to \{0,1\}^{d'}$ such that

$$\mathsf{BCP}(A_i, B_j) = \min_{t\in[k]} \mathsf{CP}(f_t(A_i) \cup f_t(B_j)) \quad \forall i, j \in [\lfloor n/n' \rfloor]. \tag{20}$$

More specifically, we pick n' to be the largest number in a log-dense sequence that is smaller than $n^{0.1}$. Then, we apply Theorem 31 to classically construct a bipartite graph $G(A \cup B, E)$ with n' vertices in each side and a realization τ. By choosing $\delta = \epsilon/2$ in Theorem 31, the graph G has $|E| = \Omega(n'^{2-\epsilon/2})$ edges. And we can get $2n'$ 0/1-strings of length $(\log n')^{O(2/\epsilon)}$:

$$\tau_i^A = \tau(u_i) \quad \forall u_i \in A, \quad \text{and} \quad \tau_i^B = \tau(v_i) \quad \forall v_i \in B. \tag{21}$$

In order to cover the complete bipartite graph, we run the classical algorithm (Lemma 33) to find k permutations $\pi_1, \ldots, \pi_k : [n'] \to [n']$, where k is a parameter to be chosen later.

Then, we can define the mappings as follows:

$$f_t(u) = \begin{cases} x_v \circ \left(\tau^A_{\pi_t(w)}\right)^{d+1} & \text{if } 1 \le u \le n' \\ y_v \circ \left(\tau^B_{\pi_t(w)}\right)^{d+1} & \text{if } n' < u \le 2n' \end{cases} \qquad \forall t \in [k], u \in [2n'], \tag{22}$$

where $\circ$ means string concatenation and $(s)^{d+1}$ denotes $d+1$ copies of the string s. For a point $p \in A_i \cup B_j$, $u \in [2n']$ is the index in this union-set, $v \in [n]$ is the index in the ground set A or B, and $w \in [n']$ is the index in the subset A_i or B_j. Further, if $1 \le u \le n'$, then $w := u$; otherwise, $w := u - n'$.

2. Oracle construction

For $i, j \in [r], t \in [k]$, we then construct the input oracle $\mathcal{M}_{i,j,t}$ for the problem $\mathsf{CP}(f_t(A_i) \cup f_t(B_j))$. For a query index $u \in [2n']$,

$$M_{i,j,t} |u\rangle |0\rangle = |u\rangle |f_t(u)\rangle . \tag{23}$$

With the help of the input oracle $\mathcal{O}_{\mathsf{BCP}}$, we can implement $\mathcal{M}_{i,j,t}$ in the following way:

1. Prepare an ancilla qubit $|b\rangle$ such that $b = 1$ if $u > n'$.
2. Transform $|u\rangle$ to $|v\rangle$, the index of the point in A or B, based on the value of b. Note that the index is unique. Hence, this transformation is unitary and can be easily achieved by a small quantum circuit.
3. Query $\mathcal{O}_{\mathsf{BCP}}$ with input $|b\rangle |v\rangle$. Assume $b = 0$. Then,

$$|b\rangle |v\rangle |0\rangle \xmapsto{\mathcal{O}_{\mathsf{BCP}}} |b\rangle |v\rangle |x_v\rangle . \tag{24}$$

4. Similar to the second step, the index w of the point in A_i and B_j can be computed from v by a unitary transformation:

$$|b\rangle |v\rangle |x_v\rangle \mapsto |b\rangle |w\rangle |x_v\rangle \tag{25}$$

5. Since each w corresponds to a unique string $\tau^A_{\pi_t(w)}$, we can attach $d+1$ copies of this string to the remaining quantum registers:

$$|b\rangle |w\rangle |x_v\rangle \mapsto |b\rangle |w\rangle |x_v\rangle \left| \left(\tau^A_{\pi_t(w)}\right)^{d+1} \right\rangle . \tag{26}$$

6. By recovering u from w, we get the final state:

$$|u\rangle |f_t(u)\rangle = |u\rangle \left| x_v, \left(\tau^A_{\pi_t(w)}\right)^{d+1} \right\rangle . \tag{27}$$

3. Query process

By Equations (19) and (20), we have

$$\mathsf{BCP}(A,B) = \min_{i,j\in[r],t\in[k]} \mathsf{CP}(f_t(A_i)\cup f_t(B_j)). \tag{28}$$

Hence, we can use quantum minimum-finding algorithm in Theorem 15 over the sub-problems to find the minimum solution. For each sub-problem, we can run the algorithm for CP with $\mathcal{M}_{i,j,t}$ as the input oracle.

4. Post-processing

In case that n is not divisible by n', let the remaining points in A and B be A_{res}, B_{res}, respectively. Then, we can use Grover search to find the closest pair between A_{res} and B, and between B_{res} and A. Then, compare the answer to the previously computed result and pick the smaller one.

Correctness

In this reduction, we do not change the constructions of the mappings $\{f_i\}_{i\in[k]}$. By [29], Equation (28) is correct in the classical setting. Hence, it also holds in the quantum setting, and we can use Grover search to find the minimum solution. However, since the algorithm $\mathcal{A}$ for CP has success probability 2/3, for each tuple $(i,j,t)\in[r]\times[r]\times[k]$, we need to run $\mathcal{A}^{\mathcal{M}_{i,j,t}}$ $O(\log n)$ times to boost the success probability to at least $1-\frac{1}{n}$. Then, by the union bound, the probability that all queries in the Grover search are correct is at least 99/100. Hence, by Theorem 9, the overall success probability is at least 2/3.

Running Time of the Reduction

The running time of the pre-processing step consists of two parts: (1) constructing the graph G and its realization τ; (2) finding k permutations. For the first part, by Theorem 31, it can be done in $n'^{2+o(1)}$ time. For the second part, we pick $k = O(\frac{2n'^2\log n'}{n'^{2-\epsilon/2}}) = O(n'^{\epsilon/2}\log n')$, and by Lemma 33, it can be done in $O(n'^6\log n')$ time. Hence, the total running time of pre-processing step is $n'^{2+o(1)} + O(n'^6\log n') = \widetilde{O}(n^{0.6})$.

The oracle construction can be done "on-the-fly". More specifically, given the strings $\{\tau_i^A,\tau_i^B\}_{i\in[n']}$, and permutations $\{\pi_i\}_{i\in[k]}$, for each query index u, we can simulate the oracle $\mathcal{M}_{i,j,t}$ defined in Equation (23) in $c(\mathcal{M}_{i,j,t}) = O(d_2) = (\log n')^{\Omega(1)} = \widetilde{O}(1)$ time.

In the query process, for each CP instance indexed by (i,j,t), suppose $\mathcal{A}^{\mathcal{M}_{i,j,t}}$ gets the answer in time $q(n') = n'$. Moreover, for each time $\mathcal{A}$ querying the input oracle $\mathcal{M}_{i,j,t}$, we need to spend $c(\mathcal{M}_{i,j,t})$ time to simulate the oracle. And we also have $O(\log n)$ runs for each instance. Hence, the total running time for each CP is at most

$$n'^{1-\epsilon}\cdot\widetilde{O}(1)\cdot O(\log n) = \widetilde{O}(n'^{1-\epsilon}). \tag{29}$$

Then, we use Grover's search algorithm over $r^2\cdot k$ instances, which can be done by querying $\widetilde{O}(\sqrt{r^2\cdot k})$ instances by Theorem 9. Therefore, for any $\epsilon>0$, we have

$$\widetilde{O}(\sqrt{r^2k})\cdot q(n')^{1-\epsilon}\cdot c(\mathcal{M}_{i,j,t})\cdot O(\log n) = \widetilde{O}(\sqrt{(n/n')^2k}\cdot(n')^{1-\epsilon}) \tag{30}$$

$$\leq \widetilde{O}(n\cdot(n')^{-\epsilon}) \leq \widetilde{O}(n\cdot n^{-\epsilon/2}) \leq n^{1-\delta}, \tag{31}$$

where the first inequality follows from $k = O(n'^{\epsilon/2}\log n')$ as shown in [29] and the last inequality follows by setting $\delta = \epsilon/10$.

For the post-processing step, the sizes of A_{res} and B_{res} are at most n'. The running time is

$$O(\sqrt{n \cdot n'} \cdot \log n) \leq \widetilde{O}(n^{0.55}). \tag{32}$$

Therefore, for any $\epsilon > 0$, there exists a $\delta > 0$ such that the Equation (30) holds and the total reduction time is $O(n^{1-\delta})$. By Definition 25, BCP_{n,d_1} can be quantum fine-grained reduced to CP_{n,d_2}. This completes the proof of this lemma. ◀

Finally, we show that $(\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}, n)$ by quantizing the reduction in [29] following the same idea.

▶ **Lemma 34.** *For $d = \Theta(\log n)$,*

$$(\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}, n), \tag{33}$$

where $d' = \Theta(\log n)$.

Proof. For an OV instance with sets of vectors A and B, let $\mathcal{O}_{\mathsf{OV}}$ be the input oracle such that:

$$\mathcal{O}_{\mathsf{OV}} |i\rangle |0\rangle = \begin{cases} |i\rangle |a_i\rangle & \text{if } i \in A, \\ |i\rangle |b_i\rangle & \text{if } i \in B. \end{cases} \tag{34}$$

where $a_i, b_i \in \{0,1\}^d$.

Then, similar to the classical reduction, we can construct mappings $f_A, f_B : \{0,1\}^d \to \{0,1\}^{5d}$ such that

$$f_A(a_i)_{5j-4:5j} = \begin{cases} 11000 & \text{if } a_i(j) = 0 \\ 00110 & \text{if } a_i(j) = 1 \end{cases} \quad \forall j \in [d], \tag{35}$$

and

$$f_B(b_i)_{5j-4:5j} = \begin{cases} 10100 & \text{if } b_i(j) = 0, \\ 01001 & \text{if } b_i(j) = 1. \end{cases} \quad \forall j \in [d]. \tag{36}$$

By the classical reduction, we have

$$\mathsf{OV}(A, B) = 1 \text{ if and only if } \mathsf{BCP}(f_A(A), f_B(B)) = 2d \tag{37}$$

under Hamming distance.

Also, note that we can simulate the input oracle $\mathcal{O}_{\mathsf{BCP}}$ by first querying the oracle $\mathcal{O}_{\mathsf{OV}}$ to get the vector, then applying the corresponding mapping f_A or f_B, which can be done in $c(\mathcal{O}_{\mathsf{BCP}}) = O(d)$ time. Let the running time of the algorithm for BCP be $q(n) = n$. Then for any $\epsilon > 0$,

$$q(n)^{1-\epsilon} \cdot c(\mathcal{O}_{\mathsf{BCP}}) = n^{1-\epsilon} \cdot \Theta(\log n) \leq n^{1-\delta} \tag{38}$$

for some small $\delta > 0$. Hence, by Definition 25, $(\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}, n)$. ◀

Proof of Theorem 26. We can prove the theorem by contradiction following Lemma 27, Lemma 34, and Lemma 28. Specifically, suppose that there exists an $\epsilon > 0$, for all $d = \Theta(\log n)$, there exists a quantum algorithm which can solve OV in time $O(n^{1-\epsilon})$. Then, we can obtain a quantum algorithm for $\mathsf{CNF\text{-}SAT}$, which runs in time $O(2^{n/2(1-\epsilon)})$ by Lemma 27. This contradicts QSETH. The proof for CP is the same. ◀

3.3 Quantum lower bound for BCP in nearly-constant dimensions under QSETH

A byproduct of the previous subsection is a quantum lower bound for BCP in higher dimensions (i.e., $d = \text{polylog}(n)$) under QSETH (Lemma 34). In this subsection, we show that this quantum lower bound for BCP even holds for nearly-constant dimensions (i.e., $d = c^{\log^*(n)}$). The main result of this subsection is the following theorem.

▶ **Theorem 35.** *Assuming QSETH, there is a constant c such that* BCP *in $c^{\log^*(n)}$ dimensions requires $n^{1-o(1)}$ time for any quantum algorithm.*

We will "quantize" the results by Chen [17] to prove this theorem. More specifically, we first show a quantum fine-grained self-reduction of OV from $\log n$ dimensions with binary entries to $2^{\log^*(n)}$ dimensions with integer entries ($\mathbb{Z}$-OV). Then, we give a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP in nearly-constant dimensions.

▶ **Definition 36** (Integral Orthogonal Vector, $\mathbb{Z}$-OV)**.** *Given two sets A, B of n vectors in $\mathbb{Z}^d$, find a pair of vectors $a \in A$ and $b \in B$ such that $\langle a, b\rangle = 0$, where the inner product is taken in $\mathbb{Z}$.*

We use $\mathbb{Z}\text{-OV}_{n,d}$ to denote $\mathbb{Z}$-OV with n vectors of d dimension in each set. We then recap a theorem in [17]:

▶ **Theorem 37** ([17, Theorem 4.1])**.** *Let b, ℓ be two sufficiently large integers. There is a classical reduction $\psi_{b,\ell} : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^\ell$ and a set $V_{b,\ell} \subseteq \mathbb{Z}$, such that for every $x, y \in \{0,1\}^{b\cdot\ell}$,*

$$\langle x, y\rangle = 0 \;\Leftrightarrow\; \langle \psi_{b,\ell}(x), \psi_{b,\ell}(y)\rangle \in V_{b,\ell} \tag{39}$$

and

$$0 \le \psi_{b,\ell}(x)_i < \ell^{6^{\log^*(b)}\cdot b} \tag{40}$$

for all possible x and $i \in [\ell]$. Moreover, the computation of $\psi_{b,\ell}(x)$ takes $\text{poly}(b\cdot\ell)$ time, and the set $V_{b,\ell}$ can be constructed in $O\left(\ell^{O(6^{\log^*(b)}\cdot b)} \cdot \text{poly}(b\cdot\ell)\right)$ time.

Note that the size of $V_{b,\ell}$ is at most $\ell^{2\cdot 6^{\log^*(b)}\cdot b+1}$. The following lemma gives a quantum fine-grained reduction from OV to $\mathbb{Z}$-OV:

▶ **Lemma 38.** *For $d = \Theta(\log n)$,*

$$(\text{OV}_{n,d}, n) \le_{\text{QFG}} (\mathbb{Z}\text{-OV}_{n,d'}, n). \tag{41}$$

where $d' = 2^{O(\log^* n_2)}$.

Proof. Consider an $\text{OV}_{n,d}$ with $d = c\cdot\log n$, where c is an arbitrary constant. We choose $\ell := 7^{\log^* n}$ and $b := d/\ell$. Then, we can apply Theorem 37 to get the mapping function $\psi_{b,\ell}$ and the set $V_{b,\ell}$. For each $v \in V_{b,\ell}$, we'll construct an instance of $\mathbb{Z}\text{-OV}_{n,\ell+1}$ as follows:

1. Let $|i\rangle$ be the input query index of $\mathbb{Z}\text{-OV}_{n,\ell+1}$.
2. Query $\text{OV}_{n,d}$'s input oracle $\mathcal{O}_{\text{OV}}$ and get the vector $|i, x\rangle$.
3. Compute the mapping $\psi_{b,\ell}$ and get $|i, x\rangle|\psi_{b,\ell}(x)\rangle$.
4. If $x \in A$, then attach 1 to the end of the register: $|i, x\rangle|\psi_{b,\ell}(x), 1\rangle$. If $x \in B$, then attach $-v$ to the end: $|i, x\rangle|\psi_{b,\ell}(x), -v\rangle$.
5. Use $\mathcal{O}_{\text{OV}}$ to erase x and return the final input state $|i\rangle|\psi_{b,\ell}(x), 1\rangle$ or $|i\rangle|\psi_{b,\ell}(x), -v\rangle$.

For each instance, we can use the quantum oracle for $\mathbb{Z}\text{-OV}_{n,\ell+1}$ to check the orthogonality. $\text{OV}_{n,d}$ is YES if and only if there exists a YES-instance of $\mathbb{Z}\text{-OV}_{n,\ell+1}$.

Correctness

The correctness follows from Equation (39):

$$\langle x, y\rangle = 0 \Leftrightarrow \langle \psi_{b,\ell}(x), \psi_{b,\ell}(y)\rangle = v \in V_{b,\ell} \Leftrightarrow \langle [\psi_{b,\ell}(x),\ 1], [\psi_{b,\ell}(y),\ -v]\rangle = 0. \tag{42}$$

Reduction time

Note that for $\ell = 7^{\log^* n}$ and $b = d/\ell$, we have:

$$\log\left(\ell^{O(6^{\log^*(d)}\cdot b)}\right) = \log \ell \cdot O\left(6^{\log^*(d)} \cdot (d/\ell)\right) \tag{43}$$

$$= O\left(\log^*(n) \cdot 6^{\log^* n} \cdot c \log n / 7^{\log^* n}\right) \tag{44}$$

$$= o(\log n). \tag{45}$$

This implies that $|V_{b,\ell}| \leq \ell^{2\cdot 6^{\log^*(b)}\cdot b+1} \leq n^{o(1)}$. Hence, the number of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ instances is $n^{o(1)}$ and the running time for compute $V_{b,\ell}$ is $n^{o(1)}$. And for each input query, the oracle for $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ can be simulated in $c(\mathcal{O}_{\mathbb{Z}\text{-}\mathsf{OV}}) = \mathrm{poly}(d) = \mathrm{poly}(\log n)$ time. We can show that for every $\epsilon > 0$, if $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ can be decided in $n^{1-\epsilon}$ time, then

$$\sum_{v\in V_{b,\ell}} n^{1-\epsilon} \cdot c(\mathcal{O}_{\mathbb{Z}\text{-}\mathsf{OV}}) = n^{o(1)} \cdot n^{1-\epsilon} \cdot \mathrm{poly}(\log n) \leq n^{1-\delta} \tag{46}$$

for some $\delta > 0$, which satisfies the definition of quantum fine-grained reduction (Definition 25).

Therefore, $\mathsf{OV}_{n,O(\log n)}$ is quantum fine-grained reducible to $\mathbb{Z}$-$\mathsf{OV}_{n,2^{O(\log^*(n))}}$. ◀

Then, we give a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP:

▶ **Lemma 39.** *For $d = 2^{O(\log^* n)}$,*

$$(\mathbb{Z}\text{-}\mathsf{OV}_{n,d}, n) \leq_{\mathrm{QFG}} (\mathsf{BCP}_{n,d'}). \tag{47}$$

where $d' = d^2 + 2$.

Proof. We remark here that this proof closely follows that for Theorem 4.3 in [17]. We nonetheless give it here as some details are different.

For an $\mathbb{Z}$-$\mathsf{OV}_{n,d}$ instance with $(k \cdot \log n)$-bit entries, we construct a BCP instance as follows:

1. For $x \in A$, construct a vector $x' \in \mathbb{Z}^{d^2}$ such that $x'_{i,j} = x_i \cdot x_j$. Here, we index a d^2-dimensional vector by $[d] \times [d]$. Similarly, for $y \in B$, construct a vector $y' \in \mathbb{Z}^{d^2}$ such that $y'_{i,j} = -y_i \cdot y_j$.
2. Choose $W := (d^2+1) \cdot n^{4k}$. For each x', construct a vector $x'' \in \mathbb{R}^{d^2+2}$ such that

$$x'' = \left[x',\ \sqrt{W - \|x'\|_2^2},\ 0\right]. \tag{48}$$

For each y', construct a vector $y'' \in \mathbb{R}^{d^2+2}$ such that

$$y'' = \left[y',\ 0,\ \sqrt{W - \|y'\|_2^2}\right]. \tag{49}$$

Then, we claim that the $\mathbb{Z}$-OV instance is YES if and only if the BCP instance has the minimum distance $\leq \sqrt{2W}$.

Correctness

First note that $\|x'\|_2^2 \leq d^2 \cdot (2^{k \log n})^4 = d^2 \cdot n^{4k}$. Hence, $W - \|x'\|_2^2 > 0$ and $W - \|y'\|_2^2 > 0$. For any x'' and y'' in the new constructed instance of BCP, we have

$$\begin{aligned} \|x'' - y''\|_2^2 &= \|x''\|_2^2 + \|y''\|_2^2 - 2 \cdot \langle x'', y'' \rangle && (50) \\ &= 2 \cdot W - 2 \cdot \langle x', y' \rangle && (51) \\ &= 2 \cdot W - 2 \cdot \sum_{(i,j) \in [d] \times [d]} x_i \cdot x_j \cdot (-y_j \cdot y_j) && (52) \\ &= 2 \cdot W + 2 \cdot (\langle x, y \rangle)^2. && (53) \end{aligned}$$

Hence,

$$\langle x, y \rangle = 0 \;\Leftrightarrow\; \|x'' - y''\|_2^2 = 2W. \tag{54}$$

Reduction time

We can see from the above description that the input mapping function is simple and can be computed by a small quantum circuit in $O(d^2) = O(2^{O(\log^*(n))})$ time. Hence, we have $c(\mathcal{O}_{\mathsf{BCP}}) = O(2^{O(\log^*(n))})$. Also, by Definition 25, it's easy to check that this is indeed a quantum fine-grained reduction from $\mathbb{Z}$-OV to BCP. ◀

Now Theorem 35 follows immediately from Lemma 38 and Lemma 39:

Proof of Theorem 35. Let $\epsilon > 0$ be some constant. Suppose we can solve $\mathsf{BCP}_{n,c^{\log^*(n)}}$ in $n^{1-\epsilon}$ time for all constant $c > 0$. Then, by Lemma 38 and Lemma 39, we can also solve $\mathsf{OV}_{n,c' \log n}$ in $n^{1-\delta}$ time for some $\delta > 0$ and any $c' > 0$. However, this contradicts QSETH by Theorem 26. Therefore, assuming QSETH, there exists a constant c such that $\mathsf{BCP}_{n,c^{\log^*(n)}}$ requires $n^{1-o(1)}$ time. ◀

4 Closest pair in constant dimension

In this section, we show that there exist almost-optimal quantum algorithms for CP in constant dimension. The main result is the following theorem, which is a direct consequence of Corollary 55 and Theorem 56.

▶ **Theorem 40.** *For any constant dimension, the quantum time complexity for* CP *is* $\widetilde{\Theta}(n^{2/3})$.

Our approach to solve CP is first reducing to the decision version of the problem, and then apply quantum walk algorithms to solve the decision version. We define the decision version of CP, CP_ϵ, as follows.

▶ **Definition 41** (CP_ϵ)**.** *Given a set of points $P \subset \mathbb{R}^d$ and $\epsilon \in \mathbb{R}$, find a pair $a, b \in P$ such that $\|a - b\| \leq \epsilon$ if there is one and returns* no *is no such pair exists.*

The reduction from CP to CP_ϵ is given by the following lemma.

▶ **Lemma 42.** *Let m be the number of bits needed to encode each coordinate as a bit string and d be the dimension. Given an oracle $\mathcal{O}$ for CP_ϵ, there exists an algorithm $A^{\mathcal{O}}$ that runs in time and query complexity $O(m + \log d)$ that solves the* CP.

Proof. Let (P, δ) be an instance of the CP. We first pick an arbitrary pair $a_0, b_0 \in P$ and compute $\Delta(a_0, b_0)$. Then, we set ϵ to be $\Delta(a_0, b_0)/2$ and run the oracle $\mathcal{O}$ to check whether there exists a distinct pair with distance less than $\Delta(a_0, b_0)/2$ or not. If there exists such a pair, which we denote as (a_1, b_1), then we set $\epsilon = \Delta(a_1, b_1)$ and call $\mathcal{O}$ to check again. If there is no such pair, then we set $\epsilon = 3\Delta(a_0, b_0)/4$ and call $\mathcal{O}$. We run this binary search for $m + \log d$ iterations. Finally, the algorithm outputs the closest pair. ◀

In classical setting, point location is an important step in solving the closest-pair problem, especially the dynamic version. For the quantum algorithm, as walking on the Markov chain, we repeatedly delete a point and add a new point. Hence, in each step, the first thing is to determine the location of the new added point.

For simplicity, we assume that $m = O(\log n)$, which is the number of digits of each coordinate of the points. By translation, we can further assume that all the points are lying in $[0, L]^d$, where $L = O(2^m) = \text{poly}(n)$.

Since we are considering CP_ϵ, one simple way of point location is to discretize the whole space into a hypergrid, which is defined as follows:

▶ **Definition 43.** *Let $d, \epsilon, L > 0$. A hypergrid $G_{d,\epsilon,L}$ in the space $[0, L]^d$ consists of all ϵ-boxes*

$$g := [a_1, b_1) \times [a_2, b_2) \times \cdots \times [a_d, b_d), \tag{55}$$

such that $b_1 - a_1 = \cdots = b_d - a_d = \epsilon/\sqrt{d}$ [VII], and a_i is divisible by ϵ for all $i \in [d]$.

For each point $p_i \in [0, L]^d$, we can identify the ϵ-box that contains it using the function $\text{id}(p_i) : [0, L]^d \to \{0, 1\}^{d \log(L/\epsilon)}$:

$$\text{id}(p_i) = \left(\lfloor p_i(1)/w \rfloor, \lfloor p_i(2)/w \rfloor, \ldots, \lfloor p_i(d)/w \rfloor \right), \tag{56}$$

where $w = \frac{\epsilon}{\sqrt{d}}$ is the width of the ϵ-box. The number of bits to store $\text{id}(p_i)$ is $d \cdot \log(L/w) = O(d \cdot \log(L))$. Since all the points in an ϵ-box have the same id, we also use this $g(\text{id}(p))$ to denote this ϵ-box containing p.

For the ease of our analysis, we define the neighbors of a hypergrid.

▶ **Definition 44.** *Let $\epsilon \in \mathbb{R}$. Let g_1, g_2 be two ϵ-boxes in a hypergrid where $\text{id}(g_1) = (x_1, \ldots, x_d)$ and $\text{id}(g_2) = (x'_1, \ldots, x'_d)$. We say that g_1 and g_2 are each other's* ϵ-neighbor *if*

$$\sqrt{\sum_{i=1}^{d} \|x_i - x'_i\|^2} \leq \epsilon \tag{57}$$

Note that the number of ϵ-neighbors of a ϵ-box is at most $(2\sqrt{d} + 1)^d$. We also have the following observation:

▶ **Observation 45.** *Let $p_1, p_2 \in [0, L]^d$ be any two distinct points.*

- *If p_1 and p_2 are in the same ϵ-box, then $\Delta(p_1, p_2) \leq \epsilon$.*
- *If $\Delta(p_1, p_2) \leq \epsilon$, then $g(\text{id}(p_1))$ must be an ϵ-neighbor of $g(\text{id}(p_2))$.*

To solve CP_ϵ with quantum walk, we need data structures to keep track of the pairs that have distance at most ϵ. The desired data structure should have size $\widetilde{O}(n^{2/3})$, insertion/deletion time $O(\log n)$, and one should be able to check whether there exist pairs of distance at most ϵ in time $O(\log n)$. In addition, as pointed out in [5], the data structure should have the following two properties:

[VII] The diagonal length of an ϵ-box is ϵ.

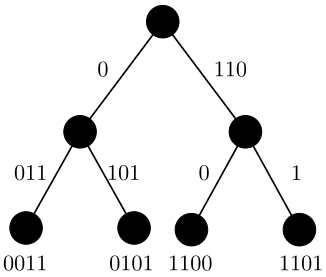


Figure 2 The uniquely represented radix tree that stores the keys $\{0011, 0101, 1100, 1101\}$.

- the data structure should have the bounded *worst-case* performance rather than *average-case* performance;
- the representation of the data structure should be history-independent, i.e., the data is uniquely represented regardless of the order of insertions and deletions.

We need the first property since the data structure may take too long for some operations, and this is not acceptable. The second property is required because, otherwise, the interference of quantum states would be messed up. In [5], a hash table and a skip list is used to for solving the element distinctness problem using quantum walks. In [11], a simpler data structure, namely, a radix tree, is used to achieve the same performance. More details of using a radix tree to solve the element distinctness can be found in [28]. Similar to the quantum data structure model in [5, 11, 28], we need the *quantum random access gate* to efficiently access data from a quantum memory, whose operation is defined as:

$$|i, b, z_1, \ldots, z_m\rangle \mapsto |i, z_i, z_1, \ldots, z_{i-1}, b, z_{i+1}, z_m\rangle\,, \tag{58}$$

where $|z_1, \ldots, z_m\rangle$ is some data in a quantum memory with m qubits. We assume this operation takes $O(\log m)$ time.

In the remainder of this section, we present two quantum algorithms for solving CP_ϵ. The data structures of both versions are based on the *augmented radix tree*, which we discuss in detail in the following subsection.

4.1 Radix tree for at most one solution

The purpose of the augmented radix tree is to quickly locate the points in an ϵ-box given its id. An ordinary radix tree is a binary tree that organizes a set of keys which are represented as binary strings. Each edge is labeled by a substring of a key and each leaf is labeled by a key such that concatenating all the labels on the path from the root to a leaf yields the key for this leaf. In addition, for each internal node, the labels of the two edges connecting to two children start with different bit. Note that in this definition, we implicitly merge all internal nodes that have only one child. The radix tree is uniquely represented for any set of keys. An example of a radix tree is shown as Figure 2.

Our basic radix tree is essentially the one in [11, 28] with modification on the nodes' internal structure. We highlight the extra information stored in the radix tree. First we use a *local counter* to store the number of points in this ϵ-box; second, we use a *flag* in each leaf node to indicate whether there is a point in this ϵ-box that is in some pair with distance at most ϵ. The flag bit in an internal node is the OR of the ones in its children. The local counter in each internal node is the sum of the local counters in its children. We also store at

most two points that are in the ϵ-box corresponding to this node. More precisely, let S be a subset of indices of the input points. We use $\tau(S)$ to denote the radix tree associated with S. Then, $\tau(S)$ consists of at most $r\lceil \log r \rceil$ nodes. Each node consists of the following registers:

$$\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{C} \times \mathcal{F} \times \mathcal{P}_1 \times \mathcal{P}_2, \tag{59}$$

where $\mathcal{D}$ stores the id of an ϵ-box for a leaf (and a substring of an id for an internal node) using $O(d \log(L/\epsilon))$ bits. $\mathcal{M}_1, \mathcal{M}_2$, and $\mathcal{M}_3$ use $O(\log n)$ bits to store the pointers to its parent, left child, and right child, respectively as well as the labels of the three edges connecting them to this node, $O(\log n)$ bits to store the labels of the three edges incident to it. $\mathcal{C}$ uses $O(\log n)$ bits to store the local counter. $\mathcal{F}$ stores the flag bit. $\mathcal{P}_1$ and $\mathcal{P}_2$ stores the coordinates of at most two points in this ϵ-box, which takes $O(d \log L)$ bits. The two points are stored in ascending order of their indices.

We need to pay attention to the layout of $\tau(S)$ in memory. We use three times more bits than needed to store $\tau(S)$, this will ensure that there are always more than $1/3$ of the bits that are free. We divide the memory into cells where each cell is large enough to store one leaf node of $\tau(S)$. Besides $\tau(S)$, we also store a bitmap $\mathcal{B}$, which takes $O(\log n)$ bits to encode the current free cells (with "1" indicating occupied and "0" indicating free). To make the radix tree history-independent, we use a quantum state which is the uniform superposition of basis states $|\tau(S), B\rangle$ for all possible valid layout of $\tau(S)$ and it corresponds to the bitmap $\mathcal{B}$.

Insertion and deletion from $\tau(S)$ takes $O(\log n)$ time. Checking the presence of an ϵ-close pair takes constant time – we just need to read the flag bit in the root. Preparing the uniform superposition of all $i \in S$ can be done in $O(\log n)$ time by performing a controlled-rotation on each level of the radix tree where the angles are determined by the local counters in the two children of a node.

In the following subsections, we present the two versions of our algorithms. The first version invokes the quantum walk framework only once and its data structure maintains the existence of an ϵ-close pair. The second version uses a much simpler data structure, but it is only capable of handling CP_ϵ with a unique solution. Hence it requires invoking the quantum walk framework multiple times to solve the general CP_ϵ. These two quantum algorithms have almost the same time complexity.

4.2 Single-shot quantum walk with complicated data structure

To handle multiple solutions, our data structure is a composition of an augmented radix tree, a hash table, and a skip list. We give a high-level overview of our data structure as follows. Recall that by the discretization of the space into ϵ-boxes, it is possible that a pair of points in different ϵ-boxes have distance at most ϵ, but one only needs to check $(2\sqrt{d}+1)^d$ ϵ-neighbors to detect such a case. We maintain a list of points for each nonempty ϵ-box in an efficient way. A hash table is used to store the tuple (i, p_i) which is used to quickly find the point p_i, given its index i. The points are also stored in a skip list for each nonempty ϵ-box, ordered by its index i, which allows for quick insertion and deletion of points. Each ϵ-box is encoded into a unique key, and a radix tree is used to store such key-value pairs, where the value is associated with a skip list. The flag bits in this radix tree maintain the presence of an ϵ-close pair.

In the following, we present the details of the data structure and show it has all the desired properties.

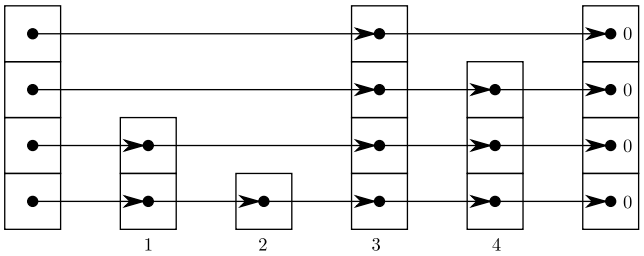


Figure 3 An example of a skip list that stores $\{1, 2, 3, 4\}$.

Hash table

The hash table we use is almost the same as the one used in [5], except that we do not store the $\lfloor \log r \rfloor$ counters in each bucket to facilitate the diffusion operator (which is handled easily here in the quantum walk on a Johnson graph). Our hash table has r buckets, where each bucket contains $\lceil \log n \rceil$ entries. We use a fixed hash function $h(i) = \lfloor ir/n \rfloor + 1$ to hash $\{1, \ldots, n\}$ to $\{1, \ldots, r\}$. That is, for $j \in [r]$, the j-th bucket contains the entries for (i, p_i) in ascending order of i, where $i \in S$ and $h(i) = j$.

The entry for (i, p_i) contains the tuple (i, p_i) and $\lceil \log n \rceil + 1$ pointers to other entries. These pointers are used in the skip list which we will describe below. The memory size of each entry is hence $O(\log^2 n + d \log L)$ and there are $O(r \log n)$ entries. Therefore, the hash table uses $O(rd \log^3(n + dL))$ qubits.

It is possible that more than $\lceil \log n \rceil$ points are hashed into the same bucket. However, as shown in [5], this probability is small.

Skip list

The skip list we use closely follows that in [5], except that the elements p_i in our skip list is ordered by its index i. We construct a skip list for each ϵ-box containing at least one point to store the points in it. For each $i \in S$, p_i belongs to exactly one skip list. Also, for $i \in S$, we randomly assign a level $\ell_i \in [0, \ldots, \ell_{\max}]$ where $\ell_{\max} = \lceil \log n \rceil$. The skip list associated with a ϵ-box has $\ell_{\max} + 1$ lists, where the level-ℓ list consists of all $i \in S$ such that $\ell_i \geq \ell$ and p_i is in this ϵ-box. Hence, the level-0 list consists of all $i \in S$ for p_i in this ϵ-box. Each element of the level-ℓ list has a specific pointer to the next element in this level, or to 0 if there is no next element. Each skip list contains a start entry that does not contain any (i, p_i) information but $\ell_{\max} + 1$ pointers to the first element of the each level. This start entry is stored in a leaf node of the augmented radix tree (which we will describe below) corresponding to this ϵ-box. In each skip list, we do not allocate memory for each node. Instead, each pointer is pointing to an entry of the hash table. The pointers are stored in the hash table (for the internal entries of each level) and in the radix tree (for the start entry). An example of a skip list is shown in Figure 3.

Given $i \in S$, we can search for p_i as follows. We start from the start entry of the level-$\ell_{\max}$ list and traverse each element until we find the last element $j_{\ell_{\max}}$ such that $j_{\ell_{\max}} < i$. Repeat this for levels $\ell_{\ell_{\max}-1}, \ldots, \ell_0$ and at each level start from the element that ended the previous level. At level-0, we obtain the element j_0. Then, the next element of j_0 is where p_i should be located (if it is stored in this skip list) or be inserted.

Each $i \in S$ is randomly assigned a level ℓ_i at the beginning of computation that does not change during the computation. More specifically, $\ell_i = \ell$ with probability $1/2^{\ell+1}$ for $\ell < \ell_{\max}$ and with probability $1/2^{\ell_{\max}}$ for $\ell = \ell_{\max}$. This can be achieved using $\ell_{\max}$

hash functions $h_1, \ldots, h_{\ell_{\max}} : [n] \to \{0,1\}$. In this way, each $i \in [n]$ has level $\ell < \ell_{\max}$ if $h_1(i) = \cdots = h_\ell(i) = 1$ but $h_{\ell+1}(i) = 0$; and it has level $\ell_{\max}$ if $h(i) = \cdots = h_{\ell_{\max}}(i) = 1$. In this quantum algorithm, we use an extra register to hold the state $|h_1, \ldots, h_{\ell_{\max}}\rangle$ which is initialized to a uniform superposition of all possible such functions from a d-wise independent family of hash functions (see [5, Theorem 1]) for $d = \lceil 4 \log n + 1 \rceil$. During the execution of the quantum algorithm, a hash function from the hashing family is chosen depending on the state in this register.

At first glance, the skip list has the same role as the hash table – finding p_i given index i. However, they have very different purposes in our algorithm. Recall that each nonempty ϵ-box is associated with a skip list, which is used to quickly insert and delete a point in this ϵ-box. The number of points in this ϵ-box can be as small as one and as large as r (in the extreme case where all the points are in the same ϵ-box). Hence, we cannot afford to have a fixed length data structure (such as a hash table or a sorted array) to store these points. In addition, to support quick insertion and deletion, a skip list is a reasonable choice (against an ordinary list). The purpose of the hash table can be viewed as a uniquely represented memory storing all the r points that can be referred to by the skip lists.

Augmented radix tree

We augment the radix tree described in Section 4.1 to handle multiple solution. In this augmented radix tree, we do not need the registers $\mathcal{P}_1$ and $\mathcal{P}_2$. Instead, we use $\lceil \log n \rceil$ pointers $\mathcal{L}_1, \ldots, \mathcal{L}_{\lceil \log n \rceil}$ as the start entry of a skip list. These pointers uses $O(\log^2 n)$ bits. In addition, we use an *external counter* in the leaf nodes to record whether there is a point in other ϵ-boxes that is at most ϵ-away from a point in this ϵ-box, which uses $O(\log n)$ bits. More formally, let $\tau'(S)$ be the augmented radix tree associated with S. Each node of $\tau'(S)$ consists of the following registers

$$\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{E} \times \mathcal{C} \times \mathcal{F} \times \mathcal{E} \times \mathcal{L}_1 \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}. \tag{60}$$

Next, we present how to perform the required operations on S with our data structure.

Checking for ϵ-close pairs

To check the existence of an ϵ-close pair, we just read the flag in the root of the radix tree. If the flag is set, there is at most one ϵ-close pair in S, and no such pairs otherwise. This operation takes $O(1)$ time.

Insertion

Given (i, p_i), we perform the insertion with the following steps:

1. Insert this tuple into the hash table.
2. Compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to. Denote this ϵ-box by $g(\mathrm{id}(p_i))$.
3. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau'(S)$, if so, insert i into the skip list corresponding to $g(\mathrm{id}(p_i))$; otherwise, first create a uniform superposition of the addresses of all free cells into another register, then create a new tree node in the cell determined by this address register and insert it into the tree. The pointers for the start entry of the skip list is initially set to 0. Insert i into this skip list. Let $\tau'(S, g(\mathrm{id}(p_i))$ denote the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$.
4. Increase the local counter $\mathcal{C}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.
5. Use Procedure 1 to update the external counters $\mathcal{E}$ and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in the leaf nodes corresponding to the neighbor ϵ-boxes of $g(\mathrm{id}(p_i))$.

Note that the first step takes at most $O(\log n)$ time. The second step can be done in $O(d)$ time. In Procedure 1, the number of ϵ-neighbors to check is at most $(2\sqrt{d}+1)^d$.

Procedure 1 Updating nodes for insertion.

input : (i, p_i), The leaf node in $\tau'(S)$ corresponding to the ϵ-box $g(\mathrm{id}(p_i))$, denoted by, $\tau'(S, g(\mathrm{id}(p_i)))$.

if *the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, \mathrm{id}(p_i))$ **then**
  for *all* ϵ*-box* g' *that is a* ϵ*-neighbor (see Definition 44) of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, g')$ *and the distance between* p_i *and the point in* g' *is at most* ϵ **do**
    Increase the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1;
    Increase the external counter $\mathcal{E}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1;
    if *the external counter* $\mathcal{E}$ *in* $\tau'(S, g')$ *was increased from 0 to 1* **then**
      Set the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
  if *the external counter* $\mathcal{E} > 1$ *in* $\tau'(S, g(\mathrm{id}(p_i)))$ **then**
    Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
    Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  end
else if *the local counter* $\mathcal{C} = 2$ *in* $\tau'(S, \mathrm{id}(p_i))$ **then**
  Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
  Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g(\mathrm{id}(p_i)))$ to the root of $\tau'(S)$;
  Set the external counter $\mathcal{E} = 0$ in $\tau'(S, \mathrm{id}(p_i))$;
  Let i' be the other index (than i) stored in the skip list corresponding to $g(\mathrm{id}(p_i))$;
  for *all* ϵ*-box* g' *that is a* ϵ*-neighbor of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, g')$ *and the distance between* $p_{i'}$ *and the point in* g' *is at most* ϵ **do**
    Decrease the external counter of $\tau'(S, g')$ by 1;
    if *the external counter* $\mathcal{E}$ *in* $\tau'(S, g')$ *was decreased from 1 to 0* **then**
      Unset the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
end

To obtain a uniform superposition of the addresses of all free cells, we first create a uniform superposition of all possible addresses to access to the bitmap $|\mathcal{B}\rangle$. We also use an auxiliary register that is initialized to $|0\rangle$. Then, the quantum random access gate defined in Equation (58) is applied on the register holding the uniform superposition of all addresses, the auxiliary register, and the bitmap register, which is effectively a SWAP operation on the second register and the corresponding bit in $|\mathcal{B}\rangle$. The auxiliary register remains $|0\rangle$ if and

only if the address in the first register is free. Since the size of memory space is chosen so that the probability of seeing a free cell is at least 1/3 (and we know exactly this probability based on how many cells have already been used), we can add an extra register and apply a one-qubit rotation to make the amplitude of the second register being $|0\rangle$ exactly 1/2. Hence, using *one* iteration of the oblivious amplitude amplification (which is a generalized version of Grover's search algorithm. See [12] and [34]) with the second register being the indicator, we obtain the uniform superposition of the addresses of all free cells. This cost if $O(\log n)$.

In [5], it was shown that with high probability, insertion into the skip list can be done in $O(d + \log^4(n + L))$ time. Hence, with high probability, the insertion costs $O(d + \log^4(n + L) + d(2\sqrt{d}+1)^d)$ time, where $O(d(2\sqrt{d}+1)^d)$ is the time for checking neighbors. To further reduce the running time, we can just stop the skip list's insertion and deletion procedures after $O(d + \log^4(n + L))$ time, which only causes little error (see Lemma 47).

Deletion

Given (i, p_i), we perform the following steps to delete this tuple from our data structure.

1. Compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to, and denote this ϵ-box by $g(\mathrm{id}(p_i))$.
2. Using $\mathrm{id}(p_i)$ as the key, we find the leaf node in $\tau'(S)$ that is corresponding to $g(\mathrm{id}(p_i))$.
3. Remove i from the skip list, and decrease the local counter $\mathcal{C}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.
4. Use Procedure 2 to update the external counters $\mathcal{E}$ and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in leaf nodes corresponding to the neighbor ϵ-boxes of $g(\mathrm{id}(p_i))$.
5. If the local counter $\mathcal{C} = 0$ in this leaf node, remove $\tau'(S, g(\mathrm{id}(p_i)))$ from $\tau'(S)$, and update the bitmap $\mathcal{B}$ in $\tau'(S)$ that keeps track of all free memory cells.
6. Remove (i, p_i) from the hash table.

Note that the first step can be done in $O(d)$ time. The second step can be done in $O(\log n)$ time. Procedure 2 has the same time complexity with Procedure 1. Hence, the cost for the deletion procedure is the same as that for insertion.

Finding a ϵ-close pair

We just read the flag in the root of the radix tree and then go to a leaf whose flag is 1. Check the local counter $\mathcal{C}$ of the node. if it is at least 2, output the first two elements in skip list. Otherwise, we find the ϵ-neighbor of the current node whose $C = 1$ and then output the points in that ϵ-neighbor and the current node.

Uniqueness

The uniqueness of our data structure follows from the analysis of [5, 11, 28]. More specifically, the hash table is always stored in the same way, as each $i \in S$ is stored in the same bucket for the fixed hash function and in each bucket, elements are stored in ascending order of i. The skip list is uniquely stored once the hash functions $h_1, \ldots, h_{\ell_{\max}}$ is determined. The shape of the radix tree is unique for S, but each node can be stored in different locations in memory. We use a uniform superposition of all possible memory organizations (by keeping track of the bitmap for free cells) to keep the quantum state uniquely determined by S.

Correctness

In the following, we argue that our data structure has the desired properties. First, we prove the correctness.

Procedure 2 Updating nodes for deletion.

input : (i, p_i), The leaf node in $\tau'(S)$ corresponding to the ϵ-box $g(\mathrm{id}(p_i))$, denoted by, $\tau'(S, g(\mathrm{id}(p_i)))$.

if *the local counter* $\mathcal{C} = 0$ *in* $\tau'(S, \mathrm{id}(p_i))$ **then**
  Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
  Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  Set the external counter $\mathcal{E} = 0$ in $\tau'(S, \mathrm{id}(p_i))$;
  for *all* ϵ*-box* g' *that is a* ϵ*-neighbor (see Definition 44) of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, g')$ *and the distance between* p_i *and the point in* g' *is at most* ϵ **do**
    Decrease the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1;
    if *the external counter* $\mathcal{E}$ *in* $\tau'(S, g')$ *was decreased from 1 to 0* **then**
      Unset the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
else if *the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, \mathrm{id}(p_i))$ **then**
  Let i' be the only index stored in the skip list corresponding to $g(\mathrm{id}(p_i))$;
  for *all* ϵ*-box* g' *that is a* ϵ*-neighbor of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C} = 1$ *in* $\tau'(S, g')$ *and the distance between* $p_{i'}$ *and the point in* g' *is at most* ϵ **do**
    Increase the external counter $\mathcal{E}$ of $\tau'(S, g')$ by 1;
    Increase the external counter $\mathcal{E}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1;
    if *the external counter* $\mathcal{E}$ *in* $\tau'(S, g')$ *was increased from 0 to 1* **then**
      Set the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
  if *the external counter* $\mathcal{E} = 0$ *in* $\tau'(S, g(\mathrm{id}(p_i)))$ **then**
    Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
    Update the flag $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  end
end

▶ **Lemma 46.** *The flag bit in the root of* $\tau'(S)$ *is set if and only if there exist distinct* $i, j \in S$ *such that* $|p_i - p_j| \leq \epsilon$.

Proof. We show that after each insertion and deletion, the data structure maintains the following conditions, and then lemma follows.

1. The flag bit of each leaf node of $\tau'(S)$ is set if and only if either its local counter is at least 2, or its external counter is at least 1.
2. The external counter of a leaf node $\tau'(S, g')$ is nonzero if and only if g' contains only one point p, and there exists another p' in another ϵ-box g'' such that $|p - p'| \leq \epsilon$.

It is easy to check that the first condition is maintained for each insertion and deletion. We show the second condition holds during insertions and deletion. For insertions, we consider the first case where a point p is just inserted into the ϵ-grid g', and p is its only point. The first for-loop in Procedure 1 updates other ϵ-boxes that have only one point to maintain the second condition. We consider the second case where g' already contains p' and p is just inserted, then the external counter in g' should be 0, and the second for-loop in Procedure 1 updates other ϵ-boxes that have only one point using the information of p'. This maintains the second condition. For deletions, there are also two cases. First, consider p is the only one point in g' and it is just deleted. We use the first for-loop in Procedure 2 to update the ϵ-boxes that has only one point using the information of p to maintain the second condition. Second, there is another point p' left in g' after deleting p. In this case, we start to check the external counter in g'. We use the second for-loop in Procedure 2 to check other ϵ-boxes that have only one point using the information of p' and update the corresponding external counter to maintain the second condition. ◀

Imperfection of the data structures and error analysis

Our data structure is not perfect. As indicated by Ambainis [5], there are two possibilities that it will fail. First, the hash table might overflow. Second, it might take more that $\lceil \log n \rceil$ time to search in a skip list. To resolve the first problem, we fix the number of entries in each bucket to be $\lceil \log n \rceil$ and treat any overflow as a failure. For the second problem, we stop the subroutine for accessing the skip list after $O(\log n)$ steps, and it causes an error in some cases. The original error analysis can be directly used in our case, as our hash table doesn't change the structure or the hash function, and our skip lists can be viewed as breaking the skip list in [5] into several pieces (one for each nonempty ϵ-box), and each insertion/deletion only involves one of them. Hence, the discussion in [5, Section 6] can be directly adapted to our case:

▶ **Lemma 47** (Adapted from [5]). *Let $|\psi\rangle$ be the final state of our algorithm (with imperfect data structures) and $|\psi'\rangle$ be the final state with the perfect data structure. Then $\| |\psi\rangle - |\psi'\rangle \| \leq O(1/\sqrt{n})$.*

Sketch of proof. There are two places where the data structure may give error: first, the hash table may have overflow, and second, the algorithm cannot find the required element in the skip lists in the desired time. The original proof showed that the probability that any of these errors happens is negligible, and thus the two-norm distance between $|\psi\rangle$ and $|\psi'\rangle$ can be bounded. Here, our data structure combining hash table, skip list, and radix tree, only has errors from hash tables and skip lists. The radix tree which has no error can be viewed as applying additional unitaries on $|\psi\rangle$ and $|\psi'\rangle$, and this does not change the distance between the two states. Since the probability that the errors from hash tables and skip lists happen are negligible by following the same analysis in [5], we can conclude that the two-norm distance between $|\psi\rangle$ and $|\psi'\rangle$ is small. ◀

Time complexity for CP_ϵ

We use the quantum walk framework reviewed in Section 2.5 to solve CP_ϵ. We first build the Johnson graph for CP_ϵ, which is similar to that for ED in Section 2.5. The vertices of the Johnson graph are $S \subseteq [n]$ with $|S| = n^{2/3}$ and S is marked if there exist distinct $i, j \in S$ such that $\Delta(p_i, p_j) \leq \epsilon$. We use $|S, d(S)\rangle$ to represent the quantum state corresponding to S, where $d(S)$ is the data structure of S defined in Section 4.1. Consider the three operations:

- **Steup:** with cost S, preparing the initial state

$$|\pi\rangle = \frac{1}{\sqrt{\binom{n}{n^{2/3}}}} \sum_{S\subseteq[n]:|S|=n^{2/3}} \sqrt{\pi_S}\,|S, d(S)\rangle\,. \tag{61}$$

- **Update:** with cost U, applying the transformation

$$|S, d(S)\rangle\,|0\rangle \mapsto |S, d(S)\rangle \sum_{S'\subseteq[n]:|S\cap S'|=n^{2/3}-1} \sqrt{p_{SS'}}\,|S', d(S')\rangle\,, \tag{62}$$

  where $p_{SS'} = \frac{1}{n^{2/3}(n-n^{2/3})}$.
- **Checking:** with cost C, applying the transformation:

$$|S, d(S)\rangle \mapsto \begin{cases} -\,|S, d(S)\rangle & \text{if } S \in M \\ |S, d(S)\rangle & \text{otherwise,} \end{cases} \tag{63}$$

  where M is the set of marked states.

We have the following result.

▶ **Theorem 48.** *There exists a quantum algorithm that with high probability solves* CP_ϵ *with time complexity* $O(n^{2/3}(d + \log^4(n+L) + d(2\sqrt{d}+1)^d))$.

Proof. The Johnson graph has $\lambda \geq 1/n^{2/3}$ and the Markov chain has spectral gap $\delta \geq 1/n^{2/3}$. For the setup operation, $\mathsf{S} = O(n^{2/3}(d + \log^4(n+L) + d(2\sqrt{d}+1)^d))$, since preparing the uniform superposition for all $|S\rangle$ costs $O(\log n)$ Hadamard gates and we need to do $n^{2/3}$ insertions to prepare the data structure. Each insertion costs $O(d+\log^4(n+L)+d(2\sqrt{d}+1)^d)$. For the update operation, we can implement the quantum walk operator as described in [28]: we use $|S, d(S)\rangle\,|i,j\rangle$ to represent $|S, d(S)\rangle\,|S', d(S')\rangle$ where S' is obtained from S by adding index i and removing index j. Then the diffusion can be implemented by preparing a uniform superposition of all $i \in S$ and a uniform superposition of all $j \notin S$, which takes time $O(\log n)$, and the "SWAP" operation can be implemented by a unitary that maps $|S, d(S)\rangle\,|i,j\rangle$ to $|S', d(S')\rangle\,|j,i\rangle$. In this way, the update operation uses $O(1)$ insertion and deletion to construct $d(S')$ from $d(S)$, and hence $\mathsf{U} = O(d + \log^4(n+L) + d(2\sqrt{d}+1)^d)$. The checking operation can be done in $O(1)$ time with the data structure. Therefore, by Lemma 22, the time complexity is $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})) = O(n^{2/3}(d + \log^4(n+L) + d(2\sqrt{d}+1)^d))$. ◀

By Lemma 42, we have the following corollary.

▶ **Corollary 49.** *There exists a quantum algorithm that with high probability solves CP with time complexity* $O(n^{2/3} \cdot (d + \log^4(n+L) + d(2\sqrt{d}+1)^d) \cdot (m + \log d))$.

▶ Remark 50. For $d = O(1)$ dimension and $m = O(\log n)$ digits of each coordinate of the points, the running time of the single-shot quantum algorithm is $O(n^{2/3} \cdot \log^5 n)$.

4.3 Multiple-trial quantum walks with simple data structure

In the previous subsection, we provide a quantum algorithm which solves CP_ϵ in time $O(n^{2/3}(d + \log^4 n + d(2\sqrt{d}+1)^d))$, where the logarithmic cost is mainly from the cost of the skip list. In this section we present a quantum algorithm which only requires the radix tree, and thus improve the running time. The caveat is that, with only the radix tree data structure, the insertion would fail if there are more than one ϵ-close pairs. As a result,

we need to keep shrinking the size of the problem using [5, Algorithm 3] until there is a unique solution with high probability, and then run the $\widetilde{O}(n^{2/3})$ quantum algorithm for this unique-solution case.

In the following, we first show how to solve the unique-solution CP_ϵ, and then show the reduction from the multiple-solution case to the unique-solution case.

▶ **Lemma 51.** *There exists a quantum algorithm that with high probability solves the unique-solution* CP_ϵ *with time complexity* $O(n^{2/3}(\log n + d(2\sqrt{d}+1)^d))$.

Data structure for unique-solution

We use the radix tree $\tau(S)$ for S defined in Section 4.1. In the following, we describe the necessary operations on $\tau(S)$.

Checking for ϵ-close pair

To check the existence of an ϵ-close pair, we just read the flag bit in the root of $\tau(S)$, which takes $O(1)$ time.

Insertion

Given (i, p_i), we perform the following steps for insertion. First compute the id, $\mathrm{id}(p_i)$, of the ϵ-box which p_i belongs to. Denote this ϵ-box by $g(\mathrm{id}(p_i))$. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau(S)$. There are two cases:

- $\mathrm{id}(p_i)$ is already in $\tau(S)$: insert p_i into $\tau(S, g(\mathrm{id}(p_i)))$, increase the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1 and also set the flag. Then update the flag and local counter of the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ to the root.
- $\mathrm{id}(p_i)$ is not in $\tau(S)$: create a new leaf node for $\mathrm{id}(p_i)$ and insert it into $\tau(S)$. Insert p_i into this new leaf node, and increase the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1. Then, check the ϵ-neighbors g' of $\tau(S, g(\mathrm{id}(p_i)))$ that contains only one point p' and set both flags if p_i is ϵ-close to p', and in this case, update the flag bit and local counter on the nodes along the paths from $\tau(S, g(\mathrm{id}(p_i)))$ and g'.

Deletion

Given (i, p_i), we first compute the id, $\mathrm{id}(p_i)$ of the ϵ-box that p_i belongs to, and locate the corresponding leaf node $\tau(S, g(\mathrm{id}(p_i)))$. Decrease the local counter in $\tau(S, g(\mathrm{id}(p_i)))$ by 1 and update the local counter in the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ to the root. Check the number of points stored in $\tau(S, g(\mathrm{id}(p_i)))$. There are two possibilities:

- There are two points in $\tau(S, g(\mathrm{id}(p_i)))$: unset the flag in $\tau(S, g(\mathrm{id}(p_i)))$ and update the flag bit in the nodes along the path to the root and delete p_i from $\tau(S, g(\mathrm{id}(p_i)))$.
- p_i is the only point in $\tau(S, g(\mathrm{id}(p_i)))$: check the ϵ-neighbors g' of $\tau(S, g(\mathrm{id}(p_i)))$ that contains only one point p' and unset both flags if p_i is ϵ-close to p', and in this case, update the flag bit on the nodes along the path from $\tau(S, g(\mathrm{id}(p_i)))$ and g' to the root. Delete p_i from $\tau(S, g(\mathrm{id}(p_i)))$ and delete $\tau(S, g(\mathrm{id}(p_i)))$ from $\tau(S)$.

As in Section 4.2, we use a bitmap register $|\mathcal{B}\rangle$ to keep track of the free cells in $\tau(S)$. For insertion, we maintain a uniform superposition of all possible free cells to insert a new radix tree node. For deletion, we update the bitmap $|\mathcal{B}\rangle$ accordingly. This ensures the uniqueness of the quantum data structure.

The correctness of the data structure is straightforward, and the time complexity is $O(\log n + d(2\sqrt{d}+1)^d)$ for both insertion and deletion. Also, preparing a uniform superposition for all $i \in S$ costs $O(\log n)$ using the local counter in each node. By a similar analysis of Theorem 48, we prove Lemma 51 as follows.

Proof of Lemma 51. The algorithm uses the framework in Lemma 22 with the data structure we just described in this subsection, where $\mathsf{U} = O(\log n + d(2\sqrt{d}+1)^d))$, $\mathsf{C} = O(1)$ and $\mathsf{S} = O(n^{2/3}(\log n + d(2\sqrt{d}+1)^d))$. Therefore, the running time of the algorithm is as claimed. ◀

Next, we show how to reduce multiple-solution CP_ϵ to unique-solution CP_ϵ. A high-level overview of Ambainis's reduction in [5] is the following. We run the algorithm for unique-solution CP_ϵ several times on some random subsets of the given input. If the given subset contains solutions, then with constant probability there exists a subset which contains exactly one solution.

▶ **Definition 52** ([5, 27]). *Let $\mathcal{F}$ be a family of permutations on $f : [n] \to [n]$. $\mathcal{F}$ is ϵ-approximate d-wise independent if for any $x_1, \ldots, x_d \in [n]$ and for all $y_1, \ldots, y_d \in [n]$,*

$$\frac{1-\epsilon}{n\cdot(n-1)\cdot(n-d+1)} \leq \Pr\left[\bigwedge_{i=1}^{n} f_i(x_i) = y_i\right] \leq \frac{1+\epsilon}{n\cdot(n-1)\cdot(n-d+1)}. \tag{64}$$

▶ **Lemma 53** ([5, 27]). *Let n be an even power of a prime number. For any $t \leq n$, $\epsilon > 0$, there exists an ϵ-approximate t-wise independent family $\mathcal{F} = \{\pi_j | j \in [R]\}$ of permutations $\pi_j : [n] \to [n]$ such that:*

- $R = O\left(\left(n^{t^2}\cdot\epsilon^{-t}\right)^{3+o(1)}\right)$;
- *given i, j, $\pi_j(i)$ can be computed in time $O(t\log^2 n)$.*

The multiple-solution algorithm from [5] is as follows:

Algorithm 3 The algorithm for multiple ϵ-close pair.

input : Let (S, ϵ) be the input, and $|S| = n$.

Let $T_1 = S$ and $j = 1$;

while $|T_j| > n^{2/3}$ **do**

  Run the algorithm described in Lemma 51 on T_j, and Measure the final state. If there is a pair with distance less than ϵ, output the pair and stop ;

  Let q_j be an even power of a prime with $|T_j| \leq q_j \leq (1+\frac{1}{8})|T_j|$. Select a random permutation π_j on $[q_j]$ from the $\frac{1}{n}$-approximately $4\log n$-wise independent family of permutations as in Lemma 53 ;

  Let

$$T_{j+1} := \left\{\pi_1^{-1}\cdot\pi_2^{-1}\cdots\pi_j^{-1}(i), \quad i \in \left[\left\lceil\frac{4q_j}{5}\right\rceil\right]\right\}. \tag{65}$$

  $j \leftarrow j+1$;

end

If $|T_j| \leq n^{\frac{2}{3}}$, then run Grover's search algorithm on T_j for a pair with distance at most ϵ ;

We have the main result of this subsection:

▶ **Theorem 54.** *There exists a quantum algorithm that with high probability solves* CP_ϵ *with time complexity* $O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n) = O(n^{2/3} \cdot \log^4 n)$ *for* $d = O(1)$.

Proof. We prove the running time of the algorithm here. For the correctness, one can check [5] for the detail.

By Equation (65), the size of T_{j+1} will be at most

$$\frac{4}{5} \cdot (1 + \frac{1}{8})|T_j| = \frac{9}{10}|T_j|. \tag{66}$$

Therefore, the while-loop takes at most $O(\log n)$ iterations in the worst case. Let $n_j = |T_j|$ be the size of the instance in the j-th iteration. Then, the unique-solution algorithm in Procedure 3 runs in $O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d))$-time (Lemma 51), given an $O(1)$-time access to the set T_j. However, in Procedure 3 each element of the random permutation can be computed in time $O(\log^3 n)$ according to Lemma 53 with $t = 4 \log n$, which means the unique-solution algorithm will take $O(\log^3 n)$ time for each query to T_j. Note that we will not actually compute the whole set T_{j+1}, as shown in Procedure 3, which takes too much time. Hence, the running time for the j-th iteration is $O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d) \cdot \log^3 n)$. And the total running time for the while-loop is

$$\sum_{j=1}^{O(\log n)} O(n_j^{2/3} \cdot (\log n_j + d(2\sqrt{d}+1)^d) \cdot \log^3 n) \tag{67}$$

$$\leq O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n) \cdot \sum_{j=0}^{O(\log n)} \left(\frac{9}{10}\right)^{2j/3} \tag{68}$$

$$= O(n^{2/3} \cdot (\log n + d(2\sqrt{d}+1)^d) \cdot \log^3 n), \tag{69}$$

where the first inequality follows from $n_j \leq (\frac{9}{10})^{j-1} \cdot n$. Finally, Procedure 3 runs in time $O(n^{2/3} \log n)$. This completes the proof of the running time. ◀

To conclude the quantum algorithms for solving CP in constant dimension, we have the following corollary that is a direct consequence of either Theorem 48 or Theorem 54.

▶ **Corollary 55.** *For any* $d = O(1)$*, there exists a quantum algorithm that, with high probability, solves* $\mathsf{CP}_{n,d}$ *in time* $\widetilde{O}(n^{2/3})$.

4.4 Quantum lower bound for CP in constant dimensions

We can easily get an $\Omega(n^{2/3})$ lower bound for the quantum time complexity of CP in constant dimension by reducing the element distinctness problem (ED) to CP.

▶ **Theorem 56** (Folklore). *The quantum time complexity of* CP *is* $\Omega(n^{2/3})$.

Proof. We reduce ED to one dimensional CP by mapping the point i with value $f(i)$ in ED the point $f(i) \in \mathbb{R}$ in CP. If the closest pair has distance zero, we know there is a collision $f(i) = f(j)$. If the closest pair has distance greater or equal to one, we know there is no collision. Therefore, ED's $\Omega(n^{2/3})$ query lower bound by [1] translates into $\Omega(n^{2/3})$ time lower bound for CP. ◀

5 Bichromatic closest pair in constant dimensions

Classically, bichromatic closest pair problem is harder than the closest pair problem. In constant dimension, the best algorithms for the closest pair problem are "nearly linear", while the algorithm by [3] for bichromatic closest pair problem is "barely subquadratic", running in $O(n^{2-1/\Theta(d)})$-time. In quantum, we found that BCP is still harder than CP in constant dimension. In particular, we cannot adapt the quantum algorithm in previous section for solving BCP because the data structure cannot distinguish the points from two sets efficiently. We can only get a sub-linear time quantum algorithm for BCP using different approach, which is a quadratic speed-up for the classical algorithm.

Nevertheless, we show that we can find an approximate solution for BCP with multiplicative error $1+\xi$ with quantum time complexity $\widetilde{\Theta}(n^{2/3})$. The following theorem is a direct consequence of Theorems 64 and 67.

▶ **Theorem 57.** *For any fixed dimension and error ξ, there is a quantum algorithm which can find an approximate solution for* BCP *with multiplicative error $1+\xi$ in time $\widetilde{O}(n^{2/3})$. Moreover, all quantum algorithms which can find an approximate solution for* BCP *with arbitrary multiplicative error requires time $\Omega(n^{2/3})$.*

Similar to solving CP, we reduce BCP to its decision version of the problem, and then apply quantum algorithms to solve the decision problem. We define the decision problem as BCP_ϵ.

▶ **Definition 58** (BCP_ϵ)**.** *In BCP_ϵ, we are given two sets A, B of n points $\in \mathbb{R}^d$ and a distance measure Δ. The goal is to find a pair of points $a \in A$, $b \in B$ such that $\Delta(a,b) \leq \epsilon$ if it exists and returns* no *if no such pair exists.*

To address the approximate version of BCP, we also define the approximation version of BCP_ϵ as follows:

▶ **Definition 59** ($(1+\xi)$-BCP_ϵ)**.** *In $(1+\xi)$-BCP_ϵ, we are given two sets A, B of n points $\in \mathbb{R}^d$, a distance measure Δ, and ξ. The goal is to do the following*

1. *If there exists a pair of points $a \in A$, $b \in B$ such that $\Delta(a,b) \leq \epsilon$, output the pair (a,b).*
2. *If for all pairs of points $a \in A$, $b \in B$, $\Delta(a,b) > (1+\xi)\epsilon$, returns* no.

Again, we consider $\Delta(a,b) = \|a-b\|$ as the distance measure in this work. We show that BCP reduce to BCP_ϵ in time $O(m + \log d)$, where m is the number of digits of each coordinate and d is the dimension.

▶ **Lemma 60.** *Given an oracle $\mathcal{O}$ for $(1+\xi)$-BCP_ϵ, there exists an algorithm $A^{\mathcal{O}}$ that runs in time and query complexity $O(m + \log d)$ solves the $(1+\xi)$-*BCP*.*

Proof. Let (A, B, δ) be an instance of the $(1+\xi)$-BCP. We first pick an arbitrary pair $a_0 \in A, b_0 \in B$ and computes $\Delta(a_0, b_0)$. Then, we set ϵ to be $\Delta(a_0,b_0)/2$ and run the oracle $\mathcal{O}$ to check whether there exists a distinct pair which distance is less than $\Delta(a_0,b_0)/2$ or not. If there exists such a pair, which we denote as (a_1, b_1), then we set $\epsilon = \Delta(a_1, b_1)$ and call $\mathcal{O}$ to check again. If there is no such a pair, then we set $\epsilon = 3\Delta(a_0,b_0)/4$ and call $\mathcal{O}$. We continuously run this binary search for $m + \log d$ iterations. Finally, the algorithm outputs the bichromatic closest pair. ◀

In the subsections, we present a quantum algorithm for solving $(1+\xi)$-BCP and a quantum algorithm for exact BCP. To complement the algorithmic results, we also give quantum lower bound for BCP.

5.1 Quantum algorithm for $(1+\xi)$-BCP

The quantum algorithm is based on the quantum walk framework on a tensor product of Johnson graphs. To begin with, we define the Johnson graphs J_A and J_B for A and B, respectively. The vertices of J_A, denoted by X_A, is defined as the set $\{S \subseteq A : |S| = n^{2/3}\}$. There is an edge connecting S and S' if and only if $|S \cap S'| = n^{2/3} - 1$. The Markov chain M_A is defined on X_A with $p_{SS'} = \frac{1}{n^{2/3}(n-n^{2/3})}$ when S and S' are connected by an edge. The Johnson graph for J_B for B and its corresponding Markov chain can be defined similarly. The *tensor product* $M_A \otimes M_B$ is defined as the Markov chain based on $X_A \times X_B$ defined as

$$X_A \times X_B := \{(S_A, S_B) : S_A \in X_A, S_B \in X_B\}, \tag{70}$$

with transition probability

$$p_{(S_A,S_B)(S'_A,S'_B)} = p_{S_A S'_A} \cdot p_{S_B S'_B}. \tag{71}$$

A state (S_A, S_B) is marked if there exists a pair $a \in S_A$ and $b \in S_B$ such that $\Delta(a, b) \le \epsilon$.

Now, we examine the properties of $M_A \otimes M_B$. It is easy to see that $\lambda = \frac{\binom{n-1}{n^{2/3}-1}^2}{\binom{n}{n^{2/3}}^2} = \frac{1}{n^{2/3}}$. Let δ_A and δ_B be the spectral gap of M_A and M_B respectively. As a result of [6, Lemma 21.17], $\delta \ge \min\{\delta_A, \delta_B\} = \frac{1}{n^{2/3}}$. By Lemma 22, the cost for solving $(1+\xi)$-BCP$_\epsilon$ is $O(\mathsf{S} + n^{1/3}(n^{1/3}\mathsf{U} + \mathsf{C}))$, where S, U and C are the cost of quantum operations defined in Section 4.2. Before describing the data structure to achieve meaningful S, U, and C, we first introduce a finer discretization scheme. In Section 4, we used a hypergrid consisting of ϵ-boxes. Here, we discretize the space $[0, L]^d$ as a hypergrid consisting of $\frac{\xi\epsilon}{2\sqrt{d}}$-boxes. The following lemma guarantees that distance between a $\frac{\xi\epsilon}{2\sqrt{d}}$-box and its ϵ-neighbor is at most $(1+\xi)\epsilon$.

▶ **Lemma 61.** *Let g and g' be $\frac{\xi\epsilon}{2\sqrt{d}}$-boxes. If g and g' are ϵ-neighbors, then for all $p \in g$ and $p' \in g'$, $\Delta(p, p') \le (1+\xi)\epsilon$.*

Proof. Recall the definition of the id function in Equation (56). $\mathrm{id}(g)$ can be treated as a point, and we can measure the distance between $\mathrm{id}(g)$ and other points. The lemma can be proven via the triangle inequality:

$$\Delta(p, p') \le \Delta(p, \mathrm{id}(g)) + \Delta(\mathrm{id}(g), \mathrm{id}(g')) + \Delta(p', \mathrm{id}(g') \le \frac{\xi\epsilon}{2} + \epsilon + \frac{\xi\epsilon}{2} \le (1+\xi)\epsilon. \tag{72}$$

◀

In our algorithm, we need to search for all ϵ-neighbors that contain the other color to report an ϵ-close pair (with an multiplicative error ξ). It's easy to see that the number of neighbors of a box is bounded in terms of d and ξ:

▷ Claim 62. For each $\frac{\xi\epsilon}{2\sqrt{d}}$-box, the number of ϵ-neighbors is at most $\left(4\sqrt{d}/\xi + 1\right)^d$.

Based on this finer discretization scheme, we use the data structure defined in Section 4.2 but with simple modifications on the radix tree. Instead of using $\mathcal{L}_1, \ldots, \mathcal{L}_{\lceil \log n \rceil}$ as the start entry of the skip list, we use $\lceil \log n \rceil$ pointers for both sets A and B. We also need local counters $\mathcal{C}^A$ and $\mathcal{C}^B$ for both colors. Now, each node in the radix tree has the following registers:

$$\begin{aligned}&\mathcal{D} \times \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \times \mathcal{E}^A \times \mathcal{E}^B \times \mathcal{C}^A \times \mathcal{C}^B \times \\ &\mathcal{F} \times \mathcal{L}_1^A \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}^A \times \mathcal{L}_1^B \times \cdots \times \mathcal{L}_{\lceil \log n \rceil}^B.\end{aligned} \tag{73}$$

The points in A (or B, respectively) is organized by the skip list for A (or B, respectively). The insertion and deletion operations are similar to the data structure in Section 4.2, but in the procedure for updating the local and external counters and checking ϵ-neighbors, we need to consider points of the other color. We formally describe the two procedures as follows.

Insertion

Given a point (i, p_i, x), where $x \in \{A, B\}$ denotes the color. We perform the insertion with the following steps:

1. Insert this tuple into the hash table corresponding to x.
2. Compute the id, $\mathrm{id}(p_i)$, of the $\frac{\xi\epsilon}{\sqrt{d}}$-box which p_i belongs to and denote it by $g(\mathrm{id}(p_i))$.
3. Using $\mathrm{id}(p_i)$ as the key, check if this key is already in $\tau'(S)$, if so, insert i into the skip list for color x corresponding to $g(\mathrm{id}(p_i))$; otherwise, first create a uniform superposition of the addresses of all free cells into another register, then create a new tree node in the cell determined by this address register and insert it into the tree. The pointer for the start entry of the skip list is initially set to 0. Insert i into this skip list. Let $\tau'(S, g(\mathrm{id}(p_i))$ denote the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$.
4. Increase the local counter $\mathcal{C}^x$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.
5. Use Procedure 4 to update the external counters $\mathcal{E}^x, \mathcal{E}^{\bar{x}}$ (here $\bar{x}$ denotes the other color than x) and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$, the leaf nodes which are corresponding to the ϵ-neighbors of $g(\mathrm{id}(p_i))$, and their parent nodes.

Note that the first step takes at most $O(\log n)$ time. The second step can be done in $O(d)$ time. In Procedure 4, the number of ϵ-neighbors to check is at most $(\frac{4\sqrt{d}}{\xi} + 1)^d$ by Claim 62.

Deletion

Given (i, p_i, x), we perform the following steps to delete this tuple from our data structure.

1. Compute the id, $\mathrm{id}(p_i)$, of the $\frac{\epsilon\xi}{\sqrt{d}}$-box which p_i belongs to and denote it by $g(\mathrm{id}(p_i))$.
2. Using $\mathrm{id}(p_i)$ as the key, we find the leaf node in $\tau'(S)$ that is corresponding to $g(\mathrm{id}(p_i))$.
3. Remove i from the skip list for color x, and decrease the local counter $\mathcal{C}^x$ in $\tau'(S, g(\mathrm{id}(p_i)))$ by 1.
4. Use Procedure 2 to update the external counters $\mathcal{E}^x$ and $\mathcal{E}^{\bar{x}}$ (here $\bar{x}$ denote the other color than x) and flags $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$ as well as in leaf nodes corresponding to the ϵ-neighbors of $g(\mathrm{id}(p_i))$.
5. If both local counters $\mathcal{C}^x, \mathcal{C}^{\bar{x}}$ in this leaf node are 0, remove $\tau'(S, g(\mathrm{id}(p_i)))$ from $\tau'(S)$, and update the bitmap $\mathcal{B}$ in $\tau'(S)$ that keeps track of all free memory cells.
6. Remove (i, p_i, x) from the hash table.

Note that the first step can be done in $O(d)$ time. The second step can be done in $O(\log n)$ time. Procedure 5 has the same time complexity with Procedure 4. Hence, the cost for the deletion procedure is the same with that for insertion.

Checking for $(1 + \xi)\epsilon$-close pairs

To check the existence of an $(1 + \xi)\epsilon$-close pair, we just read the flag in the root of the radix tree. If the flag is set, there is at most one ϵ-close pair in S, and no such pairs otherwise. This operation takes $O(1)$ time.

Procedure 4 Updating nodes for insertion for the bichromatic case.

input : (i, p_i, x), the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$, denoted by $\tau'(S, g(\mathrm{id}(p_i)))$.

Let $\bar{x} \in \{A, B\}$ and $\bar{x} \neq x$;
if $\mathcal{C}^x = 1$ *in* $\tau'(S, \mathrm{id}(p_i))$ *and* $\mathcal{C}^{\bar{x}} = 0$ **then**
  for *all* ϵ*-neighbor* g' *(see Definition 44) of* $g(\mathrm{id}(p_i))$ *where* $\mathcal{C}^{\bar{x}} \geq 1$ *in* $\tau'(S, g')$ **do**
    Increase $\mathcal{E}^x$ of $\tau'(S, g')$ by 1;
    Increase $\mathcal{E}^{\bar{x}}$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1;
    if $\mathcal{E}^x$ *in* $\tau'(S, g')$ *was increased from 0 to 1* **then**
      Set the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
  if $\mathcal{E}^{\bar{x}} \geq 1$ *in* $\tau'(S, g(\mathrm{id}(p_i)))$ **then**
    Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
    Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  end
else if $\mathcal{C}^x = 1$ *and* $\mathcal{C}^{\bar{x}} \geq 1$ *in* $\tau'(S, g(\mathrm{id}(p_i)))$ **then**
  Set the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
  Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g(\mathrm{id}(p_i)))$ to the root of $\tau'(S)$;
  Set $\mathcal{E}^{\bar{x}} = 0$ in $\tau'(S, \mathrm{id}(p_i))$;
  for *all* g' *that is an* ϵ*-neighbor of* $g(\mathrm{id}(p_i))$ *where the the local counter* $\mathcal{C}^{\bar{x}} \geq 1$ *in* $\tau'(S, g')$ **do**
    Decrease $\mathcal{E}^x$ of $\tau'(S, g')$ by 1;
    if $\mathcal{E}^x$ *in* $\tau'(S, g')$ *was decreased from 1 to 0* **then**
      Unset the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
end

Finding a $(1+\xi)\epsilon$-close pair

We just read the flag in the root of the radix tree and then go to a leaf which flag is 1. Check the local counters of the node. If both local counters are at least 1, output the first elements in skip lists for A and the first element in the skip list for B. Otherwise, check the external counters. Suppose $\mathcal{E}^A$ is non-zero. Then we find the ϵ-neighbor of the current node whose $\mathcal{C}^B > 0$ and output the first point in the skip list of A of the current node and the first element in the skip list of B of the ϵ-neighbor.

We have the following result.

▶ **Theorem 63.** *For any fixed dimension and fixed* ξ*, there exists a quantum algorithm that, with high probability, can solve* $(1+\xi)$-BCP_ϵ *in time* $O(n^{2/3}(d + \log^4(n+L) + d(\frac{4\sqrt{d}}{\xi} + 1)^d))$.

Procedure 5 Updating nodes for deletion for the bichromatic case.

input : (i, p_i, x) from A, the leaf node in $\tau'(S)$ corresponding to $g(\mathrm{id}(p_i))$, which we denote as $\tau'(S, g(\mathrm{id}(p_i)))$.

Let $\bar{x} \in \{A, B\}$ and $\bar{x} \neq x$;
if $\mathcal{C}^x$ *and* $\mathcal{C}^{\bar{x}}$ *in* $\tau'(S, \mathrm{id}(p_i)) = 0$ **then**
  Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
  Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  Set $\mathcal{E}^x = 0$ and $\mathcal{E}^{\bar{x}} = 0$ in $\tau'(S, \mathrm{id}(p_i))$;
  for *all* g' *that is an* ϵ*-neighbor (see Definition 44) of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C}^{\bar{x}} \geq 1$ *in* $\tau'(S, g')$ **do**
    Decrease $\mathcal{E}^x$ of $\tau'(S, g')$ by 1;
    if $\mathcal{E}^x$ *in* $\tau'(S, g')$ *was decreased from 1 to 0* **then**
      Unset the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
else if $\mathcal{C}^x = 0$ *and* $\mathcal{C}^{\bar{x}} \geq 1$ **then**
  for *all* g' *that is an* ϵ*-neighbor of* $g(\mathrm{id}(p_i))$ *where the local counter* $\mathcal{C}^x \geq 1$ *in* $\tau'(S, g')$ **do**
    Increase $\mathcal{E}^{\bar{x}}$ of $\tau'(S, g')$ by 1;
    Increase $\mathcal{E}^x$ of $\tau'(S, g(\mathrm{id}(p_i)))$ by 1;
    if $\mathcal{E}^{\bar{x}}$ *in* $\tau'(S, g')$ *was increased from 0 to 1* **then**
      Set the flag $\mathcal{F}$ in $\tau'(S, g')$;
      Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, g')$ to the root of $\tau'(S)$;
    end
  end
  if $\mathcal{E}^x = 0$ *in* $\tau'(S, g(\mathrm{id}(p_i)))$ **then**
    Unset the flag $\mathcal{F}$ in $\tau'(S, g(\mathrm{id}(p_i)))$;
    Update the flags $\mathcal{F}$ in the nodes along the path from $\tau'(S, \mathrm{id}(p_i))$ to the root of $\tau'(S)$;
  end
end

Proof. The proof closely follows the analysis for Theorem 48, and the correctness of the data structure and the time complexity of its operations follow from the discussion in Section 4.2. Note that our algorithm will output a pair which belong to the same $\frac{\xi\epsilon}{2\sqrt{d}}$-box or two of them that are ϵ-neighbors. Based on Lemma 61, two points which corresponding hyercubes are ϵ-neighbors have distance at most $(1+\xi)\epsilon$. Therefore, our algorithm could output a pair of points which distance is at most $(1+\xi)\epsilon$. Another difference is that here we need to search at most $(4\sqrt{d}/\xi + 1)^d$ neighbors during insertions and deletions. As a result, $\mathsf{U} = O(d + \log^4(n+L) + d(4\sqrt{d}/\xi+1)^d)$, and $\mathsf{S} = O(n^{2/3}(d + \log^4(n+L) + d(4\sqrt{d}/\xi+1)^d)$. Again, $\mathsf{C} = O(1)$, $\delta \geq 1/n^{2/3}$, and $\lambda \geq 1/n^{2/3}$. Therefore, by Lemma 22, the total cost is $O(\mathsf{S} + \frac{1}{\sqrt{\lambda}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})) = O(n^{2/3}(d + \log^4(n+L) + (4\sqrt{d}/\xi+1)^d))$. ◀

By Lemma 60 and the above Theorem 63, we have the following theorem:

▶ **Theorem 64.** *For an fixed dimension and fixed ξ, there exists a quantum algorithm that, with high probability, can solve $(1+\xi)$-BCP in time $\widetilde{O}(n^{2/3})$.*

5.2 Quantum algorithm for solving BCP exactly

In this subsection, we present a quantum algorithm for solving BCP exactly. The main idea of this algorithm is to partition A into smaller subsets. Then we build data structures which support nearest-neighbor search on all subsets in superposition. We use the quantum minimum finding algorithm to find the smallest distances from B to each subset, among which we use the quantum minimum finding algorithm again to find the smallest distance.

Unlike the data structure for solving CP, the data structure for BCP does not have to be uniquely represented, as no insertion and deletion are performed in this algorithm. The data structure can have expected running time instead of the worst-case running time. The total worst-case running time can be bounded by standard techniques. The nearest-neighbor search data structure we use is from [19], and is reformulated in the following lemma.

▶ **Lemma 65** ([19]). *For any fixed dimension, there exists a data structure for n points in $\mathbb{R}^d$ that can be built in expected time complexity $O(n^{\lceil d/2\rceil+\delta})$ for arbitrarily small δ and the nearest-neighbor search can be performed in worst-case time complexity $O(\log n)$.*

This data structure is based on the Voronoi diagram and its triangulation in higher dimensions. Using this data structure, we have a quantum algorithm for solving BCP exactly, which yields the following theorem.

▶ **Theorem 66.** *There exists a quantum algorithm that, with high probability, solves BCP for dimension d with time complexity $\widetilde{O}\left(n^{1-\frac{1}{2d}+\delta}\right)$ for arbitrarily small δ.*

Proof. We first partition A into $\lceil n/r\rceil$ subsets $S_1,\ldots,S_{\lceil n/r\rceil}$, where $|S_i|=r$ for $i\in\left[\lceil n/r\rceil\right]$. (The value of r will be determined later). For all $i\in\left[\lceil n/r\rceil\right]$, we can find a closest pair between S_i and B as follows. First, a data structure as in Lemma 65 for S_i is built in expected time $O\left(r^{\lceil d/2\rceil+\delta}\right)$, which supports nearest-neighbor search in time $O(\log n)$. Then, we use the quantum minimum finding subroutine (Theorem 9) which uses the distance reported by the nearest-neighbor search as the oracle. The closest pair between S_i and B can be found in time complexity $\widetilde{O}(\sqrt{n})$. Note that the time complexity for building the data structure is not bounded for the worst case. However, using Markov's inequality, we know that with high probability, say, at least $9/10$, the time complexity is bounded by $O\left(r^{\lceil d/2\rceil+\delta}\right)$. Hence, fixing a constant $c\geq 10$, and stop the data structure construction after $c\cdot r^{\lceil d/2\rceil+\delta}$ steps. With at most $1/10$ probability, the construction will fail and this event can be detected by checking the solution returned by the quantum minimum finding subroutine. We run $O(\log n)$ instances of above procedure in parallel and use take the quantum minimum of all the $O(\log n)$ results. The probability that all these instances fail is at most $(1/10)^{O(\log n)}=O(1/n)$. We refer to the above procedure as the "inner search", and its time complexity is $O\left(r^{\lceil d/2\rceil+\delta}+\sqrt{n}\right)$.

Next, we use the distance of the output of the inner search as the oracle and perform another quantum minimum finding subroutine for $i\in\left[\lceil n/r\rceil\right]$. We refer to this procedure as the "outer search". The probability that the closest pair between A and B lies in S_i and B is r/n. As a result, the number of the oracle queries for the quantum minimum finding subroutine is $\widetilde{O}(\sqrt{n/r})$. The time complexity for each query is $O\left(r^{\lceil d/2\rceil+\delta}+\sqrt{n}\right)$. Therefore,

the total time complexity is $\widetilde{O}\big((r^{\lceil d/2\rceil+\delta}+\sqrt{n})\cdot\sqrt{n/r}\big)$. A simple calculation shows that this achieves the minimum (ignoring the δ term in the exponent) when $r = n^{1/d}/(d-1)^{2/d}$, which yields the total time complexity

$$\widetilde{O}\left(n^{1-\frac{1}{2d}+\delta}\right). \tag{74}$$

The failure probability for each query is at most $O(1/n)$. Therefore, the total failure probability is at most $O(\sqrt{n/r}/n) = O(n^{-(1/2-1/2d)})$ for $d > 1$, which can be smaller than any constant. ◀

5.3 Quantum lower bound for BCP in constant dimensions

Now, we give a lower bound for $(1+\xi)$-BCP, which trivially holds for BCP.

▶ **Theorem 67.** *The quantum query complexity for solving* BCP *is* $\Omega(n^{2/3})$. *Furthermore, the quantum query complexity for solving* $(1+\xi)$-BCP *with an arbitrary* ξ *is also* $\Omega(n^{2/3})$.

Proof. Recall that we have shown in Section 4.4 that ED reduces to CP by viewing ED as one-dimensional CP with the minimum distance 0. It is not hard to see that ED also reduces to approximate CP with multiplicative error $1+\xi$ since 0 times $1+\xi$ is still 0. For simplicity, we denote approximate CP with multiplicative error $1+\xi$ as $(1+\xi)$-CP. Given a set S as a $(1+\xi)$-CP instance, we choose $A, B \subset S$ uniformly at random such that $A = S \setminus B$ and $|A| = |B|$. Then, with $1/2$ probability, a closest pair in S has one point in A and another in B. Therefore, if (a, b) be a valid solution for $(1+\xi)$-BCP on (A, B), (a, b) is also a a valid solution for $(1+\xi)$-CP on S with probability $1/2$.

It is obvious that following the same proof, CP reduces to BCP. Hence, the quantum query complexity for BCP and $(1+\xi)$-BCP are both $\Omega(n^{2/3})$. This completes the proof. ◀

6 Orthogonal vectors in constant dimensions

▶ **Theorem 68.** *The time complexity of* $\mathsf{OV}_{n,d}$ *(Definition 10) in quantum query model is* $\Theta(\sqrt{n})$ *when the dimension* d *is constant .*

Proof. We show lower and upper bounds for $\mathsf{OV}_{n,d}$:

Lower bound

We reduce the search problem to an instance of 2-dimensional OV. Let all vectors in A be $(0, 1)$. We map an element of the search instance with value 0 as a vector in B with value $(0, 1)$ in $\mathsf{OV}_{n,2}$, and 1 as $(1, 0)$. An orthogonal pair must contain the vector in B with value $(1, 0)$ in this construction. Therefore, if we find an orthogonal pair, we find the corresponding marked (value 1) element in the search instance. The $\Omega(\sqrt{n})$ lower bound of Grover's search algorithm gives an $\Omega(\sqrt{n})$ lower bound to $\mathsf{OV}_{n,d}$.

Upper bound

The vectors only have 2^d possible values, $\{0, 1\}^d$, in the d-dimensional OV. For a particular value $v \in \{0, 1\}^d$, we can use Grover search to check whether there exist vector $a \in A$ such that $a = v$ in time $O(\sqrt{n})$, and similarly for vectors in B. Therefore we can, for all $v \in \{0, 1\}^d$, check whether there exist $a \in A$ such that $a = v$ and $b \in B$ such that $b = v$ in $O(2^{d+1}\sqrt{n})$ time, recording the results as two 2^d bit strings S_A and S_B. Then we check all

2^{2d} pairs of values (v, w) whether $\langle v, w\rangle = 0$, $S_A(v) = 1$, and $S_B(w) = 1$. When we found such a pair (v, w), we use Grover's search algorithm again to output a corresponding pair of vectors. The total running time is $O(2^{d+1}\sqrt{n} + 2^{2d} + 2\sqrt{n}) = \widetilde{O}(\sqrt{n})$. ◀

References

1 Scott Aaronson and Yaoyun Shi. Quantum Lower Bounds for the Collision and the Element Distinctness Problems. *J. ACM*, 51(4):595–605, July 2004.

2 Amir Abboud, Ryan Williams, and Huacheng Yu. More Applications of the Polynomial Method to Algorithm Design. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230, 2015.

3 Pankaj K. Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean Minimum Spanning Trees and Bichromatic Closest Pairs. *Discrete & Computational Geometry*, 6(3):407–422, September 1991.

4 A. Ambainis. Quantum Search Algorithms. *SIGACT News*, 35(2):22–35, June 2004. doi:10.1145/992287.992296.

5 Andris Ambainis. Quantum Walk Algorithm for Element Distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

6 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

7 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the Fine-Grained Complexity of Empirical Risk Minimization: Kernel Methods and Neural Networks. In *Advances in Neural Information Processing Systems*, pages 4308–4318, 2017.

8 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of Useful Work. *IACR Cryptology ePrint Archive*, 2017:203, 2017.

9 C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

10 Jon Louis Bentley and Michael Ian Shamos. Divide-and-Conquer in Multidimensional Space. In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 220–230. ACM, 1976.

11 Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum Algorithms for the Subset-Sum Problem. In *Post-Quantum Cryptography*, pages 16–33. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-38616-9_2.

12 Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential Improvement In Precision for Simulating Sparse Hamiltonians. *Forum of Mathematics, Sigma*, 5, 2017. doi:10.1017/fms.2017.2.

13 Sergei N Bespamyatnikh. An Optimal Algorithm for Closest-Pair Maintenance. *Discrete & Computational Geometry*, 19(2):175–195, 1998.

14 Harry Buhrman, Christoph Durr, Mark Heiligman, Peter Hoyer, Frédéric Magniez, Miklos Santha, and Ronald De Wolf. Quantum Algorithms for Element Distinctness. In *Proceedings 16th Annual IEEE Conference on Computational Complexity*, pages 131–137. IEEE, 2001.

15 Harry Buhrman, Subhasree Patro, and Florian Speelman. The Quantum Strong Exponential-Time Hypothesis. *arXiv preprint*, 2019. arXiv:1911.05686.

16 Timothy M Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1246–1255. Society for Industrial and Applied Mathematics, 2016.

17 Lijie Chen. On the Hardness of Approximate and Exact (Bichromatic) Maximum Inner Product. *arXiv preprint*, 2018. arXiv:1802.02325.

18 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–40. SIAM, 2019.

19 Kenneth L Clarkson. A Randomized Algorithm for Closest-Point Queries. *SIAM Journal on Computing*, 17(4):830–847, 1988.

20 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

21 Evgeny Dantsin, Vladik Kreinovich, and Alexander Wolpert. On Quantum Versions of Record-breaking Algorithms for SAT. *SIGACT News*, 36(4):103–108, December 2005. doi:10.1145/1107523.1107524.

22 R. David, K. S., and B. Laekhanukit. On the Complexity of Closest Pair via Polar-Pair of Point-Sets. *SIAM Journal on Discrete Mathematics*, 33(1):509–527, 2019.

23 Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint*, 1996. arXiv:quant-ph/9607014.

24 Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-eighth ACM Symposium on Theory of Computing - STOC '96*. ACM Press, 1996. doi:10.1145/237814.237866.

25 Timon Hertli. *Improved Exponential Algorithms for SAT and ClSP*. PhD thesis, ETH Zurich, 2015.

26 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, March 2001. doi:10.1006/jcss.2000.1727.

27 Toshiya Itoh, Tatsuya Nagatani, and Jun Tarui. Explicit Construction for k-Wise Nearly Random Permutations by Iterated Feistel Transform. Workshop on Randomness and Computation, 2005.

28 Stacey Jeffery. *Frameworks for Quantum Algorithms*. PhD thesis, University of Waterloo, 2014.

29 CS Karthik and Pasin Manurangsi. On Closest Pair in Euclidean Metric: Monochromatic is as Hard as Bichromatic. *10th Innovations in Theoretical Computer Science*, 2019.

30 S. Khuller and Y. Matias. A Simple Randomized Sieve Algorithm for the Closest-Pair Problem. *Information and Computation*, 118(1):34–37, 1995.

31 Victor Klee. On the Complexity of d-Dimensional Voronoi Diagrams. *Archiv der Mathematik*, 34(1):75–80, 1980. doi:10.1007/bf01224932.

32 Jon Kleinberg and Eva Tardos. *Algorithm Design*. Pearson Education India, 2006.

33 Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. Search via Quantum Walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.

34 Chris Marriott and John Watrous. Quantum Arthur–Merlin games. *Computational Complexity*, 14(2):122–152, 2005. doi:10.1007/s00037-005-0194-x.

35 Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact Discovery of Time Series Motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

36 Alexandros Nanopoulos, Yannis Theodoridis, and Yannis Manolopoulos. C2P: Clustering based on Closest Pairs. In *VLDB*, 2001.

37 Michael A Nielsen and Isaac Chuang. Quantum Computation and Quantum Information, 2002.

38 Ramamohan Paturi and Pavel Pudlák. On the Complexity of Circuit Satisfiability. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 241–250. ACM, 2010.

39 Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An Improved Exponential-Time Algorithm for k-SAT. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

40 Michael O Rabin. Probabilistic Algorithms Algorithms and Complexity: New Directions and Recent Results, 1976.

41 Kunihiko Sadakane, Norito Sugawara, and Takeshi Tokuyama. Quantum Algorithms for Intersection and Proximity Problems. In Peter Eades and Tadao Takaoka, editors, *Algorithms and Computation*, pages 148–159, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

42 Dominik Scheder and John P Steinberger. PPSZ for General k-SAT-Making Hertli's Analysis Simpler and 3-SAT Faster. In *32nd Computational Complexity Conference (CCC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

43 T Schoning. A Probabilistic Algorithm for k-SAT and Constraint Satisfaction Problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 410–414. IEEE, 1999.

44 M. I. Shamos and D. Hoey. Closest-Point Problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 151–162, 1975.

45 Mario Szegedy. Quantum Speed-Up of Markov Chain Based Algorithms. In *45th Annual IEEE symposium on foundations of computer science*, pages 32–41. IEEE, 2004.

46 Virginia Vassilevska Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

47 Nilton Volpato and Arnaldo Moura. A fast quantum algorithm for the closest bichromatic pair problem, 2010.

48 Ryan Williams. Pairwise comparison of bit vectors. Theoretical Computer Science Stack Exchange. URL: `https://cstheory.stackexchange.com/q/37369`.

49 Ryan Williams. A New Algorithm for Optimal 2-Constraint Satisfaction and Its Implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

50 Ryan Williams and Huacheng Yu. Finding Orthogonal Vectors In Discrete Structures. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1867–1877. SIAM, 2014.

51 Raymond Chi-Wing Wong, Yufei Tao, Ada Wai-Chee Fu, and Xiaokui Xiao. On Efficient Spatial Matching. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 579–590, 2007.

52 A. C. . Yao. Lower Bounds for Algebraic Computation Trees with Integer Inputs. In *30th Annual Symposium on Foundations of Computer Science*, pages 308–313, 1989.

Limits of Preprocessing

Yuval Filmus
Technion – Israel Institute of Technology, Haifa, Israel
yuvalfi@cs.technion.ac.il

Yuval Ishai
Technion – Israel Institute of Technology, Haifa, Israel
yuvali@cs.technion.ac.il

Avi Kaplan
Technion – Israel Institute of Technology, Haifa, Israel
kavi@cs.technion.ac.il

Guy Kindler
Hebrew University of Jerusalem, Jerusalem, Israel
gkindler@cs.huji.ac.il

Abstract

It is a classical result that the inner product function cannot be computed by an AC^0 circuit [17, 1, 22]. It is conjectured that this holds even if we allow arbitrary preprocessing of each of the two inputs separately. We prove this conjecture when the preprocessing of one of the inputs is limited to output $n + n/(\log^{\omega(1)} n)$ bits. Our methods extend to many other functions, including pseudorandom functions, and imply a (weak but nontrivial) limitation on the power of encoding inputs in low-complexity cryptography. Finally, under cryptographic assumptions, we relate the question of proving variants of the main conjecture with the question of learning AC^0 under simple input distributions.

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

Keywords and phrases circuit, communication complexity, IPPP, preprocessing, PRF, simultaneous messages

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.17

Funding *Yuval Filmus*: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC.
Yuval Ishai: Supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.
Avi Kaplan: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC, and ERC Project NTSC (742754).

Acknowledgements We thank Andrej Bogdanov, Mika Göös, Sajin Koroth, Dinesh Krishnamoorthy, Srikanth Srinivasan, and anonymous reviewers for helpful discussions, comments, and pointers.

1 Introduction

Can preprocessing help in computation? This question, which arises in several areas of complexity theory, can be formalized in many ways. We consider the following version:

> Suppose that $f(x, y)\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a function hard for AC^0. Are there functions $\alpha, \beta\colon \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ such that $f(x, y)$ can be computed from $\alpha(x), \beta(y)$ using an AC^0 circuit?

We think of α, β as functions that preprocess the inputs x, y in order to make the computation of f easier. Alternatively, one can think of $\alpha(x)$ and $\beta(y)$ as messages sent simultaneously by two parties to an AC^0 referee, whose goal is to compute $f(x, y)$. An alternative formulation is:

> Let $\mathcal{F}$ be a collection of hard functions $f_x \colon \{0,1\}^n \to \{0,1\}$ indexed by $x \in \{0,1\}^n$. Is there a function $\beta \colon \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ such that each $f_x(y) \in \mathcal{F}$ can be computed from $\beta(y)$ using an AC^0 circuit?

The two formulations are equivalent due to the completeness of circuit evaluation: if f_x can be computed efficiently from $\beta(y)$, then the function $f(x, y) = f_x(y)$ can be computed efficiently from $\beta(y)$ and the description of the circuit for f_x.

A simple example where preprocessing does help is when the function $f(x, y)$ depends only on the Hamming weights of x and y (e.g., $|x| > |y|$). Another simple example is any equivalence relation (e.g., graph isomorphism), where the two parties can send to the referee canonical representatives of the equivalence class of their respective inputs.

In contrast to the above examples, it is widely believed that for $f(x, y) = \sum_{i=1}^{n} x_i y_i \bmod 2$ (also known as *mod-2 inner product*) the answer to the above questions is negative. Following Rothblum [27], we refer to this as the *inner product with preprocessing* (IPPP) conjecture. Our main result proves a weak version of the IPPP conjecture, ruling out the utility of preprocessing when the output of β is short:

▶ **Theorem 1** (Main theorem, informal). *Let f be the mod-2 inner product function, or alternatively any exponentially-secure cryptographic pseudorandom function, and let $m = n + n/(\log^{\omega(1)} n)$. There are no functions $\alpha \colon \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ and $\beta \colon \{0,1\}^n \to \{0,1\}^m$ for which $f(x, y)$ can be computed from $\alpha(x), \beta(y)$ using an AC^0 circuit.*

Our result is in fact more general, applying to a broad class of other functions, and ruling out not only AC^0 circuits, but also bounded depth circuits of subexponential size. In particular, it applies to any function with large *statistical query dimension* [23, 7].

Our main theorem implies a modest but meaningful limitation on the power of preprocessing in low-complexity cryptography. There is a large body of work on minimizing the complexity of pseudorandom functions (PRFs) [19]; see [9] for a survey. A recent work of Boneh et al. [10] proposed a relaxed notion of PRF, dubbed "encoded-input PRF", that allows an arbitrary polynomial-time encoding of the input. This is motivated by several applications of low-complexity PRFs for which the relaxed notion suffices. The result of Linial et al. [24] rules out the existence of PRFs (with better than quasipolynomial security) in the complexity class AC^0. A natural question is whether one can circumvent this impossibility by encoding the input. We show that such an encoding (if it exists) must have a nontrivial stretch.

As a final contribution, we relate the question of fully settling variants of the IPPP conjecture to another wide-open question: learning AC^0 under "simple" input distributions, such as polynomial-time samplable distributions, or uniform distributions over linear subspaces of $\mathbb{F}_2^n$. Under cryptographic assumptions from [6, 10], we show that either (1) the known quasipolynomial time learning algorithm for AC^0 under the uniform distribution [24] cannot be extended to other simple distributions, even with subexponential time; or (2) IPPP-style hardness conjectures are true. Put differently, progress on learning AC^0 (even under simple distributions and in subexponential time) would lead to proving IPPP-style conjectures under cryptographic assumptions. The latter currently seems difficult. The idea behind this connection is that the functions α and β corresponding to a refutation of an IPPP-style conjecture define a *reduction* from breaking "rounded inner-product" style (weak) PRF candidates to learning AC^0 under simple distributions.

1.1 Related Work

The power of preprocessing is relevant to many problems in computer science. For instance, the broad goal of *data structures* is to preprocess x into a polynomially longer $\hat{y} = \beta(y)$, such that queries of the form $f(x, y)$ can be answered by reading few bits of $\hat{y}$. In our case, we replace "reading few bits of $\hat{y}$" by "computing an AC^0 function of $\hat{y}$". Below we survey several settings in complexity theory and cryptography that motivate this kind of questions.

Communication complexity – Polynomial hierarchy

Communication complexity contains analogs of the familiar complexity classes of computational complexity. For example, P^{cc} consists of all two-party functions which can be computed by exchanging polylogarithmically many bits, and $\mathsf{NP}^{\mathsf{cc}}$ consists of all two-party functions which can be *verified* using polylogarithmically many bits.

An $\mathsf{NP}^{\mathsf{cc}}$ protocol for a function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ proceeds as follows: an oracle sends the two parties the index of a combinatorial rectangle $X \times Y \subseteq \{0,1\}^n \times \{0,1\}^n$ on which $f = 1$, and the two parties verify that their inputs x, y belong to the rectangle: $x \in X$ and $y \in Y$; the complexity of the protocol is the length of the index. Equivalently, $f \in \mathsf{NP}^{\mathsf{cc}}$ if it can be written as a disjunction of $2^{\mathsf{polylog}(n)}$ combinatorial rectangles, that is, functions of the form "$x \in A$ and $y \in B$".

Babai, Frankl and Simon [4] extended this by defining an analog of the polynomial hierarchy, $\mathsf{PH}^{\mathsf{cc}}$. A function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ belongs to $\mathsf{PH}^{\mathsf{cc}}$ if it can be expressed as a constant depth circuit of quasipolynomial size $2^{\mathsf{polylog}(n)}$ whose leaves are combinatorial rectangles. Equivalently, $f \in \mathsf{PH}^{\mathsf{cc}}$ if it can be expressed as a constant depth circuits of quasipolynomial size whose leaves are arbitrary functions depending arbitrarily on one of the inputs. This is an instance of our main question, with a slight difference: $\mathsf{PH}^{\mathsf{cc}}$ allows the circuits to have quasipolynomial size.

Existing lower bound methods in communication complexity only go as far as $\mathsf{P}^{\mathsf{NPcc}}$ [20]. Nevertheless, it is a folklore conjecture that the inner product function IP lies outside $\mathsf{PH}^{\mathsf{cc}}$. This is considered as one of the most important outstanding open problems in the field.

Razborov [26] showed that a function whose matrix representation is rigid enough doesn't belong to $\mathsf{PH}^{\mathsf{cc}}$ (see also [35]), thus giving one potential avenue to prove lower bounds against $\mathsf{PH}^{\mathsf{cc}}$. Recently, in a surprising result, Alman and Williams [3] (see also [16]) showed that the inner product (or Hadamard) matrix isn't as rigid as was previously believed; however, their result doesn't rule out the use of Razborov's approach for proving the IPPP conjecture.

Communication complexity – Simultaneous messages and compression

As noted above, the question we study can be naturally cast as a computationally bounded variant of the *simultaneous messages* (SM) model in communication complexity [36, 5]. In this model, $k \geq 2$ parties send their messages to a referee, who should immediately output the value of the function. In our case, $k = 2$ and the referee is limited to be an AC^0 circuit. On the other hand, the two parties are computationally unbounded, and the message sent by each party can be longer than its input.

A different communication complexity model that considers AC^0-bounded parties is the compression model from [15, 11]. In this model, there is an AC^0 party whose goal is to compute the parity of its n-bit input x using the help of a computationally unbounded party, while minimizing the communication. This model is very different from ours; in particular, the model is trivialized if one allows n bits of communication.

Circuit complexity – Graph complexity

Pudlák, Rödl and Savický [25] developed the concept of *graph complexity* as a new approach to circuit lower bounds. Given a graph, we attempt to build it up from "axioms" using union, intersection, and complementation. In the particular case of *bipartite complexity*, the graph to be constructed is bipartite, and the axioms are complete bipartite graphs respecting the bipartition of the target graph.

A bipartite graph naturally defines a Boolean function with two inputs: the inputs are one vertex from each side, and the output is whether the edge exists. This correspondence shows that bipartite complexity is the same as a circuit whose leaves are combinatorial rectangles. Alternatively, we can allow each leaf to depend arbitrarily on one of the inputs, thus recovering our model of study.

Bipartite complexity can be studied for various circuit classes. One recent highlight is the work of Tal [29], in which he shows that bipartite formulas computing IP must have quadratic size.

Circuit complexity – $\mathsf{AC^0} \circ \mathsf{MOD_2}$

Our understanding of $\mathsf{AC^0}[p]$ circuits lacks compared to $\mathsf{AC^0}$ circuits. While we have strong lower bounds against $\mathsf{AC^0}[p]$ circuits, the existing correlation bounds are significantly weaker, and this is a barrier for constructing pseudorandom generators for $\mathsf{AC^0}[p]$. Observing all of this, Servedio and Viola [28] suggest considering a weakening of $\mathsf{AC^0}[2]$, in which all parity gates appear in the bottom layer. They conjecture that inner product cannot be computed by such circuits, and prove their conjecture for depth-3 circuits. Akavia et al. [2] give cryptographic applications for lower bounds against this class, and Cheragchi et al. [12] give superlinear lower bounds for inner product.

$\mathsf{AC^0}$ circuits with parity gates at the bottom are the same as $\mathsf{AC^0}$ circuits with *linear* preprocessing, namely where the preprocessing functions α, β are linear over $\mathbb{F}_2$. In other words, the conjecture of Serverdio and Viola is a special case of our conjecture, in which it suffices to rule out linear α, β.

Cryptography

Our formulation of the IPPP conjecture is a close variant of the IPPP conjecture made by Rothblum [27], where it was used to construct circuits resilient to $\mathsf{AC^0}$ leakage. (The flavor of IPPP from [27] is different from ours in that it restricts α, β to be polynomial-time computable and assumes hardness of approximation as opposed to just worst-case hardness.) In a recent work of Bogdanov et al. [8], a similar result was obtained unconditionally.

As discussed above, our main question is strongly relevant to the goal of implementing cryptographic primitives in $\mathsf{AC^0}$. The work of Boneh et al. [10] poses the question of implementing an "encoded-input pseudorandom function" in $\mathsf{AC^0}$, namely a pseudorandom function family $f_k(x)$, where each function f_k can be computed in $\mathsf{AC^0}$ given an encoding of the input x. This is essentially the same as asking whether our main question can be answered affirmatively for some $f(k, x)$ such that $f_k(x)$ is a pseudorandom function family.

Extractors

As part of his study of extractors for $\mathsf{NC^0}$ and $\mathsf{AC^0}$ sources, Viola [34] constructed a function $f\colon \{0,1\}^n \to \{0,1\}$ such that the distribution $(\mathbf{x}, f(\mathbf{x}))$ (with $\mathbf{x}$ uniform) is hard for $\mathsf{AC^0}$ to sample, even approximately. In particular, his results imply that the function $F\colon [n+1] \times \{0,1\}^n \to \{0,1\}$ given by

$$F(i,y) = \begin{cases} y_i & i \in [n], \\ f(y) & i = n+1, \end{cases}$$

cannot be computed by an AC^0 circuit from $\alpha(x), \beta(y)$, where $\alpha\colon [n+1] \to \{0,1\}^{\mathsf{poly}(n)}$ and $\beta\colon \{0,1\}^n \to \{0,1\}^n$. Therefore $F(x,y)$ requires exactly $n+1$ bits of preprocessing of y.

1.2 Overview of techniques

We outline the technique used for proving Theorem 1. The main tool we use is the LMN inequality [24, 30], which states that AC^0 functions can be approximated by low degree functions. Let us illustrate the main idea behind the proof by sketching the proof of the following special case.

▶ **Theorem 2.** *Let $\alpha\colon \{0,1\}^n \to \{0,1\}^*$ and let $\beta\colon \{0,1\}^n \to \{0,1\}^n$, and suppose that C is a bounded-depth circuit satisfying $C(\alpha(x), \beta(y)) = \mathsf{IP}(x,y)$ for all $x, y \in \{0,1\}^n$. Then C has exponential size.*

Since the inner product function is injective in each of its inputs, the preprocessing function β must be bijective.

For each $x \in \{0,1\}^n$, we can plug in the values $\alpha(x)$ to obtain constant-depth circuit C_x, of size at most that of C, satisfying $C_x(y) = \mathsf{IP}(x, \beta^{-1}(y))$ for all $y \in \{0,1\}^n$.

For any two $x \neq z$, the functions $f_x(y) = \mathsf{IP}(x,y)$ and $f_z(y) = \mathsf{IP}(z,y)$ are orthogonal (this is the well-known orthogonality of the Fourier characters). This property is crucially maintained by the functions C_x, C_z, which are also orthogonal.

Suppose that C has small size. We are thus in the following situation: we have 2^n orthogonal functions C_x, each of which can be approximated by a low degree function (by LMN), and so close to a low-dimensional subspace x of $\mathbb{R}[\{0,1\}^n]$. This is, however, impossible.

The argument works in much the same way for any function $f(x,y)$ which is injective in its first input and whose "slices" $f_x(y) = f(x,y)$ are approximately orthogonal on average. A short hybrid argument shows that PRFs fit the bill.

It is more challenging to extend the argument to functions β with larger output $\{0,1\}^m$. The basic idea is to complete the functions C_x, which are a priori defined only on 2^n of the 2^m possible inputs, to total functions which are still approximately orthogonal. Therefore if C has small size then one of the functions C_x will be far from V. On the other hand, since C_x agrees with a function computed by an AC^0 circuit on a 2^{n-m} fraction of the input, and that function is close to V. These two properties contradict each other.

This sketch explains why we can only expect to handle this way $m = n + o(n)$: if m is any larger, then the correlation of C_x with the output of the circuit is too small, and so we cannot reach any contradiction.

Organization

After brief preliminaries (Section 2), we state our main results in Section 3, including Theorem 1 above. The connection to learning AC^0 functions under simple input distributions appears in Section 4. We prove our main technical theorem in Section 5, which is followed by applications to encoded-input PRFs (Section 6) and rounded inner product (Section 7).

2 Preliminaries

2.1 Definitions and notation

Simultaneous messages protocols

A (two-party) simultaneous messages (SM) protocol consists of two players, which we refer to as Alice and Bob, and a referee, which we refer to as Carol, that together compute a function. Alice and Bob each send a message, which is based on the input, to Carol, who then computes a function of the two messages received. Formally, we have the following definitions:

▶ **Definition 3** (Simultaneous messages protocols). *Let $X, Y, \widehat{X}, \widehat{Y}, Z$ be finite nonempty sets. A* simultaneous messages protocol *(shortly,* SM protocol *or* SMP*) $\mathcal{P}$ is a triplet of functions (A, B, C), where $A\colon X \to \widehat{X}$, $B\colon Y \to \widehat{Y}$, and $C\colon \widehat{X} \times \widehat{Y} \to Z$. We call C the* referee function*.*

▶ **Definition 4** (SM protocol admittance). *Let $f\colon X \times Y \to Z$ be a function. We say that f* admits an SM protocol *(A, B, C) if $f(x, y) = C(A(x), B(y))$ for every $(x, y) \in X \times Y$. In that case, we also say that (A, B, C)* computes *f.*

Inner product space of Boolean functions

For the purpose of utilizing Fourier analysis, we will consider the inner product space of all functions $\{-1, 1\}^n \to \mathbb{R}$ with the following inner product:

$$\langle f, g\rangle = \mathop{\mathrm{E}}_{\boldsymbol{x}\sim\{-1,1\}^n}[f(\boldsymbol{x})g(\boldsymbol{x})] = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x)\cdot g(x).$$

It is a known fact that the aforementioned inner product space has as orthonormal basis the set of all parity functions $\{\chi_S\}_{S\subseteq[n]}$, defined by $\chi_S(x) = \prod_{i\in S} x_i$.

The Inner Product function

The *inner product modulo* 2 function $\mathsf{IP}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined by

$$\mathsf{IP}(x, y) = \sum_{i=1}^{n} x_i y_i \pmod 2.$$

Note that each x, when fixed, corresponds to a parity function on a subset S of y's coordinates, a subset which is determined by x. This correspondence is actually a bijection mapping elements of $\{0,1\}^n$ to subsets of $2^{[n]}$. Thus, when we switch to ± 1 notation, we can identify each $x \in \{-1, 1\}^n$ with the parity function $\chi_{S(x)}(y)$ that results when fixing x in $\mathsf{IP}(x, y)$.

One-to-one condition

Some functions have a certain property, formally defined below, rendering them harder to compute with preprocessing. Restricting attention to these functions seems to help in obtaining lower bounds.

▶ **Definition 5** (One-to-one condition). *Let $f\colon X \times Y \to Z$ be a function. We say that f* satisfies the left one-to-one condition *if for every $x \neq x' \in X$ there exists $y \in Y$ such that $f(x, y) \neq f(x', y)$. Similarly, we say that f* satisfies the right one-to-one condition *if for every $y \neq y' \in Y$ there exists $x \in X$ such that $f(x, y) \neq f(x, y')$. Finally, we say that f* satisfies the one-to-one condition *if f satisfies both the left and right one-to-one conditions.*

▶ **Proposition 6** (One-to-one preprocessing). *Let $f\colon X \times Y \to Z$ be a function. Then:*

- *If f satisfies the left one-to-one condition, then for every SM protocol (A, B, C) that f admits, A computes a one-to-one mapping.*
- *If f satisfies the right one-to-one condition, then for every SM protocol (A, B, C) that f admits, B computes a one-to-one mapping.*

2.2 Known facts

The following are known facts we will need later.

▶ **Theorem 7** (Tal's LMN improvement (LMNT) [24, 30]). *Let f be a Boolean function with n variables computable by an unbounded fan-in circuit of depth h and size M, and let t be any integer. Then,*

$$\|f^{\geq t}\|^2 \leq 2 \cdot 2^{-t/O_h(\log M)^{h-1}}.$$

▶ **Lemma 8** (Lemma 3.6 in [21]). *Let* $\mathrm{H}\colon [0,1] \to \mathbb{R}$ *be the* binary entropy function *defined by*

$$\mathrm{H}(p) = -p \log p - (1-p)\log(1-p).$$

Then, for any $0 < a \leq 1/2$ and $n \in \mathbb{N}$,

$$\binom{n}{\leq an} \leq 2^{\mathrm{H}(a)n}.$$

3 Main results

Our main technical result, proved in Section 5, states that a "large" collection of functions that are "close" to being orthonormal, is computationally hard for SM protocols in which the referee is an unbounded fan-in circuit of constant depth, and one player is limited to "short" preprocessing output length.

▶ **Theorem 9** (Main Theorem). *Let $f\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ be a Boolean function, let $0 \leq k \leq n/2 - 1$, and let $0 \leq t \leq n + k$ be an integer. Denote $f_x(y) \triangleq f(x,y)$. Suppose the following hold:*

- *f satisfies the right one-to-one condition.*
- *There exists a subset $X \subseteq \{-1,1\}^n$ of size $|X| \geq 13 \cdot 2^{2(k+1)} \cdot \binom{n+k}{\leq t}$ such that*

$$\mathop{\mathrm{E}}_{\boldsymbol{x} \neq \boldsymbol{x}' \sim X}\left[\langle f_{\boldsymbol{x}}, f_{\boldsymbol{x}'}\rangle^2\right] \leq \frac{2^{2k}}{36|X|^2}.$$

- *f admits an SM protocol $\mathcal{P} = (A, B, C)$ such that $B\colon \{-1,1\}^n \to \{-1,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M.*

Then:

$$M \geq 2^{\Omega_h\left(\left[\frac{t}{k}\right]^{1/(h-1)}\right)}.$$

One may wonder about the necessity of satisfying the one-to-one condition. While it may still be the case that the same result (or even better) follows without this assumption, if we could obtain it that way, then we could easily exhibit a function with stronger lower bounds. As an example, consider taking the inner product function and computing it only on a prefix of the input while ignoring other bits – relying on the main theorem's consequence, we could extend it to an arbitrary $\Omega(n)$ lower bound on the preprocessing output length.

The (somewhat cumbersome) second requirement on a function $f\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ specified in Theorem 9 can be replaced by a slightly stronger requirement, yet one involving the more familiar measure borrowed from the study of learning by statistical queries [23]. Define the *statistical query dimension* of f with respect to the *uniform* distribution to be the size of the largest set $D \subseteq \{-1,1\}^n$ such that $|\langle f_x, f_{x'}\rangle| \leq 1/|D|$ for every $x \neq x' \in D$. More details on statistical query dimension can be found in [7].

A simple consequence of Theorem 9 is Theorem 1. Here we state a more general version that refers to statistical query dimension. At a high level, the theorem says that computing with preprocessing a function having exponential statistical query dimension remains as hard for AC^0 as without, given that one player is limited to output a string whose length stretches the input length by an additive sublinear term:

▶ **Proposition 10** (Formal version of Theorem 1)**.** *Let $k = o(n)$, and suppose that a function $f\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ satisfies the one-to-one condition and has statistical query dimension of $2^{\Omega(n)}$. If f admits an SM protocol (A,B,C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M, then:*

$$M \geq 2^{\Omega_h\left(\left[\frac{n}{k}\right]^{1/(h-1)}\right)}.$$

Proof. Let $D \subseteq \{-1,1\}^n$ be a set of size $2^{\Omega(n)}$ such that $|\langle f_x, f_{x'}\rangle| \leq 1/|D|$ for every $x \neq x' \in D$. One can easily find an $0 < \alpha \leq 1/2$ for which $\mathrm{H}(\alpha)$ is small enough, such that setting $t = \alpha(n+k)$ gives

$$13 \cdot 2^{2(k+1)} \cdot \binom{n+k}{\leq \alpha(n+k)} \underset{\text{Lemma 8}}{\leq} 13 \cdot 2^{2(k+1)} \cdot 2^{\mathrm{H}(\alpha)(n+k)} \leq |D|/6.$$

Now, let $X \subseteq D$ of size $|X| = 13 \cdot 2^{2(k+1)} \cdot \binom{n+k}{\leq t}$. We have:

$$\begin{aligned}\mathop{\mathrm{E}}_{\boldsymbol{x}\neq\boldsymbol{x}'\sim X}\left[\langle f_{\boldsymbol{x}}, f_{\boldsymbol{x}'}\rangle^2\right] &\leq \mathop{\mathrm{E}}_{\boldsymbol{x}\neq\boldsymbol{x}'\sim D}\left[\langle f_{\boldsymbol{x}}, f_{\boldsymbol{x}'}\rangle^2\right] \leq \mathop{\mathrm{E}}_{\boldsymbol{x}\neq\boldsymbol{x}'\sim D}\left[\left|\langle f_{\boldsymbol{x}}, f_{\boldsymbol{x}'}\rangle\right|\right]^2 \\ &\leq \frac{1}{|D|^2} \leq \frac{1}{36|X|^2} \leq \frac{2^{2k}}{36|X|^2}.\end{aligned}$$

Thus, by Theorem 9,

$$M \geq 2^{\Omega_h\left(\left[\frac{t}{k}\right]^{1/(h-1)}\right)} = 2^{\Omega_h\left(\left[\frac{\alpha(n+k)}{k}\right]^{1/(h-1)}\right)} = 2^{\Omega_h\left(\left[\frac{n}{k}\right]^{1/(h-1)}\right)}.$$ ◀

Since IP satisfies the one-to-one condition and has the largest possible statistical query dimension of 2^n, we get the following corollary.

▶ **Corollary 11.** *Let $k \leq n^\alpha$ for some $0 \leq \alpha < 1$, and suppose that IP admits an SM protocol (A,B,C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M. Then:*

$$M \geq 2^{\Omega_h\left(n^{\frac{1-\alpha}{h-1}}\right)}.$$

▶ **Corollary 12.** *Let $k \leq \frac{n}{\log^\beta n}$ for every $\beta > 0$ (for large enough n), and suppose that IP admits an SM protocol (A,B,C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M. Then:*

$$M \geq 2^{\Omega_h(\log^c n)} \text{ for every } c > 0.$$

We now present an application of our main theorem to cryptography. We show that exponentially secure PRFs (in fact, even *weak* PRFs) are not computable in AC^0, even if one allows an arbitrary sublinear-stretch encoding of the input. This implies a limitation on the power of encoded-input PRFs in AC^0 [10].

▶ **Definition 13** (pseudorandom functions). *Let $\mathcal{K}$ be a keys domain, and let $F\colon \mathcal{K}\times\{0,1\}^n \to \{0,1\}$ be a family of functions; denote $F_k(x) \triangleq F(k,x)$. For integer m and $\epsilon \in [0,1]$, we say that F is a (strong) (ϵ,m)-*pseudorandom function function family *(shortly (ϵ,m)-*PRF*) if for every (non-uniform) circuit distinguisher D^f of size at most m, the following holds:*

$$\left|\Pr_{\boldsymbol{k}\sim\mathcal{K}}\left[D^{F_{\boldsymbol{k}}}(1^n)=1\right] - \Pr_{\boldsymbol{f}}\left[D^{\boldsymbol{f}}(1^n)=1\right]\right| \leq \epsilon.$$

If the distinguisher is limited to querying the oracle on random and independent inputs, then we say that F is a weak (ϵ,m)-PRF*.*

For simplicity, we will consider the case in which $\mathcal{K} = \{0,1\}^n$ (under the uniform distribution).

We prove the following result in Section 6:

▶ **Theorem 14** (Lower bound for exponentially secure weak PRFs). *Let $k \leq n^\alpha$ for some $0 \leq \alpha < 1$, and suppose that $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a strong $2^{\Omega(n)}$-PRF, or alternatively a weak $2^{\Omega(n)}$-PRF satisfying the right one-to-one condition. If F admits an SM protocol (A,B,C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M, then:*

$$M \geq 2^{\Omega_h\left(n^{\frac{1-\alpha}{h-1}}\right)}.$$

(Similar results hold for $2^{n^{\Omega(1)}}$-PRFs, with slightly worse bounds on M.)

As before, the reason we require a weak PRF to satisfy the right one-to-one condition is that its "effective" input size could be much smaller than n. For example, imagine a weak PRF which ignores the right half of its input. A distinguisher would need $2^{\Omega(n)}$ random samples to notice this. The right one-to-one condition is automatically satisfied by strong PRFs: if $f_k(x) = f_k(x')$ for all (or even most) keys k, then it is easy to distinguish f_k from a random function by querying the input function at x, x'.

We prove similar results for a class of functions obtained by applying a "rounding predicate" to inner-product modulo q.

▶ **Definition 15** (Rounded inner product). *For an integer $q \geq 2$ and a set $R \subseteq \{0,1,\ldots,q-1\}$ we define the (q,R)-*rounded inner product function $\mathsf{IP}^{[q,R]}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *by*

$$\mathsf{IP}^{[q,R]}(x,y) = \begin{cases} 0 & \sum_{i=1}^n x_i y_i \pmod q \in R, \\ 1 & \text{otherwise.} \end{cases}$$

One reason for our interest in this class is that some instances, such as rounded inner product modulo 6, are conjectured to be (weak) pseudorandom functions with near-exponential security [10]. Under such a conjecture, the desired negative result would follow from our results on (weak) PRFs. However, the results about rounded inner product functions are unconditional, and apply also to instances that are provably not (weak) PRFs.

We prove the following result in Section 7:

▶ **Theorem 16** (Lower bound for rounded inner product). *Let $q \geq 2$ be even, and let $R \subseteq \{0, 1, \ldots, q-1\}$ such that $|R| = q/2$. Let $k \leq n^\alpha$ for some $0 \leq \alpha < 1$, and suppose that $\mathsf{IP}^{[q,R]}$ admits an SM protocol (A, B, C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+k}$ and C is an unbounded fan-in circuit of depth h and size M. Then:*

$$M \geq 2^{\Omega_h\left(n^{\frac{1-\alpha}{h-1}}\right)}.$$

4 Conditional Limits of Preprocessing and Learning AC^0

We were unable to settle the main IPPP conjecture or prove similar results on the limits of preprocessing for other explicit functions. Moreover, our current techniques seems insufficient. A second-best alternative is to settle such questions under widely believed conjectures from complexity theory or cryptography. While we are also unable to show such a conditional result (and view this as an interesting goal), we can relate this challenge to another intriguing question: learning AC^0 under "simple" distributions.

The work of Linial et al. [24] shows that AC^0 can be learned in quasipolynomial time under the uniform distribution. It is open whether the same holds for PAC learning under arbitrary distributions. The question is still open even when restricted to simple input distributions, such as uniform distributions over linear subspaces of $\mathbb{F}_2^n$, and even if "quasipolynomial" is relaxed to "subexponential." In fact, we are not aware of hardness results that apply to any simple distributions or beyond quasipolynomial time. See [13, 31] for weaker conditional hardness results, and [14] for a survey of known learning algorithms for AC^0.

We observe that positive results on learning AC^0 under simple distributions can be used to base hardness of IPPP-style problems on cryptographic assumptions from [6, 10]. Equivalently, cryptographic assumptions imply that either (1) AC^0 cannot be learned under simple distributions in subexponential time, or (2) IPPP-style hardness conjectures are true. While both (1) and (2) seem highly plausible, strong versions of them may turn out to be false. Moreover, to the best of our knowledge, neither (1) nor (2) are known to be implied by standard conjectures in cryptography or complexity theory. A direct proof that either (1) or (2) hold also seems unlikely. For these reasons, we believe that the above connection is meaningful, and can potentially lead to future progress on either IPPP-style questions or learning AC^0 under simple distributions.

4.1 The conjectures

We will show connections between the following types of conjectures:

- Cryptographic assumptions:
  - (C1) Subexponential hardness of Learning With Rounding (LWR) [6]: for some $\epsilon > 0$ and polynomials $p = p(n)$, $q = q(n)$, the function $f_k(x) = \mathsf{Round}(\langle k, x\rangle \pmod{2q})$ is a $2^{\Omega(n^\epsilon)}$-secure weak PRF, where $k \in \{0, 1, \ldots, p-1\}^n$ and $x \in \{0,1\}^n$. Here, $\mathsf{Round}(y)$ returns 0 or 1 depending on whether y is closer to 0 or to the modulus $2q$.
  - (C2) Subexponential hardness of LWR mod 6 [10]: for some $\epsilon > 0$, $f_k(x) = \mathsf{Round}(\langle k, x\rangle \pmod 6)$ is a $2^{\Omega(n^\epsilon)}$-secure weak PRF, where $k, x \in \{0,1\}^n$.
- Hardness of learning conjectures:
  - (L1) AC^0 cannot be learned in subexponential time under all polynomial-time samplable input distributions.
  - (L2) AC^0 cannot be learned in subexponential time under all $\mathbb{F}_2$-linear input distributions.

  Here, learning in subexponential time refers to a $2^{n^{o(1)}}$-time learning algorithm in the standard PAC model [32].

- IPPP-style conjectures:
  - (P1) Integer-IP does not admit an SM protocol (A, B, C) where the referee C is in AC^0 and the parties A and B are polynomial-time. Here Integer-IP is the (non-boolean) inner product of two n-bit vectors over the integers.
  - (P2) Integer-IP is not in $\mathsf{AC}^0 \circ \mathsf{MOD}_2$.

  Similarly, we define $(\mathsf{P1})^m$ and $(\mathsf{P2})^m$ as variants where Integer-IP is replaced by inner product modulo m. Note that $(\mathsf{P1})^2$ is the worst-case variant of Rothblum's IPPP conjecture [27] and $(\mathsf{P2})^2$ is the IPPP with linear preprocessing conjecture made by Servedio and Viola [28].

4.2 The connections

We now establish simple connections between the previous conjectures.

▶ **Theorem 17.** *The following implications hold:*

1. $(\mathsf{C1}) \Rightarrow (\mathsf{L1}) \vee (\mathsf{P1})$
2. $(\mathsf{C1}) \Rightarrow (\mathsf{L2}) \vee (\mathsf{P2})$
3. $(\mathsf{C2}) \Rightarrow (\mathsf{L1}) \vee (\mathsf{P1})^2 \vee (\mathsf{P1})^3$
4. $(\mathsf{C2}) \Rightarrow (\mathsf{L2}) \vee (\mathsf{P2})^2 \vee (\mathsf{P2})^3$

Proof. To prove (1), suppose that both (L1) and (P1) are false. We use the SM protocol implied by $\neg(\mathsf{P1})$ to convert the learning algorithm implied by $\neg(\mathsf{L1})$ into an attack against the LWR assumption in (C1). Let $f(a, b)$ be the Integer-IP function. By $\neg(\mathsf{P1})$, there is an SM protocol (A, B, C) for f where C is in AC^0 and the parties A and B are polynomial-time. Letting $f'(k, x)$ be the rounded inner product function defined by polynomials p, q as in (C1), we get a similar SM protocol (A', B', C') for f' in the following natural way: A' expands each $k_i \in \{0, 1, \ldots, p-1\}$ to a binary length-p vector of weight k_i and invokes A; B' expands each $x_i \in \{0, 1\}$ to the length-p vector $(x_i, \ldots, x_i)$ and invokes B; and C' invokes C to compute the integer inner product $\langle k, x\rangle$, reduces the result modulo q, and rounds. Since p and q are polynomials, C' can indeed be implemented in AC^0. Now consider the message $\hat{k}$ sent by A' on a uniformly random input k, and let $C'_{\hat{k}}$ be the AC^0 circuit obtained by restricting C' to this first message. Let X be the (polynomial-time samplable) input distribution defined by the message sent by B' on a uniformly random input x. Using the subexponential time learning algorithm implied by $\neg(\mathsf{L1})$ to learn $C'_{\hat{k}}$ on input distribution X, we get a subexponential time algorithm breaking (C1) as required.

The proofs of the other parts of the theorem follow similarly, noting that if neither $(\mathsf{P1})^2$ nor $(\mathsf{P1})^3$ hold (resp., neither $(\mathsf{P2})^2$ nor $(\mathsf{P2})^3$ hold), then f' computing rounded inner product modulo 6 admits an SM protocol with referee in AC^0 and polynomial-time parties (resp., parties computing an $\mathbb{F}_2$-linear function with polynomial stretch). ◀

5 Proof of Main Theorem

The following two results will be needed for proving the main theorem, Theorem 9.

▶ **Proposition 18** (High-degree spectral concentration bound)**.** *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function, and let $0 \le \epsilon \le 1/2$. Then, for every integer $0 \le t \le n$ such that $\|f^{\le t}\| \le \epsilon$, if an unbounded fan-in circuit of depth h and size M agrees with f on at least $1/2 + \epsilon$ fraction of inputs, then*

$$M \ge 2^{\Omega_h\left(\left[\frac{t}{1-2\log\epsilon}\right]^{1/(h-1)}\right)}.$$

Proof. Let $0 \le t \le n$ be an integer such that $\|f^{\le t}\| \le \epsilon$, and suppose that an unbounded fan-in circuit of depth h and size M computes a function F that agrees with f on at least $1/2 + \epsilon$ fraction of inputs.

On one hand, we have

$$\langle F, f\rangle = 2\Pr[F = f] - 1 \underset{\text{assumption}}{\ge} 2(1/2 + \epsilon) - 1 = 2\epsilon.$$

On the other hand, we have

$$\langle F, f\rangle = \langle F^{\le t}, f^{\le t}\rangle + \langle F^{>t}, f^{>t}\rangle \underset{\text{Cauchy–Schwarz}}{\le} \|f^{\le t}\| + \|F^{>t}\| \underset{\text{LMNT}}{\le} \epsilon + \sqrt{2 \cdot 2^{-t/O_h(\log M)^{h-1}}}.$$

Thus,

$$2\epsilon \le \epsilon + \sqrt{2 \cdot 2^{-t/O_h(\log M)^{h-1}}} \implies M \ge 2^{\Omega_h\left(\left[\frac{t}{1-2\log\epsilon}\right]^{1/(h-1)}\right)}. \qquad \blacktriangleleft$$

In what follows, we will use the following notation:

- For a set X, we write $\boldsymbol{i} \ne \boldsymbol{j} \sim X$ to mean that $(\boldsymbol{i}, \boldsymbol{j})$ is chosen uniformly at random from the set $\{(i, j) \in X \times X : i \ne j\}$.
- Given an inner product space V, a subspace $U \le V$, and a vector $v \in V$, we denote the projection of v onto U by $\mathrm{proj}_U(v)$.

▶ **Lemma 19** (The Projection Lemma). *Let V be an inner product space over $\mathbb{R}$. Let $\{v_i\}_{i\in X} \subseteq V$ be a set of* unit *vectors indexed by X, and suppose that*

$$\mathop{\mathrm{E}}_{\boldsymbol{i}\ne\boldsymbol{j}\sim X}\left[\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle^2\right] \le \frac{1}{36|X|^2}.$$

Then, for every subspace $U \le V$, there exists $i \in X$ such that $\|\mathrm{proj}_U(v_i)\|^2 = O\left(\frac{\dim U}{|X|}\right)$.

Proof. Let $U \le V$ be a subspace, and denote $D \triangleq \dim U$.

By Cauchy–Schwartz,

$$\mathop{\mathrm{E}}_{\boldsymbol{i}\ne\boldsymbol{j}\sim X}\left[|\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle|\right] \le \mathop{\mathrm{E}}_{\boldsymbol{i}\ne\boldsymbol{j}\sim X}\left[\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle^2\right]^{1/2} \le \frac{1}{6|X|}.$$

By Markov's inequality,

$$\Pr_{\boldsymbol{i}\sim X}\left[\mathop{\mathrm{E}}_{\boldsymbol{j}\sim X\setminus\{\boldsymbol{i}\}}\left[\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle^2\right] > \frac{1}{12|X|^2}\right] \le 12|X|^2 \cdot \mathop{\mathrm{E}}_{\boldsymbol{i}\ne\boldsymbol{j}}\left[\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle^2\right] \le \frac{1}{3}.$$

and similarly,

$$\Pr_{\boldsymbol{i}\sim X}\left[\mathop{\mathrm{E}}_{\boldsymbol{j}\sim X\setminus\{\boldsymbol{i}\}}\left[|\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle|\right] > \frac{1}{2|X|}\right] \le 2|X| \cdot \mathop{\mathrm{E}}_{\boldsymbol{i}\ne\boldsymbol{j}}\left[|\langle v_{\boldsymbol{i}}, v_{\boldsymbol{j}}\rangle|\right] \le \frac{1}{3},$$

which implies that at least $1/3$ of the indices $i \in X$ satisfy

$$\mathop{\mathrm{E}}_{\boldsymbol{j}\sim X\setminus\{i\}}\left[\langle v_i, v_{\boldsymbol{j}}\rangle^2\right] \le \frac{1}{12|X|^2} \quad \text{and} \quad \mathop{\mathrm{E}}_{\boldsymbol{j}\sim X\setminus\{i\}}\left[|\langle v_i, v_{\boldsymbol{j}}\rangle|\right] \le \frac{1}{2|X|},$$

or equivalently,

$$\sum_{j\in X\setminus\{i\}} \langle v_i, v_j\rangle^2 \le \frac{1}{12|X|} \quad \text{and} \quad \sum_{j\in X\setminus\{i\}} |\langle v_i, v_j\rangle| \le \frac{1}{2}. \tag{1}$$

Put these indices in a set Y, and let $W \triangleq \operatorname{span}(\{v_i : i \in Y\})$.

Let $w \in W$ such that $\|w\| \leq 1$, and write $w = \sum_{i \in Y} c_i v_i$ with $c_i \in \mathbb{R}$. Then for $i \in Y$,

$$\langle w, v_i \rangle = \sum_{j \in Y} c_j \langle v_i, v_j \rangle = c_i + \sum_{j \in Y \setminus \{i\}} c_j \langle v_i, v_j \rangle.$$

Multiply this by c_i, and sum over all $i \in Y$ to obtain

$$1 \geq \|w\|^2 = \sum_{i \in Y} c_i^2 + \sum_{i \neq j} c_i c_j \langle v_i, v_j \rangle.$$

Since $2|c_i c_j| \leq c_i^2 + c_j^2$, it follows that

$$1 \geq \sum_{i \in Y} c_i^2 - \frac{1}{2} \sum_{i \neq j} (c_i^2 + c_j^2) |\langle v_i, v_j \rangle| = \sum_{i \in Y} c_i^2 \Big(1 - \sum_{j \in Y \setminus \{i\}} |\langle v_i, v_j \rangle| \Big) \underset{\text{Eq. (1)}}{\geq} \frac{1}{2} \sum_{i \in Y} c_i^2,$$

which implies $\sum_{i \in Y} c_i^2 \leq 2$.[1] Since $(a+b)^2 \leq 2a^2 + 2b^2$, for every $i \in Y$ we have

$$\begin{aligned} \langle w, v_i \rangle^2 &\leq 2c_i^2 + 2 \Big(\sum_{j \in Y \setminus \{i\}} c_j \langle v_i, v_j \rangle \Big)^2 \underset{\text{Cauchy–Schwarz}}{\leq} 2c_i^2 + 2 \sum_{j \in Y \setminus \{i\}} c_j^2 \cdot \sum_{j \in Y \setminus \{i\}} \langle v_i, v_j \rangle^2 \\ &\leq 2c_i^2 + 4 \sum_{j \in Y \setminus \{i\}} \langle v_i, v_j \rangle^2 \underset{\text{Eq. (1)}}{\leq} 2c_i^2 + \frac{1}{3|X|}. \end{aligned}$$

Taking expectation over $i \in Y$, we deduce

$$\begin{aligned} \mathop{\mathrm{E}}_{i \sim Y} \Big[\langle w, v_i \rangle^2 \Big] &\leq \mathop{\mathrm{E}}_{i \sim Y} \left[2c_i^2 + \frac{1}{3|X|} \right] = \frac{2}{|Y|} \sum_{i \in Y} c_i^2 + \frac{1}{3|X|} \leq \frac{4}{|Y|} + \frac{1}{3|X|} \\ &\underset{|Y| \geq |X|/3}{\leq} \frac{12}{|X|} + \frac{1}{3|X|} \leq \frac{13}{|X|}. \end{aligned}$$

Now let $u_1, \ldots, u_D$ be an orthonormal basis for U, and for every $k \in [D]$, let $w_k \in W$ be the projection of u_k onto W (notice that $\|w_k\| \leq 1$). We have

$$\begin{aligned} \mathop{\mathrm{E}}_{i \sim Y} \Big[\|\operatorname{proj}_U(v_i)\|^2 \Big] &= \mathop{\mathrm{E}}_{i \sim Y} \left[\sum_{k \in [D]} \langle v_i, u_k \rangle^2 \right] = \mathop{\mathrm{E}}_{i \sim Y} \left[\sum_{k \in [D]} \langle v_i, w_k \rangle^2 \right] \\ &= \sum_{k \in [D]} \mathop{\mathrm{E}}_{i \sim Y} \Big[\langle v_i, w_k \rangle^2 \Big] \leq \frac{13D}{|X|}, \end{aligned}$$

which implies there exists $i \in Y$ such that $\|\operatorname{proj}_U(v_i)\|^2 \leq \frac{13D}{|X|} = O\left(\frac{D}{|X|}\right)$, as desired. ◀

We can now prove our main theorem.

Proof of Theorem 9. The proof follows several steps.

[1] Note that the argument implies that the vectors $\{v_i\}_{i \in Y}$ are linearly independent; otherwise, we can find representations of w for which $\sum_{i \in Y} c_i^2$ is arbitrary large.

STEP 1: Since f satisfies the left one-to-one condition, by Proposition 6, B computes a one-to-one mapping; hence, we can extend it to a permutation $\tau\colon \{-1,1\}^{n+k} \to \{-1,1\}^{n+k}$ as follows:

$$\tau(y_1,\ldots,y_{n+k}) = \begin{cases} B(y_1,\ldots,y_n) & \text{if } y_{n+1} = \cdots = y_{n+k} = 1, \\ \text{arbitrary choice} & \text{otherwise,} \end{cases}$$

where by arbitrary choice we mean one of the $(2^{n+k}-2^n)!$ possible ways of completing the definition so as to yield a permutation. Define $\sigma = \tau^{-1}$ and note that σ is a permutation as well.

STEP 2: For every $x \in \{-1,1\}^n$ and $R \subseteq \{n+1,\ldots,n+k\}$, define $f_x^R\colon \{-1,1\}^{n+k} \to \{-1,1\}^{n+k}$ by

$$f_x^R(y_1,\ldots,y_{n+k}) = \begin{cases} f_x(y_1,\ldots,y_n) & \text{if } y_{n+1} = \cdots = y_{n+k} = 1, \\ \chi_{S(x)}(y_1,\ldots,y_n)\cdot\chi_R(y_{n+1},\ldots,y_{n+k}) & \text{otherwise.} \end{cases}$$

What can we say about these functions?

- Fix $x \in \{-1,1\}^n$, and denote by C_x the circuit obtained from C when Alice is given x as input. Now, consider $y \in \{-1,1\}^{n+k}$.
  - If $y_{n+1} = \cdots = y_{n+k} = 1$, then $f_x^R(y) = f_x(y)$ by definition; hence, by the correctness of $\mathcal{P}$ and the definition of σ, we have that f_x^R agrees with $C_x \circ \sigma^{-1}$ on all such y's.
  - Otherwise, let $i \in \{n+1,\ldots,n+k\}$ such that $y_i = -1$. For every $R \subseteq \{n+1,\ldots,n+k\}$ that contains i, we have that $C_x \circ \sigma^{-1}$ agrees with exactly one of $f_x^R, f_x^{R\setminus\{i\}}$ on the input $(y_1,\ldots,y_{n+i-1},-1,y_{n+i+1},\ldots,y_{n+k})$; thus, for exactly half the subsets $R \subseteq \{n+1,\ldots,n+k\}$, f_x^R agrees with $C_x \circ \sigma^{-1}$ on y. Therefore,

$$\Pr_{\boldsymbol{R}\sim 2^{[n+k]\setminus[n]}}\left[f_x^{\boldsymbol{R}}(y) = C_x(\sigma^{-1}(y))\right] = \frac{1}{2}.$$

  This holds for any y such that $(y_{n+1},\ldots,y_{n+k}) \neq (1,\ldots,1)$; hence,

$$\mathop{\mathrm{E}}_{\substack{\boldsymbol{y}\sim\{-1,1\}^{n+k} \\ \exists j\in[k]\colon \boldsymbol{y}_{n+j}=-1}}\left[\Pr_{\boldsymbol{R}\sim 2^{[n+k]\setminus[n]}}[f_x^{\boldsymbol{R}}(\boldsymbol{y}) = C_x(\sigma^{-1}(\boldsymbol{y}))]\right] = \frac{1}{2},$$

  which implies there exists $R(x) \subseteq [n+k]\setminus[n]$ such that

$$\Pr_{\substack{\boldsymbol{y}\sim\{-1,1\}^{n+k} \\ \exists j\in[k]\colon \boldsymbol{y}_{n+j}=-1}}\left[f_x^{R(x)}(\boldsymbol{y}) = C_x(\sigma^{-1}(\boldsymbol{y}))\right] \geq \frac{1}{2},$$

  It follows that the fraction of inputs on which $f_x^{R(x)}$ and $C_x \circ \sigma^{-1}$ agree is at least

$$\frac{2^n + (1/2)\cdot\left(2^{n+k}-2^n\right)}{2^{n+k}} = \frac{1}{2} + \frac{1}{2^{k+1}},$$

  which is also the fraction of inputs on which $F_x^{R(x)} \triangleq f_x^{R(x)} \circ \sigma$ and C_x agree.

- The second thing we observe is that for any $x \neq x' \in X$,

$$\begin{aligned}
\left\langle F_x^{R(x)}, F_{x'}^{R(x')} \right\rangle &= \left\langle f_x^{R(x)} \circ \sigma, f_{x'}^{R(x')} \circ \sigma \right\rangle = \left\langle f_x^{R(x)}, f_{x'}^{R(x')} \right\rangle \\
&= \mathop{\mathrm{E}}_{\boldsymbol{y} \sim \{-1,1\}^{n+k}} \left[f_x^{R(x)}(\boldsymbol{y}) \cdot f_{x'}^{R(x')}(\boldsymbol{y}) \right] \\
&= \mathop{\mathrm{E}}_{\boldsymbol{y} \sim \{-1,1\}^{n}} \left[f_x(\boldsymbol{y}) \cdot f_{x'}(\boldsymbol{y}) \right] \cdot 2^{-k} \\
&\quad + \sum_{\substack{z \in \{-1,1\}^k \\ \exists j \in [k] \colon z_j = -1}} \chi_{R(x)}(z) \cdot \chi_{R(x')}(z) \cdot \mathop{\mathrm{E}}_{\boldsymbol{y} \sim \{-1,1\}^{n}} \left[\chi_{S(x)}(\boldsymbol{y}) \cdot \chi_{S(x')}(\boldsymbol{y}) \right] \cdot 2^{-k} \\
&= \langle f_x, f_{x'} \rangle \cdot 2^{-k} + \sum_{\substack{z \in \{-1,1\}^k \\ \exists j \in [k] \colon z_j = -1}} \chi_{R(x)}(z) \cdot \chi_{R(x')}(z) \cdot \underbrace{\langle \chi_{S(x)}, \chi_{S(x')} \rangle}_{0} \cdot 2^{-k} \\
&= \langle f_x, f_{x'} \rangle \cdot 2^{-k},
\end{aligned}$$

  which implies

$$\mathop{\mathrm{E}}_{\boldsymbol{x} \neq \boldsymbol{x}' \sim X} \left[\left\langle F_{\boldsymbol{x}}^{R(\boldsymbol{x})}, F_{\boldsymbol{x}'}^{R(\boldsymbol{x}')} \right\rangle^2 \right] \leq 2^{-2k} \cdot \mathop{\mathrm{E}}_{\boldsymbol{x} \neq \boldsymbol{x}' \sim X} \left[\langle f_x, f_{x'} \rangle^2 \right] \underset{\text{assumption}}{\leq} 2^{-2k} \cdot \frac{2^{2k}}{36|X|^2} = \frac{1}{36|X|^2}.$$

STEP 3: Let V be the inner product space of all functions $\{-1,1\}^{n+k} \to \mathbb{R}$, and let $U \leq V$ be the subspace of all functions of degree up to t, which is spanned by $\{\chi_T\}_{|T| \leq t}$ and has dimension $\dim U = \binom{n+k}{\leq t}$. By the Projection Lemma, there exists $x^* \in X$ such that

$$\left\| F_{x^*}^{R(x^*)^{\leq t}} \right\|^2 \leq \frac{13\binom{n+k}{\leq t}}{|X|} \underset{\text{assumption}}{\leq} \frac{1}{2^{2(k+1)}} \implies \left\| F_{x^*}^{R(x^*)^{\leq t}} \right\| \leq \frac{1}{2^{k+1}}.$$

Since $\left\| F_{x^*}^{R(x^*)^{\leq t}} \right\| \leq \frac{1}{2^{k+1}}$ and C_{x^*} is an unbounded fan-in circuit of depth h and size $\leq M$ that agrees with $F_{x^*}^{R(x^*)}$ on at least $\frac{1}{2} + \frac{1}{2^{k+1}}$ fraction of inputs, by Proposition 18,

$$M \geq 2^{\Omega_h\left(\left[\frac{t}{1-2\log\left(2^{-(k+1)}\right)}\right]^{1/(h-1)}\right)} = 2^{\Omega_h\left(\left[\frac{t}{2k+3}\right]^{1/(h-1)}\right)} = 2^{\Omega_h\left(\left[\frac{t}{k}\right]^{1/(h-1)}\right)}. \qquad \blacktriangleleft$$

6 Encoded-input pseudorandom functions

The goal of this section is to prove Theorem 14, which shows that weak PRFs are hard for our model.

▶ **Proposition 20** (Expected inner product bound for weak PRFs). *Let $\delta \in (0,1]$, and suppose that $F \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a weak $(m, \frac{1}{m})$-PRF for $m = \Omega\big((1/\delta)^2 \ln(4/\delta) \cdot n\big)$. Then:*

$$\mathop{\mathrm{E}}_{\boldsymbol{k}, \boldsymbol{k}' \sim \mathcal{K}} \left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'} \rangle^2 \right] \leq 4\delta.$$

Proof. We switch notation to $F \colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$. The proof follows a hybrid argument.

Consider the following algorithm $M(f,g)$ which is given access to a pair of functions f, g and operates as follows:

1. M chooses uniformly and independently $N = 32(1/\delta)^2 \ln(4/\delta)$ random inputs $\vec{\boldsymbol{x}} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)})$.

2. M estimates $\langle f, g\rangle$ with the following estimator:

$$\hat{\boldsymbol{\theta}} = \frac{1}{N}\sum_{i\in[N]} f(\boldsymbol{x}^{(i)})g(\boldsymbol{x}^{(i)}).$$

3. M outputs 1 if $\hat{\boldsymbol{\theta}}^2 > \delta/2$, and 0 otherwise.

Suppose that f, g are randomly chosen. Denote $\boldsymbol{Z}_i = \boldsymbol{f}(\boldsymbol{x}^{(i)})\boldsymbol{g}(\boldsymbol{x}^{(i)})$ for every $i \in [N]$, and $\boldsymbol{Z} = \sum_{i\in[N]} \boldsymbol{Z}_i$. We have $\mathrm{E}_{\boldsymbol{f},\boldsymbol{g}}[\boldsymbol{Z}_i] = 0$ for every $i \in [N]$, implying $\mathrm{E}_{\boldsymbol{f},\boldsymbol{g}}[\boldsymbol{Z}] = 0$.[2] Thus,

$$\Pr_{\boldsymbol{f},\boldsymbol{g}}[M(\boldsymbol{f},\boldsymbol{g}) = 1] = \Pr_{\boldsymbol{f},\boldsymbol{g}}\left[\left|\hat{\boldsymbol{\theta}}\right| > \sqrt{\delta/2}\right] = \Pr\left[\left|\boldsymbol{Z} - \mathrm{E}[\boldsymbol{Z}]\right| > N\sqrt{\delta/2}\right] \underset{\text{Hoeffding}}{\leq} 2e^{-N\delta/4}.$$

To establish the hybrid argument, we define two distinguishers:

- Algorithm $A^f(1^n)$: runs $M(f, \boldsymbol{g})$, where $\boldsymbol{g}$ is chosen uniformly at random by A. This means that whenever M wishes to access $\boldsymbol{g}$, A chooses a random answer and passes it to M; to be consistent, A records past answers.
- Algorithm $B^g(1^n)$: runs $M(F_{\boldsymbol{k}'}, g)$, where $\boldsymbol{k}'$ is chosen uniformly at random by B. This means that B draws $\boldsymbol{k}'$ once at the beginning, and that F is accessible.

Observe that $\Pr_{\boldsymbol{k}}[A^{F_{\boldsymbol{k}}}(1^n) = 1] = \Pr_{\boldsymbol{g}}[B^{\boldsymbol{g}}(1^n) = 1]$. Thus,

$$\begin{aligned}
&\left|\Pr_{\boldsymbol{f},\boldsymbol{g}}[M(\boldsymbol{f},\boldsymbol{g}) = 1] - \Pr_{\boldsymbol{k},\boldsymbol{k}'}[M(F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}) = 1]\right| \\
&= \left|\Pr_{\boldsymbol{f}}[A^{\boldsymbol{f}}(1^n) = 1] - \Pr_{\boldsymbol{k}}[B^{F_{\boldsymbol{k}}}(1^n) = 1]\right| \\
&\leq \left|\Pr_{\boldsymbol{f}}[A^{\boldsymbol{f}}(1^n) = 1] - \Pr_{\boldsymbol{k}}[A^{F_{\boldsymbol{k}}}(1^n) = 1]\right| + \left|\Pr_{\boldsymbol{k}}[B^{F_{\boldsymbol{k}}}(1^n) = 1] - \Pr_{\boldsymbol{g}}[B^{\boldsymbol{g}}(1^n) = 1]\right|.
\end{aligned}$$

Both A^f and B^g require circuits of size $m = O(Nn) = O\big((1/\delta)^2 \ln(4/\delta) \cdot n\big)$. Thus, by definition,

$$\left|\Pr_{\boldsymbol{f},\boldsymbol{g}}[M(\boldsymbol{f},\boldsymbol{g}) = 1] - \Pr_{\boldsymbol{k},\boldsymbol{k}'}[M(F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}) = 1]\right| \leq \tfrac{2}{m} \leq \tfrac{2}{N},$$

which implies

$$\Pr_{\boldsymbol{k},\boldsymbol{k}'}[M(F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}) = 1] \leq \Pr_{\boldsymbol{f},\boldsymbol{g}}[M(\boldsymbol{f},\boldsymbol{g}) = 1] + \tfrac{2}{N} \leq 2e^{-N\delta/4} + \tfrac{2}{N}.$$

Consider now running $M(F_{\boldsymbol{k}}, F_{\boldsymbol{k}'})$ with $\boldsymbol{k}, \boldsymbol{k}'$ chosen uniformly at random.

- By the analysis above: $\Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\hat{\boldsymbol{\theta}}^2 > \delta/2\right] \leq 2e^{-N\delta/4} + \frac{2}{N}$.
- Applying Hoeffding's inequality once more,

$$\begin{aligned}
\Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\left|\hat{\boldsymbol{\theta}}^2 - \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2\right| > \delta/2\right] &= \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\left|\hat{\boldsymbol{\theta}} - \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle\right| > \frac{\delta/2}{\left|\hat{\boldsymbol{\theta}} + \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle\right|}\right] \\
&\leq \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\left|\hat{\boldsymbol{\theta}} - \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle\right| > \delta/4\right] \\
&\leq 2e^{-N\delta^2/32}.
\end{aligned}$$

[2] To ease notation, we shall omit references to $\vec{\boldsymbol{x}}$ when writing probabilities and expectations, yet we should keep in mind that these are taken with respect to the random choice of $\vec{\boldsymbol{x}}$ as well.

Thus,

$$\begin{aligned}\Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 > \delta\right] &= \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 - \hat{\boldsymbol{\theta}}^2 + \hat{\boldsymbol{\theta}}^2 > \delta\right] \\ &\leq \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 - \hat{\boldsymbol{\theta}}^2 > \delta/2\right] + \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\hat{\boldsymbol{\theta}}^2 > \delta/2\right] \\ &\leq 2e^{-N\delta^2/32} + 2e^{-N\delta/4} + \tfrac{2}{N} \\ &\underset{\delta\in(0,1]}{\leq} 4e^{-N\delta^2/32} + \tfrac{2}{N}.\end{aligned}$$

Therefore,

$$\begin{aligned}\mathop{\mathrm{E}}_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2\right] &= \mathop{\mathrm{E}}_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 \,\middle|\, \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 > \delta\right] \cdot \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 > \delta\right] \\ &\quad + \mathop{\mathrm{E}}_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 \,\middle|\, \langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 \leq \delta\right] \cdot \Pr_{\boldsymbol{k},\boldsymbol{k}'}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2 \leq \delta\right] \\ &\leq 1 \cdot (4e^{-N\delta^2/32} + \tfrac{2}{N}) + \delta \cdot 1 \\ &= \delta + 4e^{-N\delta^2/32} + \tfrac{2}{N} \\ &\leq 2\delta + 2\delta = 4\delta,\end{aligned}$$

the last inequality holding since $N = 32(1/\delta)^2 \ln(4/\delta) \geq 1/\delta$. ◀

▶ **Proposition 21** (General lower bound for weak PRFs). *There exist a constant $0 < a \leq 1/2$ such that for every $0 \leq r \leq n/2 - 1$ and integer $0 \leq t \leq a(n+r)$ the following holds: If $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ satisfies the right one-to-one condition and is a weak $(m, \frac{1}{m})$-PRF for*

$$m = \Omega\left(n \cdot 2^{4r} \cdot \binom{n+r}{t}^4 \left[r + \log\binom{n+r}{t}\right]\right),$$

and F admits an SM protocol (A, B, C) such that $B\colon \{0,1\}^n \to \{0,1\}^{n+r}$ and C is an unbounded fan-in circuit of depth h and size M, then

$$M \geq 2^{\Omega_h\left(\left[\frac{t}{r}\right]^{1/(h-1)}\right)}.$$

Proof. Let us define:

$$s \triangleq 13 \cdot 2^{2(r+1)} \cdot \binom{n+r}{t} \quad , \quad \delta \triangleq \frac{2^{2r-2}}{36s^2} = \frac{1}{36 \cdot 169 \cdot 2^{2r+6} \cdot \binom{n+r}{t}^2}.$$

We have:

$$(1/\delta)^2 \ln(1/\delta) = \Theta\left(2^{4r} \cdot \binom{n+r}{t}^4 \left[r + \log\binom{n+r}{t}\right]\right).$$

Thus, assuming $m = \Omega\big((1/\delta)^2 \ln(4/\delta) \cdot n\big)$, by assumption and Proposition 20, we get

$$\mathop{\mathrm{E}}_{\boldsymbol{k}\neq\boldsymbol{k}'\sim\{0,1\}^n}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2\right] \leq 4\delta = \frac{2^{2r}}{36s^2}.$$

In particular, there exists a set $X \subseteq \{0,1\}^n$ of size $|X| = s$ such that

$$\mathop{\mathrm{E}}_{\boldsymbol{k}\neq\boldsymbol{k}'\sim X}\left[\langle F_{\boldsymbol{k}}, F_{\boldsymbol{k}'}\rangle^2\right] \leq \frac{2^{2r}}{36s^2}.$$

We need to justify why $s \le 2^n$. As shown in the proof of Proposition 10, there exists $0 < a \le 1/2$ such that

$$13 \cdot 2^{2(r+1)} \cdot \binom{n+r}{\le a(n+r)} \le 2^n;$$

hence, for $t = a(n+r)$ we have $s \le 2^n$.[3] Thus, by Theorem 9,

$$M \ge 2^{\Omega_h\left(\left[\frac{t}{r}\right]^{1/(h-1)}\right)}.$$ ◀

Theorem 14 is an immediate corollary.

7 Rounded inner product

In this section we prove Theorem 16, an IPPP-style theorem (with sublinear stretch) for a class of functions obtained by applying a "rounding predicate" to an inner product modulo q. We remind the reader that these functions are given in Definition 15.

The following proposition will be useful.

▶ **Proposition 22** (Inner product convergence). *Let $q \ge 2$ be an integer. Then, for every $r \in \{0, \dots, q-1\}$,*

$$\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\{0,1\}^{2n}}\left[\sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{y}_i \pmod q = r\right] \xrightarrow[n\to\infty]{} \frac{1}{q}.$$

Moreover, there exists $0 < c < 1$ such that for every $r \in \{0, \dots, q-1\}$, for large enough n,

$$\Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\{0,1\}^{2n}}\left[\sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{y}_i \pmod q = r\right] = \frac{1}{q} \pm O(c^n).$$

Proof. For the finite state space $Q = \{0, \dots, q-1\}$ of remainders modulo q, we define a sequence of random variables $\boldsymbol{Z}_0, \boldsymbol{Z}_1, \dots, \boldsymbol{Z}_n$ by

$$\boldsymbol{Z}_i = \begin{cases} 0 & i = 0, \\ \boldsymbol{Z}_{i-1} + \boldsymbol{x}_i\boldsymbol{y}_i \pmod q & i \in [n], \end{cases}$$

where $(\boldsymbol{x}, \boldsymbol{y}) \sim \{0,1\}^{2n}$. We are interested in $\lim_{n\to\infty} \Pr[\boldsymbol{Z}_n = r \mid \boldsymbol{Z}_0 = 0]$. For every $i \in [n]$, we have

$$\Pr[\boldsymbol{Z}_i \mid \boldsymbol{Z}_{i-1}] = \Pr[\boldsymbol{Z}_i \mid \boldsymbol{Z}_{i-1}, \dots, \boldsymbol{Z}_0],$$

and for every $i \in [n]$ and $u \in Q$, we have

$$\Pr[\boldsymbol{Z}_i = u \mid \boldsymbol{Z}_{i-1} = u] = 3/4,$$
$$\Pr[\boldsymbol{Z}_i = u + 1 \pmod q \mid \boldsymbol{Z}_{i-1} = u] = 1/4.$$

Thus, the sequence $(\boldsymbol{Z}_i)$ forms a Markov chain, which we claim is *ergodic*. To see that, consider a walk of q steps; then, for every $u, v \in Q$, we have $\Pr[\boldsymbol{Z}_{i+q} = u \mid \boldsymbol{Z}_i = v] \ge (\frac{1}{4})^q > 0$. Since $(\boldsymbol{Z}_i)$ is a finite ergodic Markov chain, it follows that there exists a unique stationary

[3] Note that for $a \le 1/2$, the function $t \mapsto \binom{n+r}{\le t}$ is monotone on $[0, a(n+r)]$.

distribution π over Q such that for every $r \in Q$, we have $\lim_{n\to\infty} \Pr[\boldsymbol{Z}_n = r \mid \boldsymbol{Z}_0 = 0] = \pi(r)$. It is easy to verify that $\pi^* = (\frac{1}{q}, \ldots, \frac{1}{q})$ is a distribution over Q which satisfies $\pi^* = \pi^* P$, where P is the transition matrix of the Markov chain, given by

$$P = \begin{bmatrix} 3/4 & 1/4 & 0 & \dots & 0 \\ 0 & 3/4 & 1/4 & \dots & 0 \\ 0 & 0 & 3/4 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/4 & 0 & 0 & \dots & 3/4 \end{bmatrix}.$$

It follows that $\pi = \pi^*$, hence

$$\lim_{n\to\infty} \Pr_{(\boldsymbol{x},\boldsymbol{y})\sim\{0,1\}^{2n}} \left[\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{y}_i \pmod q = r \right] = \lim_{n\to\infty} \Pr[\boldsymbol{Z}_n = r \mid \boldsymbol{Z}_0 = 0] = \frac{1}{q}.$$

The second part of the claim follows from known properties of convergence to a stationary distribution. ◀

We are now ready to prove the lower bound for computing rounded inner products in our setting.

Proof of Theorem 16. Let $y, z \in \{0,1\}^n$ be such that the Hamming distance between y and z is at least $n/3$, and let S_y and S_z be the subsets of $[n]$ characterized by y and z, respectively. Without loss of generality, we may assume that $|S_y \setminus S_z| \geq n/6$, and let us denote $J \triangleq S_y \setminus S_z$.

For $x \in \{0,1\}^n$, let us write $x = (u, v)$ with $u \in \{0,1\}^J$ and $v \in \{0,1\}^{[n]\setminus J}$. Fix a v now. Define

$$a_v \triangleq \sum_{i\in[n]\setminus J} v_i y_i \quad , \quad b_v \triangleq \sum_{i\in[n]\setminus J} v_i z_i,$$

and observe that a_v, b_v are also fixed. We have

$$\sum_{i=1}^{n} x_i y_i \pmod q = \left(\sum_{i\in J} u_i + \sum_{i\in[n]\setminus J} v_i y_i \right) \pmod q = \left(\sum_{i\in J} u_i + a_v \right) \pmod q,$$

$$\sum_{i=1}^{n} x_i z_i \pmod q = \sum_{i\in[n]\setminus J} v_i z_i \pmod q = b_v \pmod q.$$

It follows that there exists a subset $R_v \subseteq \{0, 1, \ldots, q-1\}$ of size $q/2$ such that

$$\mathsf{IP}^{[q,R]}((u,v),y) = \mathsf{IP}^{[q,R]}((u,v),z) \iff \sum_{i\in J} u_i \pmod q \in R_v.$$

Therefore, by Proposition 22, for any fixed v and large enough n,

$$\begin{aligned} \Pr_{\boldsymbol{u}\sim\{0,1\}^J} \left[\mathsf{IP}^{[q,R]}((\boldsymbol{u},v),y) = \mathsf{IP}^{[q,R]}((\boldsymbol{u},v),z) \right] &= \Pr_{\boldsymbol{u}\sim\{0,1\}^J} \left[\sum_{i\in J} \boldsymbol{u}_i \pmod q \in R_v \right] \\ &= |R_v| \cdot \left(\frac{1}{q} \pm O(c^n) \right) = \frac{1}{2} \pm O(c^n), \end{aligned}$$

which implies

$$\Pr_{\boldsymbol{x}} \left[\mathsf{IP}^{[q,R]}(\boldsymbol{x},y) = \mathsf{IP}^{[q,R]}(\boldsymbol{x},z) \right] = \mathop{\mathrm{E}}_{\boldsymbol{v}} \left[\Pr_{\boldsymbol{u}} \left[\mathsf{IP}^{[q,R]}((\boldsymbol{u},v),y) = \mathsf{IP}^{[q,R]}((\boldsymbol{u},v),z) \right] \right] = \frac{1}{2} \pm O(c^n).$$

Considering $\mathsf{IP}^{[q,R]}(x,y)$ and $\mathsf{IP}^{[q,R]}(x,z)$ as functions of x, and switching to $\{0,1\}^n \to \{-1,1\}$ notation, we get

$$\left\langle \mathsf{IP}^{[q,R]}(\cdot,y), \mathsf{IP}^{[q,R]}(\cdot,z) \right\rangle = 2 \Pr_{\boldsymbol{x}}\left[\mathsf{IP}^{[q,R]}(\boldsymbol{x},y) = \mathsf{IP}^{[q,R]}(\boldsymbol{x},z)\right] - 1 = \pm O(c^n).$$

Finally, we have:

- $\mathsf{IP}^{[q,R]}$ satisfies the right one-to-one condition.
- The Gilbert–Varshamov bound [18, 33] tells us there exists $\mathcal{C} \subseteq \{0,1\}^n$ of size $2^{\Omega(n)}$ and minimal Hamming distance $n/3$. By the analysis above, there exists a constant $K > 0$ such that (for large enough n) $|\langle f_x, f_{x'} \rangle| \leq Kc^n$ for every $x \neq x' \in \mathcal{C}$. Define $s = \min\left\{|\mathcal{C}|, \frac{1}{6K} 2^{\log(1/c)n}\right\}$, and let $0 < \alpha \leq 1/2$ be such that $\mathrm{H}(\alpha)$ is small enough so setting $t = \alpha(n+k)$ gives us

$$13 \cdot 2^{2(k+1)} \cdot \binom{n+k}{\leq \alpha(n+k)} \underset{\text{Lemma 8}}{\leq} 2^{\mathrm{H}(\alpha)(n+k)+2(k+1)+4} \leq s.$$

  It follows that any set $X \subseteq \mathcal{C}$ of size s satisfies both $13 \cdot 2^{2(k+1)} \cdot \binom{n+k}{\leq \alpha(n+k)} \leq |X|$ and

$$|X| \leq \frac{1}{6K} 2^{\log(1/c)n} \implies Kc^n \leq \frac{1}{6|X|},$$

  which implies

$$\mathop{\mathrm{E}}_{\boldsymbol{x} \neq \boldsymbol{x}' \sim X}\left[\langle f_{\boldsymbol{x}}, f_{\boldsymbol{x}'} \rangle^2\right] \leq K^2 c^{2n} \leq \frac{1}{36|X|^2} \leq \frac{2^{2k}}{36|X|^2}.$$

Thus, by Theorem 9,

$$M \geq 2^{\Omega_h\left(\left[\frac{t}{k}\right]^{1/(h-1)}\right)} = 2^{\Omega_h\left(n^{\frac{1-\alpha}{h-1}}\right)}.$$ ◀

References

1. Miklós Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.
2. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $\mathsf{AC}^0 \circ \mathsf{MOD}_2$. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260, 2014. doi:10.1145/2554797.2554821.
3. Josh Alman and R. Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 641–652, 2017. doi:10.1145/3055399.3055484.
4. László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 337–347, 1986. doi:10.1109/SFCS.1986.15.
5. László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003. doi:10.1137/S0097539700375944.
6. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 719–737, 2012. doi:10.1007/978-3-642-29011-4_42.

7 Avrim Blum, Merrick L. Furst, Jeffrey C. Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 253–262. ACM, 1994. doi:10.1145/195058.195147.

8 Andrej Bogdanov, Yuval Ishai, and Akshayaram Srinivasan. Unconditionally secure computation against low-complexity leakage. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 387–416, 2019. doi:10.1007/978-3-030-26951-7_14.

9 Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography*, pages 79–158. Springer, Cham, 2017. doi:10.1007/978-3-319-57048-8_3.

10 Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: - new simple PRF candidates and their applications. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, 2018. doi:10.1007/978-3-030-03810-6_25.

11 Arkadev Chattopadhyay and Rahul Santhanam. Lower bounds on interactive compressibility by constant-depth circuits. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 619–628, 2012. doi:10.1109/FOCS.2012.74.

12 Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $\mathsf{AC}^0 \circ \mathsf{MOD}_2$ lower bounds for the boolean inner product. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 35:1–35:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.35.

13 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 441–448, 2014. doi:10.1145/2591796.2591820.

14 Ning Ding, Yanli Ren, and Dawu Gu. PAC learning depth-3 AC^0 circuits of bounded top fanin. In *International Conference on Algorithmic Learning Theory, ALT 2017, 15-17 October 2017, Kyoto University, Kyoto, Japan*, pages 667–680, 2017. URL: http://proceedings.mlr.press/v76/ding17a.html.

15 Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 711–720, 2006. doi:10.1145/1132516.1132615.

16 Zeev Dvir and Benjamin Edelman. Matrix rigidity and the croot-lev-pach lemma. *CoRR*, abs/1708.01646, 2017. arXiv:1708.01646.

17 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 260–270, 1981. doi:10.1109/SFCS.1981.35.

18 Edgar N. Gilbert. A comparison of signalling alphabets. *The Bell system technical journal*, 31(3):504–522, 1952.

19 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

20 Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 86:1–86:15, 2016. doi:10.4230/LIPIcs.ICALP.2016.86.

21 Robert M. Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

22 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. doi:10.1145/12130.12132.

23 Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 392–401. ACM, 1993. doi:10.1145/167088.167200.

24 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 574–579, 1989. doi:10.1109/SFCS.1989.63537.

25 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Inf.*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

26 Alexander A. Razborov. On rigid matrices. *preprint*, 1989.

27 Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569, 2012. doi:10.1007/978-3-642-32009-5_32.

28 Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:144, 2012. URL: http://eccc.hpi-web.de/report/2012/144.

29 Avishay Tal. The bipartite formula complexity of inner-product is quadratic. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:181, 2016. URL: http://eccc.hpi-web.de/report/2016/181.

30 Avishay Tal. Tight bounds on the fourier spectrum of AC^0. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 15:1–15:31, 2017. doi:10.4230/LIPIcs.CCC.2017.15.

31 Salil P. Vadhan. On learning vs. refutation. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 1835–1848, 2017. URL: http://proceedings.mlr.press/v65/vadhan17a.html.

32 Leslie G. Valiant. A theory of the learnable. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 436–445, 1984. doi:10.1145/800057.808710.

33 R. R. Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

34 Emanuele Viola. Extractors for circuit sources. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 220–229, 2011. doi:10.1109/FOCS.2011.20.

35 Henning Wunderlich. On a theorem of razborov. *Computational Complexity*, 21(3):431–477, 2012. doi:10.1007/s00037-011-0021-5.

36 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979. doi:10.1145/800135.804414.

Sign Rank vs Discrepancy

Hamed Hatami
McGill University, Montreal, Canada
hatami@cs.mcgill.ca

Kaave Hosseini
Carnegie Mellon University, Pittsburgh, PA, USA
kaave.hosseini@gmail.com

Shachar Lovett
University of California San Diego, CA, USA
slovett@ucsd.edu

Abstract

Sign-rank and discrepancy are two central notions in communication complexity. The seminal work of Babai, Frankl, and Simon from 1986 initiated an active line of research that investigates the gap between these two notions. In this article, we establish the strongest possible separation by constructing a boolean matrix whose sign-rank is only 3, and yet its discrepancy is $2^{-\Omega(n)}$. We note that every matrix of sign-rank 2 has discrepancy $n^{-O(1)}$.

Our result in particular implies that there are boolean functions with $O(1)$ *unbounded error* randomized communication complexity while having $\Omega(n)$ *weakly unbounded error* randomized communication complexity.

2012 ACM Subject Classification Theory of computation → Communication complexity

Keywords and phrases Discrepancy, sign rank, Unbounded-error communication complexity, weakly unbounded error communication complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.18

Funding *Hamed Hatami*: Supported by an NSERC grant.
Kaave Hosseini: Supported by NSF grant CCF-1614023.
Shachar Lovett: Supported by NSF grant CCF-1614023.

1 Introduction

Sign-rank and discrepancy are arguably the most important analytic notions in the area of communication complexity. Let A be a matrix with $\{-1,1\}$ entries (we refer to these matrices as boolean matrices in this paper). The *discrepancy* of A is the minimum over all input distribution of the maximum correlation that A has with a rectangle (for a formal definition see Section 2). It was introduced by Chor and Goldreich [8], and has become one of the most commonly used measures in communication complexity to prove lower bounds for randomized protocols. The *sign-rank* of A is the minimal rank of a real matrix whose entries have the same sign pattern as A. This natural and fundamental notion was first introduced by Paturi and Simon [16] in the context of the unbounded error communication complexity. Since then, its applications have extended beyond communication complexity to areas such as circuit complexity [17, 6], learning theory [12, 13, 10], and even connections to algebraic geometry [23].

Boolean matrices in communication complexity correspond to boolean functions: give an n-bits two player function $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$, it corresponds to the $2^n \times 2^n$ matrix $A_{x,y} = f(x,y)$. The notions of discrepancy and sign-rank for f correspond to its respective matrix. The main informal question motivating this work is:

▶ **Problem 1.** *Does every function of low sign-rank have an efficient randomized protocol?*

If the answer is negative, then the next question is, does it at least have large discrepancy (small discrepancy is one technique to prove randomized communication complexity lower bounds, but there are functions showing separations between the two measures, for example set-disjointness [7]).

▶ **Problem 2.** *Does every function of low sign-rank have large discrepancy?*

In order to build some intuition towards more quantitative questions, lets consider some well-known examples:

- Greater-than: we interpret x, y as integers in $\{1, \ldots, 2^n\}$ and define $f(x, y) = 1$ if $x \leq y$ and $f(x, y) = -1$ otherwise. This function has sign-rank 2 and requires $\Theta(\log n)$ bits of randomized communication [15]. Moreover, its discrepancy is $n^{-\Theta(1)}$, which proves the communication lower bound.
- Set-disjointness: we interpret x, y as subsets of $[n]$, and define $f(x, y) = 1$ if x, y are disjoint and $f(x, y) = -1$ otherwise. This function has sign-rank $O(n)$ and requires communication complexity of $\Theta(n)$ bits. However, this cannot be shown using discrepancy, as the discrepancy of set-disjointness is $n^{-O(1)}$ [7].
- Sherstov [21] constructed a function with sign-rank $O(n)$ and discrepancy $2^{-\Omega(n)}$.

Thus, it seems that functions with logarithmic sign-rank can already be very complicated, both in terms of their randomized communication complexity and also in terms of their discrepancy. However, the situation is less clear for functions of *constant* sign-rank.

▶ **Problem 3.** *Does every function of constant sign-rank have an efficient randomized protocol? in particular, does it have large discrepancy?*

Our main result is a sounding no, already for sign-rank 3.

▶ **Theorem 4** (Main Theorem; informal version)**.** *There exists a function $f : \{0,1\}^n \times \{0,1\}^n \to \{-1, 1\}$ of sign-rank 3 and discrepancy $2^{-\Omega(n)}$. In particular, f has $\Omega(n)$ randomized communication complexity.*

The sign-rank 3 in Theorem 4 is tight. We show in Section 3 that functions of sign-rank 1 or 2 are very simple combinatorially, and in particular have discrepancy $n^{-O(1)}$ and randomized communication complexity $O(\log n)$.

The function f in Theorem 4 is simple to define: the sign on an inner product in dimension 3. Concretely, let $M \approx 2^{n/3}$. Alice gets a vector $\mathbf{a} \in [-M, M]^3$ and Bob gets a vector $\mathbf{b} \in [-M, M]^3$. Define

$$f(\mathbf{a}, \mathbf{b}) = \text{sign}\langle \mathbf{a}, \mathbf{b} \rangle,$$

where $\text{sign} : \mathbb{R} \to \{-1, 1\}$ is the sign function, mapping positive inputs to 1 and zero or negative inputs to -1; and $\langle \cdot, \cdot \rangle$ is inner product over the integers. It is obvious from the definition that f has sign-rank 3. We prove that its discrepancy is exponentially small. The actual function we study is a mild restriction of this function, convenient for the proof. See Theorem 7 for details.

1.1 Connections to communication complexity

Theorem 4 is motivated by its applications in communication complexity. Consider a communication problem $f : \{0,1\}^n \times \{0,1\}^n \to \{-1, 1\}$ in Yao's two party model. Given an error parameter $\epsilon \in [0, 1/2]$, let $R_\epsilon(f)$ be the smallest communication cost of a *private-coin*

randomized communication protocol that on *every* input produces the correct answer with probability at least $1-\epsilon$. Here private-coin refers to the assumption that players each have their own unlimited *private* source of randomness. Three natural complexity measures arise from $R_\epsilon(f)$.

1. The quantity $R_{1/3}(f)$ is called the *bounded-error randomized communication complexity* of f. The particular choice of $1/3$ is not important as long as one is concerned with an error that is bounded away from both 0 and $1/2$ since in this case the error can be reduced by running the protocol constantly many times and outputting the majority answer.
2. The *weakly unbounded error randomized communication complexity* of f is defined as
$$\mathrm{PP}(f) = \inf_{0\le\epsilon\le 1/2}\left\{R_\epsilon(f) + \log\frac{1}{1-2\epsilon}\right\},$$
that includes an additional penalty term, which increases as ϵ approaches $\frac{1}{2}$. The purpose of this error term is to capture the range where ϵ is "moderately" bounded away from $\frac{1}{2}$.
3. Finally the *unbounded error communication complexity* of f is defined as the smallest communication cost of a private-coin randomized communication protocol that computes every entry of f with an error probability that is *strictly* smaller than $\frac{1}{2}$. In other words, the protocol only needs to outdo a random guess, which is always correct with probability $\frac{1}{2}$. We have
$$\mathrm{UPP}(f) = \lim_{\epsilon\nearrow\frac{1}{2}} R_\epsilon(f).$$

In their seminal paper, Babai, Frankl and Simon [2] associated a complexity class to each measure of communication complexity. While in the theory of Turing machines, a complexity that is polynomial in the size of input bits is considered efficient, in the realm of communication complexity, poly-logarithmic complexity plays this role, and communication complexity classes are defined accordingly. Here, the communication complexity classes $\mathrm{BPP}^{\mathrm{cc}}$, $\mathrm{PP}^{\mathrm{cc}}$, and $\mathrm{UPP}^{\mathrm{cc}}$ correspond to the class of communication problems $\{f_n\}_{n=0}^{\infty}$ with polylogarithmic $R_{1/3}(f_n)$, $\mathrm{PP}(f_n)$, and $\mathrm{UPP}(f_n)$, respectively.

Note that while $\mathrm{BPP}^{\mathrm{cc}}$ requires a strong bound on the error probability, and $\mathrm{UPP}^{\mathrm{cc}}$ only requires an error that beats the random guess, $\mathrm{PP}^{\mathrm{cc}}$ corresponds to the natural requirement that the protocol beats the $\frac{1}{2}$ bound by a margin that is quasi-polynomially large. That is, $\mathrm{PP}^{\mathrm{cc}}$ is the class of communication problems f_n that satisfy $R_{\frac{1}{2}-2^{-\log^c(n)}}(f_n) \le \log^c(n)$ for some positive constant c. We have the containment $\mathrm{BPP}^{\mathrm{cc}} \subseteq \mathrm{PP}^{\mathrm{cc}} \subseteq \mathrm{UPP}^{\mathrm{cc}}$.

It turns out that both $\mathrm{UPP}(f)$ and $\mathrm{PP}(f)$ have elegant algebraic formulations. Paturi and Simon [16] proved that UPP essentially coincides with the sign-rank of f:
$$\log \mathbf{rk}_{\pm}(f) \le \mathrm{UPP}(f) \le \log \mathbf{rk}_{\pm}(f) + 2.$$

Similar to the way that sign-rank captures the complexity measure $\mathrm{UPP}(f)$, discrepancy captures $\mathrm{PP}(f)$. The classical result relating randomized communication complexity and discrepancy, due to Chor and Goldreich [8], is the inequality
$$R_\epsilon(f) \ge \log\frac{1-2\epsilon}{\mathrm{Disc}(f)}.$$
This in particular implies $\mathrm{PP}(f) \ge -\log \mathrm{Disc}(f)$. Klauck [9] showed that the opposite is also true; more precisely, that
$$\mathrm{PP}(f) = O\left(-\log \mathrm{Disc}(f) + \log(n)\right).$$
Thus, a direct corollary of Theorem 4 is the following separation between unbounded error and weakly bounded error communication complexity.

▶ **Corollary 5.** *There exists a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ *with* $\mathrm{UPP}(f) = O(1)$ *and* $\mathrm{PP}(f) = \Omega(n)$.

Another closely related notion to sign-rank is approximate rank. Given $\alpha > 1$, the α-approximate rank of a boolean matrix A is the minimal rank of a real matrix B, of the same dimensions as A, that satisfies $1 \leq A_{i,j}B_{i,j} \leq \alpha$ for all i, j. The α-approximate rank of a boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ is the α-approximate rank of the associated $2^n \times 2^n$ boolean matrix. Observe that

$$\mathbf{rk}_{\pm}(f) = \lim_{\alpha \to \infty} \mathbf{rk}^{\alpha}(f).$$

Given this, a natural question is whether sign-rank can be separated from α-approximate rank. This is also a consequence of Theorem 4(in fact to be precise, this is rather a corollary of Theorem 7 which is the formal version of Theorem 4).

▶ **Corollary 6.** *There exists a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$ *with* $\mathbf{rk}_{\pm}(f) = 3$ *and* $\mathbf{rk}^{\alpha}(f) = \Omega(2^{n/4}/(\alpha n)^2)$ *for any* $\alpha > 1$.

Corollary 6 follows from Theorem 4 and the fact that

$$\mathbf{rk}^{\alpha}(f) \geq \Omega\left(\alpha^{-2}\mathrm{Disc}(f)^{-2}\right),$$

which is a combination of the results of Linial and Shraibman [14, Theorem 18] and Lee and Shraibman [11, Theorem 1].

1.2 Related works

The question of separating sign-rank from discrepancy (or equivalently, separating unbounded from weakly unbounded communication complexity) has been studied in a number of papers.

When Babai et al. [2] introduced the complexity classes $\mathrm{BPP}^{\mathrm{cc}} \subseteq \mathrm{PP}^{\mathrm{cc}} \subseteq \mathrm{UPP}^{\mathrm{cc}}$, they noticed that the set-disjointness problem separates $\mathrm{BPP}^{\mathrm{cc}}$ from $\mathrm{PP}^{\mathrm{cc}}$, but they left open the question of separating $\mathrm{UPP}^{\mathrm{cc}}$ from $\mathrm{PP}^{\mathrm{cc}}$, or equivalently sign-rank from discrepancy. This question remained unanswered for more than two decades until finally Buhrman et al. [5] and independently Sherstov [18] showed that there are n-bit boolean function f such that $\mathrm{UPP}(f) = O(\log n)$ but $\mathrm{PP}(f) = \Omega(n^{1/3})$ and $\mathrm{PP}(f) = \Omega(\sqrt{n})$, respectively. The bounds on $\mathrm{PP}(f)$ were strengthened in subsequent works [19, 20, 22, 21] with the final recent separation from [21] giving a function f with $\mathrm{UPP}(f) = O(\log n)$ and maximal possible $\mathrm{PP}(f) = \Omega(n)$. Despite this line of work, no improvement was made on the $O(\log(n))$ bound on $\mathrm{UPP}(f)$. In fact, to the best of our knowledge, prior to this work, it was not even known whether there are functions with $\mathrm{UPP}(f) = O(1)$ and $R_{1/3}(f) = \omega(\log(n))$. To recall, Corollary 5 gives a function f with $\mathrm{UPP}(f) = O(1)$ and $\mathrm{PP}(f) = \Omega(n)$.

A different aspect is the study of sign-rank of matrices. Matrices of sign-rank 1 and 2 are simple combinatorially, while matrices with sign-rank 3 seem to be much more complex. First, it turns out that deciding whether a matrix has sign-rank 3 is NP-hard, a result that was shown by Basri et al. [3] and independently by Bhangale and Kopparty [4]. In fact, deciding if a matrix has sign-rank 3 is $\exists\mathbb{R}$-complete, where $\exists\mathbb{R}$ is the existential first-order theory of the reals, a complexity class lying between NP and PSPACE. This $\exists\mathbb{R}$-completeness result is implicit in both [3] and [4], as observed by [1].

1.3 Proof overview

We give a proof overview of Theorem 4. Let us first slightly modify f in a way that will convenient for the proof.

Let $N \approx 2^{n/4}$. Alice gets three integers x_1, x_2, z and Bob gets two integers y_1, y_2, where $x_1, x_2, y_1, y_2 \in [N]$ and $z \in [-3N^2, 3N^2]$. We shorthand $x = [x_1, x_2]$ and $y = [y_1, y_2]$, so that Alice's input is $[x, z]$ and Bob's input is y. Note that $x, y \in [N]^2$. Define

$$f([x, z], y) = \text{sign}(z - \langle x, y \rangle) = \text{sign}(z - x_1 y_1 - x_2 y_2).$$

The following is our main technical result.

▶ **Theorem 7** (Main result; formal version). *Let f be as above. Then* $\text{Disc}(f) = O(n \cdot 2^{-n/8})$.

We remark that the function f here is a restriction of the function f described before Theorem 4, and therefore, Theorem 7 implies Theorem 4.

To prove Theorem 7, it is useful to think about our discrepancy bound in the language of communication complexity. We prove Theorem 7 in two steps. Below we denote random variables with bold letters.

Step 1: constructing a hard distribution

First, we define a hard distribution ν. Alice and Bob receive uniformly random integers $\mathbf{x}, \mathbf{y} \in [N]^2$ respectively where $N \approx 2^{n/4}$. The inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ is a random variable over $[2N^2]$. Alice also receives another random variable $\mathbf{z}$ over $[-3N^2, 3N^2]$, whose distribution we will explain shortly. The players want to decide whether $\langle \mathbf{x}, \mathbf{y} \rangle \geq \mathbf{z}$. We define $\mathbf{z}$ in such a way that

- $\langle \mathbf{x}, \mathbf{y} \rangle - \mathbf{z} \in [-2N, 2N)$,
- $\langle \mathbf{x}, \mathbf{y} \rangle \geq \mathbf{z}$ happens with probability $\frac{1}{2}$,
- $\langle \mathbf{x}, \mathbf{y} \rangle - \mathbf{z}$ is extremely close in total variation distance to $\langle \mathbf{x}, \mathbf{y} \rangle - \mathbf{z} - 2N$ (which is always negative), even when restricted to arbitrary large combinatorial rectangles.

To construct $\mathbf{z}$, we first define another independent random variable $\mathbf{k}$ and then let $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}$, or $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N$, with equal probabilities. We choose $\mathbf{k} = \mathbf{k}_1 + \mathbf{k}_2$ for $\mathbf{k}_1, \mathbf{k}_2$ independent uniform elements from $[N]$ so that $\mathbf{k}$ is smooth enough for the analysis to go through. Note that the range of $\mathbf{z}$ is really just $[-2N, 2N^2 + 2N]$, and we picked the range of z in the definition of f as $z \in [-3N^2, 3N^2]$ for its simpler shape.

Step 2: translation invariance of $\mathbf{k}$

We bound the discrepancy $\text{Disc}_\nu(f)$ as follows. Fix a combinatorial rectangle $A \times B \subset ([N]^2 \times [-3N^2, 3N^2]) \times [N]^2$. We want to bound the correlation of f with $1_A 1_B$ under the distribution ν. This boils down to showing that after conditioning on the input being in $A \times B$, the distribution $(\langle \mathbf{x}, \mathbf{y} \rangle - \mathbf{z})|_{A,B}$ has small total variation distance with its translation by $2N$. We prove a stronger statement, and show that in fact this is true even if we fix $\mathbf{x} = x$ to a typical x (and therefore choosing $A \subset \{x\} \times [-3N^2, 3N^2]$), namely, after conditioning $\mathbf{x} = x$, and $\mathbf{y} \in B$, the distribution of $(\langle x, \mathbf{y} \rangle - \mathbf{z})|_{\mathbf{y} \in B}$ has small total variation distance with its translation by $2N$. To prove the claim we appeal to Fourier analysis and estimate the Fourier coefficients of the random variable, and verify that the only potentially large Fourier coefficients correspond to Fourier characters that are almost invariant under translations by $2N$. Computing these Fourier coefficients involves computing some partial exponential sums whose details may be seen in Lemma 10 and Lemma 11. At a high level, these boils down to showing that if $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^2$ are two random independent variables, uniform over large sets, then their inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ has well-behaved spectral properties.

Paper organization

We give preliminary definitions need for the proof in Section 2. We discuss the structure of matrices of sign-rank 1 and 2 in Section 3. We prove or main result, Theorem 7, in Section 4.

2 Preliminaries

Notations

To simplify the presentation, we often use $\lesssim$ or $\approx$ instead of the big-O notation. That is, $x \lesssim y$ means $x = O(y)$, and $x \approx y$ means $x = \Theta(y)$. For integers $N \leq M$ we denote $[N, M] = \{N, \ldots, M\}$, and we shorthand $[N] = [1, N]$.

Discrepancy

Let $\mathcal{X}, \mathcal{Y}$ be finite sets, and ν be a probability distribution on $\mathcal{X} \times \mathcal{Y}$. The discrepancy of a function $f : \mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$ with respect to ν and a combinatorial rectangle $A \times B \subseteq \mathcal{X} \times \mathcal{Y}$ is defined as

$$\mathrm{Disc}_\nu^{A\times B}(f) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\nu}\left[f(\mathbf{x},\mathbf{y})1_A(\mathbf{x})1_B(\mathbf{y})\right].$$

The discrepancy of f with respect to ν is defined as

$$\mathrm{Disc}_\nu(f) = \max_{A,B} \mathrm{Disc}_\nu^{A\times B}(f),$$

and finally the discrepancy of f is defined as

$$\mathrm{Disc}(f) = \min_\nu \mathrm{Disc}_\nu(f).$$

Probability

We denote random variables with bold letters. Given a random variable $\mathbf{r}$, let $\mu = \mu_{\mathbf{r}}$ denote its distribution. The conditional distribution of $\mathbf{r}$, conditioned on $\mathbf{r} \in S$ for some set S, is denoted by $\mu|_S$. Given a finite set S, we denote the uniform measure on S by μ_S. If $\mathbf{r}$ is uniformly sampled from S, we denote it by $\mathbf{r} \sim S$.

Fourier analysis

The proof of Theorem 7 is based on Fourier analysis over cyclic groups. We introduce the relevant notation in the following. Let p be a prime. For $f, g : \mathbb{Z}_p \to \mathbb{C}$ define

$$\langle f, g\rangle = \frac{1}{p}\sum_{x\in\mathbb{Z}_p} f(x)\overline{g}(x),$$

and

$$f * g(z) = \frac{1}{p}\sum_{x\in\mathbb{Z}_p} f(x)g(z-x).$$

Let $\mathrm{e}_p : \mathbb{Z}_p \to \mathbb{C}$ denote the function $\mathrm{e}_p : x \mapsto e^{2\pi i x/p}$. For $a \in \mathbb{Z}_p$ define the character $\chi_a : x \mapsto \mathrm{e}_p(-ax)$. The Fourier expansion of $f : \mathbb{Z}_p \to \mathbb{C}$ is the sum

$$f(x) = \sum_{x\in\mathbb{Z}_p} \widehat{f}(a)\chi_a(x),$$

where $\widehat{f}(a) = \langle f, \chi_a \rangle$. Note that by definition,

$$\widehat{f}(a) = \frac{1}{p} \sum_{x \in \mathbb{Z}_p} f(x) \mathrm{e}_p(ax).$$

It follows from the properties of the characters that

$$f * g(z) = \sum_{a \in \mathbb{Z}_p} \widehat{f}(a)\widehat{g}(a)\chi_a(z),$$

showing that $\widehat{f * g}(a) = \widehat{f}(a)\widehat{g}(a)$. In particular, if $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are independent random variables taking values in $\mathbb{Z}_p$, and if $\mathbf{x} = \mathbf{x}_1 + \ldots + \mathbf{x}_k$, then

$$\widehat{\mu_{\mathbf{x}}}(a) = p^{k-1} \prod_{i=1}^{k} \widehat{\mu_{\mathbf{x}_i}}(a).$$

Number theory estimates

Fix a prime p. Given an integer x, we denote the distance of x to the closest multiple of p (and abusing standard notation) by

$$\|x\|_p = \min\{|x - zp| : z \in \mathbb{Z}\}.$$

We will often use the estimate

$$|\mathrm{e}_p(x) - 1| \approx \frac{\|x\|_p}{p},$$

which follows from the easy estimate that $4|y| \leq |e^{2\pi i y} - 1| \leq 8|y|$ for $y \in [-1/2, 1/2]$, and taking $y = \frac{\mathrm{sign}(x)\|x\|_p}{p}$.

3 Sign-rank 1 and 2

In this section we demonstrate that boolean matrices with sign-rank 1 or 2 are very simple combinatorially. Let A be an $N \times N$ boolean matrix for $N = 2^n$. If A has sign-rank 1, then there exist nonzero numbers $a_1, \ldots, a_N, b_1, \ldots, b_N \in \mathbb{R}$ such that $A_{i,j} = \mathrm{sign}(a_i b_j)$. In particular, if we partition the a_i and the b_j to the positive and negative numbers, we see that A can be partitioned into 4 monochromatic sub-matrices. This implies that $\mathrm{Disc}(A) = \Omega(1)$.

Assume next that A has sign-rank 2. Then there exist vectors $u_1, \ldots, u_N, v_1, \ldots, v_N \in \mathbb{R}^2$ such that $A_{i,j} = \mathrm{sign}(\langle u_i, v_j \rangle)$. By applying a rotation to the vectors, we may assume that their coordinates are all nonzero. Next, by scaling the vectors, we may assume that $u_i = (\pm 1, a_i)$ and $v_j = (b_j, \pm 1)$ for all i, j. Next, partition the a_i and the b_j to the positive and negative numbers. Consider without loss of generality the sub-matrix in which $u_i = (1, a_i)$ and $v_j = (b_j, -1)$ for all i, j (the other three cases are equivalent). In this sub-matrix, $A_{i,j} = \mathrm{sign}(a_i - b_j)$. By removing repeated rows and columns, we get that the sub-matrix is an upper triangular matrix, with 1 on or above the diagonal and -1 below the diagonal. That is, the sub-matrix is equivalent to the matrix corresponding to the Greater-Than boolean function on at most n bits. Nisan [15] showed that the bounded-error communication complexity of this matrix is $O(\log n)$, which in particular implies that the discrepancy is at least $n^{-O(1)}$. This implies that also $\mathrm{Disc}(A) \geq n^{-O(1)}$.

4 Sign-rank 3 vs. discrepancy

We now turn to prove Theorem 7. To recall, Alice's input is the pair $[x,z]$ with $x \in [N]^2, z \in [-3N^2, 3N^2]$, and Bob's input is $y \in [N]^2$. The hard distribution ν is defined as follows. First, sample $\mathbf{x}, \mathbf{y}$ uniformly and independently from $[N]^2$. Next, sample $\mathbf{k}_1, \mathbf{k}_2 \in [N]$ uniformly and independently, and let $\mathbf{k} = \mathbf{k}_1 + \mathbf{k}_2$. Define $\mathbf{z}$ as follows: choose $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}$ or $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N$, each with probability $1/2$. Observe that in the former case $\langle \mathbf{x}, \mathbf{y} \rangle < \mathbf{z}$ and hence $f([\mathbf{x}, \mathbf{z}], \mathbf{y}) = 1$; and in the latter case $\langle \mathbf{x}, \mathbf{y} \rangle \geq \mathbf{z}$ and hence $f([\mathbf{x}, \mathbf{z}], \mathbf{y}) = -1$. Thus f is balanced:

$$\Pr[f([\mathbf{x}, \mathbf{z}], \mathbf{y}) = 1] = \Pr[f([\mathbf{x}, \mathbf{z}], \mathbf{y}) = -1] = 1/2.$$

In order to prove the theorem, we bound the correlation of f with a rectangle $A \times B$, where $A \subseteq [N]^2 \times [-3N^2, 3N^2]$ and $B \subseteq [N]^2$. For $x \in [N]^2$, let

$$A_x = \{z : [x, z] \in A\}.$$

We have

$$\begin{aligned} \mathrm{Disc}_\nu^{A \times B}(f) &= \mathbb{E}_{([\mathbf{x},\mathbf{z}],\mathbf{y}) \sim \nu} \left[f([\mathbf{x}, \mathbf{z}], \mathbf{y}) 1_A(\mathbf{x}, \mathbf{z}) 1_B(\mathbf{y}) \right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y} \sim [N]^2} 1_B(\mathbf{y}) \mathbb{E}_{\mathbf{z}|\mathbf{x},\mathbf{y}} \left[f([\mathbf{x}, \mathbf{z}], \mathbf{y}) 1_{A_\mathbf{x}}(\mathbf{z}) \right]. \end{aligned}$$

Recall the definition of f and that $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}$ or $\mathbf{z} = \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N$ with equal probabilities. In the former case f evaluates to 1, and it the latter case it evaluates to -1. We thus have

$$\begin{aligned} \mathrm{Disc}_\nu^{A \times B}(f) &= \frac{1}{2} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{k}} \left[f([\mathbf{x}, \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}], \mathbf{y}) 1_B(\mathbf{y}) 1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}) \right] \\ &\quad + \frac{1}{2} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{k}} \left[f([\mathbf{x}, \langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N], \mathbf{y})) 1_B(\mathbf{y}) 1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N) \right] \\ &= \frac{1}{2} \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{k}} \left[1_B(\mathbf{y}) 1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}) - 1_B(\mathbf{y}) 1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N) \right] \\ &= \frac{|B|}{2N^2} \mathbb{E}_\mathbf{x} \mathbb{E}_{\mathbf{y} \sim B} \mathbb{E}_\mathbf{k} \left[1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}) - 1_{A_\mathbf{x}}(\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k} - 2N) \right]. \end{aligned}$$

For $x \in [N]^2$ let ν_x^B denote the distribution of $\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}$ conditioned on $\mathbf{x} = x, \mathbf{y} \in B$. With this notation,

$$\begin{aligned} \mathrm{Disc}_\nu^{A \times B}(f) &= \frac{|B|}{2N^2} \mathbb{E}_\mathbf{x} \mathbb{E}_{\mathbf{w} \sim \nu_\mathbf{x}^B} \left[1_{A_\mathbf{x}}(\mathbf{w}) - 1_{A_\mathbf{x}}(\mathbf{w} - 2N) \right] \\ &= \frac{|B|}{2N^2} \mathbb{E}_\mathbf{x} \sum_{w \in \mathbb{Z}} 1_{A_\mathbf{x}}(w) \nu_\mathbf{x}^B(w) - 1_{A_\mathbf{x}}(w - 2N) \nu_\mathbf{x}^B(w) \\ &= \frac{|B|}{2N^2} \mathbb{E}_\mathbf{x} \sum_{w \in \mathbb{Z}} 1_{A_\mathbf{x}}(w) \nu_\mathbf{x}^B(w) - 1_{A_\mathbf{x}}(w) \nu_\mathbf{x}^B(w + 2N) \\ &\leq \frac{|B|}{2N^2} \mathbb{E}_\mathbf{x} \sum_{w \in \mathbb{Z}} \left| \nu_\mathbf{x}^B(w) - \nu_\mathbf{x}^B(w + 2N) \right|, \end{aligned}$$

which no longer depends on A. The random variable $\langle \mathbf{x}, \mathbf{y} \rangle + \mathbf{k}$ is in the range $[-3N^2, 3N^2]$ so we embed $[-3N^2, 3N^2]$ into $\mathbb{Z}_p$ for some prime $p \in [6N^2 + 1, 12N^2]$. We consider ν_x^B as a distribution over $\mathbb{Z}_p$, and thus we can rewrite

$$\begin{aligned} \mathrm{Disc}_\nu^{A \times B}(f) &\leq \frac{p|B|}{2N^2} \mathbb{E}_\mathbf{x} \mathbb{E}_{\mathbf{w} \sim \mathbb{Z}_p} |\nu_\mathbf{x}^B(\mathbf{w}) - \nu_\mathbf{x}^B(\mathbf{w} + 2N)| \\ &\lesssim |B| \cdot \mathbb{E}_\mathbf{x} \mathbb{E}_{\mathbf{w} \sim \mathbb{Z}_p} |\nu_\mathbf{x}^B(\mathbf{w}) - \nu_\mathbf{x}^B(\mathbf{w} + 2N)|. \end{aligned}$$

The following lemma, whose proof is deferred to the next section, completes the proof.

▶ **Lemma 8.** *Let $\tilde{N} \approx N$. Then $\mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{w}\sim\mathbb{Z}_p}|\nu_{\mathbf{x}}^B(\mathbf{w}) - \nu_{\mathbf{x}}^B(\mathbf{w}+\tilde{N})| \lesssim \frac{\log N}{\sqrt{|B|N^3}}$.*

By invoking Lemma 8 for $\tilde{N} = 2N$ we obtain

$$\mathrm{Disc}(f) \le \mathrm{Disc}_{\nu}^{A\times B}(f) \lesssim |B|\frac{\log N}{\sqrt{|B|N^3}} \le \sqrt{\frac{|B|}{N^3}}\log N \le N^{-\frac{1}{2}}\log N \lesssim n2^{-n/8}.$$

4.1 Invariance of $\nu_{\mathbf{x}}^B$ under translation

The goal of this section is to prove Lemma 8. We will prove that for a typical x, the measure ν_x^B is almost invariant under the translations by $\tilde{N} \approx N$. First we compute the Fourier expansion of this measure.

▶ **Lemma 9.** *For all $x \in [N]^2$ and $a \in \mathbb{Z}_p$, we have*

$$\widehat{\nu_x^B}(a) = \frac{1}{p}\mathrm{e}_p(2a)\left(\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right)^2 \mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle x,\mathbf{y}\rangle)\right].$$

Proof. Recall that ν_x^B is the distribution of $\langle x,\mathbf{y}\rangle + \mathbf{k}_1 + \mathbf{k}_2$ where $\mathbf{y}\sim B$ and $\mathbf{k}_1,\mathbf{k}_2 \sim [N]$. Therefore for all $a \in \mathbb{Z}_p$,

$$\widehat{\nu_x^B}(a) = p^2\widehat{\mu_{\langle x,\mathbf{y}\rangle}}(a)\widehat{\mu_{\mathbf{k}_1}}(a)\widehat{\mu_{\mathbf{k}_2}}(a) = p^2\widehat{\mu_{\langle x,\mathbf{y}\rangle}}(a)\widehat{\mu_{[N]}}(a)^2,$$

where to recall $\mu_{[N]}$ is the uniform distribution on $[N]$. First, we compute the Fourier coefficients of $\mu_{\langle x,\mathbf{y}\rangle}$:

$$\widehat{\mu_{\langle x,\mathbf{y}\rangle}}(a) = \frac{1}{p}\sum_{t\in\mathbb{Z}_p}\mu_{\langle x,\mathbf{y}\rangle}(t)\mathrm{e}_p(at) = \frac{1}{p}\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle x,y\rangle)\right].$$

Next, we compute the Fourier coefficients of $\mu_{[N]}$:

$$\widehat{\mu_{[N]}}(a) = \frac{1}{p}\sum_{t=1}^{N}\frac{1}{N}\mathrm{e}_p(at) = \frac{\mathrm{e}_p(a)}{pN}\cdot\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1},$$

where we have computed the partial sum of the geometric series $\{\mathrm{e}_p(at)\}_{t=1,\dots,N}$. The lemma follows. ◀

With the Fourier coefficients $\widehat{\nu_x^B}(a)$ computed in Lemma 9, we can analyze the distance of $\nu_{\mathbf{x}}^B$ from its translation by $\tilde{N} \approx N$.

Proof of Lemma 8. Let $\mathbf{w}\sim\mathbb{Z}_p$. Recall that $\mathbf{x}\sim[N]^2$ and that $\tilde{N}\approx N$. Using the Fourier expansion of $\nu_{\mathbf{x}}^B$ we can write

$$s := \mathbb{E}_{\mathbf{x},\mathbf{w}}|\nu_{\mathbf{x}}^B(\mathbf{w}) - \nu_{\mathbf{x}}^B(\mathbf{w}+\tilde{N})| = \mathbb{E}_{\mathbf{x},\mathbf{w}}\left|\sum_{a\in\mathbb{Z}_p}\widehat{\nu_{\mathbf{x}}^B}(a)\left(\chi_a(\mathbf{w}) - \chi_a(\mathbf{w}+\tilde{N})\right)\right|.$$

We may now use Lemma 9 and substitute the Fourier coefficient $\widehat{\nu_{\mathbf{x}}^B}(a)$,

$$s = \frac{1}{p}\mathbb{E}_{\mathbf{x},\mathbf{w}}\left|\sum_{a\in\mathbb{Z}_p}e_p(2a)\left(\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right)^2\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle\mathbf{x},\mathbf{y}\rangle)\right](1-\mathrm{e}_p(-\tilde{N}a))\chi_a(\mathbf{w})\right|.$$

Squaring both sides, and applying Cauchy-Schwarz and then Parseval's identity, we get

$$\begin{aligned} s^2p^2 &\leq \mathbb{E}_{\mathbf{x},\mathbf{w}}\left|\sum_{a\in\mathbb{Z}_p} \mathrm{e}_p(2a)\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle\mathbf{x},\mathbf{y}\rangle)\right]\left(\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right)^2(1-\mathrm{e}_p(-\tilde{N}a))\chi_a(\mathbf{w})\right|^2 \\ &= \mathbb{E}_{\mathbf{x}}\sum_{a\in\mathbb{Z}_p}\left|\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle\mathbf{x},\mathbf{y}\rangle)\right]\right|^2\left|\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right|^4|1-\mathrm{e}_p(-\tilde{N}a)|^2 \\ &= \sum_{a\in\mathbb{Z}_p}\left(\mathbb{E}_{\mathbf{x}}\left|\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle\mathbf{x},\mathbf{y}\rangle)\right]\right|^2\right)\left|\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right|^4|1-\mathrm{e}_p(\tilde{N}a)|^2. \end{aligned}$$

Recalling that $p\approx N^2$, note that for $a\neq 0$ it holds that

$$\left|\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right| \approx \frac{\|Na\|_p}{N\|a\|_p} \lesssim \min\left(1,\frac{N}{\|a\|_p}\right)$$

and

$$|\mathrm{e}_p(\tilde{N}a)-1| \approx \frac{\left\|\tilde{N}a\right\|_p}{p} \lesssim \min\left(1,\frac{\|a\|_p}{N}\right),$$

both of which follow from trivial upper bounds $\|Na\|_p \leq N\|a\|_p$ and $\|x\|_p \leq p \approx N^{\frac{1}{2}}$. Let us denote $E_a(B) := \mathbb{E}_{\mathbf{x}}\left|\mathbb{E}_{\mathbf{y}\sim B}\left[\mathrm{e}_p(a\langle\mathbf{x},\mathbf{y}\rangle)\right]\right|^2$. We break the sum into two parts and for each part use a different estimate for $E_a(B)$ using Lemma 10 below.

$$\begin{aligned} s^2 &\lesssim \frac{1}{p^2}\sum_{\|a\|_p<N} E_a(B)|\mathrm{e}_p(\tilde{N}a)-1|^2 + \frac{1}{p^2}\sum_{\|a\|_p\geq N} E_a(B)\left|\frac{1}{N}\frac{\mathrm{e}_p(Na)-1}{\mathrm{e}_p(a)-1}\right|^4 \\ &\lesssim \frac{1}{p^2}\sum_{\|a\|_p<N} E_a(B)\left(\frac{\|a\|_p}{N}\right)^2 + \frac{1}{p^2}\sum_{\|a\|_p\geq N} E_a(B)\left(\frac{N}{\|a\|_p}\right)^4 \\ &\lesssim \frac{1}{p^2}\sum_{\|a\|_p<N}\frac{N^2}{\|a\|_p^2}\cdot\frac{\log^2 N}{|B|}\left(\frac{\|a\|_p}{N}\right)^2 + \frac{1}{p^2}\sum_{\|a\|_p\geq N}\frac{\|a\|_p^2}{N^2}\cdot\frac{\log^2 N}{|B|}\left(\frac{N}{\|a\|_p}\right)^4 \\ &\lesssim \frac{\log^2 N}{N^2|B|}\left(\sum_{\|a\|_p<N}\frac{1}{N^2}+\sum_{\|a\|_p\geq N}\frac{1}{\|a\|_p^2}\right) \\ &\lesssim \frac{\log^2 N}{N^2|B|}\left(N\cdot\frac{1}{N^2}+\sum_{t\geq N}\frac{1}{t^2}\right) \\ &\lesssim \frac{\log^2 N}{N^2|B|}\frac{1}{N} = \frac{\log^2 N}{|B|N^3}. \end{aligned}$$

◀

4.2 Uniformity of product sets over $\mathbb{Z}_p$

Recall that $E_a(B) := \mathbb{E}_{\mathbf{x}\sim[N]^2}\left|\mathbb{E}_{\mathbf{y}\sim B}\left[\chi_a(\langle\mathbf{x},\mathbf{y}\rangle)\right]\right|^2$. The following lemma provides estimates for it.

▶ **Lemma 10.** $E_a(B) \lesssim \max\left(\frac{\|a\|_p^2}{N^2},\frac{N^2}{\|a\|_p^2}\right)\cdot\frac{\log^2 N}{|B|}$.

Proof. We have

$$\begin{aligned} E_a(B) &= \frac{1}{|B|^2}\mathbb{E}_{\mathbf{x}\sim[N]^2}\left|\sum_{y\in B}\chi_a(\langle \mathbf{x}, y\rangle)\right|^2 \\ &= \frac{1}{|B|^2}\sum_{y',y''\in B}\mathbb{E}_{\mathbf{x}\sim[N]^2}\chi_a(\langle \mathbf{x}, y'-y''\rangle) \\ &\leq \frac{1}{|B|^2}\sum_{y',y''\in B}\left|\mathbb{E}_{\mathbf{x}\sim[N]^2}\chi_a(\langle \mathbf{x}, y'-y''\rangle)\right|. \end{aligned}$$

Let $B-B=\{y'-y'' : y', y''\in B\}\subset\mathbb{Z}_p^2$. Any element $y\in B-B$ can be expressed as $y=y'-y''$ for $y', y''\in B$ in at most $|B|$ ways. Thus we can bound

$$E_a(B) \leq \frac{1}{|B|}\sum_{y\in B-B}\left|\mathbb{E}_{\mathbf{x}\sim[N]^2}\chi_a(\langle \mathbf{x}, y\rangle)\right|.$$

Since $B-B\subseteq[N]^2-[N]^2\subseteq[-N,N]^2$, we can simplify the above to

$$\begin{aligned} E_a(B) &\leq \frac{1}{N^2|B|}\sum_{y\in[-N,N]^2}\left|\sum_{x\in[N]^2}\chi_a(\langle x, y\rangle)\right| \\ &= \frac{1}{N^2|B|}\sum_{y_1,y_2\in[-N,N]}\left|\sum_{x_1,x_2\in[N]}\chi_a(x_1y_1)\cdot\chi_a(x_2y_2)\right| \\ &= \frac{1}{N^2|B|}\sum_{y_1,y_2\in[-N,N]}\left|\sum_{x_1\in[N]}\chi_a(x_1y_1)\right|\left|\sum_{x_2\in[N]}\chi_a(x_2y_2)\right| \\ &= \frac{1}{N^2|B|}\left(\sum_{y\in[-N,N]}\left|\sum_{x\in[N]}\chi_a(xy)\right|\right)^2 \\ &\lesssim \frac{1}{N^2|B|}\left(\sum_{y\in[0,N]}\left|\sum_{x\in[N]}\chi_a(xy)\right|\right)^2. \end{aligned}$$

Note that for a fixed $y\neq 0$, $\sum_{x\in[N]}\chi_a(xy)$ is a sum of a geometric series which satisfies $\left|\sum_{x\in[N]}\chi_a(xy)\right| = \left|\frac{\mathrm{e}_p(Nay)-1}{\mathrm{e}_p(ay)-1}\right|$, and hence

$$\sum_{y\in[0,N]}\left|\sum_{x\in[N]}\chi_a(xy)\right| \leq N+\sum_{y\in[N]}\left|\frac{\mathrm{e}_p(Nay)-1}{\mathrm{e}_p(ay)-1}\right| \lesssim N+\sum_{y\in[N]}\frac{\|Nay\|_p}{\|ay\|_p}.$$

Invoking Lemma 11 below finishes the proof. ◀

▶ **Lemma 11.** *Let $p\geq N^2$ be prime and let $a\in\mathbb{Z}_p\setminus\{0\}$. Then*

$$\sum_{y\in[N]}\frac{\|Nay\|_p}{\|ay\|_p} \lesssim \max\left(\|a\|_p+\frac{p}{N}, \frac{p}{\|a\|_p}\right)\cdot\log p.$$

We need the following simple claim in the proof of Lemma 11.

▷ Claim 12. Let $\mathbf{r}$ be a random variable which takes values in $[K]$. Let $g:[K]\to\mathbb{R}$. Then

$$\mathbb{E}_{\mathbf{r}}g(\mathbf{r}) = g(K)+\sum_{i=1}^{K-1}(g(i)-g(i+1))\Pr[\mathbf{r}\leq i].$$

Proof.

$$\begin{aligned}\mathbb{E}_{\mathbf{r}} g(\mathbf{r}) &= \sum_{i=1}^{K} g(i)\Pr[\mathbf{r}=i] \\ &= \sum_{i=1}^{K} g(i)\left(\Pr[\mathbf{r}\le i]-\Pr[\mathbf{r}\le i-1]\right) \\ &= g(K)+\sum_{i=1}^{K-1}\left(g(i)-g(i+1)\right)\Pr[\mathbf{r}\le i].\end{aligned}$$

◁

Proof of Lemma 11. We separate the proof to two cases of $\|a\|_p < N$ and $\|a\|_p \ge N$. Consider an integer k with $\|a\|_p \le k \le p$. We start by estimating the size of the set

$$S_k = \{y \in [N] : \|ya\|_p \le k\}.$$

Note that if $y \in S_k$, then $ya \in ph + [-k,k]$ for some integer $h \ge 0$. Since $y \in [N]$, we have $h \le \frac{N\|a\|_p + k}{p}$, and hence there are at most $\frac{N\|a\|_p}{p} + 1$ such values of h. Fixing h, we have $y \in \frac{ph}{\|a\|_p} + [-k/\|a\|_p, k/\|a\|_p]$, and there are at most $\frac{2k}{\|a\|_p} + 1 \le \frac{3k}{\|a\|_p}$ such values of y. We conclude that

$$|S_k| \le \left(\frac{N\|a\|_p}{p}+1\right)\times\frac{3k}{\|a\|_p} \le \frac{3Nk}{p}+\frac{3k}{\|a\|_p} \lesssim \frac{k}{N}+\frac{k}{\|a\|_p}.$$

Note that this bound obviously holds also for $k \ge p$.

Now to compute $\sum_{y\in[N]} \frac{\|Nay\|_p}{\|ay\|_p}$ we separate to two cases depending on whether $\|a\|_p \ge N$ or not, and then use Claim 12.

The case $\|a\|_p \ge N$. First, note that in this case we can bound $|S_k| \lesssim \frac{k}{N}$. Also to bound $\frac{\|Nay\|_p}{\|ay\|_p}$, for $y \in S_{\|a\|_p}$, we use the bound $\frac{\|Nay\|_p}{\|ay\|_p} \le N$, otherwise we use the bound $\|Nay\|_p \le p$. We get

$$\sum_{y\in[N]} \frac{\|Nay\|_p}{\|ay\|_p} \le \sum_{y\in S_{\|a\|_p}} N + p\sum_{y\in[N]}\frac{1}{\|ay\|_p}.$$

To compute $\sum_{y\in[N]} \frac{1}{\|ay\|_p}$ we use Claim 12. Let $\mathbf{u} \sim [N]$ be uniformly chosen, and set the random variable $\mathbf{r}$ to be $\mathbf{r} = \|a\mathbf{u}\|_p$. Set $g(x) = \frac{1}{x}$. Then we have

$$\begin{aligned}\frac{1}{N}\sum_{y\in[N]}\frac{1}{\|ay\|_p} &= \mathbb{E}_{\mathbf{r}} g(\mathbf{r}) \\ &= g(p)+\sum_{i=1}^{p-1}\left(g(i)-g(i+1)\right)\Pr[\mathbf{r}\le i] \\ &= \frac{1}{p}+\sum_{i=1}^{p-1}\left(\frac{1}{i}-\frac{1}{i+1}\right)\frac{|S_i|}{N} \\ &\lesssim \frac{1}{p}+\sum_{i=1}^{p-1}\frac{1}{i^2}\cdot\frac{i}{N^2} \\ &\lesssim \frac{\log p}{N^2}.\end{aligned}$$

Overall we get

$$\sum_{y\in[N]} \frac{\|Nay\|_p}{\|ay\|_p} \leq \sum_{y\in S_{\|a\|_p}} N + p\sum_{y\in[N]} \frac{1}{\|ay\|_p} \lesssim \|a\|_p + \frac{p\log p}{N}.$$

The case $\|a\|_p < N$. Here we use the estimate $|S_k| \lesssim \frac{k}{\|a\|_p}$. Also similar to the previous case, for $y \in S_N$ we use the bound $\frac{\|Nay\|_p}{\|ay\|_p} \leq N$, otherwise we use the bound $\frac{\|Nay\|_p}{\|ay\|_p} \leq \frac{p}{\|ay\|_p}$. Similar to the previous case, we have

$$\begin{aligned}
\frac{1}{N}\sum_{y\in[N]} \frac{1}{\|ay\|_p} &= g(p) + \sum_{i=1}^{p-1} (g(i) - g(i+1))\Pr[\mathbf{r} \leq i] \\
&= \frac{1}{p} + \sum_{i=1}^{p-1} \left(\frac{1}{i} - \frac{1}{i+1}\right)\frac{|S_i|}{N} \\
&\lesssim \frac{1}{p} + \sum_{i=1}^{p-1} \frac{1}{i^2}\cdot\frac{i}{\|a\|_p N} \\
&\lesssim \frac{\log p}{\|a\|_p N}.
\end{aligned}$$

So we have

$$\begin{aligned}
\sum_{y\in[N]} \frac{\|Nay\|_p}{\|ay\|_p} &\leq \sum_{y\in S_N} N + p\sum_{y\in[N]} \frac{1}{\|ay\|_p} \\
&\lesssim \frac{N^2}{\|a\|_p} + \frac{p\log p}{\|a\|_p} \\
&\lesssim \frac{p\log p}{\|a\|_p}.
\end{aligned}$$

The lemma follows. ◀

We remark that the following more general statement regarding uniformity of product sets follows by a similar proof to Lemma 10 which we record here as it may be of independent interest.

▶ **Lemma 13.** *Let $p \geq N^2$ be prime, and let $B \subseteq [N]^d$ for some positive integer d. Then*

$$\mathbb{E}_{\mathbf{x}\sim[N]^d} |\mathbb{E}_{\mathbf{y}\sim B} \chi_a(\langle \mathbf{x}, \mathbf{y}\rangle)|^2 \lesssim \max\left(\|a\|_p^d, \frac{p^d}{\|a\|_p^d}\right)\cdot\frac{\log^d p}{|B|N^d}.$$

References

1 Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank versus vc dimension. In *Conference on Learning Theory*, pages 47–80, 2016.

2 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 337–347. IEEE, 1986.

3 Ronen Basri, Pedro F Felzenszwalb, Ross B Girshick, David W Jacobs, and Caroline J Klivans. Visibility constraints on features of 3d objects. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1231–1238. IEEE, 2009.

4 Amey Bhangale and Swastik Kopparty. The complexity of computing the minimum rank of a sign pattern matrix. *arXiv preprint*, 2015. arXiv:1503.04486.

5 Harry Buhrman, Nikolay Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 24–32. IEEE, 2007.

6 Mark Bun and Justin Thaler. Improved bounds on the sign-rank of AC^0. In *43rd International Colloquium on Automata, Languages, and Programming*, volume 55 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 37, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

7 Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *ACM SIGACT News*, 41(3):59–85, 2010.

8 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

9 Hartmut Klauck. Lower bounds for quantum communication complexity. In *Proceedings 2001 IEEE International Conference on Cluster Computing*, pages 288–297. IEEE, 2001.

10 Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{O(n^{1/3})}$. *J. Comput. System Sci.*, 68(2):303–318, 2004. doi:10.1016/j.jcss.2003.07.007.

11 Troy Lee and Adi Shraibman. An approximation algorithm for approximation rank. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 351–357. IEEE, 2009.

12 Nati Linial, Shahar Mendelson, Gideon Schechtman, and Adi Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007. doi:10.1007/s00493-007-2160-5.

13 Nati Linial and Adi Shraibman. Learning complexity vs. communication complexity. *Combin. Probab. Comput.*, 18(1-2):227–245, 2009. doi:10.1017/S0963548308009656.

14 Nati Linial and Adi Shraibman. Lower bounds in communication complexity based on factorization norms. *Random Structures & Algorithms*, 34(3):368–394, 2009.

15 Noam Nisan. The communication complexity of threshold gates. *Combinatorics, Paul Erdos is Eighty*, 1:301–315, 1993.

16 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *Journal of Computer and System Sciences*, 33(1):106–123, 1986.

17 Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of AC^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010. doi:10.1137/080744037.

18 Alexander A Sherstov. Halfspace matrices. *Computational Complexity*, 17(2):149–178, 2008.

19 Alexander A Sherstov. The pattern matrix method. *SIAM Journal on Computing*, 40(6):1969–2000, 2011.

20 Alexander A Sherstov. Optimal bounds for sign-representing the intersection of two halfspaces by polynomials. *Combinatorica*, 33(1):73–96, 2013.

21 Alexander A. Sherstov. The hardest halfspace. *CoRR*, abs/1902.01765, 2019. arXiv:1902.01765.

22 Justin Thaler. Lower bounds for the approximate degree of block-composed functions. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

23 Hugh E. Warren. Lower bounds for approximation by nonlinear manifolds. *Trans. Amer. Math. Soc.*, 133:167–178, 1968. doi:10.2307/1994937.

On the Complexity of Modulo-q Arguments and the Chevalley–Warning Theorem

Mika Göös
Stanford University, CA, USA
https://theory.stanford.edu/~mika/
goos@stanford.edu

Pritish Kamath
Toyota Technological Institute at Chicago, IL, USA
https://pritishkamath.github.io
pritish@ttic.edu

Katerina Sotiraki
Massachusetts Institute of Technology, Cambridge, MA, USA
http://www.mit.edu/~katesot/
katesot@mit.edu

Manolis Zampetakis
Massachusetts Institute of Technology, Cambridge, MA, USA
http://www.mit.edu/~mzampet/
mzampet@mit.edu

Abstract

We study the search problem class PPA_q defined as a modulo-q analog of the well-known *polynomial parity argument* class PPA introduced by Papadimitriou (JCSS 1994). Our first result shows that this class can be characterized in terms of PPA_p for prime p.

Our main result is to establish that an *explicit* version of a search problem associated to the Chevalley–Warning theorem is complete for PPA_p for prime p. This problem is *natural* in that it does not explicitly involve circuits as part of the input. It is the first such complete problem for PPA_p when $p \geq 3$.

Finally we discuss connections between Chevalley-Warning theorem and the well-studied *short integer solution* problem and survey the structural properties of PPA_q.

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Total NP Search Problems, Modulo-q arguments, Chevalley–Warning Theorem

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.19

Funding *Mika Göös*: Work done while at IAS. Supported by NSF grant CCF-1412958.
Pritish Kamath: Work done while at MIT. Supported in part by NSF Awards CCF-1733808 and IIS-1741137 and MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646.
Katerina Sotiraki: Supported in parts by NSF/BSF grant #1350619, an MIT-IBM grant, and a DARPA Young Faculty Award, MIT Lincoln Laboratories and Analog Devices.
Manolis Zampetakis: Supported by a Google PhD Fellowship.

Acknowledgements We thank Christos Papadimitriou, Robert Robere, Dmitry Sokolov and Noah Stephens-Davidowitz for helpful discussions. We also thank anonymous referees for valuable suggestions.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 19; pp. 19:1–19:42

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

1 Introduction

The study of *total* NP *search problems* (TFNP) was initiated by Megiddo and Papadimitriou [32] and Papadimitriou [33] to characterize the complexity of search problems that have a solution for every input and where a given solution can be efficiently checked for validity. Meggido and Papadimitriou [32] showed that the notion of NP-hardness is inadequate to capture the complexity of total NP search problems. By now, this theory has flowered into a sprawling jungle of widely-studied syntactic complexity classes (such as PLS [28], PPA/PPAD/PPP [33], CLS [18]) that serve to classify the complexities of many relevant search problems.

The goal of identifying *natural*[1] complete problems for these complexity classes lies in the foundation of this sub-field of complexity theory and not only gives a complete picture of the computational complexity of the corresponding search problems, but also provides a better understanding of the complexity classes. Such natural complete problems have also been an essential middle-step for proving the completeness of other important search problems, the same way that the NP-completeness of Sat is an essential middle step in showing the NP-completeness of many other natural problems. Some known natural complete problems for TFNP subclasses are: the PPAD-completeness of NashEquilibrium [17], the PPA-completeness of ConsensusHalving, NecklaceSplitting and HamSandwich problems [20, 21] and the PPP-completeness of natural problems related to lattice-based cryptography [36]. Finally, the theory of total search problems has found connections beyond its original scope to areas like communication complexity and circuit lower bounds [23], cryptography [9, 29, 16] and the Sum-of-Squares hierarchy [30].

Our main result is to identify the first natural complete problem for the classes PPA_q, a variant of the class PPA. We also illustrate the relevance of these classes through connections with important search problems from combinatorics and cryptography.

Class PPA_q. The class PPA_q was defined, in passing, by Papadimitriou [33, p. 520]. It is a modulo-q analog of the well-studied *polynomial parity argument* class PPA (which corresponds to $q = 2$). The class embodies the following combinatorial principle:

If a bipartite graph has a node of degree not a multiple of q,
then there is another such node.

In more detail, PPA_q consists of all total NP search problems reducible[2] to the problem $\textsc{Bipartite}_q$ defined as follows. An instance of this problem is a balanced bipartite graph $G = (V \cup U, E)$, where $V \cup U = \{0,1\}^n$ together with a designated vertex $v^\star \in V \cup U$. The graph G is implicitly given via a circuit C that computes the neighborhood of every node in G. Let $\deg(v)$ be the degree of the node v in G. A valid solution is a node $v \in \{0,1\}^n$ such that, either

▷ $v = v^\star$ satisfying $\deg(v) \equiv 0 \pmod q$ [*Trivial Solution*] ; or

▷ $v \neq v^\star$ satisfying $\deg(v) \not\equiv 0 \pmod q$.

In Section 2 we provide some other total search problems ($\textsc{Lonely}_q$, $\textsc{Leaf}_q$) that are reducible to and from $\textsc{Bipartite}_q$. Any one of these problems could be used to define PPA_q. In fact, $\textsc{Lonely}_q$ and $\textsc{Leaf}_q$ are natural variants of the standard problems Lonely and Leaf which are used to define the class PPA.

[1] Following the terminology of many TFNP papers, including [24, 20, 21, 36], a natural problem is one that does not have explicitly a circuit or a Turing machine as part of the input.

[2] Here, we consider a *many-one reduction*, which is a polynomial time algorithm with one oracle query to the said problem. In contrast, a *Turing reduction* allows polynomially many oracle queries. See Subsection 1.5 for a comparison.

Our contributions. We illustrate the importance of the complexity classes PPA_q by showing that many important search problems whose computational complexity is not well understood belong to PPA_q (see §1.6 for details). These problems span a wide range of scientific areas, from algebraic topology to cryptography. For some of these problems we conjecture that PPA_q-completeness is the right notion to characterize their computational complexity. The study of PPA_q is also motivated from the connections to other important and well-studied classes like PPAD.

In this paper, we provide a systematic study of the complexity classes PPA_q. Our main result is the identification of the first natural complete problem for PPA_q together with some structural results. Below we give a more precise overview of our results.

§1.1 (Details in Section 3): We characterize PPA_q in terms of PPA_p for prime p.

§1.2 (Details in Section 4): Our main result is that an *explicit*[3] version of the Chevalley-Warning theorem is complete for PPA_p for prime p. This problem is *natural* in that it does not involve circuits as part of the input and is the first known natural complete problem for PPA_p when $p \geq 3$.

§1.3 (Details in Section 5): As a consequence of the PPA_p-completeness of our natural problem, we show that restricting the input circuits in the definition of PPA_p to just constant depth arithmetic formulas doesn't change the power of the class.

§1.4 (Details in Section 6): We show a connection between PPA_q and the Short Integer Solution (SIS) problem from the theory of lattices. This connection implies that SIS with constant modulus q belongs to $\mathsf{PPA}_q \cap \mathsf{PPP}$, but also provides a polynomial time algorithm for solving SIS when the modulus q is constant and has only 2 and 3 as prime factors.

§1.5 (Details in Section 7): We sketch how existing results already paint a near-complete picture of the relative power of PPA_p relative to other TFNP subclasses (via inclusions and oracle separations). We also show that PPA_q is closed under Turing reductions.

In §1.6, we include a list of open problems that illustrate the broader relevance of PPA_q. We note that a concurrent and independent work by Hollender [25] also establishes the structural properties of PPA_q corresponding to §1.1 and §1.5.

1.1 Characterization via Prime Modulus

We show, in Section 3, that every class PPA_q is built out of the classes PPA_p for p a prime. To formalize this result, we recall the operator "&" defined by Buss and Johnson [13, §6]. For any two syntactic complexity classes M_0, M_1 with complete problems S_0, S_1, the class $\mathsf{M}_0 \,\&\, \mathsf{M}_1$ is defined via its complete problem $S_0 \,\&\, S_1$ where, on input $(x, b) \in \{0,1\}^* \times \{0,1\}$, the goal is to find a solution for x interpreted as an instance of problem S_b. Namely, if $b = 0$ then the output has to be a solution of S_0 with input x, and otherwise it has to be a solution of S_1 with input x. Intuitively speaking, $\mathsf{M}_1 \,\&\, \mathsf{M}_2$ combines the powers of both M_1 and M_2. Note that $\mathsf{M}_1 \cup \mathsf{M}_2 \subseteq \mathsf{M}_1 \,\&\, \mathsf{M}_2$. We can now formally express our characterization result (where $p|q$ is the set of primes p dividing q).

▶ **Theorem 1.** $\mathsf{PPA}_q = \&_{p|q}\, \mathsf{PPA}_p$.

[3] Following the terminology in [8], by *explicit* we mean that the system of polynomials, which is the input of the computational problems we define, are given as a sum of monic monomials.

A special case of Theorem 1 is that $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$ for every prime power p^k. Showing the inclusion $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$ is the crux of our proof. This part of the theorem can be viewed as a total search problem analog of the counting class result of Beigel and Gill [7] stating that $\mathsf{Mod}_{p^k}\mathsf{P} = \mathsf{Mod}_p\mathsf{P}$; "an unexpected result", they wrote at the time. Throughout this paper, we use q to denote any integer ≥ 2 and p to denote a prime integer.

1.2 A Natural Complete Problem via Chevalley-Warning Theorem

There have been several works focusing on completeness results for the class PPA (i.e. PPA_2). Initial works showed the PPA-completeness of (non-natural) total search problems corresponding to topological fixed point theorems [24, 1, 19]. Closer to our paper, Belovs et al. [8] show the PPA-completeness of computational analogs of Combinatorial Nullstellensatz and the Chevalley–Warning Theorem, but which explicitly involve a circuit as part of the input. More recently, breakthrough results showed PPA-completeness of problems without a circuit or a Turing Machine in the input such as Consensus-Halving, Necklace-Splitting and Ham-Sandwich [20, 21] resolving an open problem since the definition of PPA in [33].

Our main contribution is to provide a natural complete problem for PPA_p, for every prime p; thereby also yielding a new complete problem for PPA. Our complete problem is an extension of the problem $\textsc{Chevalley}_p$, defined by Papadimitriou [33], which is a search problem associated to the celebrated Chevalley-Warning Theorem. We first present an abstract way to understand the proof of the Chevalley-Warning Theorem that motivates the definition of our natural complete problem for PPA_p.

1.2.1 Max-Degree Monic Monomials and Proof of Chevalley-Warning Theorem

In 1935, Claude Chevalley [15] resolved a hypothesis stated by Emil Artin, that all finite fields are quasi-algebraically closed. Later, Ewald Warning [37] proved a slight generalization of Chevalley's theorem. This generalized statement is usually referred to as the Chevalley-Warning Theorem (CWT, for short). Despite its initial algebraic motivation, CWT has found profound applications in combinatorics and number theory as we discuss in §1.4 (and Section 6).

We now explain the statement of the Chevalley-Warning Theorem, starting with some notations. For any field $\mathbb{F}$ and any polynomial f in a polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ we use $\deg(f)$ to represent the degree of f. We use $\boldsymbol{x}$ to succinctly denote the set of all variables $(x_1, \ldots, x_n)$ (the number of variables will always be n) and $\boldsymbol{f}$ to succinctly denote a system of polynomials $\boldsymbol{f} = (f_1, \ldots, f_m) \in \mathbb{F}[\boldsymbol{x}]^m$. We will often abuse notations to use $\boldsymbol{x}$ to also denote assignments over $\mathbb{F}_p^n$. For instance, let $\mathcal{V}_{\boldsymbol{f}} \coloneqq \left\{\boldsymbol{x} \in \mathbb{F}_p^n : f_i(\boldsymbol{x}) = 0 \text{ for all } i \in [m]\right\}$ be the set of all common roots of $\boldsymbol{f}$.

▶ **Chevalley-Warning Theorem** ([15, 37]). *For any prime*[4] p *and polynomial system* $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ *satisfying*

$$\sum_{i=1}^{m} \deg(f_i) < n, \qquad \text{(CW Condition)}$$

it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$.

[4] While most of the results in this section generalize to prime powers, we only consider prime fields for simplicity.

Given a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, the key idea in the proof of the Chevalley-Warning Theorem is the polynomial

$$\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \coloneqq \prod_{i=1}^{m} \left(1 - f_i(\boldsymbol{x})^{p-1}\right) \pmod{\{x_i^p - x_i\}_i} .$$

Note that $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. Thus, $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x} \in \mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod{p}$. The following definition informally describes a special type of monomial of $\mathsf{CW}_{\boldsymbol{f}}$ that is of particular interest in the proof. For the precise definition, we refer to Section 4.

▶ **Definition 2** (Max-Degree Monic Monomials (Informal)). *Let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$. A monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ refers to a monic monomial obtained when symbolically expanding $\mathsf{CW}_{\boldsymbol{f}}$ as a sum of monic monomials. A monic monomial is said to be of max-degree if it is $\prod_{j=1}^{n} x_j^{p-1}$.*

In the above definition, it is important to consider the *symbolic expansion* of $\mathsf{CW}_{\boldsymbol{f}}$ and ignore any cancellation of coefficients that might occur. Observe that, although the expansion of $\mathsf{CW}_{\boldsymbol{f}}$ is exponentially large in the description size of $\boldsymbol{f}$, each monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ can be succinctly described as a combination of monic monomials of the polynomials $f_1, \ldots, f_m$. We formally discuss this in Section 4.

Using the definition of max-degree monic monomials, we state the main technical lemma underlying the proof of CWT (with proof in Section 4).

▶ **Chevalley–Warning Lemma.** *For any prime p and $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$,*

$$|\mathcal{V}_{\boldsymbol{f}}| \equiv (-1)^n \cdot |\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\}| \pmod{p} \qquad \text{(CW Lemma)}$$

The Chevalley-Warning Theorem now follows by observing that if $\sum_{i=1}^{m} \deg(f_i) < n$ then the number of max-degree monic monomials of $\mathsf{CW}_{\boldsymbol{f}}$ is zero. Hence, we get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$.

1.2.2 Proofs of Cancellation

From the proof sketch of CWT in the previous section, a slight generalization of CWT follows. In particular, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod{p}$ if and only if

$$\left|\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\}\right| \equiv 0 \pmod{p}, \qquad \text{(Extended CW Condition)}$$

Any condition on $\boldsymbol{f}$ that implies the (Extended CW Condition) can replace (CW Condition) in the Chevalley-Warning Theorem. Note that the (Extended CW Condition) is equivalent to all the max-degree monic monomials in $\mathsf{CW}_{\boldsymbol{f}}$ cancelling out. Thus, we call any such condition on $\boldsymbol{f}$ that implies (Extended CW Condition) to be a "*proof of cancellation*" for the system $\boldsymbol{f}$.

We can now reinterpret the result of Belovs et al. [8] in this framework of "proof of cancellation" conditions. In particular, [8] considers the case $p = 2$ and defines the problem PPA-Circuit-Chevalley, in which a "proof of cancellation" is given in a specific form of circuits. These circuits describe the system $(f_1, \ldots, f_m)$ in the PPA-Circuit-Chevalley problem. It is then shown that PPA-Circuit-Chevalley is PPA_2-complete.

1.2.3 Computational Problems Based on Chevalley-Warning Theorem

Every "proof of cancellation" that is *syntactically refutable* can be used to define a total search problem that lies in PPA_p. By *syntactically refutable* we mean that whenever the "proof of cancellation" is false, there exists a small witness that certifies so. In this section,

we define three computational problems with their corresponding "proof of cancellation": (1) the $\textsc{Chevalley}_p$ problem defined by [33], (2) the $\textsc{GeneralChevalley}_p$ problem that is a generalization of $\textsc{Chevalley}_p$, and (3) the problem $\textsc{ChevalleyWithSymmetry}_p$ that we show to be PPA_p-complete. All these problems are defined for every prime modulus p and are natural in the sense that they do not explicitly involve a circuit or a Turing Machine in their input. In particular, the polynomial systems in the input are *explicit* in that they are given as a sum of monic monomials.

1.2.3.1 Chevalley

This is the direct computational analog of the Chevalley-Warning Theorem and was defined by Papadimitriou [33] as the following total search problem:

$\underline{\textsc{Chevalley}_p}$
Given an explicit polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$, output one of the following:
▷ [*Refuting witness*] (CW Condition) is not satisfied.
▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \setminus \{\boldsymbol{x}^\star\}$.
We will particularly consider a special case where all the f_i's have zero constant term (*zecote*, for short). In this case, $\boldsymbol{x}^\star = \mathbf{0} \in \mathcal{V}_{\boldsymbol{f}}$, so there is no need to explicitly include $\boldsymbol{x}^*$ in the input.

1.2.3.2 General Chevalley

As mentioned already, we can define a search problem corresponding to any syntactically refutable condition that implies the (Extended CW Condition). One such condition is to directly assert that

$$\{\text{max-degree monic monomials of } \mathsf{CW}_{\boldsymbol{f}}\} \;=\; \emptyset. \qquad \text{(General CW Condition)}$$

In particular, note that (CW Condition) implies this condition. Moreover, this condition is syntactically refutable by a max-degree monic monomial, which is efficiently representable as a combination of at most $m(p-1)$ monomials of the f_i's. Thus, we can define the following total search problem generalizing $\textsc{Chevalley}_p$.

$\underline{\textsc{GeneralChevalley}_p}$
Given an explicit polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$, output one of the following:
▷ [*Refuting Witness*] A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$.
▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \setminus \{\boldsymbol{x}^\star\}$.

While $\textsc{GeneralChevalley}_p$ generalizes $\textsc{Chevalley}_p$, it does not capture the full generality of (Extended CW Condition). However (Extended CW Condition) is not syntactically refutable (in fact, it is $\mathsf{Mod}_p\mathsf{P}$–complete to decide[5] if the final coefficient of the max-degree monomial is 0).

A natural question then is whether $\textsc{GeneralChevalley}_p$, or even $\textsc{Chevalley}_p$, could already be PPA_p–complete. We believe this to be unlikely because (General CW Condition) seems to fail in capturing other simple conditions that are syntactically refutable and yet imply (Extended CW Condition). Namely, consider a permutation permutation $\sigma \in S_n$ of

[5] Circuit-SAT can be encoded as satisfiability of a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ by including a polynomial for each gate along with $\left\{x_i^2 - x_i = 0\right\}$ to ensure Booleanity. Thus, number of satisfiable assignments to the Circuit-SAT is $\equiv |\mathcal{V}_{\boldsymbol{f}}| \pmod p$, which is $0 \pmod p$ iff the final coefficient of the max-degree monomial is 0.

the variables $x_1, \ldots, x_n$ of order p (i.e. σ^p is the identity permutation). Suppose that for every $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{f}}}$, it holds that $\sigma(\boldsymbol{x}) \in \overline{\mathcal{V}_{\boldsymbol{f}}} \setminus \{\boldsymbol{x}\}$; in other words $\boldsymbol{x}, \sigma(\boldsymbol{x}), \sigma^2(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})$ are all distinct and in $\overline{\mathcal{V}_{\boldsymbol{f}}}$ (where, $\sigma(\boldsymbol{x})$ denotes the assignment obtained by permutating the variables of the assignment $\boldsymbol{x}$ according to σ). This implies that the elements of $\overline{\mathcal{V}_{\boldsymbol{f}}}$ can be partitioned into groups of size p (given by the orbits of the action σ) and hence $|\overline{\mathcal{V}_{\boldsymbol{f}}}| \equiv 0 \pmod p$. Hence, such a σ provides a syntactically refutable proof that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$ and hence that (Extended CW Condition) hold.

Hence, we further generalize $\textsc{GeneralChevalley}_p$ into a problem that incorporates this additional "proof of cancellation" in the form of a permutation $\sigma \in S_n$.

1.2.3.3 Chevalley with Symmetry

We consider a union of two polynomial systems $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$. Even if both $\boldsymbol{g}$ and $\boldsymbol{h}$ satisfy (CW Condition), the combined system $\boldsymbol{f} := (g_1, \ldots, g_{m_g}, h_1, \ldots, h_{m_h})$ might not satisfy (CW Condition) and it might even be the case that $|\mathcal{V}_{\boldsymbol{f}}|$ is not a multiple of p. Thus, we need to bring in some additional conditions.

We start by observing that since $|\mathcal{V}_{\boldsymbol{f}}| + |\overline{\mathcal{V}_{\boldsymbol{f}}}| = p^n$, it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$ if and only if $|\overline{\mathcal{V}_{\boldsymbol{f}}}| \equiv 0 \pmod p$. Also note that, $|\overline{\mathcal{V}_{\boldsymbol{f}}}| = |\overline{\mathcal{V}_{\boldsymbol{g}}}| + |(\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}})|$.

If $\boldsymbol{g}$ satisfies the (General CW Condition) then we have that $|\mathcal{V}_{\boldsymbol{g}}| \equiv |\overline{\mathcal{V}_{\boldsymbol{g}}}| \equiv 0 \pmod p$. A simple way to enforce that $|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}| \equiv 0 \pmod p$ is to enforce a "symmetry", namely that its elements can be grouped into groups of size p each. We impose this grouping with a permutation $\sigma \in S_n$ of the variables $x_1, \ldots, x_n$ of order p such that for any $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, it holds that $\sigma(\boldsymbol{x}) \in (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \setminus \{\boldsymbol{x}\}$; or in other words that $\boldsymbol{x}, \sigma(\boldsymbol{x}), \sigma^2(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})$ are all distinct and contained in $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

We now define the following natural total search problem.

$\textsc{ChevalleyWithSymmetry}_p$
Given two explicit polynomial systems $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$, and an $\boldsymbol{x}^\star \in \mathcal{V}_{\boldsymbol{f}}$ (where $\boldsymbol{f} := (\boldsymbol{g}, \boldsymbol{h})$) and a permutation $\sigma \in S_n$ of order p, output one of the following:
▷ [*Refuting Witness* – 1] A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{g}}$.
▷ [*Refuting Witness* – 2] $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \notin (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \setminus \{\boldsymbol{x}\}$.
▷ $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}} \setminus \{\boldsymbol{x}^\star\}$.

The above problem is natural, because the input consists of a system of polynomial in an explicit form, i.e. as a sum of monic monomials, together with a permutation in S_n given say in one-line notation. Also, observe that when $\boldsymbol{h}$ is empty, the above problem coincides with $\textsc{GeneralChevalley}_p$ (since $\overline{\mathcal{V}_{\boldsymbol{h}}} = \emptyset$ when $\boldsymbol{h}$ is empty). Our main result is the following (proved in Section 4).

▶ **Theorem 3.** *For any prime p, $\textsc{ChevalleyWithSymmetry}_p$ is PPA_p-complete.*

1.3 Complete Problems via Small Depth Arithmetic Formulas

While the $\textsc{ChevalleyWithSymmetry}_p$ problem may seem somewhat contrived, the importance of its PPA_p-completeness is illustrated by our next result (proved in Section 5) showing that we can reformulate any of the proposed definitions of PPA_p, by restricting the circuit in the input to be just constant depth arithmetic formulas with gates $\times \pmod p$ and $+ \pmod p$ (we call this class $\mathsf{AC}^0_{\mathbb{F}_p}$). This result is analogous to the NP-completeness of SAT which basically shows that $\textsc{CircuitSAT}$ remains NP-complete even if we restrict the input circuit to be a (CNF) formula of depth 2.

▶ **Theorem 4.** *LONELY$_p$/BIPARTITE$_p$/LEAF$_p$ with $\mathsf{AC}^0_{\mathbb{F}_p}$ input circuits are PPA_p–complete.*

We hope that this theorem will be helpful in the context of proving PPA_p-hardness of other problems. There it would be enough to consider only constant depth arithmetic formulas (and hence NC^1 Boolean formulas) in the definitions of PPA_p as opposed to unbounded depth circuits. Such a simplification has been a key-step for proving hardness results for other TFNP subclasses, e.g. in the PPAD-hardness proofs of APPROXIMATE-NASH (cf. [35]).

1.4 Applications of Chevalley-Warning

Apart from its initial algebraic motivation, the Chevalley-Warning theorem has been used to derive several non-trivial combinatorial results. Alon et al. [3] show that adding an extra edge to any 4-regular graph forces it to contain a 3-regular subgraph. More generally, they prove that certain types of "almost" regular graphs contain regular subgraphs. Another application of CWT is in proving *zero-sum theorems* similar to the Erdös-Ginzburg-Ziv Theorem. A famous such application is the proof of Kemnitz's conjecture by Reiher [34].

We define two computational problems that we show are reducible to CHEVALLEY$_p$ and suffice for proving most of the combinatorial applications of the Chevalley-Warning Theorem mentioned above (for a certain range of parameters n and m). Both involve finding solutions to a system of linear equations modulo q, given as $\boldsymbol{A}\boldsymbol{x} \equiv \mathbf{0} \pmod q$ for $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$.

▷ BIS$_q$: Find $\boldsymbol{x} \in \{0,1\}^n$ satisfying $\boldsymbol{x} \neq \mathbf{0}$ and $\boldsymbol{A}\boldsymbol{x} \equiv \mathbf{0} \pmod q$.

▷ SIS$_q$: Find $\boldsymbol{x} \in \{-1,0,1\}^n$ satisfying $\boldsymbol{x} \neq \mathbf{0}$ and $\boldsymbol{A}\boldsymbol{x} \equiv \mathbf{0} \pmod q$.

The second problem is a special case of the well-known *short integer solution* problem in ℓ_∞ norm. Note that, when $n > m \cdot \log_2 q$, the totality of SIS$_q$ is guaranteed by pigeonhole principle; that is, SIS$_q$ is in PPP in this range of parameters. We are interested in identifying the range of parameters that places this problem in PPA_q – see Definitions 40 and 41 for the precise range of parameters n and m that we consider. In Theorem 42, we prove a formal version of the following:

▶ **Theorem** (Informal)**.** *For a certain range of parameters n, m, it holds that*

1. *For all primes p : BIS$_p$ and SIS$_p$ are Karp-reducible to CHEVALLEY$_p$, hence are in PPA_p.*
2. *For all q : BIS$_q$ and SIS$_q$ are Turing-reducible to any PPA_q–complete problem.*
3. *For all k : BIS$_{2^k}$ is solvable in polynomial time.*
4. *For k and ℓ : SIS$_{2^k 3^\ell}$ is solvable in polynomial time.*

Even though the SIS$_q$ problem is well-studied in lattice theory, not many results are known in the regime where q is a constant and the number of variables depends linearly on the number of equations. Part (1) of the above theorem establishes a reduction from SIS$_p$ to CHEVALLEY$_p$ for prime p. Part (2) follows by a bootstrapping method that allows us to combine algorithms for SIS$_{q_1}$ and SIS$_{q_2}$ to give an algorithm for SIS$_{q_1 q_2}$ (for a certain regime for parameters n and m). Finally Parts (3) and (4) results follow by using this bootstrapping method along with the observation that Gaussian elimination provides valid solutions for BIS$_2$ (hence also SIS$_2$) and for SIS$_3$.

1.5 Structural properties

Relation to other classes

Buss and Johnson [13, 27] had defined a class PMOD_q which turns out to be slightly weaker than PPA_q (refer to Section 7). Despite this slight difference between the definitions of PPA_q and PMOD_q, we can still deduce statements about PPA_q from the work of [27]. In particular, it follows that PPAD $\subseteq \mathsf{PPA}_q$ (refer to Subsection 7.1).

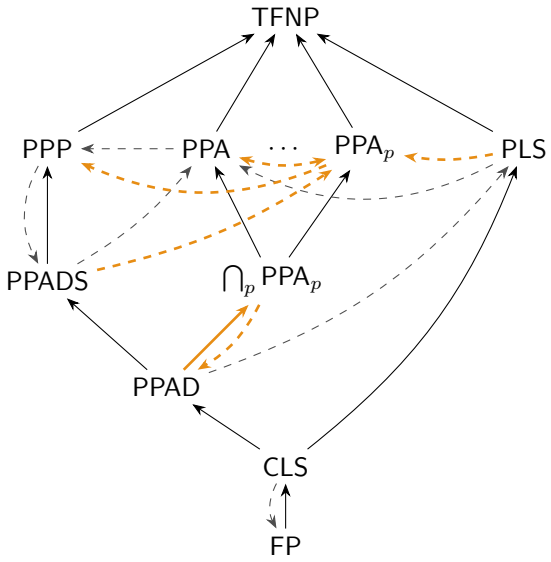


Figure 1 The landscape of TFNP subclasses. A solid arrow $\mathsf{M}_1 \to \mathsf{M}_2$ denotes $\mathsf{M}_1 \subseteq \mathsf{M}_2$, and a dashed arrow $\mathsf{M}_1 \dashrightarrow \mathsf{M}_2$ denotes an oracle separation: $\mathsf{M}_1^{\mathcal{O}} \not\subseteq \mathsf{M}_2^{\mathcal{O}}$ relative to some oracle $\mathcal{O}$. The relationships involving PPA_p are highlighted in yellow. See Section 7 for details.

More broadly, a near-complete picture of the power of PPA_q relative to other subclasses of TFNP is summarized in Figure 1. These relationships (inclusions and oracle separations) mostly follow from prior work in proof complexity [6, 12, 27, 23] (refer to Subsection 7.2).

Closure under Turing reductions

Recall that TFNP subclasses are defined as the set of all total search problems that are *many-one* reducible (aka Karp–reducible) to the corresponding complete problems. One can ask whether more power is gained by allowing *Turing reductions*, that is, polynomially many oracle queries to the corresponding complete problem. Buss and Johnson [13] showed that PLS, PPAD, PPADS, PPA are closed under Turing reductions (with a notable exception of PPP, which remains open). We show this for PPA_p when p is a prime.

▶ **Theorem 5.** $\mathsf{FP}^{\mathsf{PPA}_p} = \mathsf{PPA}_p$ *for every prime* p.

By contrast, it follows from [13, §6] that PPA_q is not closed under *black-box* Turing reductions for non-prime powers q. See Subsection 7.3 for details.

1.6 Open questions

Factoring

It has been shown that Factoring reduces to PPP-complete problems as well as to PPA-complete problems [11, 26], albeit under randomized reductions (which can be derandomized assuming the Generalized Reimann Hypothesis). It has been asked whether in fact Factoring could be reduced to PPAD-complete problems [26]. As a step towards this problem, we propose the following question.

▶ **Open Problem 1.** *Is* Factoring *in* PPA_p *for all primes* p *(perhaps under randomized reductions)?*

This is clearly an easier problem since $\mathsf{PPAD} \subseteq \mathsf{PPA}_p$. Interestingly, note that there exists an oracle $\mathcal{O}$ relative to which $\bigcap_p \mathsf{PPA}_p^{\mathcal{O}} \not\subseteq \mathsf{PPAD}^{\mathcal{O}}$. Thus, the above problem, even if established for all prime p, is still weaker than showing that FACTORING reduces to PPAD-complete problems.

Necklace Splitting

The q-NECKLACE-SPLITTING problem is defined as follows: There is an open necklace[6] with $q \cdot a_i$ beads of color i, for $i \in [n]$. The goal is to cut the necklace in $(q-1) \cdot n$ places and partition the resulting substrings into k collections, each containing precisely a_i beads of color i for each $i \in [n]$.

The fact that such a partition exists was first shown in the case of $q = 2$ by Goldberg and West [22] and by Alon and West [4]. Later, Alon [2] proved it for all $q \geq 2$. As mentioned before, Filos-Ratsikas and Goldberg [21] showed that the 2-NECKLACE-SPLITTING problem is PPA-complete. Moreover, they put forth the following question (which we strengthen further).

▶ **Open Problem 2.** *Is q-NECKLACE-SPLITTING in PPA_q? More strongly, is it PPA_q-complete?*

While we do not know how to prove/disprove this yet, we point out that it was also shown in [21] that 2^k-NECKLACE-SPLITTING is in fact in PPA_2. This is actually well aligned with this conjecture since we showed that $\mathsf{PPA}_{2^k} = \mathsf{PPA}_2$ (Theorem 1).

Bárány-Shlosman-Szücs theorem

Alon's proof of the q-Necklace-Splitting theorem [2] was topological and used a certain generalization of the Borsuk-Ulam theorem due to Bárány, Shlosman and Szücs [14]. Since the computational BORSUK-ULAM problem is PPA-complete, we could ask a similar question about this generalization.

▶ **Open Problem 3.** *Is BÁRÁNY-SHLOSMAN-SZÜCS$_p$ problem in PPA_p (perhaps even PPA_p-complete)?*

Applications of Chevalley-Warning Theorem

We conclude with some interesting directions for further exploring the connections of CHEVALLEY with other computational problems.

▶ **Open Problem 4.** *Does SIS_q admit worst-to-average case reductions to other lattice problems in our range of parameters? Or is it average-case hard assuming standard cryptographic assumptions, e.g. the "learning with errors" assumption?*

If resolved positively, the above would serve as evidence of the average-case hardness for the class PPA_p, similar to the evidence that we have for PPA by reduction from FACTORING.

▶ **Open Problem 5.** *For all primes p, is CHEVALLEY$_p$ reducible to BIS_p?*

▶ **Open Problem 6.** *For all q, is there a non-trivial regime of parameters n, m where BIS_q is solvable in polynomial time?*

---

[6] an "open necklace" means that the beads form a string, not a cycle

2 The class PPA_q

Search Problems in FNP and TFNP

A search problem in FNP is defined by a polynomial time computable relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, that is, for every (x, y), it is possible to decide whether $(x, y) \in \mathcal{R}$ in $\mathrm{poly}(|x|, |y|)$ time. A solution to the search problem on input x is a y such that $|y| = \mathrm{poly}(|x|)$ and $(x, y) \in \mathcal{R}$. For convenience, define $\mathcal{R}(x) \coloneqq \{y : (x, y) \in \mathcal{R}\}$. A search problem is *total* if for every input $x \in \{0,1\}^*$, there exists $y \in \mathcal{R}(x)$ such that $|y| \leq \mathrm{poly}(|x|)$. TFNP is the class of all total search problems in FNP.

Reducibility among search problems

A search problem $\mathcal{R}_1$ is *Karp-reducible* (or *many-one reducible*) to a search problem $\mathcal{R}_2$, or $\mathcal{R}_1 \preceq \mathcal{R}_2$ for short, if there exist polynomial-time computable functions f and g such that given any instance x of $\mathcal{R}_1$, $f(x)$ is an instance of $\mathcal{R}_2$ such that for any $y \in \mathcal{R}_2(f(x))$, it holds that $g(x, f(x), y) \in \mathcal{R}_1(x)$.

On the other hand, we say that $\mathcal{R}_1$ is *Turing-reducible* to $\mathcal{R}_2$, or $\mathcal{R}_1 \preceq_T \mathcal{R}_2$ for short, if there exists a polynomial-time oracle Turing machine that on input x to $\mathcal{R}_1$, makes oracle queries to $\mathcal{R}_2$, and outputs a $y \in \mathcal{R}_1(x)$. In this paper, we primarly deal with Karp-reductions, except in Subsection 7.3, where we compare the two different notions of reductions in the context of PPA_q.

PPA_q via complete problems

We describe several total search problems (parameterized by q) that we show to be inter-reducible. PPA_q is then defined as the set of all search problems reducible to either one of the search problems defined below.

Recall that Boolean circuits take inputs of the form $\{0,1\}^n$ and operate using $(\wedge, \vee, \neg)$ gates. In addition, we'll also consider circuits acting on inputs in $[q]^n$. We interpret the input to be of the form $(\{0,1\}^{\lceil \log q \rceil})^n$, where the circuit will be evaluated only on inputs where each block of $\lceil \log q \rceil$ bits represents a element in $[q]$. In the case where q is a prime, we could also represent the circuit as $C : \mathbb{F}_q^n \to \mathbb{F}_q^n$ with arbitrary gates of the form $g : \mathbb{F}_q^2 \to \mathbb{F}_q$. However, we can simulate any such gate with $\mathrm{poly}(q)$ many $+$ and $\times$ operations (over $\mathbb{F}_q$) along with a constant (1) gate. Hence, in the case of prime q, we'll assume that such circuits are composed of only $(+, \times, 1)$ gates.

▶ **Definition 6** ($\textsc{Bipartite}_q$).

Principle: *A bipartite graph with a non-multiple-of-q degree node has another such node.*

Object: *Bipartite graph* $G = (V \cup U, E)$. *Designated vertex* $v^* \in V$

Inputs: $\triangleright$ $C : \{0,1\}^n \to (\{0,1\}^n)^k$, *with* $(\{0,1\}^n)^k$ *interpreted as a k-subset of* $\{0,1\}^n$
$\triangleright$ $v^* \in \{0\} \times \{0,1\}^{n-1}$ *(usually* 0^n*)*

Encoding: $V \coloneqq \{0\} \times \{0,1\}^{n-1}$, $U \coloneqq \{1\} \times \{0,1\}^{n-1}$,
$E \coloneqq \{(v, u) : v \in V \cap C(u)$ *and* $u \in U \cap C(v)\}$

Solutions: v^* *if* $\deg(v^*) \equiv 0 \pmod q$ *and*
$v \neq v^*$ *if* $\deg(v) \not\equiv 0 \pmod q$

▶ **Definition 7** ($\textsc{Lonely}_q$).

Principle: *A q-dimensional matching on a non-multiple-of-q many vertices has an isolated node.*

Object: *q-dimensional matching $G = (V, E)$.*
Designated vertices $V^* \subseteq V$ with $|V^*| \leq q - 1$

Inputs: $\triangleright$ $C : [q]^n \to [q]^n$
$\triangleright$ $V^* \subseteq [q]^n$ *with* $|V^*| \leq q - 1$

Encoding: *$V \coloneqq [q]^n$. For distinct $v_1, \ldots, v_q$, edge $e \coloneqq \{v_1, \ldots, v_q\} \in E$ if $C(v_i) = v_{i+1}$, $C(v_q) = v_1$*

Solutions: *$v \in V^*$ if $\deg(v) = 1$ and*
$v \notin V^*$ if $\deg(v) = 0$

▶ **Definition 8** ($\textsc{Leaf}_q$).

Principle: *A q-uniform hypergraph with a non-multiple-of-q degree node has another such node.*

Object: *q-uniform hypergraph $G = (V, E)$. Designated vertex $v^* \in V$*

Inputs: $\triangleright$ $C : \{0,1\}^n \to (\{0,1\}^{nq})^q$*; Interpret $(\{0,1\}^{nq})^q$ as q many q-subsets of $\{0,1\}^n$*
$\triangleright$ $v^* \in \{0,1\}^n$ *(usually 0^n)*

Encoding: *$V \coloneqq \{0,1\}^n$. For distinct $v_1, \ldots, v_q$, edge $e \coloneqq \{v_1, \ldots, v_q\} \in E$ if $e \in C(v)$ for all $v \in e$*

Solutions: *v^* if $\deg(v) \equiv 0 \pmod q$ and*
$v \neq v^*$ if $\deg(v) \not\equiv 0 \pmod q$

We remark that $\textsc{Lonely}_q$ and $\textsc{Leaf}_q$ are modulo-q analogs of the PPA-complete problems Lonely and Leaf [33, 5]. We prove the following theorem in Appendix A.

▶ **Theorem 9.** *The problems $\textsc{Bipartite}_q$, $\textsc{Lonely}_q$ and $\textsc{Leaf}_q$ are inter-reducible.*

▶ Remark 10 (Simplifications in describing reductions.). We will use the following simple conventions repeatedly, in order to simplify the descriptions of reductions between different search problems.

1. We will often use "algorithms", instead of "circuits" to encode our hypergraphs. It is standard to simulate polynomial-time algorithms by polynomial sized circuits.
2. While our definitions require vertex sets to be of a very special form, e.g. $\{0,1\}^n$ or $[q]^n$, it will hugely simplify the description of our reductions to let vertex sets be of arbitrary sizes. This is not a problem as long as the vertex set is efficiently indexable, that is, elements of V must have a $\mathrm{poly}(n)$ length representation and we must have a poly-time computable bijective map $\varphi : V \to [|V|]$, whose inverse is also poly-time computable. We could then use φ to interpret the first $|V|$ elements of $\{0,1\}^n$ (or $[q]^n$) as vertices in V. Note that, we need to ensure that no new solutions are introduced in this process. In the case of $\textsc{Bipartite}_q$ or $\textsc{Leaf}_q$, we simply leave the additional vertices isolated and they don't contribute any new solutions. In the case of $\textsc{Lonely}_q$ we need to additionally ensure that $|V| \equiv 0 \pmod q$, so that we can easily partition the remaining vertices into q-uniform hyperedges thereby not introducing any new solutions.
3. The above simplification gives us that all our problems have an *instance-extension property* (cf. [10]) – this will be helpful in proving Theorem 5.
4. To simplify our reductions even further, we'll often describe the edges/hyperdges directly instead of specifying how to compute the neighbors of a given vertex. This is only for simplicity and it will be easy to see how to compute the neighbors of any vertex locally.

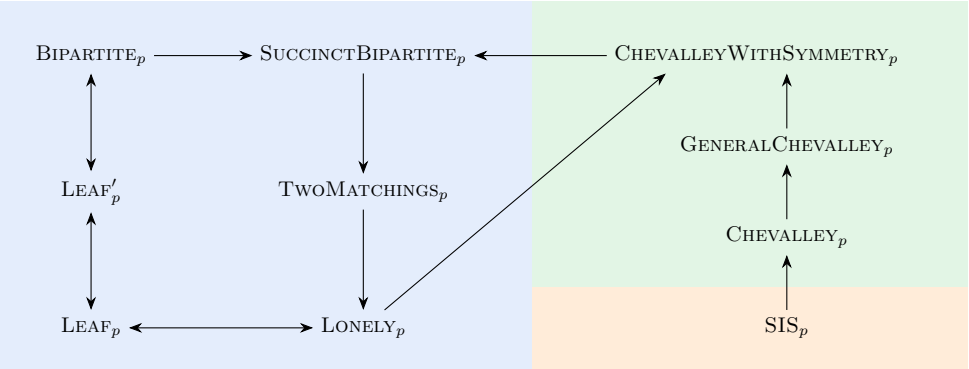


Figure 2 Total search problems studied in this work. An arrow $A \to B$ denotes a reduction $A \preceq B$ that we establish. Problems in the blue region are non-natural problems, which are all complete for PPA_p. Problems in the green region are natural problems of which CHEVALLEYWITHSYMMETRY$_p$ is the one we show to be PPA_p–complete. The problem in the orange region is a cryptographically relevant problem.

3 Characterization via Primes

In this section we prove Theorem 1, namely $\mathsf{PPA}_q = \&_{p|q}\,\mathsf{PPA}_p$. The theorem follows by combining the following two ingredients.

§3.1: $\mathsf{PPA}_{qr} = \mathsf{PPA}_q \,\&\, \mathsf{PPA}_r$ for any coprime q and r.

§3.2: $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$ for any prime power p^k.

3.1 Coprime case

$\mathsf{PPA}_{qr} \supseteq \mathsf{PPA}_q \,\&\, \mathsf{PPA}_r$

We show that LONELY$_q$ & LONELY$_r$ reduces to LONELY$_{qr}$. Recall that an instance of LONELY$_q$ & LONELY$_r$ is a tuple (C, V^*, b) where (C, V^*) describes an instance of either LONELY$_q$ or LONELY$_r$ as chosen by $b \in \{0,1\}$. Suppose wlog that $b = 0$, so the input encodes a q-dimensional matching $G = (V, E)$ over $V = [q]^n$ with designated vertices $V^* \subseteq V$, $|V^*| \not\equiv 0 \pmod q$. We can construct a qr-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertices $\overline{V} \coloneqq V \times [r]$ as follows: For every hyperedge $e \coloneqq \{v_1, \ldots, v_q\} \in E$, we include the hyperedge $e \times [r]$ in $\overline{E}$. We let the designated vertices of $\overline{G}$ be $\overline{V}^* \coloneqq V^* \times [r]$. Note that $|\overline{V}^*| \not\equiv 0 \pmod{qr}$. It is easy to see that a vertex (v, i) is isolated in G' iff v is isolated in G. This completes the reduction since $\overline{V}$ is efficiently indexable, and the neighbors of any vertex in $\overline{V}$ are locally computable using black-box access to C.

$\mathsf{PPA}_{qr} \subseteq \mathsf{PPA}_q \,\&\, \mathsf{PPA}_r$

We show that BIPARTITE$_{qr}$ reduces to BIPARTITE$_q$ & BIPARTITE$_r$. Our input instance of BIPARTITE$_{qr}$ is a circuit $C : \{0,1\}^n \to (\{0,1\}^n)^k$ that encodes a bipartite graph $G = (V \cup U, E)$ with a designated node $v^* \in V$. If $\deg(v^*) \equiv 0 \pmod{qr}$, then we already have solved the problem and no further reduction is necessary. Otherwise, if $\deg(v^*) \not\equiv 0 \pmod{qr}$, we have, by the coprime-ness of q and r, that either $\deg(v^*) \not\equiv 0 \pmod q$ or $\deg(v^*) \not\equiv 0 \pmod r$. In the first case (the second case is analogous), we can simply view (G, v^*) as an instance of BIPARTITE$_q$, since vertices with degree $\not\equiv 0 \pmod q$ in G are also solutions to BIPARTITE$_{qr}$.

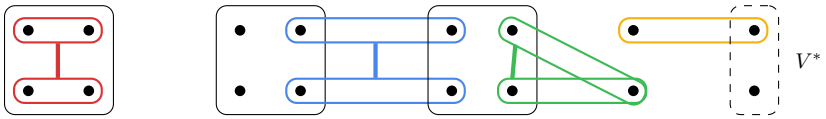


Figure 3 Illustration of the proof of $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$ for $p = 2$, $k = 2$, $n = 2$, $t = 1$. In black, we indicate the 4-dimensional matching G. In color, we highlight some of the vertices of $\overline{G}$ and the edges between them. The vertices of $\overline{G}$ in red, blue and green are paired up and hence are non-solutions; whereas the vertex in yellow is isolated and not in $\overline{V}^*$ and hence a solution.

3.2 Prime power case

$\mathsf{PPA}_{p^k} \supseteq \mathsf{PPA}_p$ follows immediately from our proof of $\mathsf{PPA}_{qr} \supseteq \mathsf{PPA}_q$ & PPA_r, which didn't require that q and r be coprime. It remains to show $\mathsf{PPA}_{p^k} \subseteq \mathsf{PPA}_p$. We exploit the following easy fact.

▶ **Fact 11.** *For all primes p, it holds that,*

$$\text{for integers } t, c > 0: \quad \binom{c \cdot p^t}{p^t} \equiv 0 \pmod{p} \quad \text{if and only if} \quad c \equiv 0 \pmod{p} \tag{3.1}$$

$$\text{for integer } k > 0: \quad \binom{p^k}{i} \equiv 0 \pmod{p} \quad \text{for all} \quad 0 < i < p^k \tag{3.2}$$

We reduce $\textsc{Lonely}_{p^k}$ to $\textsc{Lonely}_p$. Our instance of $\textsc{Lonely}_{p^k}$ is (C, V^*) where C implicitly encodes a p^k-dimensional matching $G = (V = [p^k]^n, E)$ and a designated vertex set $V^* \subseteq V$ such that $|V^*| \not\equiv 0 \pmod{p^k}$.

Let p^t, $0 \le t < k$, be the largest power of p that divides $|V^*|$. Through local operations we construct a p-dimensional matching hypergraph $\overline{G} = (\overline{V}, \overline{E})$ over vertices $\overline{V} \coloneqq \binom{V}{p^t}$ (set of all size-p^t subsets of V) with designated vertices $\overline{V}^* \coloneqq \binom{V^*}{p^t}$. From Eq. 3.1, we get that $|\overline{V}| \equiv 0 \pmod{p}$ and $|\overline{V}^*| \not\equiv 0 \pmod{p}$.

We will describe an algorithm that on vertex $\overline{v} \in \overline{V}$ outputs a hyperedge of p vertices that contains $\overline{v}$ (if any). To this end, first fix an algorithm that for any set $e \coloneqq \{u_1, \ldots, u_{p^k}\} \subseteq V$ and for any $1 \le i \le p^t$, computes some "canonical" partition of the set $\binom{e}{i}$ into subsets of size p, and moreover assigns a canonical cyclic order within each such subset. This is indeed possible because of Eq. 3.2, since $t < k$.

Given a vertex $\overline{v} \coloneqq \{v_1, \ldots, v_{p^t}\} \in \overline{V}$,

- ▷ Compute all edges $e_1, \ldots, e_\ell \in E$ that include some $v \in \overline{v}$.
- ▷ For edge e_j, define $S_j \coloneqq e_j \cap \overline{v}$ and let $S_j^1, \ldots, S_j^{p-1}$ be the remaining subsets in the same partition as S_j in the canonical partition of $\binom{e_j}{|S_j|}$, listed in the canonical cyclic order starting at S_j. Also, let S_0 be the set of untouched vertices in $\overline{v}$. Observe that $\overline{v} = S_0 \cup S_1 \cup \ldots \cup S_\ell$.
- ▷ Output neighbors of $\overline{v}$ as the vertices $\overline{v}_1, \ldots, \overline{v}_{p-1}$ where $\overline{v}_i \coloneqq S_0 \cup S_1^i \cup \ldots \cup S_\ell^i$.

It is easy to see that $\overline{v}$ is isolated in $\overline{G}$ iff all $v \in \overline{v}$ are isolated in G. Moreover, any isolated vertex in $\overline{V} \setminus \overline{V}^*$ contains at least one isolated vertex in $V \setminus V^*$; and a non-isolated vertex in $\overline{V}^*$ contains at least one non-isolated vertex in V^* (in fact p^t many).

The edges of $\overline{G}$ can indeed be computed efficiently with just black-box access to C. In order to complete the reduction, we only need that $\overline{V}$ is efficiently indexable. This is indeed standard; see [31, §2.3] for a reference. See Figure 3 for an illustration of the proof.

▶ Remark 12. Note that the size of the underlying graph blows up polynomially in our reduction. We do not know whether a reduction exists that avoids such a blow-up, although we suspect that the techniques of [6] can be used to show that some blow-up is necessary for black-box reductions.

4 A Natural Complete Problem

We start with some notations that will be useful for the presentation of our results.

Notations. For any polynomial $g \in \mathbb{F}_p[\boldsymbol{x}]$, we define $\deg(g)$ to be the degree of g. We define *the expansion to monic monomials* of g as $\sum_{\ell=1}^{L} t_\ell(\boldsymbol{x})$, where $t_\ell(\boldsymbol{x})$ is a monic monomial in $\mathbb{F}_p[\boldsymbol{x}]$, i.e. a monomial with coefficient 1. For example, the expansion of the polynomial $g(x_1, x_2) = x_1 \cdot (2x_1 + 3x_2)$ is given by $x_1^2 + x_1^2 + x_1x_2 + x_1x_2 + x_1x_2$.

For a polynomial system $\boldsymbol{f} := (f_1, \ldots, f_m) \in \mathbb{F}_p[\boldsymbol{x}]^m$, its affine variety $\mathcal{V}_{\boldsymbol{f}} \subseteq \mathbb{F}_p^n$ is defined as $\mathcal{V}_{\boldsymbol{f}} := \{\boldsymbol{x} \in \mathbb{F}_p^n \mid \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}\}$. Let $\overline{\mathcal{V}_{\boldsymbol{f}}} := \mathbb{F}_p^n \setminus \mathcal{V}_{\boldsymbol{f}}$. If the constant term of each f_i is 0, we say that $\boldsymbol{f}$ is *zecote*, standing for "Zero Constant Term" (owing to lack of known terminology and creativity on our part).

4.1 The Chevalley-Warning Theorem

We repeat the formal statement of Chevalley-Warning Theorem together with its proof.

▶ **Chevalley-Warning Theorem** ([15, 37]). *For any prime p and a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ satisfying $\sum_{i=1}^{m} \deg(f_i) < n$* (CW Condition)*, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

We describe the proof of CWT through Lemma 14. Even though there are direct proofs, the following presentation helps motivate the generalizations we study in future sections. Given a polynomial system $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, a key idea in the proof is the polynomial $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) := \prod_{i=1}^{m} \mathsf{CW}_{f_i}(\boldsymbol{x})$ where each $\mathsf{CW}_{f_i}(\boldsymbol{x}) := (1 - f_i(\boldsymbol{x})^{p-1})$. Observe that $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. The following definition describes the notion of a max-degree monomial of $\mathsf{CW}_{\boldsymbol{f}}$ that plays an important role in the proof.

▶ **Definition 13** (Max-Degree Monic Monomials). *For any prime p, let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ and let the expansion into monic monomials of $\mathsf{CW}_{f_i}(\boldsymbol{x})$ be $\sum_{\ell=1}^{r_i} t_{i,\ell}(\boldsymbol{x})$. Let also $U_i = \{(i, \ell) \mid \ell \in [r_i]\}$ and $U = \bigtimes_{i=1}^{m} U_i$, we define the following quantities.*

1. *A monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ is a product $t_S(\boldsymbol{x}) = \prod_{i=1}^{m} t_{s_i}(\boldsymbol{x})$ for $S = (s_1, \ldots, s_m) \in U$.*
2. *A max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{f}}$ is any monic monomial $t_S(\boldsymbol{x})$, such that $t_S(\boldsymbol{x}) \equiv \prod_{j=1}^{n} x_j^{p-1} \left(\mathrm{mod}\ \{x_i^p - x_i\}_{i \in [n]}\right)$.*
3. *We define $\mathcal{M}_{\boldsymbol{f}}$ to be the set of max-degree monic monomials of $\mathsf{CW}_{\boldsymbol{f}}$, i.e. $\mathcal{M}_{\boldsymbol{f}} := \{S \in U \mid t_S$ is a max-degree monic monomial of $\mathsf{CW}_{\boldsymbol{f}}\}$.*

In words, the monomials $t(S)$ are precisely the ones that arise when symbolically expanding $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x})$. We illustrate this with an example: Let $p = 3$ and $f_1(x_1, x_2) = x_1 + x_2$ and $f_2(x_1, x_2) = x_1^2$. Then modulo $\{x_1^3 - x_1, x_2^3 - x_2\}$, we have

$$\begin{aligned} \mathsf{CW}_{(f_1,f_2)}(x_1, x_2) &= (1 - (x_1 + x_2)^2)(1 - (x_1^2)^2) \\ &= (1 - x_1^2 - 2x_1x_2 - x_2^2) \cdot (1 - x_1^2) \\ &= (1 + x_1^2 + x_1^2 + x_1x_2 + x_2^2 + x_2^2) \cdot (1 + x_1^2 + x_1^2) \end{aligned}$$

Thus there are 18 ($= 6 \times 3$) monic monomials in the system (f_1, f_2). The monomial corresponding to $S = ((1, 5), (2, 2))$ is a maximal monomial since the 5-th term in CW_{f_1} is x_2^2 and 2-nd term in CW_{f_2} is x_1^2. Using the above definitions, we now state the main technical lemma of the proof of CWT.

▶ **Lemma 14** (Main Lemma in the proof of CWT). *For any prime p and any system of polynomials $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$, it holds that $|\mathcal{V}_{\boldsymbol{f}}| \equiv (-1)^n |\mathcal{M}_{\boldsymbol{f}}| \pmod p$.*

Proof. As noted earlier, $\mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) = 1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ and is 0 otherwise. Thus, it follows that $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x}\in\mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod p$. For any monic monomial $m(\boldsymbol{x}) = \prod_{j=1}^n x_j^{d_j}$, it holds that $\sum_{\boldsymbol{x}\in\mathbb{F}_p^n} m(\boldsymbol{x}) = 0$ if $d_j < p-1$ for some x_j. On the other hand, for the monic max-degree monomial $m(\boldsymbol{x}) = \prod_{j=1}^n x_j^{p-1}$, it holds that $\sum_{\boldsymbol{x}\in\mathbb{F}_p^n} m(\boldsymbol{x}) = (p-1)^n$. Thus, we get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv \sum_{\boldsymbol{x}\in\mathbb{F}_p^n} \mathsf{CW}_{\boldsymbol{f}}(\boldsymbol{x}) \pmod p \equiv \sum_{S\in U} \sum_{\boldsymbol{x}\in\mathbb{F}_p^n} t_S(\boldsymbol{x}) \pmod p \equiv (-1)^n |\mathcal{M}_{\boldsymbol{f}}| \pmod p$. ◀

The proof of Chevalley-Warning Theorem follows easily from Lemma 14.

Proof of Chevalley-Warning Theorem. We have that $\deg(\mathsf{CW}_{\boldsymbol{f}}) \le (p-1)\sum_{i=1}^m \deg(f_i)$. Thus, if $\boldsymbol{f}$ satisfies (CW Condition), then $\deg(\mathsf{CW}_{\boldsymbol{f}}) < (p-1)n$ and hence $|\mathcal{M}_{\boldsymbol{f}}| = 0$. CWT now follows from Lemma 14. ◀

4.2 The Chevalley-Warning Theorem with Symmetry

In this section, we formalize the intuition that we built in Sections 1.2.2 and 1.2.3 to prove the more general statements to lead to the same conclusion as the Chevalley-Warning Theorem.

First, we prove a theorem that argues about the cardinality of $\mathcal{V}_{\boldsymbol{f}}$ directly using some symmetry of the system of polynomials $\boldsymbol{f}$. Then, combining this symmetry-based argument with the (General CW Condition) we get the generalization of the Chevalley-Warning Theorem. Our natural PPA_p-complete problem is based on this generalization.

The theorem statements are simplified using the definition of *free action* of a group. For a permutation over n elements $\sigma \in S_n$, we define $\langle\sigma\rangle$ to be the sub-group generated by σ and $|\sigma|$ to be the order of $\langle\sigma\rangle$. For $\boldsymbol{x} \in \mathbb{F}_p^n$, $\sigma(\boldsymbol{x})$ denotes the assignment obtained by permutating the variables of the assignment $\boldsymbol{x}$ according to σ.

▶ **Definition 15** (Free Group Action)**.** *Let $\sigma \in S_n$ and $\mathcal{V} \subseteq \mathbb{F}_p^n$, then we say that $\langle\sigma\rangle$ acts freely on $\mathcal{V}$ if, for every $\boldsymbol{x} \in \mathcal{V}$, it holds that $\sigma(\boldsymbol{x}) \in \mathcal{V}$ and $\boldsymbol{x} \neq \sigma(\boldsymbol{x})$.*

Our first theorem highlights the use of symmetry in arguing about the size of $|\mathcal{V}_{\boldsymbol{f}}|$.

▶ **Theorem 16.** *Let $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ be a system of polynomials. If there exists a permutation $\sigma \in S_n$ with $|\sigma| = p$ such that $\langle\sigma\rangle$ acts freely on $\overline{\mathcal{V}}_{\boldsymbol{f}}$, then $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

Proof. Since σ acts freely on $\overline{\mathcal{V}}_{\boldsymbol{f}}$, we can partition $\overline{\mathcal{V}}_{\boldsymbol{f}}$ into orbits of any $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{f}}$ under actions of $\langle\sigma\rangle$, namely sets of the type $\left\{\sigma^i(\boldsymbol{x})\right\}_{i\in[p]}$ for $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{f}}$. Since $\langle\sigma\rangle$ acts freely on $\overline{\mathcal{V}}_{\boldsymbol{f}}$, each such orbit has size p. Thus, we can conclude that $\left|\overline{\mathcal{V}}_{\boldsymbol{f}}\right| \equiv 0 \pmod p$ from which the theorem follows. ◀

▶ Remark 17. For any polynomial system $\boldsymbol{f}$ and any permutation σ, we can check in linear time if $|\sigma| = p$ and we can syntactically refute that $\langle\sigma\rangle$ acts freely on $\overline{\mathcal{V}}_{\boldsymbol{f}}$ with an $\boldsymbol{x} \in \mathbb{F}_p^n \setminus \{\boldsymbol{0}\}$ such that $\boldsymbol{f}(\sigma(\boldsymbol{x})) = \boldsymbol{0}$ or $\sigma(\boldsymbol{x}) = \boldsymbol{x}$.

We now state and prove an extension of CWT that captures both the argument from Lemma 14 and the symmetry argument from Theorem 16.

▶ **Theorem 18** (Chevalley-Warning with Symmetry Theorem)**.** *Let $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ and $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$ be two systems of polynomials, and $\boldsymbol{f} \coloneqq (\boldsymbol{g}, \boldsymbol{h})$. If there exists a permutation $\sigma \in S_n$ with $|\sigma| = p$ such that (1) $\mathcal{M}_{\boldsymbol{g}} = \emptyset$ and (2) $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, then $|\mathcal{V}_{\boldsymbol{f}}| = 0 \pmod p$.*

▶ Remark 19. We point to the special form of Condition 2. By definition, $\mathcal{V}_{\boldsymbol{f}} = \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$, hence if $\langle\sigma\rangle$ were to act freely on $\overline{\mathcal{V}_{\boldsymbol{g}}} \cup \overline{\mathcal{V}_{\boldsymbol{h}}}$ (or even $\mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$), then we could just use Theorem 16 to get that $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$. In the above theorem, we only require that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. Observe that Theorem 16 follows as a special case of CWT with Symmetry by setting $m_g = 0$. Additionally, by setting $m_h = 0$ we get the generalization of CWT corresponding to the (General CW Condition) as presented in Subsubsection 1.2.3.

Proof of Theorem 18. If $\mathsf{CW}_{\boldsymbol{g}}$ does not have any max- degree monic monomials, we have $|\mathcal{V}_{\boldsymbol{g}}| \equiv 0 \pmod p$ (similar to proof of CWT) and, since $\overline{\mathcal{V}_{\boldsymbol{g}}} = \mathbb{F}_p^n \setminus \mathcal{V}_{\boldsymbol{g}}$, we have $\left|\overline{\mathcal{V}_{\boldsymbol{g}}}\right| \equiv 0 \pmod p$. Also, since $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, we have $\left|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}\right| \equiv 0 \pmod p$ (similar to the proof of Theorem 16). Hence, $\left|\overline{\mathcal{V}_{\boldsymbol{f}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}}} \cup \overline{\mathcal{V}_{\boldsymbol{h}}}\right| = \left|\overline{\mathcal{V}_{\boldsymbol{g}}}\right| + \left|\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}\right| \equiv 0 \pmod p$. Thus, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$. ◀

4.3 Computational Problems Related to Chevalley-Warning Theorem

We now follow the intuition developed in the previous section and in Subsection 1.2 to formally define the computational problems $\textsc{Chevalley}_p$, $\textsc{GeneralChevalley}_p$, and $\textsc{ChevalleyWithSymmetry}_p$.

▶ **Definition 20** ($\textsc{Chevalley}_p$)**.**
Principle: *Chevalley-Warning Theorem.*
Input: $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ *: an explicit zecote polynomial system.*
Condition: $\sum_{i=1}^m \deg(f_i) < n$.
Output: $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ **Definition 21** ($\textsc{GeneralChevalley}_p$)**.**
Principle: *General Chevalley-Warning Theorem via* (General CW Condition).
Input: $\boldsymbol{f} \in \mathbb{F}_p[\boldsymbol{x}]^m$ *: an explicit zecote polynomial system.*
Output: ***0.*** *A max-degree monic monomial* $t_S(\boldsymbol{x})$ *of* $\mathsf{CW}_{\boldsymbol{f}}$*, or*
1. $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ **Definition 22** ($\textsc{ChevalleyWithSymmetry}_p$)**.**
Principle: *Chevalley-Warning Theorem with Symmetry (Theorem 18).*
Input: ▷ $\boldsymbol{g} \in \mathbb{F}_p[\boldsymbol{x}]^{m_g}$ *and* $\boldsymbol{h} \in \mathbb{F}_p[\boldsymbol{x}]^{m_h}$ *: explicit zecote polynomial systems*
▷ $\sigma \in S_n$ *: a permutation over* $[n]$.
Condition: $|\sigma| = p$.
Output: ***0. (a)*** *A max-degree monic monomial* $t_S(\boldsymbol{x})$ *of* $\mathsf{CW}_{\boldsymbol{g}}$*, or*
(b) $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ *such that* $\sigma(\boldsymbol{x}) \notin (\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}) \setminus \{\boldsymbol{x}\}$*, or*
1. $\boldsymbol{x} \in \mathbb{F}_p^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$.

▶ Remark 23. Some observations about the above computational problems follow:

1. In the problems $\textsc{GeneralChevalley}_p$ and $\textsc{ChevalleyWithSymmetry}_p$, we assume that, if the output is a max-degree monic monomial, this is given via the multiset of indices S that describes the monomial as formalized in Definition 13.
2. We have that $\textsc{Chevalley}_p \preceq \textsc{GeneralChevalley}_p \preceq \textsc{ChevalleyWithSymmetry}_p$. Thus, inclusion of $\textsc{ChevalleyWithSymmetry}_p$ in PPA_p implies that both $\textsc{Chevalley}_p$ and $\textsc{GeneralChevalley}_p$ are also in PPA_p. Also, in Section 6 we prove that SIS_p reduces to $\textsc{Chevalley}_p$, where SIS_p is a cryptographically relevant problem. This shows that the problems $\textsc{GeneralChevalley}_p$ and $\textsc{ChevalleyWithSymmetry}_p$ are at least as hard as SIS_p.

We restate our main result.

▶ **Theorem 3.** *For any prime* p*,* $\textsc{ChevalleyWithSymmetry}_p$ *is* PPA_p*-complete.*

4.4 ChevalleyWithSymmetry$_p$ is PPA$_p$–complete

4.4.1 ChevalleyWithSymmetry$_p$ is in PPA$_p$

Even though Papadimitriou [33] provided a rough proof sketch of CHEVALLEY$_p \in$ PPA$_p$, a formal proof was not given. We show that CHEVALLEYWITHSYMMETRY$_p$ is in PPA$_p$ (and so are GENERALCHEVALLEY$_p$ and CHEVALLEY$_p$). In order to do so we extend the definition of BIPARTITE$_q$ to instances where the vertices might have exponential degree and edges appear with multiplicity. The key here is to define a BIPARTITE$_q$ instance with unbounded (even exponential) degree, but with additional information that allows us to verify solutions efficiently.

▶ **Definition 24** (SUCCINCTBIPARTITE$_q$)**.**
Principle: *Similar to* BIPARTITE$_q$*, but degrees are allowed to be exponentially large, edges are allowed with multiplicities at most* $q-1$*.*
Object: *Bipartite graph* $G = (V \cup U, E)$ *s.t.* $E \subseteq V \times U \times \mathbb{Z}_q$*. Designated edge* $e^* \in E$*.*
Inputs: *Let* $V \coloneqq \{0\} \times \{0,1\}^{n-1}$ *and* $U \coloneqq \{1\} \times \{0,1\}^{n-1}$*:*
$\triangleright$ $\mathcal{C} : V \times U \to [q]$*, edge counting circuit*
$\triangleright$ $\phi_V : V \times U \times [q] \to (U \times [q])^q$*, grouping pivoted at* V
$\triangleright$ $\phi_U : V \times U \times [q] \to (V \times [q])^q$*, grouping pivoted at* U
$\triangleright$ $e^* = (v^*, u^*, k^*)$*, designated edge*
Encoding: $V \coloneqq \{0\} \times \{0,1\}^{n-1}$*,* $U \coloneqq \{1\} \times \{0,1\}^{n-1}$*,*
$E \coloneqq \{(v,u,k) : 1 \le k \le C(v,u),\ (v,u) \in V \times U\}$ *(edges with multiplicities)*
Edge (v,u,k) *is grouped with* $\{(v,u',k') : (u',k') \in \phi_V(v,u,k)\}$ *(pivoting at* v*), provided* $|\phi_V(v,u,k)| = q$*, all* $(v,u',k') \in E$ *and* $\phi_V(v,u',k') = \phi_V(v,u,k)$*.*
Edge (v,u,k) *is grouped with* $\{(v',u,k') : (v',k') \in \phi_U(v,u,k)\}$ *(pivoting at* u*), provided* $|\phi_U(v,u,k)| = q$*, all* $(v,u',k') \in E$ *and* $\phi_U(v',u,k') = \phi_V(v,u,k)$*.*
Solutions: e^* *if* e^* *is grouped, pivoting at* v^**, or if* e^* *is not grouped pivoting at* u^**, OR* $e \neq e^*$ *if* e *is not grouped pivoting at one of its ends.*

In words, SUCCINCTBIPARTITE$_p$ encodes a bipartite graph with arbitrary degree. Instead of listing the neighbors of a vertex using a circuit, we have a circuit that outputs the multiplicity of edges between any two given vertices. We are therefore unable to efficiently count the number of edges incident on any vertex. The grouping function ϕ_V aims to group edges incident on any vertex $v \in V$ into groups of size q. Similarly, ϕ_U aims to group edges incident on any vertex $u \in U$. The underlying principle is that if we have an edge e^* that is not grouped pivoting at v^* (one of its endpoints), then either e^* is not pivoted at u^* (its other endpoint) or there exists another edge that is also not grouped pivoting at one of its ends. Note that in contrast to the problems previously defined, v^* might still be an endpoint of a valid solution.

▶ **Lemma 25.** *For all primes* p*,* CHEVALLEYWITHSYMMETRY$_p \in$ PPA$_p$*.*

Proof. We reduce CHEVALLEYWITHSYMMETRY$_p$ to SUCCINCTBIPARTITE$_p$, which we show to be PPA$_p$–complete in Subsection A.1. Given an instance of CHEVALLEYWITHSYMMETRY$_p$, namely a zecote polynomial system $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$ and a permutation σ, we construct a bipartite graph $G = (U \cup V, E)$ encoded as an instance of SUCCINCTBIPARTITE$_p$ as follows.

Description of vertices. $U = \mathbb{F}_p^n$, namely all possible assignments of $\boldsymbol{x}$. The vertices of V are divided into two parts $V_1 \cup V_2$. The part V_1 contains one vertex for each monomial in the expansion of $\mathsf{CW}_{\boldsymbol{g}} = \prod_{i=1}^{m_g}(1 - g_i^{p-1})$. Since p is constant, we can efficiently list out the monomials of $1 - g_i^{p-1}$. For a fixed lexicographic ordering of the monomials of each $\mathsf{CW}_{g_i} \coloneqq 1 - g_i^{p-1}$, a monomial of $\mathsf{CW}_{\boldsymbol{g}}$ is represented by a tuple $(a_1, a_2, \ldots, a_{m_g})$ with

$0 \le a_i < L_i$, where a_i represents the index of a monomial of CW_{g_i} and L_i is the number of monomials of CW_{g_i}, where $a_i = 0$ corresponds to the constant term 1. The part $V_2 \coloneqq \binom{\mathbb{F}_p^n}{p}$, i.e. it contains a vertex for each subset of p distinct elements in $\mathbb{F}_p^n$.

Description of edges. We first describe the edges between U and V_1, namely include an edge between an assignment $\boldsymbol{x}$ and a monomial t with multiplicity $t(\boldsymbol{x})$. With these edges in place, the degree of vertices are as follows:

- $\boldsymbol{x} = 0^n$ has a single edge corresponding to the constant monomial 1, since $\boldsymbol{f}$ is zecote. We let this be the designated edge e^* in the final SuccinctBipartite$_p$ instance.
- $\boldsymbol{x} \notin \mathcal{V}_{\boldsymbol{g}}$ has $0 \pmod p$ edges (counting multiplicities). Since $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) = 0$, the sum over all monomials of $t(\boldsymbol{x})$ must be $0 \pmod p$.
- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$ has $1 \pmod p$ edges (counting multiplicities), since the sum over all $t(\boldsymbol{x})$ monomials gives $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) \equiv 1 \pmod p$.

Thus with the edges so far, the vertices (excluding 0^n), with degree $\not\equiv 0 \pmod p$ are precisely vertices $t \in V_1$ such that $\sum_{\boldsymbol{x}} t(\boldsymbol{x}) \not\equiv 0 \pmod p$ or $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \setminus \{0^n\}$. For the former case, if t contained a variable with degree less than $p-1$, then $\sum_{\boldsymbol{x}} t(\boldsymbol{x}) \equiv 0 \pmod p$. Hence, it must be that $t = \prod_{i=1}^{n} x_i^{p-1}$. In the later case, the degree of $\boldsymbol{x}$ is $1 \pmod p$ and hence $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$.

However, there is no guarantee that a vertex $\boldsymbol{x}$ with degree $1 \pmod p$ is in $\mathcal{V}_{\boldsymbol{h}}$ as well. To argue about $\boldsymbol{h}$, we add edges between U and V_2 that exclude solutions $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, on which σ acts freely (that is, $\sigma(\boldsymbol{x}) = \boldsymbol{x}$). More specifically, for $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, if $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$, we add an edge with multiplicity $p-1$ between $\boldsymbol{x}$ and $\Sigma_{\boldsymbol{x}} \in V_2$ where $\Sigma_{\boldsymbol{x}} \coloneqq \{\sigma^i(\boldsymbol{x})\}_{i \in \mathbb{Z}_p}$ (note that, in this case $|\Sigma_{\boldsymbol{x}}| = p$ since $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ and $|\sigma| = p$ is prime). Observe that, if a vertex in V_2 corresponds to a $\Sigma_{\boldsymbol{x}}$, it has p edges each with multiplicity $p-1$, one for each $\boldsymbol{x}' \in \Sigma_{\boldsymbol{x}}$ only if $\Sigma_{\boldsymbol{x}} \subseteq \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. If a vertex in V_2 does not correspond to a $\Sigma_{\boldsymbol{x}}$, then it has no edges. Thus, a vertex in V_2 has degree $\not\equiv 0 \pmod p$ iff it contains an $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

Thus, with all the edges added, vertices with degree $\not\equiv 0 \pmod p$ correspond to one of

- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$ such that $\boldsymbol{x} \neq \boldsymbol{0}$, or
- $t \in V_1$ such that $t(\boldsymbol{x})$ is a max-degree monomial or
- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) = \boldsymbol{x}$ or
- $v \in V_2$ such that $\exists\, \boldsymbol{x} \in v$ satisfying $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ and $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

These correspond precisely to the solutions of ChevalleyWithSymmetry$_p$. To summarize, the edge counting circuit C on input $(\boldsymbol{x}, t) \in U \times V_1$ outputs $t(\boldsymbol{x})$ and on input $(\boldsymbol{x}, v) \in U \times V_2$ outputs $p-1$ if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ and $v = \Sigma_{\boldsymbol{x}}$ and 0 otherwise.

Grouping Functions. The grouping functions ϕ_U and ϕ_V are defined as follows (analogous to the so-called "chessplayer algorithm" in [33]):

▷ Grouping ϕ_U (corresponding to endpoint in U):

- For $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{g}}}$: we have that there exists some i such that $\mathsf{CW}_{g_i}(\boldsymbol{x}) = 0$. Consider an edge of the form $(\boldsymbol{x}, (a_1, a_2, \ldots, a_{m_g}), k)$. We can explicitly list out the multiset containing the monomials $t_j = (a_1, a_2, \ldots, a_i \leftarrow j, \ldots, a_{m_g})$ with multiplicity $t_j(\boldsymbol{x})$, for each $1 \le j \le L_i$. Since $\mathsf{CW}_{g_i}(\boldsymbol{x}) = 0$, this multiset has size multiple of p. Hence, we can canonically divide its elements into groups of size p, counting multiplicities and ϕ_U returns the subset containing (t, k).
- For $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ such that $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$: Note that $g_i^{p-1}(\boldsymbol{x}) = 0$ for all $i \in [m_g]$. Let $v_1 \in V_1$ be the vertex corresponding to the constant monomial 1. ϕ_U groups the edge $(\boldsymbol{x}, v_1, 1)$ (of multiplicity 1) with the $p-1$ edges $(\boldsymbol{x}, \Sigma_{\boldsymbol{x}}, k)$ for $k \in [p-1]$. For any other $t \in V_1 \setminus \{v_1\}$ and an edge $(\boldsymbol{x}, t, k)$, we have that $t = (a_1, \ldots, a_{m_g})$ has $a_i \neq 0$ for some i. We define the multiset containing $t_j = (a_1, \ldots, a_i \leftarrow j, \ldots a_{m_g})$ with multiplicity $t_j(\boldsymbol{x})$ for each $1 \le j < L_i$. Since $g_i^{p-1}(\boldsymbol{x}) = 0$, this multiset has size which is a multiple of p, which we can canonically partition into groups of size p. Thus, ϕ_U on input $(\boldsymbol{x}, t, k)$ returns the group containing (t, k).

▷ Grouping ϕ_V (corresponding to endpoint in V):

- For $t \in V_1$ such that $t \neq \prod_{i=1}^{n} x_i^{p-1}$: there exists a variable x_i with degree less than $p-1$. For $\boldsymbol{x}_j = (x_1, \ldots, x_{i-1}, x_i \leftarrow j, \ldots, x_n)$ with $j \in \mathbb{F}_p$ we define the multiset $\{(\boldsymbol{x}_j, t(\boldsymbol{x}_j))\}_{j \in \mathbb{F}_p}$. Since $\sum_{j=0}^{p-1} t(\boldsymbol{x}_j) = 0$, this multiset has size multiple of p, so we can canonically partition it into groups of size p. Then, $\phi_V(\boldsymbol{x}, t, k)$ returns the group containing $(\boldsymbol{x}, k)$,
- For $v \in V_2$: if $\deg(v) = 0$, then there is no grouping to be done. Else if $\deg(v) \equiv 0 \pmod p$ then $\phi_V(\boldsymbol{x}, t, k)$ returns $\{(\boldsymbol{x}, k)\}_{\boldsymbol{x} \in v}$.

Thus, for any vertex with degree $\equiv 0 \pmod p$, we have provided a grouping function for all its edges. So, for any edge that is not grouped by grouping function at any of its endpoints, then such an endpoint must have degree $\not\equiv 0 \pmod p$ and hence point to a valid solution of the ChevalleyWithSymmetry$_p$ instance. ◀

4.4.2 ChevalleyWithSymmetry$_p$ is PPA$_p$–hard

We show a reduction from Lonely$_p$ to ChevalleyWithSymmetry$_p$. In the instance of ChevalleyWithSymmetry$_p$ that we create, we will ensure that there are no solutions of type 0 (as in Definition 22) and thus, the only valid solutions will be of type 1. In order to do so, we introduce the notions of labeling and proper labeling and prove a generalization of CWT that we call Labeled CWT (Theorem 30).

As we will see, the Labeled CWT, is just a re-formulation of the original CWT rather than a generalization. To understand the Labeled CWT we start with some examples that do not seem to satisfy the Chevalley-Warning condition, but where a solution exists.

Example 1. Consider the case where $p = 3$ and $f(x_1, x_2) = x_2 - x_1^2$. In this case the Chevalley-Warning condition is not met, since we have 2 variables and the total degree is also 2. But, let us consider a slightly different polynomial where we replace the variable x_2 with the product of two variables x_{21}, x_{22} then we get the polynomial $g(x_1, x_{21}, x_{22}) = x_{21} \cdot x_{22} - x_1^2$. Now, g satisfies (CW Condition) and hence, we conclude that the number of roots of g is a multiple of 3. Interestingly, from this fact we can argue that there exists a non-trivial solution for $f(\boldsymbol{x}) = 0$. In particular, the assignment $x_1 = 0$, $x_2 = 0$ corresponds to five assignments of the variables x_1, x_{21}, x_{22}. Hence, since $|\mathcal{V}_g| = 0 \pmod 3$, g has another root, which corresponds to a non-trivial root of f. In this example, we applied the CWT on a slightly different polynomial than f to argue about the existence of non-trivial solutions of f, even though f did not satisfy (CW Condition) itself.

Ignore Some Terms. The Labeled CWT formalizes the phenomenon observed in Example 1 and shows that under certain conditions we can *ignore some terms* when defining the degree of each polynomial. For instance, in Example 1, we can ignore the term x_1^2 when computing the degree of f and treat f as a degree 1 polynomial of 2 variables, in which case the condition of CWT is satisfied.

We describe which terms can be ignored by defining a *labeling* of the terms of each polynomial in the system. The labels take values in $\{-1, 0, +1\}$ and our final goal is to ignore the terms with label $+1$. Of course, it should not be possible to define any labeling that we want; for example we cannot ignore all the terms of a polynomial. Next, we describe the rules of a *proper labeling* that will allow us to prove the Labeled CWT. We start with a definition of a labeling.

▶ **Definition 26** (Monomial Labeling). *Let $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ and let t_{ij} be the j-th monomial of the polynomial $f_i \in \mathbb{F}[\boldsymbol{x}]$ (written in some canonical sorted order). Let $\mathcal{T}$ be the set of all pairs (i,j) such that t_{ij} is a monomial in $\boldsymbol{f}$. A labeling of $\boldsymbol{f}$ is a function $\lambda : \mathcal{T} \to \{-1, 0, +1\}$ and we say that $\lambda(i,j)$ is the label of t_{ij} according to λ.*

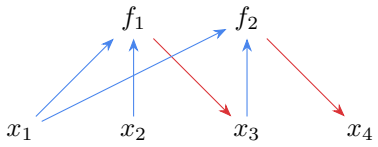
▶ **Definition 27** (Labeled Degree). *For $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ, we define the labeled degree of f_i as, $\deg^\lambda(f_i) \coloneqq \max_{j\,:\,\lambda(i,j)\neq+1} \deg(t_{ij})$, in words, maximum degree among monomials of f_i labeled either 0 or -1.*

Example 1 (continued). According to the lexicographic ordering, $f(x_1, x_2) = -x_1^2 + x_2$ and we have the monomials $t_{11} = -x_1^2$ and $t_{12} = x_2$. Hence, one possible labeling, which as we will see later corresponds to the vanilla Chevalley-Warning Theorem, is $\lambda(1,1) = \lambda(1,2) = 0$. According to this labeling, $\deg^\lambda(f) = 2$. Another possible labeling, that, as we will see, allows us to apply the Labeled CWT , is $\lambda(1,1) = +1$ and $\lambda(1,2) = -1$. In this case, the labeled degree is $\deg^\lambda(f) = 1$.

As we highlighted before, our goal is to prove the Chevalley-Warning Theorem, but with the weaker condition that $\sum_{i=1}^m \deg^\lambda(f_i) < n$ instead of $\sum_{i=1}^m \deg(f_i) < n$. Of course, we first have to restrict the space of all possible labelings by defining *proper labelings*. In order to make the condition of proper labelings easier to interpret we start by defining the notion of a labeling graph.

▶ **Definition 28** (Labeling Graph). *For $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ, we define the labeling graph $G_\lambda = (U \cup V, E)$ as a directed bipartite graph on vertices $U = \{x_1, \ldots, x_n\}$ and $V = \{f_1, \ldots, f_m\}$. The edge $(x_j \to f_i)$ belongs to E if x_j appears in a monomial t_{ir} in f_i with label $+1$, i.e. $\lambda(i,r) = +1$. Symmetrically, the edge $(f_i \to x_j)$ belongs to E if the x_j appears in a monomial t_{ir} in f_i with label -1, i.e. $\lambda(i,r) = -1$.*

Example 2. Let $p = 2$ and $f_1(x_1, x_2, x_3, x_4) = x_1x_2 - x_3$, $f_2(x_1, x_2, x_3, x_4) = x_1x_3 - x_4$. In this system, if we use the lexicographic monomial ordering we have the monomials $t_{11} = x_1x_2$, $t_{12} = -x_3$, $t_{21} = x_1x_3$, $t_{22} = -x_4$. The following figure shows the graph G_λ for the labeling $\lambda(1,1) = +1$, $\lambda(1,2) = -1$, $\lambda(2,1) = +1$ and $\lambda(2,2) = -1$.



▶ **Definition 29** (Proper Labeling). *Let $\boldsymbol{f} \in \mathbb{F}[\boldsymbol{x}]^m$ with a labeling λ. We say that the labeling λ is proper if the following conditions hold.*

(1) *For all i, either $\lambda(i,j) \in \{-1, 1\}$ for all j, or $\lambda(i,j) = 0$ for all j.*
(2) *If two monomials t_{ij}, $t_{ij'}$ contain the same variable x_k, then $\lambda(i,j) = \lambda(i,j')$.*
(3) *If $\lambda(i,j) = -1$, then t_{ij} is multilinear.*
(4) *If x_k is a variable in the monomials t_{ij}, $t_{i'j'}$, with $i \neq i'$ and $\lambda(i,j) = -1$, then $\lambda(i',j') = +1$.*
(5) *If $\lambda(i,j) \neq 0$, then there exists a j' such that $\lambda(i,j') = -1$.*
(6) *The labeling graph G_λ contains no directed cycles.*

We give an equivalent way to understand the definition of a proper labeling.

- ▷ Condition (1) : there is a partition of the polynomial system $\boldsymbol{f}$ into polynomial systems $\boldsymbol{g}$ and $\boldsymbol{h}$ such that all monomials in $\boldsymbol{g}$ are labeled in $\{+1, -1\}$ and all monomials in $\boldsymbol{h}$ are labeled 0.
- ▷ Condition (2) : each polynomial g_i in $\boldsymbol{g}$ can be written as $g_i = g_i^+ + g_i^-$, such that g_i^+ and g_i^- are polynomials on a disjoint set of variables.
- ▷ Condition (3) : Each g_i^- is multilinear.
- ▷ Condition (4) : Any variable x_k can appear in at most one of the g_i^-. Moreover, if an x_k appears in some g_i^-, it does not appear in any h_j in $\boldsymbol{h}$.
- ▷ Condition (5) : Every g_i^- involves at least one variable.
- ▷ Condition (6) : The graph G_λ is essentially between polynomials in $\boldsymbol{g}$ and the variables that appear in them, with an edge $(g_i \to x_k)$ if x_k appears in g_i^- or an edge $(x_k \to g_i)$ if x_k appears in g_i^+.
- ▷ Note that $\deg^\lambda(g_i) = \deg(g_i^-)$, whereas $\deg^\lambda(h_j) = \deg(h_j)$.

It is easy to see that the trivial labeling $\lambda(i, j) = 0$ is always proper. As we will see this special case of the Labeled CWT corresponds to the original CWT. Note that in this case the labeling graph G_λ is an empty graph. Also, given a system of polynomials $\boldsymbol{f}$ and a labeling λ, it is possible to check in polynomial time whether the labeling λ is proper or not.

Example 2 (continued). It is an instructive exercise to verify that the labeling λ specified was indeed a proper labeling of $\boldsymbol{f}$.

▶ **Theorem 30** (Labeled Chevalley-Warning Theorem)**.** *Let $\mathbb{F}_q$ be a finite field with characteristic p and $\boldsymbol{f} \in \mathbb{F}_q[\boldsymbol{x}]^m$. If λ is a proper labeling of $\boldsymbol{f}$ with $\sum_{i=1}^m \deg^\lambda(f_i) < n$, then $|\mathcal{M}_{\boldsymbol{f}}| = 0$. In particular, $|\mathcal{V}_{\boldsymbol{f}}| \equiv 0 \pmod p$.*

Proof. Note that $\mathsf{CW}_{\boldsymbol{f}}(x) = \sum_{S \subseteq [m]} \prod_{i \in S} (-1)^{|S|} f_i^{p-1}$. We'll show that every monomial appearing in the expansion of $\prod_{i \in S} f_i^{p-1}$ will have at least one variable with degree at most $p-1$. For simplicity, we focus on the case $S = [m]$ and the other cases of S follow similarly.

We index a monomial of $\prod_{i \in [m]} f_i^{p-1}$ with a tuple

$$((j_{11}, j_{12}, \ldots, j_{1(p-1)}), \ldots, (j_{m1}, \ldots, j_{m(p-1)}))$$

with $1 \le j_{i\ell} \le L_i$ where L_i is the number of monomials in the explicit representation of f_i. The coordinates $(j_{i1}, \ldots, j_{i(p-1)})$ represent the indices of the monomials chosen from each of the $p-1$ copies of f_i^{p-1}. More succinctly, we have $t = \prod_{i=1}^m \prod_{\ell=1}^{p-1} t_{i,j_{i\ell}}$.

Case 1. $\lambda(i, j_{i\ell}) \in \{0, -1\}$, for all (i, ℓ):
Here, $\deg(t) \le (p-1) \sum_{i=1}^m \deg^\lambda(f_i)$ which, by our assumption, is strictly less than $(p-1)n$. Hence, there is a variable with degree less than $p-1$.

Case 2. There is a unique i with $\lambda(i, j_{i\ell}) = +1$ for some ℓ: (warmup for case 3)
That is, for all $i' \neq i$, $\lambda(i', j_{i'\ell}) \in \{0, -1\}$. By condition (5) of proper labeling there exists a $j' \neq j_{i\ell}$ such that $\lambda(i, j') = -1$. Let x_k be a variable in the monomial $t_{ij'}$. By condition (2), x_k is not present in the monomial $t_{i,j_{i\ell}}$ and by condition (3), its degree in $(t_{i,j_{i,1}}, \ldots, t_{i,j_{i,p-1}})$ is at most $p-2$. Additionally, by condition (4), any monomial of $f_{i'}$ for $i' \neq i$ containing x_k must have label $+1$, but $\lambda(i', j_{i',\ell})$ are all in $\{0, -1\}$. Hence, x_k does not appear in any other monomial of t and its degree on t is equal to its degree in $(t_{i,j_{i,1}} \cdots t_{i,j_{i,p-1}})$, which is strictly less than $p-1$.

Case 3. $I = \{i : \lambda(i, j_{i\ell}) = +1$ for some $\ell\}$:
In the labeling graph G_λ, let $i \in I$ be such that there is no path from f_i to any other $f_{i'}$ for $i' \in I$. Such an i exists due to acyclicity of G_λ, i.e. condition (6). Let ℓ be such that $\lambda(i, j_{i\ell}) = +1$. Again, by condition (5) of proper labeling there exists a $j' \neq j_{i\ell}$ such that

$\lambda(i, j') = -1$. Let x_k be a variable in the monomial $t_{ij'}$. By condition (2), x_k is not present in the monomial $t_{i,j_{i\ell}}$ and by condition (3), its degree in $(t_{i,j_{i,1}}, \ldots, t_{i,j_{i,p-1}})$ is at most $p-2$. Additionally, by condition (4), any monomial of $f_{i'}$ for $i' \neq i$ containing x_k must have label $+1$. For $i' \notin I$, $\lambda(i', j_{i',\ell})$ are all in $\{0, -1\}$. And for $i' \in I$, variable x_k cannot appear with $+1$ label in $f_{i'}$ by our choice of f_i. Hence, x_k does not appear in any other monomial of t and its degree on t is equal to its degree on $(t_{i,j_{i,1}} \cdots t_{i,j_{i,p-1}})$, which is strictly less than $p-1$. ◀

We are now ready to prove the PPA_p-hardness of ChevalleyWithSymmetry$_p$.

▶ **Lemma 31.** *For all primes p, ChevalleyWithSymmetry$_p$ is PPA_p-hard.*

Proof. We prove that Lonely$_p \preceq$ ChevalleyWithSymmetry$_p$. Let us assume (without loss of generality from Lemma 44) that the Lonely$_p$ instance has a single distinguished vertex represented by 0^n. We'll assume that 0^n is isolated, otherwise, no further reduction is necessary.

Pre-processing. We slightly modify the given circuit $\mathcal{C}$ by defining $\mathcal{C}' : \mathbb{F}_p^n \to \mathbb{F}_p^n$ as follows:

$$\mathcal{C}'(v) = \begin{cases} v & \text{, if } \mathcal{C}^p(v) \neq v \\ \mathcal{C}(v) & \text{, otherwise} \end{cases}$$

Since p is a prime, a vertex $v \in \mathbb{F}_p^n$ has $\deg(v) = 1$ if and only if $\mathcal{C}^p(v) = v$ and $\mathcal{C}(v) \neq v$. By modifying the circuit, we changed this condition to just $\mathcal{C}'(v) \neq v$, which facilitates our reduction.

Circuit $\mathcal{C}'$ is composed of the $\mathbb{F}_p$-addition $(+)$, $\mathbb{F}_p$-multiplication $(\times)$ and the constant (1) gates. However, we require the input of ChevalleyWithSymmetry$_p$ to be a zecote polynomial system, and so we further modify the circuit $\mathcal{C}'$ to eliminate all the constant (1) gates, without changing it's behavior – this is possible because we assume $\mathcal{C}'(0^n) = 0^n$.

▷ Claim 32. Given circuit $\mathcal{C}'$ with $(+, \times, 1)$ gates, there exists circuit $\bar{\mathcal{C}}$ with $(+, \times)$ gates such that

$$\bar{\mathcal{C}}(\boldsymbol{v}) = \begin{cases} 0^n & \text{, if } \boldsymbol{v} = 0^n \\ \mathcal{C}'(\boldsymbol{v}) & \text{, otherwise} \end{cases}$$

Proof of Claim 32. We replace all instances of the (1) gate by the function $\mathbb{1}_{\{\boldsymbol{v} \neq 0^n\}}$, which we can compute using only $(+, \times)$ gates as follows: For any $x, y \in \mathbb{F}_p$, observe that $\mathbb{1}_{\{x \neq 0\}} \vee \mathbb{1}_{\{y \neq 0\}} = x^{p-1} + y^{p-1} - x^{p-1}y^{p-1}$. We can thus recursively compute $\bigvee_{i=1}^n \mathbb{1}_{\{v_i \neq 0\}}$ using only $(+, \times)$ gates. Thus, $\bar{\mathcal{C}}(\boldsymbol{v}) = \mathcal{C}'(\boldsymbol{v})$ for all $\boldsymbol{v} \neq 0^n$. And $\bar{\mathcal{C}}(0^n) = 0^n$, since $\bar{\mathcal{C}}$ is computed with only $(+, \times)$ gates. ◁

Thus, we can transform our original circuit $\mathcal{C}$ into a circuit $\bar{\mathcal{C}}$ with just $(+, \times)$ gates. For simplicity, we'll write $\bar{\mathcal{C}}$ as simply $\mathcal{C}$ from now on.

As an intermiate step in the reduction we describe a system of polynomials $\boldsymbol{f}_{\mathcal{C}}$ over $2n + s$ variables $(x_1, \ldots, x_n, z_1, \ldots, z_s, y_1, \ldots, y_n)$, where s is the size of the circuit $\mathcal{C}$. The variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ correspond to the input of $\mathcal{C}$, the variables $\boldsymbol{y} = (y_1, \ldots, y_n)$ correspond to the output and the variables $\boldsymbol{z} = (z_1, \ldots, z_s)$ correspond to the gates of $\mathcal{C}$. For an addition gate $(+)$ we include a polynomial of the form

$$f(a_1, a_2, a_3) = a_2 + a_3 - a_1,$$

where a_1 is the variable corresponding to the output of the $(+)$ gate and a_2, a_3 are the variables corresponding to its two inputs. Similarly for a multiplication $(\times)$ gate, we include a polynomial of the form

$$f(a_1, a_2, a_3) = a_2 \cdot a_3 - a_1$$

Finally, for the output of the circuit, we include the polynomial

$$f(a, y_i) = a - y_i,$$

where a is the variable corresponding to the i-th output gate of $\mathcal{C}$. It holds that

$$\mathcal{C}(\boldsymbol{x}) = \boldsymbol{y} \quad \iff \quad \boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \boldsymbol{0}.$$

We now describe the reduction from $\textsc{Lonely}_p$ to $\textsc{ChevalleyWithSymmetry}_p$. In order to do this, we need to specify a system of polynomials $(\boldsymbol{g}, \boldsymbol{h})$ and a permutation σ such that $|\sigma| = p$. In addition, we will provide a proper labeling λ for $\boldsymbol{g}$ satisfying the degree condition. We will also ensure that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. And hence, the only valid solutions for the resulting $\textsc{ChevalleyWithSymmetry}_p$ instance will be $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \mathcal{V}_{\boldsymbol{h}}$.

Definition of $\boldsymbol{g}$. The polynomial system $\boldsymbol{g}$ contains the following systems of polynomials.

$$\begin{gathered}
\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{1,2}) \\
\boldsymbol{x}_2 - \boldsymbol{x}_3 \\
\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{z}_{3,4}) \\
\boldsymbol{x}_4 - \boldsymbol{x}_5 \\
\vdots \\
\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2p-1}, \boldsymbol{x}_{2p}, \boldsymbol{z}_{2p-1,2p})
\end{gathered}$$

Note that there are $N = (2n + s)p$ variables in total.

Labeling $\boldsymbol{\lambda}$ of $\boldsymbol{g}$. For the polynomials belonging to a system of the form $\boldsymbol{f}_{\mathcal{C}}$, the labeling is equal to -1 for the monomials corresponding to the output of each gate and $+1$, otherwise. For instance, let $a_2 + a_3 - a_1$ be the i-th polynomial of $\boldsymbol{g}$ corresponding to a $(+)$ gate and let $a_1 \prec a_2 \prec a_3$, then $\lambda(i, 1) = -1$ and $\lambda(i, 2) = \lambda(i, 3) = +1$.

For the polynomials belonging to a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, the labeling is equal to -1 for the monomials with variables in $\boldsymbol{x}_{i+1}$ and $+1$ for the monomials with variables in $\boldsymbol{x}_i$.

▷ Claim 33. The labeling λ for $\boldsymbol{g}$ is proper.

Proof of Claim 33. By Definition 29, the labeling λ is proper if the following conditions hold.

Condition 1. *For all i, either $\lambda(i, j) \in \{-1, 1\}$ for all j, or $\lambda(i, j) = 0$ for all j.*

In the labeling λ, there are no labels equal to 0, so this condition holds trivially.

Condition 2. *If two monomials t_{ij}, $t_{ij'}$ contain the same variable x_k, then $\lambda(i, j) = \lambda(i, j')$.*

By construction of $\boldsymbol{g}$, no variable appears twice in the same polynomial with a different labeling. For polynomials of $\boldsymbol{f}_{\mathcal{C}}$, this holds because the output variable of a gate is not simultaneously an input variable and all input variables have the same labeling. For polynomials in a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, each polynomial contains two different variables.

Condition 3. *If $\lambda(i,j) = -1$, then t_{ij} is multilinear.*

For polynomials of $\boldsymbol{f}_{\mathcal{C}}$, only the output variable of a gate has label -1 and by definition this monomial is linear. For polynomials in a system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$, all monomials are linear, so the condition holds trivially.

Condition 4. *If x_k is a variable in the monomials t_{ij}, $t_{i'j'}$, with $i \neq i'$ and $\lambda(i,j) = -1$, then $\lambda(i',j') = +1$.*

Observe that all monomials with label -1 contain only a single variable, so we refer to a monomial x_k with label -1. For a polynomial in $\boldsymbol{f}_{\mathcal{C}}$, a monomial x_k with label -1 corresponds to the output of a gate. Hence, if x_k appears in other monomials of $\boldsymbol{f}_{\mathcal{C}}$, these monomials correspond to inputs and have label $+1$. Also, if x_k is an output variable of $\boldsymbol{f}_{\mathcal{C}}$, then it might appear in a polynomial of the form $a_1 - a_2$. However, by construction the monomials of $x_i - x_{i+1}$ that correspond to output variables of $\boldsymbol{f}_{\mathcal{C}}$ have label $+1$.

Condition 5. *If $\lambda(i,j) \neq 0$, then there exists a j' such that $\lambda(i,j') = -1$.*

By the definition of λ, all polynomials of $\boldsymbol{g}$ have a monomial with label -1. These are the monomials that correspond to the outputs of a gate for the systems of the form $\boldsymbol{f}_{\mathcal{C}}$ and the monomials that correspond to $\boldsymbol{x}_{i+1}$ for the systems of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$.

Condition 6. *The labeling graph G_λ contains no cycles.*

Each system of the form $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$ has incoming edges with variables appearing only in the i-th copy of $\boldsymbol{f}_{\mathcal{C}}$ and outgoing edges with variables appearing only in the $(i+1)$-th copy of $\boldsymbol{f}_{\mathcal{C}}$. Also, the variables appearing on the i-th copy of $\boldsymbol{f}_{\mathcal{C}}$ might appear only in the systems $\boldsymbol{x}_{i-1} - \boldsymbol{x}_i$ and $\boldsymbol{x}_i - \boldsymbol{x}_{i+1}$. Hence, G_λ has no cycles that contain vertices of two different copies of $\boldsymbol{f}_{\mathcal{C}}$ or of a copy of $\boldsymbol{f}_{\mathcal{C}}$ and a system of the form $\boldsymbol{x}_{i-1} - \boldsymbol{x}_i$.

It is left to argue that the labeling graph restricted to a copy of $\boldsymbol{f}_{\mathcal{C}}$ does not have any cycles. Let the vertices of $\boldsymbol{f}_{\mathcal{C}}$ be ordered according to the topological ordering of $\mathcal{C}$. This restricted part of G_λ corresponds exactly to the graph of $\mathcal{C}$, which by definition is a DAG. Hence, G_λ contains no cycles. ◁

We also need to show that for this labeling $\boldsymbol{g}$ satisfies the labeled Chevalley condition.

▷ Claim 34. The labeled Chevalley condition $\sum_{i=1}^{m_g} \deg^\lambda(g_i) < N$ holds for $\boldsymbol{g}$ with labeling λ.

Proof. Each polynomial of $\boldsymbol{g}$ has a unique monomial with $\lambda(i,j) = -1$ and this monomial has degree 1. Thus, $\sum_{i=1}^{m_g} \deg^\lambda(g_i) = m_g$. On the other hand, the i-th polynomial of $\boldsymbol{g}$ has exactly one variable that has not appeared in any of the previous polynomials. More specifically, the number of variables is equal to $m_g + n$, where n is the size of the input of $\mathcal{C}$. Hence, the labeled Chevalley condition holds for $\boldsymbol{g}$. ◁

Definition of $\boldsymbol{h}$. The system of polynomials $\boldsymbol{g}$ allows us to compute the p vertices given by $\mathcal{C}^i(\boldsymbol{x})$ for $i \in [p+1]$. From the definition of $\textsc{Lonely}_p$ and our pre-processing on $\mathcal{C}$, this group of p vertices is a hyperedge if and only if $\mathcal{C}(\boldsymbol{x}) \neq \boldsymbol{x}$. Since solutions of $\textsc{Lonely}_p$ are lonely vertices, we define $\boldsymbol{h}$ to exclude $\boldsymbol{x}$ such that $\mathcal{C}(\boldsymbol{x}) \neq \boldsymbol{x}$. Namely, we set $\boldsymbol{h}$ to be the system of polynomials

$$\boldsymbol{x}_1 - \boldsymbol{x}_2.$$

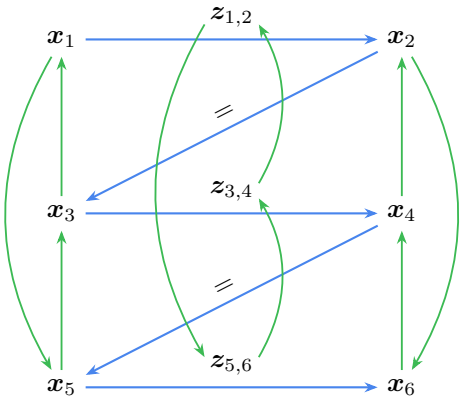
Definition of permutation $\boldsymbol{\sigma}$. In the description of $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$, we have used the following vector of variables:

$$\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{2p}, \boldsymbol{z}_{1,2}, \boldsymbol{z}_{3,4}, \ldots, \boldsymbol{z}_{2p-1,2p})$$

We define the permutation σ such that

$$\sigma(\boldsymbol{x}) = (\boldsymbol{x}_3, \boldsymbol{x}_4, \ldots, \boldsymbol{x}_{2p}, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{3,4}, \boldsymbol{z}_{5,6}, \ldots, \boldsymbol{z}_{2p-1,2p}, \boldsymbol{z}_{1,2}) \ ,$$

as illustrated in the following figure. The blue arrows indicate the polynomials $\boldsymbol{g}$ and the green arrows indicate the permutation σ in the case of $p = 3$.



▷ Claim 35. The group $\langle\sigma\rangle$ has order p and acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$.

Proof. In order to see that $|\sigma| = p$, note that the input of σ consists of $3p$ blocks of variables. The permutation σ performs a rotation of the first $2p$ blocks by two positions and of the last p blocks by one position.

All that remains is to show that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. First, we show that $\langle\sigma\rangle$ defines a group action on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, that is for all $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, it holds that $\sigma(\boldsymbol{x}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{2p-1}, \boldsymbol{x}_{2p}, \boldsymbol{z}_{1,2}, \boldsymbol{z}_{3,4}, \ldots, \boldsymbol{z}_{2p-1,2p}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, then

- $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$ implies that $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2i-1}, \boldsymbol{x}_{2i}, \boldsymbol{z}_{2i-1,2i}) = 0$ for $i \in [p]$ and $\boldsymbol{x}_{2i} = \boldsymbol{x}_{2i+1}$ for $i \in [p-1]$
- $\boldsymbol{x} \in \overline{\mathcal{V}_{\boldsymbol{h}}}$ implies that $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, that is, $\mathcal{C}(\boldsymbol{x}_1) \neq \boldsymbol{x}_1$ since $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{1,2}) = 0 \Leftrightarrow \boldsymbol{x}_2 = \mathcal{C}(\boldsymbol{x}_1)$.

Now, $\sigma(\boldsymbol{x}) = (\boldsymbol{x}_3, \boldsymbol{x}_4, \ldots, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{z}_{3,4}, \boldsymbol{z}_{5,6}, \ldots, \boldsymbol{z}_{1,2}) \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ holds because

- $\boldsymbol{f}_{\mathcal{C}}(\boldsymbol{x}_{2i-1}, \boldsymbol{x}_{2i}, \boldsymbol{z}_{2i-1,2i}) = 0$ for $i \in [p]$ and $\boldsymbol{x}_{2i} = \boldsymbol{x}_{2i+1}$ for $i \in [p-1]$, which holds from $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$. Additionally, $\boldsymbol{x}_1 = \boldsymbol{x}_{2p}$ holds because we pre-processed $\mathcal{C}$ such that $\mathcal{C}^p(\boldsymbol{x}_1) = \boldsymbol{x}_1$,
- $\boldsymbol{x}_3 \neq \boldsymbol{x}_4$, which holds because $\boldsymbol{x}_4 = \mathcal{C}(\boldsymbol{x}_3)$ for $i \in [p]$ and from the definition of $\mathcal{C}$, $\mathcal{C}(\boldsymbol{x}_1) \neq \boldsymbol{x}_1$ implies that $\boldsymbol{x}_{2i} \neq \boldsymbol{x}_{2i-1}$ for all $i \in [p]$.

Finally, if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$, by construction of $\mathcal{C}$, we have that $\boldsymbol{x}_{2k} \neq \boldsymbol{x}_{2j}$ for $k \neq j$ and thus $\sigma(\boldsymbol{x}) \neq \boldsymbol{x}$ simply because $\boldsymbol{x}_3 \neq \boldsymbol{x}_1$. Thus, we conclude that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. ◁

Putting it all together. The solution of this instance of ChevalleyWithSymmetry$_p$ cannot be a vector $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ with $\sigma(\boldsymbol{x}) \notin \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$ or $\sigma(\boldsymbol{x}) = \boldsymbol{x}$, since we know from Claim 35 that $\langle\sigma\rangle$ acts freely on $\mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}_{\boldsymbol{h}}}$. We also have from Theorem 30 that the solution also cannot be a max-degree monomial in the expansion of $\mathsf{CW}_{\boldsymbol{g}}(\boldsymbol{x}) = \prod(1 - g_i^{p-1})$. Thus, the solution must be an $\boldsymbol{x} \neq \boldsymbol{0}$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. Let $\boldsymbol{x}_1$ denote the first n coordinates of $\boldsymbol{x}$, then $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ implies that $\boldsymbol{x}_1 = \mathcal{C}(\boldsymbol{x}_1)$ and $\boldsymbol{x} \neq \boldsymbol{0}$ implies that $\boldsymbol{x}_1 \neq \boldsymbol{0}$. Hence, $\boldsymbol{x}_1$ corresponds to a lonely vertex of the Lonely$_p$ instance. ◀

5 Complete Problems via Small Depth Arithmetic Circuits

We now illustrate the significance of the PPA_p-completeness of ChevalleyWithSymmetry$_p$, by showing that we can reformulate any of the proposed definitions of PPA_p, by restricting the circuit in the input to be just constant depth arithmetic formulas with gates $\times \pmod p$ and $+ \pmod p$ (we call this class $\mathsf{AC}^0_{\mathbb{F}_p}$[7]). This result is analogous to the NP-completeness of SAT which basically shows that CircuitSAT remains NP-complete even if we restrict the input circuit to be a (CNF) formula of depth 2.

We define SuccinctBipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ to be the same as SuccinctBipartite$_p$ but with the input circuit being a formula in $\mathsf{AC}^0_{\mathbb{F}_p}$. Similarly, we define Lonely$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, Leaf$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, etc.

▶ **Theorem 36.** *For all primes p, SuccinctBipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ is PPA_p-complete.*

▶ Remark 37. In [35], a similar simplification theorem was shown for PPAD. In fact, this simplification involves only the End-of-Line problem and does not go through a natural complete problem for PPAD (see Theorem 1.5 in [35]). A similar result can be shown for other TFNP subclasses, including PPA. However, it is unclear if these techniques also apply to PPA_p classes.

Theorem 36 follows directly from the proof of Lemma 25 by observing that the reduction can be perfomed by an $\mathsf{AC}^0_{\mathbb{F}_p}$ circuit. For completeness, we include this proof in Appendix B.

Since the reductions between SuccinctBipartite$_p$ and other problems studied in this work (refer to Appendix A) can also be implemented as AC^0 circuits, we get the following corollary.

▶ **Corollary 38.** *For all primes p, Lonely$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$, Leaf$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ and Bipartite$_p[\mathsf{AC}^0_{\mathbb{F}_p}]$ are all PPA_p-complete.*

Since $+ \pmod p$ and $\times \pmod p$ can be simulated in NC^1, we also get the following corollary.

▶ **Corollary 39.** *For all primes p, Lonely$_p[\mathsf{NC}^1]$, Leaf$_p[\mathsf{NC}^1]$ and Bipartite$_p[\mathsf{NC}^1]$ are all PPA_p-complete.*

Thus, Theorem 36 allows us to consider reductions from these PPA_p-complete problems with instances encoded by a shallow formulas rather than an arbitrary circuit. We believe this could be a useful starting point for finding other PPA_p-complete problems.

6 Applications of Chevalley-Warning

For most of the combinatorial applications mentioned in Subsection 1.4, the proofs utilize restricted versions of the Chevalley-Warning Theorem that are related to finding binary or short solutions in a system of modular equations. We define two computational problems to capture these restricted cases. The first problem is about finding binary non-trivial solutions in a modular linear system of equations, which we call BIS_q. The second is a special case of the well-known short integer solution problem in ℓ_∞ norm, which we denote by SIS_q. The computational problems are defined below, where $N(q)$ denotes the sum of the exponents in the canonical prime factorization of q, e.g. $N(4) = N(6) = 2$. In particular, $N(p) = 1$ for prime p and $N(q_1 q_2) = N(q_1) + N(q_2)$ for all q_1, q_2.

[7] Note that $\mathsf{AC}^0_{\mathbb{F}_p}$ is strictly more powerful than AC^0 since the Boolean operations of $\{\wedge, \vee, \neg\}$ can be implemented in $\mathsf{AC}^0_{\mathbb{F}_p}$, but $+ \pmod p$ cannot be implemented in AC^0.

▶ **Definition 40** (BIS_q).
Input: $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$, *a matrix over* $\mathbb{Z}$
Condition: $n \geq (m+1)^{N(q)}(q-1)$
Output: $\boldsymbol{x} \in \{0,1\}^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod q$

▶ **Definition 41** (SIS_q).
Input: $\boldsymbol{A} \in \mathbb{Z}_q^{m\times n}$, *a matrix over* $\mathbb{Z}$
Condition: $n \geq ((m+1)/2)^{N(q)}(q-1)$
Output: $\boldsymbol{x} \in \{-1,0,1\}^n$ *such that* $\boldsymbol{x} \neq \boldsymbol{0}$ *and* $\boldsymbol{A}\boldsymbol{x} \equiv \boldsymbol{0} \pmod q$

SIS_q is a special case of the well-known short integer solution problem in ℓ_∞ norm from the theory of lattices. The totality of this problem is guaranteed even when $n > m\log_2 q$ by pigeonhole principle; thus, SIS_q belongs also to PPP (for this regime of parameters). However, for the parameters considered in above definitions, the existence of a solution in the BIS_q and SIS_q is guaranteed through modulo q arguments, which we formally show in the following theorem.

▶ **Theorem 42.** *For the regime of parameters n, m as in Definitions 40 and 41,*
1. *For all primes p : BIS_p, $\mathrm{SIS}_p \preceq \textsc{Chevalley}_p$.*
2. *For all q : BIS_q, $\mathrm{SIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$,*
3. *For all k : $\mathrm{BIS}_{2^k} \in \mathsf{FP}$,*
4. *For all k, ℓ : $\mathrm{SIS}_{2^k 3^\ell} \in \mathsf{FP}$.*

Proof. **Part 1.** For all primes p, BIS_p, $\mathrm{SIS}_p \preceq \textsc{Chevalley}_p$.
Given an BIS_p instance $\boldsymbol{A} = (a_{ij})$, we define a zecote polynomial system as follows

$$\boldsymbol{f} := \left\{ f_i(\boldsymbol{x}) = \sum_{j=1}^{n} a_{ij} x_j^{p-1} \ : \ i \in [m] \right\}$$

Clearly, $\deg(f_i) = p-1$, so $\sum_{i=1}^m \deg(f_i) = m(p-1)$. Since $n \geq (m+1)(p-1) > m(p-1)$, (CW Condition) is satisfied. Hence the output of $\textsc{Chevalley}_p$ is a solution $\boldsymbol{x} \neq \boldsymbol{0}$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$. This gives us that $\boldsymbol{x}^{p-1} := (x_1^{p-1}, \ldots, x_n^{p-1})$ is binary and satisfies $A\boldsymbol{x} \equiv 0 \pmod p$.

The reduction $\mathrm{SIS}_p \preceq \textsc{Chevalley}_p$ also follows in a similar fashion. Namely, we define $f_i(\boldsymbol{x}) := \sum_{j=1}^m a_{ij} x_j^{(p-1)/2}$. This satisfies the (CW Condition) because $\sum_i \deg(f_i) = m(p-1)/2 < ((m+1)/2)(p-1) \leq n$. This ensures that any $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}}$ satisfies $\boldsymbol{x}^{(p-1)/2} \in \{-1,0,1\}^n$ and $A\boldsymbol{x} \equiv 0 \pmod p$.

Part 2. For all q : BIS_q, $\mathrm{SIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$.

We show that $\mathrm{BIS}_{q_1 q_2} \preceq \mathrm{BIS}_{q_1}$ & BIS_{q_2}. Hence if $\mathrm{BIS}_{q_1} \in \mathsf{FP}^{\mathsf{PPA}_{q_1}}$ and $\mathrm{BIS}_{q_2} \in \mathsf{FP}^{\mathsf{PPA}_{q_2}}$, then $\mathrm{BIS}_{q_1 q_2} \in \mathsf{FP}^{\mathsf{PPA}_{q_1 q_2}}$. The proof of Part 2 now follows by induction.

Given a $\mathrm{BIS}_{q_1 q_2}$ instance $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$, we divide $\boldsymbol{A}$ along the columns into $n_1 = (m+1)^{N(q_1)}(q_1-1)$ submatrices denoted by $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{n_1}$, each of size at least $m \times n_2$, with $n_2 = \lfloor n/n_1 \rfloor$ (if n/n_1 is not an integer, then we let $\boldsymbol{A}_{n_1}$ has more than n_2 columns). Each $\boldsymbol{A}_i$ is an instance of BIS_{q_2}, since

$$n_2 = \lfloor n/n_1 \rfloor \geq (m+1)^{N(q_2)} \lfloor (q-1)/(q_1-1) \rfloor \geq (m+1)^{N(q_2)}(q_2-1).$$

Let $\boldsymbol{y}_i \in \{0,1\}^{n_2}$ be any solution to $\boldsymbol{A}_i \boldsymbol{y}_i \equiv 0 \pmod{q_2}$. We define the matrix $\boldsymbol{B} \in \mathbb{Z}^{m\times n_1}$ where the i-th column is equal to $\boldsymbol{A}_i \boldsymbol{y}_i / q_2$; this has integer entries since $\boldsymbol{A}_i \boldsymbol{y}_i \equiv 0 \pmod{q_2}$. Now, by our choice of n_1, we have that $\boldsymbol{B}$ is an instance of BIS_{q_1}. Let $\boldsymbol{z} = (z_1, \ldots, z_{n_1}) \in \{0,1\}^{n_1}$ be any solution to $\boldsymbol{B}\boldsymbol{z} = 0 \pmod{q_1}$.

Finally, we define $\boldsymbol{x} := (z_1\boldsymbol{y}_1, \ldots, z_{n_1}\boldsymbol{y}_{n_1}) \in \{0,1\}^n$. Observe that since $\boldsymbol{y}_i$ and $\boldsymbol{z}$ are binary, $\boldsymbol{x}$ is also binary. Additionally,

$$\boldsymbol{A}\boldsymbol{x} \;=\; \sum_{i=1}^{n_1}(\boldsymbol{A}_i\boldsymbol{y}_i)z_i \;=\; q_2\sum_{i=1}^{n_1}\frac{\boldsymbol{A}_i\boldsymbol{y}_i}{q_2}z_i \;=\; q_2\boldsymbol{B}\boldsymbol{y} \;\equiv\; \boldsymbol{0} \pmod{q_1q_2}\,.$$

Hence, $\boldsymbol{x}$ is a solution of the original $\mathrm{BIS}_{q_1q_2}$ instance $\boldsymbol{A}\boldsymbol{x} \equiv 0 \pmod{q_1q_2}$. This concludes the proof of $\mathrm{BIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$. The proof of $\mathrm{SIS}_q \in \mathsf{FP}^{\mathsf{PPA}_q}$ follows similarly, by observing that if $\boldsymbol{y}_i$ and $\boldsymbol{z}$ have entries in $\{-1,0,1\}$ then so does $\boldsymbol{x}$.

Parts 3, 4. For all k, ℓ : $\mathrm{BIS}_{2^k} \in \mathsf{FP}$ and $\mathrm{SIS}_{2^k3^\ell} \in \mathsf{FP}$.

Observe that BIS_2 (hence also SIS_2) and SIS_3 are solvable in polynomial time via Gaussian elimination. Combining this with the reduction $\mathrm{BIS}_{q_1q_2} \preceq \mathrm{BIS}_{q_1}$ & BIS_{q_2} completes the proof (similarly for SIS). ◀

Note that for a prime p and any k, we have from Theorem 1, that $\mathsf{PPA}_{p^k} = \mathsf{PPA}_p$. Additionally, Theorem 5 shows that PPA_p is closed under Turing reductions, so we have the following corollary.

▶ **Corollary 43.** *For all primes p and all k : BIS_{p^k}, $\mathrm{SIS}_{p^k} \in \mathsf{PPA}_p$.*

Even though the SIS_q problem is well-studied in lattice theory, not many results are known in the regime we consider where q is a constant. Our results show that solving $\textsc{Chevalley}_p$ is at least as hard as finding short integer solutions in p-ary lattices for a specific range of parameters. More specifically, our reduction assumes that q is a constant and, thus, it does not depend on the input lattice, and that the dimension n of lattice is related to the number of constraints in the dual as $n > ((m+1)/2)^{N(q)}(q-1)$. On the other hand, we showed (in Parts 3, 4) that there are q-ary lattice for which finding short integer solutions is easy.

7 Structural Properties of PPA_q

In this section, we prove the structural properties of PPA_q outlined in Subsection 1.5.

Relation to PMOD_q

Buss and Johnson [13, 27] defined a problem $\textsc{Mod}_q$, which is almost identical to $\textsc{Lonely}_q$, with the only difference being that the q-dimensional matching is over a power-of-2 many vertices encoded by $C : \{0,1\}^n \to \{0,1\}^n$, with no designated vertices, except when q is a power of 2 in which case we have one designated vertex. The class PMOD_q is then defined as the class of total search problems reducible to $\textsc{Mod}_q$. The restriction of number of vertices to be a power of 2, which arises as an artifact of the binary encoding of circuit inputs, makes the class PMOD_q slightly weaker than PPA_q.

To compare PPA_q and PMOD_q, we define a restricted version of $\textsc{Lonely}_q$, where the number of designated vertices is exactly k; call this problem $\textsc{Lonely}_q^k$. Clearly, $\textsc{Lonely}_q^k$ reduces to $\textsc{Lonely}_q$. We show that a converse holds, but only for prime p; see Subsection A.2 for proof.

▶ **Lemma 44.** *For all primes p and $k \in \{1, \ldots, p-1\}$, $\textsc{Lonely}_p$ reduces to $\textsc{Lonely}_p^k$.*

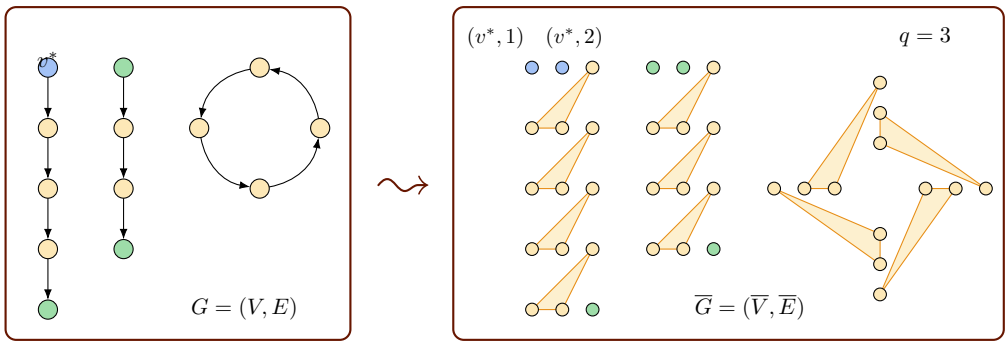
▶ **Corollary 45.** *For all primes p, $\mathsf{PPA}_p = \mathsf{PMOD}_p$.*

For composite q, however, the two classes are conceivably different. In contrast to Theorem 1, it is shown in [27] that $\mathsf{PMOD}_q = \mathsf{⅋}_{p|q}\,\mathsf{PMOD}_p$, where the operator "⅋" is defined as follows: For any two search problem classes M_0, M_1 with complete problems S_0, S_1, the class $\mathsf{M}_0 \mathbin{⅋} \mathsf{M}_1$ is defined via the complete problem $S_0 \mathbin{⅋} S_1$ defined as follows: Given $(x_0, x_1) \in \Sigma^* \times \Sigma^*$, find a solution to either x_0 interpreted as an instance of S_0 or to x_1 interpreted as an instance of S_1. In other words, $\mathsf{M}_1 \mathbin{⅋} \mathsf{M}_2$ is no more powerful than either M_1 or M_2. In particular, it holds that $\mathsf{M}_1 \mathbin{⅋} \mathsf{M}_2 = \mathsf{M}_1 \cap \mathsf{M}_2$, whereas $\mathsf{M}_1 \mathbin{\&} \mathsf{M}_2 \supseteq \mathsf{M}_1 \cup \mathsf{M}_2$. Because of this distinction, unlike Theorem 1, the proof of $\mathsf{PMOD}_{p^k} = \mathsf{PMOD}_p$ in [27] follows much more easily since for any odd prime p it holds that $2^n \not\equiv 0 \pmod p$ and hence a $\textsc{Lonely}_{p^k}$ instance readily reduces to a $\textsc{Lonely}_p$ instance.

7.1 PPAD $\subseteq$ PPA$_q$

Johnson [27] already showed that $\mathsf{PPAD} \subseteq \mathsf{PMOD}_q$ which implies that $\mathsf{PPAD} \subseteq \mathsf{PPA}_q$. We present a simplified version of that proof.

We reduce the PPAD-complete problem End-of-Line to $\textsc{Lonely}_q$. An instance of End-of-Line is a circuit C that implicitly encodes a directed graph $G = (V, E)$, with in-degree and out-degree at most 1 and a designated vertex v^* with in-degree 0 and out-degree 1.



We construct a q-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertices $\overline{V} = V \times [q]$, such that for every edge $(u \to v) \in E$, we include the hyperedge $\{(u, q), (v, 1), \ldots, (v, q-1)\}$ in $\overline{E}$. The designated vertices are $\overline{V}^* = \{(v^*, 1), \ldots, (v^*, q-1)\}$. Note that $|\overline{V}| \equiv 0 \pmod q$ and $|\overline{V}^*| = q - 1 \not\equiv 0 \pmod q$. It is easy to see that a vertex (v, i) is isolated in $\overline{G}$ if and only if v is a source or a sink in G. This completes the reduction, since $\overline{V}$ is efficiently representable and indexable and the neighbors of any vertex in $\overline{V}$ are locally computable using black-box access to C (see Remark 10).

7.2 Oracle separations

Here we explain how PPA_q can be separated from other TFNP classes relative to oracles, as summarized in Figure 1. That is, for distinct primes p, p', there exist oracles $O_1, \ldots, O_5$ such that

(1) $\mathsf{PLS}^{O_1} \not\subseteq \mathsf{PPA}_p^{O_1}$ (2) $\mathsf{PPA}_p^{O_2} \not\subseteq \mathsf{PPP}^{O_2}$ (3) $\mathsf{PPA}_{p'}^{O_3} \not\subseteq \mathsf{PPA}_p^{O_3}$

(4) $\mathsf{PPADS}^{O_4} \not\subseteq \mathsf{PPA}_p^{O_4}$ (5) $\bigcap_p \mathsf{PPA}_p^{O_5} \not\subseteq \mathsf{PPAD}^{O_5}$

The usual technique for proving such oracle separations is propositional proof complexity (together with standard diagonalization arguments) [5, 10, 13]. The main insight is that if a problem S_1 reduces to another problem S_2 in a black-box manner, then there are "efficient proofs" of the totality of S_1 starting from the totality of S_2. The discussion below assumes some familiarity with these techniques.

$\mathbf{PLS}^{O_1} \nsubseteq \mathbf{PPA}_p^{O_1}$, $\mathbf{PPA}_p^{O_2} \nsubseteq \mathbf{PPP}^{O_2}$, $\mathbf{PPA}_{p'}^{O_3} \nsubseteq \mathbf{PPA}_p^{O_3}$

Johnson [27] showed all the above separations with respect to PMOD_p. Since we showed $\mathsf{PPA}_p = \mathsf{PMOD}_p$ (Corollary 45), the same oracle separations hold for PPA_p.

$\mathbf{PPADS}^{O_4} \nsubseteq \mathbf{PPA}_p^{O_4}$

Göös et al. [23, §4.3] building on [6] showed that the contradiction underlying the PPADS-complete search problem Sink-of-Line requires $\mathbb{F}_p$-Nullstellensatz refutations of high degree. This yields the oracle separation.

$\bigcap_p \mathbf{PPA}_p^{O_5} \nsubseteq \mathbf{PPAD}^{O_5}$

For a fixed $k \geq 1$, consider the problem $S_k := \bigtriangledown_{i\in[k]} \textsc{Lonely}_{p_i}$ where p_i are the primes. Buss et al. [12] showed that the principle underlying S_i is incomparable with the principle underlying $\textsc{Lonely}_{p_{i+1}}$. This translates into an relativized separation $\bigcap_{i\in[k]} \mathsf{PPA}_{p_i} \nsubseteq \mathsf{PPA}_{p_{i+1}}$ which in particular implies $\bigcap_{i\in[k]} \mathsf{PPA}_{p_i} \nsubseteq \mathsf{PPAD}$. Finally, one can consider the problem $S := S_{k(n)}$ where $k(n)$ is a slowly growing function of the input size n. This problem is in $\bigcap_p \mathsf{PPA}_p$ since for each fixed p and for large enough input size, S reduces to the PPA_p-complete problem. On the other hand, the result of Buss et al. [12] is robust enough to handle a slowly growing $k(n)$; we omit the details.

7.3 Closure under Turing reductions

Theorem 5 says that for any prime p, the class PPA_p is closed under Turing reductions. In contrast, Buss and Johnson showed that PPA_{p_1} & PPA_{p_2}, for distinct primes p_1 and p_2, is not closed under *black-box* Turing reductions [13, 27]. In particular, they define the '$\otimes$' operator as follows. For two total search problems S_1 and S_2, the problem $S_1 \otimes S_2$ is defined as: Given $(x_0, x_1) \in \Sigma^* \times \Sigma^*$, find a solution to both x_0 (instance of S_0) and to x_1 (instance of S_1). Clearly the problem $\textsc{Lonely}_{p_1} \otimes \textsc{Lonely}_{p_2}$ can be solved with two queries to the oracle PPA_{p_1} & PPA_{p_2}. However, Buss and Johnson [13, 27] show that $\textsc{Lonely}_{p_1} \otimes \textsc{Lonely}_{p_2}$ cannot be solved with one oracle query to PPA_{p_1} & PPA_{p_2} under *black-box* reductions. In particular, this implies that PPA_q is not closed under *black-box* Turing reductions, when q is not a prime power. We now prove Theorem 5, which is equivalent to the following.

▶ **Theorem 46.** *For any prime p and total search problem S, if $S \preceq_T \textsc{Lonely}_p$, then $S \preceq_m \textsc{Lonely}_p$.*

Proof. The key reason why this theorem holds for prime p is Lemma 44: In a $\textsc{Lonely}_p$ instance, we can assume w.l.o.g. that there are exactly $p-1$ distinguished vertices.

On instance x of the problem S, suppose the oracle algorithm sequentially makes at most $t = \text{poly}(|x|)$ queries to $\textsc{Lonely}_p$ oracle. The i-th query consists of a tuple (C_i, V_i^*) where C_i encodes a p-dimensional matching graph $G_i = (V_i, E_i)$ and $V_i^* \subseteq V_i$ is the set of $p-1$ designated vertices, and let $y_i \in V_i$ be the solution returned by the $\textsc{Lonely}_p$ oracle. The query (C_i, V_i^*) is computable in polynomial time, given x and valid solutions to all previous queries. Finally, after receiving all answers the algorithm returns $L(x, y_1, \ldots, y_t)$ that is a valid solution for x in S.

We make the following simplifying assumptions.

- Each hypergraph G_i is on p^n vertices, where $n = \text{poly}(|x|)$ (thanks to instance extension property – see Remark 10).
- For any query the vertices V_i^* are always isolated in G_i (if some vertex in V_i^* were to not be isolated, the algorithm could be modified to simply not make the query).
- Exactly t queries are made irrespective of the oracle answers.

We reduce x to a single instance of $\textsc{Lonely}_p$ as follows.

Vertices. The vertices of the $\textsc{Lonely}_p$ instance will be $V = [p]^n \cup [p]^{2n} \cup \cdots \cup [p]^{tn}$, which we interpret as $\overline{V} = V_1 \cup (V_1 \times V_2) \cup (V_1 \times V_2 \times V_3) \cup \cdots \cup (V_1 \times \cdots \times V_t)$. The designated vertices will be $\overline{V}^* := V_1^*$. Note that $|\overline{V}^*| = |V_1^*| \not\equiv 0 \pmod{p}$.

Edges. We'll define the hyperedge for vertex $\overline{v} = (v_1, \ldots, v_k)$ for any $k \leq t$. Let $j \leq k$ be the last coordinate such that for all $i < j$, the vertex v_i is a valid solution for the $\textsc{Lonely}_p$ instance (C_i, V_i^*), which the algorithm creates on receiving $v_1, \ldots, v_{i-1}$ as answers to previous queries.

Case $j < k$: Let $u_1, \ldots, u_{p-1}$ be the neighbors of v_k in a canonical trivial matching over $[p]^n$; e.g. $\{[p] \times w : w \in [p]^{n-1}\}$. The neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i$.

Case $j = k$: We consider three cases, depending on whether v_k is designated, non-isolated or isolated in the $\textsc{Lonely}_p$ instance (C_k, V_k^*).

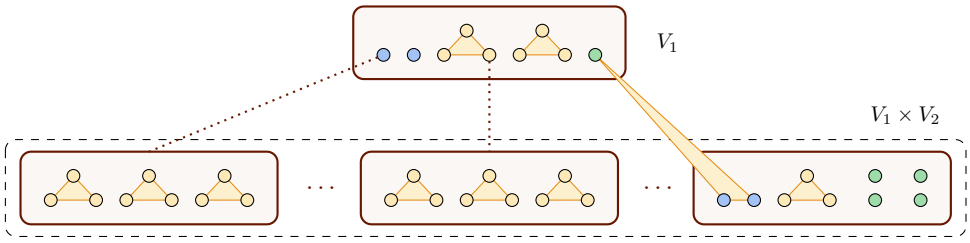
Non-isolated v_k: For $u_1, \ldots, u_{p-1}$ being the neighbors of v_k in G_k, the neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i$.

Isolated v_k: Such a v_k is a valid solution for (C_k, V_k^*).

If $k < t$: the algorithm will have a next oracle query (C_{k+1}, V_{k+1}^*). In this case, for $u_1, \ldots, u_{p-1}$ being the designated vertices in V_{k+1}^*, the neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, v_k, u_i)\}_i$.

If $k = t$: there are no more queries, and we leave $\overline{v}$ isolated.

Designated v_k: Let $u_1, \ldots, u_{p-2}$ be the other designated vertices in V_k^*. The neighbors of $\overline{v}$ are $\{(v_1, \ldots, v_{k-1}, u_i)\}_i \cup \{(v_1, \ldots, v_{k-1})\}$.



It is easy to see that our definition of edges are consistent and the only vertices which are isolated (apart from those in $\overline{V}^*$) are of the type $(y_1, \ldots, y_t)$ where each y_i is a valid solution for the $\textsc{Lonely}_p$ instance (C_i, V_i^*). Thus, given an isolated vertex $\overline{y}$, we can immediately infer a solution for x as $L(x, y_1, \ldots, y_t)$. This completes the reduction since $\overline{V}$ is efficiently representable and indexable – see Remark 10. ◀

References

1 James Aisenberg, Maria Luisa Bonet, and Sam Buss. 2-d tucker is PPA complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:163, 2015. URL: `http://eccc.hpi-web.de/report/2015/163`.

2 Noga Alon. Splitting necklaces. *Advances in Mathematics*, 63(3):247–253, 1987. `doi:10.1016/0001-8708(87)90055-7`.

3 Noga Alon, Shmuel Friedland, and Gil Kalai. Regular subgraphs of almost regular graphs. *Journal of Combinatorial Theory, Series B*, 37(1):79–91, 1984.

4 Noga Alon and Douglas B. West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986. doi:10.1090/S0002-9939-1986-0861764-9.

5 Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

6 Paul Beame and Søren Riis. More on the relative strength of counting principles. In *Proceedings of the DIMACS Workshop on Proof Complexity and Feasible Arithmetics*, volume 39, pages 13–35, 1998.

7 Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theor. Comput. Sci.*, 103(1):3–23, 1992. doi:10.1016/0304-3975(92)90084-S.

8 Aleksandrs Belovs, Gábor Ivanyos, Youming Qiao, Miklos Santha, and Siyi Yang. On the polynomial parity argument complexity of the combinatorial nullstellensatz. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 30:1–30:24, 2017. doi:10.4230/LIPIcs.CCC.2017.30.

9 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498. IEEE, 2015.

10 Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

11 Joshua Buresh-Oppenheim. On the TFNP complexity of factoring. *Manuscript*, 2006. URL: http://www.cs.toronto.edu/~bureshop/factor.pdf.

12 Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci.*, 62(2):267–289, 2001. doi:10.1006/jcss.2000.1726.

13 Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Logic*, 163(9):1163–1182, 2012. doi:10.1016/j.apal.2012.01.015.

14 I. Bárány, S. B. Shlosman, and A. Szücs. On a topological generalization of a theorem of tverberg. *Journal of the London Mathematical Society*, s2-23(1):158–164, 1981. doi:10.1112/jlms/s2-23.1.158.

15 Claude Chevalley. Démonstration d'une hypothèse de m. artin. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):73–75, December 1935. doi:10.1007/BF02940714.

16 Arka Rai Choudhuri, Pavel Hubácek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N Rothblum. Finding a nash equilibrium is no easier than breaking fiat-shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1103–1114. ACM, 2019.

17 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652.

18 Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 790–804, 2011. doi:10.1137/1.9781611973082.62.

19 Xiaotie Deng, Jack R. Edmonds, Zhe Feng, Zhengyang Liu, Qi Qi, and Zeying Xu. Understanding PPA-Completeness. In Ran Raz, editor, *31st Conference on Computational Complexity (CCC 2016)*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:25, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.23.

20 Aris Filos-Ratsikas and Paul W. Goldberg. Consensus halving is ppa-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 51–64, 2018. doi:10.1145/3188745.3188880.

21 Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *STOC (to appear)*, 2019. arXiv:1805.12559.

22 C. Goldberg and D. West. Bisection of circle colorings. *SIAM Journal on Algebraic Discrete Methods*, 6(1):93–106, 1985. doi:10.1137/0606010.

23 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 38:1–38:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

24 Michelangelo Grigni. A sperner lemma complete for ppa. *Information Processing Letters*, 77(5-6):255–259, 2001.

25 Alexandros Hollender. The classes PPA-k: Existence from arguments modulo k. In Ioannis Caragiannis, Vahab Mirrokni, and Evdokia Nikolova, editors, *Web and Internet Economics*, pages 214–227, Cham, 2019. Springer International Publishing.

26 Emil Jerábek. Integer factoring and modular square roots. *J. Comput. Syst. Sci.*, 82(2):380–394, 2016. doi:10.1016/j.jcss.2015.08.001.

27 Alan S. Johnson. Reductions and propositional proofs for total NP search problems. *UC San Diego Electronic Theses and Dissertations*, 2011. URL: https://escholarship.org/uc/item/89r774x7.

28 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

29 Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *Journal of the ACM (JACM)*, 66(5):34, 2019.

30 Pravesh K Kothari and Ruta Mehta. Sum-of-squares meets nash: lower bounds for finding any equilibrium. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1241–1248. ACM, 2018.

31 Donald L. Kreher and Douglas R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*, volume 7 of *Discrete Mathematics and Its Applications*. CRC Press, 1998.

32 Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.

33 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

34 Christian Reiher. On kemnitz'conjecture concerning lattice-points in the plane. *The Ramanujan Journal*, 13(1-3):333–337, 2007.

35 Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 258–265, 2016.

36 Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. Ppp-completeness with connections to cryptography. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 148–158, 2018. doi:10.1109/FOCS.2018.00023.

37 Ewald Warning. Bemerkung zur vorstehenden arbeit von herrn chevalley. *Abh. Math. Sem. Univ. Hamburg*, 11:76–83, 1936.

A Appendix: Reductions Between Complete Problems

In order to prove Theorem 9, we introduce an additional problem that will serve as intermediate problem in our reductions.

▶ **Definition 47** ($\textsc{Leaf}'_q$).
Principle: *Same as* $\textsc{Leaf}_q$*, but degrees are allowed to be larger (polynomially bounded).*
Object: *q-uniform hypergraph $G = (V, E)$. Designated vertex $v^* \in V$.*
Inputs: $\triangleright$ $C : \{0,1\}^n \to (\{0,1\}^{nq})^k$
where $(\{0,1\}^{nq})^k$ is interpreted as k many q-subsets of $\{0,1\}^n$
$\triangleright$ $v^* \in \{0,1\}^n$ *(usually 0^n)*
Encoding: $V \coloneqq \{0,1\}^n$.
For distinct $v_1, \ldots, v_q$, edge $e \coloneqq \{v_1, \ldots, v_q\} \in E$ if $e \in C(v)$ for all $v \in e$
Solutions: v^* *if* $\deg(v) \equiv 0 \pmod q$ *and*
$v \neq v^*$ *if* $\deg(v) \not\equiv 0 \pmod q$

Proof of Theorem 9. We show the following inter-reducibilities: (1) $\textsc{Leaf}_q \asymp \textsc{Leaf}'_q$, (2) $\textsc{Leaf}'_q \asymp \textsc{Bipartite}_q$ and (3) $\textsc{Leaf}_q \asymp \textsc{Lonely}_q$.

(1a) $\textsc{Leaf}_q \preceq \textsc{Leaf}'_q$. Each instance of $\textsc{Leaf}_q$ is trivially an instance of $\textsc{Leaf}'_q$.

(1b) $\textsc{Leaf}'_q \preceq \textsc{Leaf}_q$. We start with a $\textsc{Leaf}'_q$ instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$ with degree at most k. Let $t = \lceil k/q \rceil$. We construct a $\textsc{Leaf}_q$ instance encoding a hypergraph $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V} \coloneqq V \times [t]$, intuitively making t copies of each vertex.

In order to locally compute hyperedges, we first fix a canonical algorithm that for any vertex v and any edge $e \in E$ incident on v, assigns it a label $\ell_v(e) \in [t]$, with at most q edges mapping to the same label – e.g. sort all edges incident on v in lexicographic order and bucket them sequentially in at most t groups of at most q each. Note that we can ensure that for any vertex v at most one label gets mapped to by a non-zero, non-q number of edges. Moreover, if $\deg(v) \equiv 0 \pmod q$, then exactly q or 0 edges are assigned to any label.

We'll assume that $\deg(v^*) \not\equiv 0 \pmod q$, as otherwise, a reduction wouldn't be necessary. We let (v^*, ℓ^*) be the designated vertex of the $\textsc{Leaf}_q$ instance, where ℓ^* is the unique label that gets mapped to by a non-zero, non-q number of edges incident on v^*.

For any vertex $(v, i) \in \overline{V}$, we assign it at most q edges as follows: For each edge $e = \{v_1, \ldots, v_q\}$ such that $\ell_v(e) = i$, the corresponding hyperedge of (v, i) is

$$(v_1, \ell_{v_1}(e)), \ldots, (v_q, \ell_{v_q}(e)) \ .$$

It is easy to see that the designated vertex (v^*, ℓ^*) indeed has non-zero, non-q degree. Moreover, a vertex $\deg(v, i) \notin \{0, q\}$ in $\overline{G}$ only if v has a non-multiple-of-q degree in G. Thus, solutions to the $\textsc{Leaf}_q$ instance naturally maps to solutions to the original $\textsc{Leaf}'_q$ instance.

By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ is efficiently indexable.

(2a) $\textsc{Leaf}'_q \preceq \textsc{Bipartite}_q$. We start with a $\textsc{Leaf}'_q$ instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$. We construct a $\textsc{Bipartite}_q$ instance encoding a graph $\overline{G} = (\overline{V} \cup \overline{U}, \overline{E})$ such that $\overline{V} = V$ and $\overline{U} = \binom{V}{q}$, i.e. all q-sized subsets of V. We include the edge $(v, e) \in \overline{E}$ if $e \in E$ is incident on v. The designated vertex for the $\textsc{Bipartite}_q$ instance is v^* in $\overline{V}$.

Clearly, all vertices $e \in \overline{U}$ have degree either q or 0. For any $v \in \overline{V}$, the degree of v in $\overline{G}$ is same as its degree in G. Thus, any solution to the $\textsc{Bipartite}_q$ instance immediately gives

a solution to the original $\textsc{Leaf}'_q$ instance. By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ and $\overline{U}$ are efficiently indexable (cf. [31, §2.3] for efficiently indexing $\overline{U}$).

(2b) $\textsc{Bipartite}_q \preceq \textsc{Leaf}'_q$. We start with a $\textsc{Bipartite}_q$ instance (C, v^*) encoding a bipartite graph $\mathcal{G} = (V \cup U, E)$ with maximum degree of any vertex being at most k. We construct a $\textsc{Leaf}'_q$ instance encoding a hypergraph $\overline{G} = (\overline{V}, \overline{E})$ such that $\overline{V} = V$ with designated vertex v^*.

First, we fix a canonical algorithm that for any vertex $u \in U$ with $\deg_G(u) \equiv 0 \pmod q$ produces a partition of it's neighbors with q vertices of V in each part. Now, the set of q-uniform hyperedges incident on any vertex $v \in \overline{V}$ in $\overline{E}$ can be obtained as: for all neighbors u of v, with $\deg_G(u) \equiv 0 \pmod q$, we include a hyperedge consisting of all vertices in the same partition as v among the neighbors of u (we ignore neighbors u with $\deg(u) \not\equiv 0 \pmod q$).

Observe that $\deg_{\overline{G}}(v) \leq \deg_G(v)$ and equality holds if and only if all neighbors of v in G have degree $\equiv 0 \pmod q$. Hence for any $v \in \overline{V}$, if $\deg_{\overline{G}}(v) \neq \deg_G(v) \pmod q$, then there exists a neighbor $u \in U$ of v in G such that $\deg(u) \not\equiv 0 \pmod q$. Thus, if $v = v^*$ and $\deg_{\overline{G}}(v^*) \equiv 0 \pmod q$, then either $\deg_G(v) \equiv 0 \pmod q$ or we can find a neighbor u of v in G with $\deg(u) \not\equiv 0 \pmod q$. Similarly if for some $v \neq v^*$, we have $\deg_{\overline{G}}(v^*) \not\equiv 0 \pmod q$, then either $\deg_G(v) \not\equiv 0 \pmod q$ or we can find a neighbor u of v in G with $\deg(u) \not\equiv 0 \pmod q$. Thus, any solution to the $\textsc{Leaf}'_q$ instance gives us a solution to the original $\textsc{Bipartite}_q$ instance. This completes the reduction since $\overline{V} = \{0,1\}^n$ and the edges are locally computable with black-box access to C.

(3a) $\textsc{Leaf}_q \preceq \textsc{Lonely}_q$. We start with a $\textsc{Leaf}_q$ instance (C, v^*), where C encode a q-uniform hypergraph $G = (V, E)$ with degree at most q. If $\deg_G(v^*) = q$ or 0, then we don't need any further reduction. Else, we construct a $\textsc{Lonely}_q$ instance encoding a q-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V} = V \times [q]$. The designated vertices will be $V^* = \{(v, q-i) : 1 \leq i \leq q - \deg(v^*)\}$. Note that, $|V^*| = q - \deg_G(v^*)$ and hence $1 \leq |V^*| \leq q-1$.

In order to locally compute hyperedges, we first fix a canonical algorithm that for any vertex v and any edge $e \in E$ incident on v, assigns it a unique label $\ell_v(e) \in [q]$ – e.g. sort all edges incident on v in lexicographic order and label them sequentially in $[q]$. In fact, we can ensure that an edge incident on v get labeled within $\{1, \ldots, \deg_G(v)\}$.

For any vertex $(v, i) \in \overline{V}$, we assign it at most one hyperedge as follows:

- ▷ If $\deg_G(v) = 0$, we include the hyperedge $\{(v, i) : i \in [q]\}$.
- ▷ Else if $\deg_G(v) \geq i$, then for edge $e = \{v_1, \ldots, v_q\}$ incident on v such that $\ell_v(e) = i$, the corresponding hyperedge of (v, i) is $(v_1, \ell_{v_1}(e)), \ldots, (v_q, \ell_{v_q}(e))$.
- ▷ Else if $0 < \deg_G(v) < i$, we leave it isolated.

It is easy to see that our definition of hyperedges is consistent and that the designated vertices V^* are indeed isolated. Moreover, a vertex (v, i) is isolated in $\overline{G}$ only if $1 \leq \deg_G(v) \leq q-1$. Thus, solutions to the $\textsc{Leaf}_q$ instance naturally maps to solutions to the original $\textsc{Leaf}'_q$ instance.

By Remark 10, this completes the reduction since the edges are locally computable with black-box access to C and $\overline{V}$ is efficiently indexable.

(3b) $\textsc{Lonely}_q \preceq \textsc{Leaf}_q$. We start with a $\textsc{Lonely}_q$ instance (C, V^*), where C encode a q-dimensional matching $G = (V, E)$. We construct a $\textsc{Leaf}_q$ instance encoding a q-uniform hypergraph $\overline{G} = (\overline{V}, \overline{E})$ on vertex set $\overline{V}$ that will be specified shortly. We describe the hyperedges in $\overline{G}$ and it'll be clear how to compute the hyperedges for any vertex locally with just black-box access to C.

We start with $\overline{V} = V$. Our goal is to transform all vertices of degree 1 to degree q, while ensuring that vertices of degree 0 are mapped to vertices of degree not a multiple of q. Towards this goal we let $\overline{E}$ to be set of edges in E in addition to $q-1$ canonical q-dimensional matchings over V. For example, for a vertex $v \coloneqq (x_1, \ldots, x_n) \in V = [q]^n$, the corresponding edges in $\overline{E}$ include an edge in E (if any) and edges of the type $e_i = \{(x_1, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_n) : j \in [q]\}$ for $i \in [q-1]$ (note, this requires us to assume $n \geq q-1$). Adding the $q-1$ matchings increases the degree of each vertex by $q-1$. Therefore, vertices with initial degree 1 now have degree q and vertices with initial degree 0 now have degree $q-1$. However, a couple of issues remain in order to complete the reduction, which we handle next.

Multiplicities. An edge $e \in E$ might have gotten added twice, if it belonged to one of the canonical matchings. To avoid this issue altogether, instead of adding edges directly on V, we augment $\overline{V}$ to become $\overline{V} \coloneqq V \cup \left(\binom{V}{q} \times [q-1]\right)$, i.e. in addition to V, we have $q-1$ vertices for every potential hyperedge of G. For any edge $e \coloneqq \{v_1, \ldots, v_q\} \in E$, instead of adding it directly in $\overline{G}$, we add hyperedge $\{v, (e,1), (e,2), \ldots, (e, q-1)\}$ for each $v \in e$. Note that, all vertices $(e,i) \in \binom{V}{q} \times [q-1]$ have degree q if $e \in E$ and degree 0 if $e \notin E$, so they are non-solutions for the $\textsc{Leaf}_q$ instance. For vertices in V, we still have as before that vertices with initial degree 1 now have degree q and vertices with initial degree 0 now have degree $q-1$.

Designated vertex. In a $\textsc{Leaf}_q$ instance, we need to specify a single designated vertex $v^* \in \overline{V}$. If the $\textsc{Lonely}_q$ instance had a single designated vertex then we would be done. However, in general it is not possible to assume this (for non-prime q). Nevertheless, we provide a way to get around this. We augment $\overline{V}$ with $t = (q-1)(q-k)+1$ additional vertices to become $\overline{V} \coloneqq V \cup \left(\binom{V}{q} \times [q-1]\right) \cup \{w_{i,j} : i \in [q-k],\ j \in [q-1]\} \cup \{v^*\}$, where v^* will eventually be the single designated vertex for the $\textsc{Leaf}_q$ instance.

Let $V^* = \{u_1, \ldots, u_k\} \subseteq V$ be the set of designated vertices in the $\textsc{Lonely}_q$ instance (note $1 \leq k < q$). So far, note that $\deg_{\overline{G}}(u_i) = q-1$. The only new hyperedges we add will be among u_i's, $w_{i,j}$'s and v^*, in such a way that $\deg_{\overline{G}}(u_i)$ will become q, the degree of all $w_{i,j}$'s will also be q and degree of v^* will be $q-k$.

- ▷ For each $u \in V^*$, include $\{u, w_{1,1}, \ldots, w_{1,q-1}\}$. So far, $\deg_{\overline{G}}(u) = q$ and $\deg_{\overline{G}}(w_{1,j}) = k$.
- ▷ For each $j \in [q-1]$ and each $i \in \{2, \ldots, q-k\}$, include $\{w_{1,j}, w_{i,1}, \ldots, w_{i,q-1}\}$. So far, $\deg_{\overline{G}}(w_{i,j}) = q-1$ for all $(i,j) \in [q-k] \times [q-1]$.
- ▷ Finally, for each $(i,j) \in [q-k] \times [q-1]$, include $\{v^*, w_{i,1}, \ldots, w_{i,q-1}\}$. Now, $\deg_{\overline{G}}(w_{i,j}) = q$ for all $(i,j) \in [q-k] \times [q-1]$ and $\deg_{\overline{G}}(v^*) = q-k$.

Thus, we have finally reduced to a $\textsc{Leaf}_q$ instance encoding the graph $\overline{G} = (\overline{V}, \overline{E})$ with $\overline{V} \coloneqq V \cup \left(\binom{V}{q} \times [q-1]\right) \cup \{w_{i,j} : i \in [q-k],\ j \in [q-1]\} \cup \{v^*\}$. By Remark 10, this completes the reduction, since $\overline{V}$ is efficiently indexable (again, see [31] for a reference on indexing $\binom{V}{q}$) and the edges are locally computable using black-box access to C. ◀

A.1 Completeness of Succinct Bipartite

We introduce an intermediate problem to show PPA_p–completeness of $\textsc{SuccinctBipartite}_p$.

▶ **Definition 48** ($\textsc{TwoMatchings}_p$).

Principle: *Two p-dimensional matchings over a common vertex set, with a vertex in exactly one of the matchings, has another such vertex.*

Object: *Two p-dimensional matchings $G_0 = (V, E_0)$, $G_1 = (V, E_1)$.*
Designated vertex $v^* \in V$.

Inputs: ▷ $C_0 : \{0,1\}^n \to (\{0,1\}^n)^p$ *and* $C_1 : \{0,1\}^n \to (\{0,1\}^n)^p$
▷ $v^* \in \{0,1\}^n$
Encoding: $V \coloneqq \{0,1\}^n$. *For* $b \in \{0,1\}$, $E_b \coloneqq \{e : C_b(v) = e \text{ for all } v \in e\}$
Solutions: v^* *if* $\deg_{G_0}(v^*) \neq 1$ *or* $\deg_{G_1}(v^*) \neq 0$ *and*
$v \neq v^*$ *if* $\deg_{G_0}(v^*) \neq \deg_{G_1}(v^*)$

Observe that in the case of $p = 2$, TwoMatchings$_p$ can be readily seen as equivalent to Leaf$_2$.

▶ **Theorem 49.** *For any prime p, SuccinctBipartite$_p$ and TwoMatchings$_p$ are* PPA$_p$*–complete.*

Proof. We show Bipartite$_p \preceq$ SuccinctBipartite$_p \preceq$ TwoMatchings$_p \preceq$ Lonely$_p$.

Bipartite$_p \preceq$ SuccinctBipartite$_p$. Since p is a prime, we can assume that the designated vertex v^* has degree 1 (mod p) (similar to Lemma 44). Since the number of neighbors in a Bipartite$_p$ instance are polynomial, we can check if an edge exists and canonically group them efficiently for all vertices with degree being a multiple of p. The designated edge e^* is the unique ungrouped edge incident on v^*. Thus, valid solution edges to SuccinctBipartite$_p$ must have at least one endpoint which is a solution to the original Bipartite$_p$ instance.

SuccinctBipartite$_p \preceq$ TwoMatchings$_p$. We reduce to a TwoMatchings$_p$ instance encoding two p-dimensional matchings $\overline{G}_0 = (\overline{V}, \overline{E}_0)$ and $\overline{G}_1 = (\overline{V}, \overline{E}_1)$, over the vertex set $\overline{V} = V \times U \times [p-1]$, that is, all possible edges producible in the SuccinctBipartite$_p$ instance. The designated vertex v^* is the designated edge e^* in the SuccinctBipartite$_p$ instance.

For any edges $e_1, \ldots, e_p$, which are grouped by ϕ_V pivoted at some $v \in V$, we include the hyperedge $\{e_1, \ldots, e_p\}$ in $\overline{E}_0$. Similarly, for any edges $e_1, \ldots, e_p$, which are grouped by ϕ_U pivoted at some $u \in U$, we include the hyperedge $\{e_1, \ldots, e_p\}$ in $\overline{E}_1$. It is easy to see that points in exactly one of the two matchings $\overline{G}_0$ or $\overline{G}_1$ correspond to edges of the SuccinctBipartite$_p$ instance that are not grouped at exactly one end. Thus, we can derive a solution to SuccinctBipartite$_p$ from a solution to TwoMatchings$_p$. (Remark: while edges which are not grouped at either end are solutions to SuccinctBipartite$_p$, they do not correspond to a solution in the TwoMatchings$_p$ instance.)

TwoMatchings$_p \preceq$ Lonely$_p$. Given an instance of TwoMatchings$_p$ that encodes two p-dimensional matchings $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$, we reduce to an instance of Lonely$_p$ encoding a p-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ such that $\overline{V} = V \times [p]$. The designated vertex for the Lonely$_p$ instance is (v^*, p).

For any hyperedge $\{v_1, \ldots, v_p\}$ in E_0, we include the hyperedge $\{(v_1, i), (v_2, i), \ldots, (v_p, i)\}$ in $\overline{G}$ for each $i \in \{1, \ldots, p-1\}$. Similarly, for any hyperedge $\{v_1, \ldots, v_p\}$ in E_1, we include the hyperedge $\{(v_1, p), (v_2, p), \ldots, (v_p, p)\}$ in $\overline{G}$. If $v \in V$ is isolated in both G_0 and G_1, then we include the hyperedge $\{v\} \times [p]$.

Observe that, (v^*, p) is isolated by design. A vertex (v, i), for $i < p$ is isolated only if $\deg_{G_0}(v) = 0$ and $\deg(G_1) = 1$. Similarly, the vertex (v, p) is isolated only if $\deg_{G_0}(v) = 1$ and $\deg(G_1) = 0$. Thus, isolated vertices in the Lonely$_p$ instance correspond to solutions of the TwoMatchings$_p$ instance. ◀

A.2 Equivalence with PMOD$_p$

Proof of Lemma 44. Consider any prime p. Consider a Lonely$_p$ instance (C, V^*), where C encodes a p-dimensional matching $G = (V, E)$ and $|V^*| = \ell$. We wish to reduce to an instance of Lonely$_p^k$, where the number of designated vertices is exactly k. First, we'll

assume that all vertices in V^* are indeed isolated in G, otherwise, no reduction would be necessary. The key reason why this lemma holds for primes (and not for composites) is because ℓ has a multiplicative inverse modulo p. In particular, let $t \equiv \ell^{-1}k \pmod p$.

We construct a $\textsc{Lonely}_p^k$ instance encoding the p-dimensional matching $\overline{G} = (\overline{V}, \overline{E})$ over $\overline{V} = V \times [t]$. We let $\overline{V}^*$ to be the lexicographically first k vertices in $V^* \times [t]$. Note that $|V^* \times [t]| = t.\ell \equiv k \pmod p$. Thus, we partition the remaining vertices of $V^* \times [t]$ into p-uniform hyperedges. For any vertex $v \in V \setminus V^*$, with neighbors $v_1, \ldots, v_{p-1}$ in G, the neighbors of (v, i) in $\overline{G}$ are $(v_1, i), \ldots, (v_{p-1}, i)$ for any $i \in [t]$. Thus, a vertex (v, i) is isolated only if it is in $\overline{V}^*$ or v is isolated in G. This completes the reduction since $\overline{V}$ is efficiently indexable – see Remark 10. ◀

Proof of Corollary 45. It is easy to see that $\textsc{Mod}_q \leq \textsc{Lonely}_q$ with number of designated vertices being $k \equiv -2^n \pmod q$, since $\{0,1\}^n$ is efficiently indexable (Remark 10). Conversely, using Lemma 44, we can reduce a $\textsc{Lonely}_q$ instance to a $\textsc{Mod}_q$ instance as follows: Let the $\textsc{Lonely}_q$ instance encode a q-dimensional matching over $[q]^n$ with k designated vertices. If any of the designated vertices are not isolated, no further reduction is necessary. Otherwise, we can embed the non-designated vertices of G into the first $q^n - k$ vertices of $\{0,1\}^N$ for a choice of N satisfying $2^N > q^n$ and $2^N \equiv -k \pmod q$. Such an N is guaranteed to exist (and can be efficiently found) when q is a prime. Since $2^N - q^n + k \equiv 0 \pmod q$, we can partition the remaining vertices into q-uniform hyperedges, and thus, solutions to the $\textsc{Mod}_q$ instance readily map to solutions of the original $\textsc{Lonely}'_q$ instance. ◀

B Appendix: Proof of Theorem 36

Proof of Theorem 36. We show a reduction from $\textsc{ChevalleyWithSymmetry}_p$ to $\textsc{SuccinctBipartite}_p[\mathsf{AC}^0_{\mathbb{F}_p}]$; the theorem then follows by combining this reduction with Theorem 3. Additionally from the proof of Theorem 3 we can assume without loss of generality that the system of polynomials $\boldsymbol{f} = (\boldsymbol{g}, \boldsymbol{h})$ of the $\textsc{ChevalleyWithSymmetry}_p$ instance has the following properties.

a. Each polynomial f_i has degree at most 2.
b. Each polynomial f_i has at most 3 monomials.
c. Each polynomial f_i has at most 3 variables.

Hence, we can compute each of the polynomials g_i^{p-1} explicitly as a sum of monomials. The degree of this polynomial is $O(p)$ and the number of monomials is at most 3^p. Observe that since p is a constant, 3^p is also a constant.

Now we follow the proof of Lemma 25 that reduces $\textsc{ChevalleyWithSymmetry}_p$ to $\textsc{SuccinctBipartite}_p$. Following this proof there are two circuits that we need to replace with formulas in $\mathsf{AC}^0_{\mathbb{F}_p}$ to reduce to $\textsc{SuccinctBipartite}_p$. The first circuit is the edge counting circuit $\mathcal{C}$ and the second is the grouping function ϕ. We remind that the bipartite graph $G(U, V)$ of the $\textsc{SuccinctBipartite}_p$ instance has two parts U, V, where U is the set of all possible assignments, i.e. $\mathbb{F}_p^n$, and $V = V_1 \cup V_2$, where V_1 in the set of all monomials of the polynomial $F = \prod_{i=1}^m (1 - g_i^{p-1})$ and V_2 is the set of all p-tuples of assignments, i.e. $\left(\mathbb{F}_p^n\right)^p$.

From Edge Counting Circuit To Edge Counting Formula. As described in the proof of Lemma 25 the edge counting circuit takes as input a vertex $u \in U$ and a vertex $v \in V$ and outputs the multiplicity of the edge $\{u, v\}$ in G. Hence, the edge counting formula $\mathcal{C}$, that we want to implement, takes as input a tuple $(\boldsymbol{x}, s, \boldsymbol{a}, \boldsymbol{y})$. The vector $\boldsymbol{x}$ corresponds to the assignment in U. The vector $\boldsymbol{a}$ corresponds to the description of a monomial of F, as the product $\prod_{i=1}^m t'_{ia_i}$ where t'_{ia_i} is the a_i-th monomial of the polynomial $1 - g_i^{p-1}$. The vector

$\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p)$ and corresponds to a p-tuple in V_2. Finally, s is a selector number to distinguish between $v \in V_1$ and $v \in V_2$, namely if $s = 1$, we have $v \in V_1$ and if $s = 0$, we have that $v \in V_2$. So, the edge counting formula can be written as follows

$$\mathcal{C}(\boldsymbol{x}, s, \boldsymbol{a}, \boldsymbol{y}) = \left(\prod_{i \in \mathbb{F}_p, i \neq 1} (s - i) \right) \mathcal{C}_1(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}) + \left(\prod_{i \in \mathbb{F}_p, i \neq 0} (s - i) \right) \mathcal{C}_2(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}). \tag{B.1}$$

This way we can define the edge counting formula $\mathcal{C}_1$ for when $v \in V_1$ and the edge counting formula $\mathcal{C}_2$ for when $v \in V_2$ separately and combine them by using at most two additional layers in the arithmetic formula. Now, $\mathcal{C}_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}) = \mathbb{1}(\boldsymbol{y} = \boldsymbol{0}) \cdot \prod_{i=1}^{m} \mathcal{Q}_i(\boldsymbol{x}, a_i)$ where $\mathcal{Q}_i(\boldsymbol{x}, a_i)$ is the formula to compute the value $t_{i,a_i}(\boldsymbol{x})$. Observe that the factor $\mathbb{1}(\boldsymbol{y} = \boldsymbol{0})$ can be easily computed and is necessary since $\mathcal{C}_1$ should consider only neighbors between $\boldsymbol{x}$ and monomials in V_1. Hence, if $\boldsymbol{y}$ is not equal to $\boldsymbol{0}$, $\mathcal{C}_1$ should return 0. As we already explained the number of monomials of $1 - g_i^{p-1}$ is constant, and hence the formula $\mathcal{Q}_i(\boldsymbol{x}, a_i)$ can be easily implemented in constant depth using a selector between all different monomials similarly to Equation (B.1). Hence, $\mathcal{C}_1$ is implemented in constant depth.

The formula $\mathcal{C}_2$ has a factor $\mathbb{1}(\boldsymbol{a} = 0)$ to ensure only neighbors in V_2 have non-zero outputs. The main challenge in the description of $\mathcal{C}_2$ is that every distinct p-tuple $\boldsymbol{y}$ has $p!$ equivalent representations, but the modulo p argument of Lemma 25 applies only when edges appear to precisely one of the equivalent copies of the p-tuple. Thus, we let $\mathcal{C}_2$ add edges only to the lexicographically ordered version of $\boldsymbol{y}$. It is a simple exercise to see that sorting of $p!$ numbers, when p is constant, is possible in constant depth. We leave this folklore observation as an exercise to the reader. Once we make sure that $\boldsymbol{y}$ is lexicographically sorted, we compute a sorted representation of the set $\Sigma_{\boldsymbol{x}} = \{\boldsymbol{x}, \sigma(\boldsymbol{x}), \ldots, \sigma^{p-1}(\boldsymbol{x})\}$, where σ is the permutation in the input of the ChevalleyWithSymmetry$_p$ problem. Then, we can easily check whether the p-tuple represented by $\boldsymbol{y}$ is the same as the sorted p-tuple $\Sigma_{\boldsymbol{x}}$. Finally, we observe that edges between $\boldsymbol{x}$ and $\Sigma_{\boldsymbol{x}}$ are only used when $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}} \cap \overline{\mathcal{V}}_{\boldsymbol{h}}$ which again can be checked with constant depth formulas. If these checks pass, then $\mathcal{C}_2$ outputs $p - 1$, otherwise it outputs 0.

From Grouping Circuit to Grouping Formula. For this step we use selectors similarly to Equation (B.1) and sorting as in the description of $\mathcal{C}_2$. We consider two different cases for the grouping formula ϕ. When the first argument is in U, i.e. grouping with respect to an assignment, we call the formula ψ and when the first argument is in V, i.e. grouping with respect to monomials/p-tuples, we call the formula χ. Then, ϕ selects between ψ and χ using a selector. This adds at most two layers to ϕ.

Grouping formula for $x \in U$. First, we describe ψ with inputs $\boldsymbol{x} \in U$, $(s, \boldsymbol{a}, \boldsymbol{y}) \in V$ and r be the copy of the input edge. We have two cases with respect to whether $s = 1$ or $s = 0$. Let ψ^1 be the formula for the first case and ψ^2 be the formula for the second case. For the case $s = 1$, we need again to consider two cases: (i) $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{g}}$ and (ii) $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$. For case (i) we describe the formula ψ_1^1 and for case (ii) we define the formula ψ_2^1. It is easy to see that computing $\mathbb{1}(\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}})$ can be done using a depth 3 formula since $\boldsymbol{g}$ is given in an explicit form. Hence, once again, we can combine ψ_1^1 and ψ_2^1 using a selectors.

Case $s = 1$, $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{g}}$. The formula ψ_1^1 first computes $i^\star = \min_{i : 1 - g_i^{p-1}(\boldsymbol{x}) = 0} i$. This is doable in constant depth, since we can compute in parallel the value $\mathbb{1}(1 - g_i^{p-1}(\boldsymbol{x}) = 0)$ for all $i \in [m_1]$ and then in an extra layer compute for every i whether $1 - g_i^{p-1}(\boldsymbol{x}) = 0$ and $1 - g_j^{p-1}(\boldsymbol{x}) \neq 0$ for all $j < i$, which requires just one multiplication gate per i.

Next, we define a formula ψ^1_{1i} for all i and we use a selector to output $\psi^1_{1i^*}$. In ψ^1_{1i}, we first compute the value $C_i(\boldsymbol{x}) = \prod_{j\neq i} t_{j,a_j}(\boldsymbol{x})$. The output of ψ^1_{1i} is a p-tuple, where each of the p parts differs only on the coordinate a_i of $\boldsymbol{a}$, which corresponds to a monomial of $1 - g_i^{p-1}$, and the value r. We need to determine p different values for the tuple (a_i, r) where $a_i \in [3^p]$, $r \in \mathbb{Z}_p$. These values only depend on the evaluation of the polynomial g_i on the input $\boldsymbol{x}$, on the value a_i and on the value r.
Because of the properties of the input system of polynomials $\boldsymbol{f}$, each polynomial g_i depends only on three variables in $\mathbb{Z}_p$, let these variables be x_1, x_2, x_3 for simplicity. Then, for every i the grouping function that we want to implement is a function with input domain $\mathbb{Z}_p^3 \times [3^p] \times \mathbb{Z}_p$ and output domain $\mathbb{Z}_p^2$. The truth-table of this function has size that depends only on p and therefore we can explicitly implement this function using its truth-table in constant depth. This finishes the construction of ψ^1_{1i}.

Case $s = 1$, $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{g}}$. We remind that $\boldsymbol{a} = \boldsymbol{0}$ corresponds to the constant monomial 1 of the polynomial F. If $\boldsymbol{a} \neq \boldsymbol{0}$, this case is similar to the previous, except that we use the polynomials $\boldsymbol{g}_i^{p-1}$ instead of $1 - \boldsymbol{g}_i^{p-1}$, see also the proof of Lemma 25. If $\boldsymbol{a} = \boldsymbol{0}$, ψ^1_2 outputs the input edge $(1, \boldsymbol{a}, \boldsymbol{0}, 1)$ and $p-1$ edges of the form $(0, \boldsymbol{0}, \boldsymbol{y}, t), t \in [p-1]$ where $\boldsymbol{y}$ is the lexicographically ordered set $\Sigma_{\boldsymbol{x}}$.

Case $s = 0$. In this case, the formula ψ^2 checks whether the vector $\boldsymbol{y}$ is in lexicographic order as described in the edge counting formula $\mathcal{C}$ and $\boldsymbol{a} = \boldsymbol{0}$. It also checks if $\boldsymbol{x} \in \mathcal{V}_{\boldsymbol{f}_1} \cap \overline{\mathcal{V}}_{\boldsymbol{f}_2}$ as described before. If any of these checks fails, the output is $\boldsymbol{0}$. Otherwise, if $\boldsymbol{y} = \Sigma_{\boldsymbol{x}}$, then we output $p-1$ copies of the edge $(0, \boldsymbol{0}, \boldsymbol{y}, t), t \in [p-1]$, that connects $\boldsymbol{x}$ with $\boldsymbol{y}$, and the edge $(1, \boldsymbol{0}, \boldsymbol{0}, 1)$, that connects $\boldsymbol{x}$ with the constant term of F.

Grouping formula for vertices in V. We describe the grouping formula χ when the first argument belongs to V, i.e. the grouping with respect to monomials or p-tuples. The input again is a triple $(s, \boldsymbol{a}, \boldsymbol{y})$ representing a vertex in V, a vertex $\boldsymbol{x} \in U$ and a number $r \in \mathbb{Z}_p$ that denotes the index of the edge that we want to group, among its possible multiple copies. Again we have two cases, $s = 1$ and $s = 0$, which correspond to the formulas χ^1 and χ^2 respectively. In each case, we have to check that one of $\boldsymbol{a}$, $\boldsymbol{y}$ is equal to $\boldsymbol{0}$, which is done similarly to the previous formulas.

Case $s = 1$. In this case, the input is a monomial $t_{\boldsymbol{a}}(\boldsymbol{x}) = \prod_{i=1}^{m_1} t_{i,a_i}(\boldsymbol{x})$ and we have to find a variable that appears with degree less than $p-1$. We first construct a formula χ^1_j that computes z^k, where k is the degree of x_j in $t_{\boldsymbol{a}}(\boldsymbol{x})$. This can be done with a constant size formula that for a given index j multiplies the powers of x_j in the monomials of $1 - g_i^{p-1}$ appearing in t.
Now, we compute all values $\chi^1_j(1), \ldots, \chi^1_j(p-1)$ and we check in parallel if at least one of them is different from 1. If this is the case, then the degree of x_j in $t(\boldsymbol{x})$ is less than $p-1$. Hence, we have computed the formula $\bar{\chi}^1_j(\boldsymbol{a}) = \mathbb{1}(\text{degree of } x_j \text{ in } t_{\boldsymbol{a}} \neq p-1)$. We can find the smallest index j^* such that $\bar{\chi}^1_j(\boldsymbol{a}) = 1$ using the same construction as in ψ^1. So, we can construct a formula for each j that is equal to 1 if and only if $j = j^*$ is the smallest index such that x_{j^*} has degree less than $p-1$ in $t_{\boldsymbol{a}}$. Finally, we use a selector to find the value $C_{j^*}(\boldsymbol{x}) = x_{j^*}^{-k} t(\boldsymbol{x})$, by computing $C_j(\boldsymbol{x})$ for all j. This is done through the product of all variables that appear in $t_{\boldsymbol{a}}(\boldsymbol{x})$ excluding x_j.
It is left to implement a formula that takes as input the value $C_{j^*}(\boldsymbol{x}) \in \mathbb{Z}_p$, the value of $r \in \mathbb{Z}_p$ and the values $\chi^1_{j^*}(0), \chi^1_{j^*}(1), \ldots, \chi^1_{j^*}(p-1)$ all in $\mathbb{Z}_p$ and outputs a group of p values in $\mathbb{Z}_p^2$, which corresponds to the values of x_j and r in the output. Observe that both the input and the output size of this formula are only a function of p and, hence, constant. Therefore, we can explicitly construct a constant depth formula to capture this grouping.

Case $s = 0$. For constructing the formula χ^2 we first check whether $\boldsymbol{x} \in \overline{\mathcal{V}}_{\boldsymbol{f}_1}$ and whether $\boldsymbol{y}$ is the lexicographically sorted version of $\Sigma_{\boldsymbol{x}}$. These can both be done as we have described in the construction of the formula ψ above. If all checks pass, then we output the p edges of the form $(\boldsymbol{z}, r)$ for all $\boldsymbol{z} \in \Sigma_{\boldsymbol{x}}$, that correspond to the r-th copy of the edge between $\boldsymbol{z}$ and $\boldsymbol{y}$.

Combining the formulas ψ and χ through a selector concludes the construction of ϕ. Hence, the theorem follows by observing that the instance of $\textsc{ChevalleyWithSymmetry}_p$ that we get when reducing $\textsc{Lonely}_p$ to $\textsc{ChevalleyWithSymmetry}_p$ in Theorem 3 reduces to $\textsc{SuccinctBipartite}_p[\mathsf{AC}^0_{\mathbb{F}_p}]$. ◀

Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan
s_hirahara@nii.ac.jp

Abstract

The standard notion of promise problem is a pair of *disjoint* sets of instances, each of which is regarded as YES and NO instances, respectively, and the task of solving a promise problem is to distinguish these two sets of instances. In this paper, we introduce a set of new promise problems which are conjectured to be *non-disjoint,* and prove that hardness of these "non-disjoint" promise problems gives rise to the existence of hitting set generators (and vice versa). We do this by presenting a general principle which converts any black-box construction of a pseudorandom generator into the existence of a hitting set generator whose security is based on hardness of some "non-disjoint" promise problem (via a non-black-box security reduction).

Applying the principle to cryptographic pseudorandom generators, we introduce

The $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ Problem: Given a string x and a parameter s, distinguish whether the polynomial-time-bounded SAT-oracle Kolmogorov complexity of x is at most s, or the polynomial-time-bounded Kolmogorov complexity of x (without SAT oracle) is at least $s + O(\log |x|)$.

If $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is NP-hard, then the worst-case and average-case complexity of PH is equivalent. Under the plausible assumption that $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, the promise problem is non-disjoint. These results generalize the non-black-box worst-case to average-case reductions of Hirahara [31] and improve the approximation error from $\widetilde{O}(\sqrt{n})$ to $O(\log n)$.

Applying the principle to complexity-theoretic pseudorandom generators, we introduce a family of Meta-computational Circuit Lower-bound Problems (MCLPs), which are problems of distinguishing the truth tables of explicit functions from hard functions. Our results generalize the hardness versus randomness framework and identify problems whose circuit lower bounds characterize the existence of hitting set generators. For example, we introduce

The E vs $\mathsf{SIZE}(2^{o(n)})$ Problem: Given the truth table of a function f, distinguish whether f is computable in exponential time or requires exponential-size circuits to compute.

A *nearly-linear* $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit size lower bound for this promise problem is *equivalent* to the existence of a logarithmic-seed-length hitting set generator for $\mathsf{AC}^0 \circ \mathsf{XOR}$. Under the plausible assumption that $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$, the promise problem is non-disjoint (and thus the minimum circuit size is *infinity*). This is the first result that provides the exact characterization of the existence of a hitting set generator secure against $\mathfrak{C}$ by the worst-case lower bound against $\mathfrak{C}$ for a circuit class $\mathfrak{C} = \mathsf{AC}^0 \circ \mathsf{XOR} \subseteq \mathsf{TC}^0$. In addition, we prove that a nearly-linear size lower bound against co-nondeterministic read-once branching programs for some "non-disjoint" promise problem is sufficient for resolving $\mathsf{RL} = \mathsf{L}$.

We also establish the equivalence between the existence of a derandomization algorithm for uniform algorithms and a uniform lower bound for a problem of approximating Levin's Kt-complexity.

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

Keywords and phrases meta-complexity, pseudorandom generator, hitting set generator

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.20

Funding This work was supported by ACT-I, JST and JSPS KAKENHI Grant Number 18H04090.

Acknowledgements I thank Rahul Santhanam for helpful discussion and anonymous reviewers for helpful comments.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 20; pp. 20:1–20:47

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

1 Introduction

A *promise problem*, introduced by Even, Selman, and Yacobi [18], is a pair of disjoint sets $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ that are regarded as the sets of YES and NO instances, respectively. The problem is regarded as a problem whose instances are "promised" to come from $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$. Specifically, an algorithm A is said to *solve* a promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ if A accepts any instance $x \in \Pi_{\text{YES}}$ and rejects any instance $x \in \Pi_{\text{NO}}$; the behavior of A on any "unpromised" instance $x \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ can be arbitrary. The notion of promise problem is crucial for formalizing several important concepts and theorems in complexity theory. A canonical example is the unique satisfiability problem $(\mathsf{1SAT}, \mathsf{UNSAT})$, where $\mathsf{1SAT}$ is the set of formulas that has a unique satisfying assignment, and UNSAT is the set of unsatisfiable formulas. The promise problem is a standard Promise-UP-complete problem, and the celebrated theorem of Valiant and Vazirani [70] states that it is in fact NP-hard under randomized reductions. The reader is referred to the survey of Goldreich [23] for more background on promise problems.

Usually, it is required that Π_{YES} and Π_{NO} are disjoint, i.e., $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \varnothing$. The reason is that if there exists an instance $x \in \Pi_{\text{YES}} \cap \Pi_{\text{NO}}$, then no algorithm can solve the promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$. Indeed, if there were an algorithm A that solves $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$, then A must accept x and simultaneously reject x, which is impossible. For this reason, every definition of promise problems considered before is, to the best of our knowledge, always disjoint.

In this paper, we introduce a set of new promise problems which are *conjectured to be non-disjoint*. We will demonstrate that these "non-disjoint" promise problems are worth investigating, by showing that hardness results for our promise problems have important consequences in complexity theory. The fact that the promise problems are conjectured to be non-disjoint means that solving promise problems are conjectured to be *impossible*, no matter how long an algorithm is allowed to run. Surprisingly, our results show that if one can prove *mild hardness* results of computing "non-disjoint" promise problems, which is conjectured to be *impossible*, then one can resolve important open questions of complexity theory.

To be more specific, we consider open questions of whether there exists an explicit hitting set generator. A *hitting set generator* (HSG) $G = \{G_n \colon \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ secure against a class $\mathfrak{C}$ of algorithms is a family of functions G_n such that any algorithm $A \in \mathfrak{C}$ that accepts at least a half of the n-bit strings must accept a string $G_n(z)$ for some seed $z \in \{0,1\}^{s(n)}$ for all large $n \in \mathbb{N}$. The existence of a secure hitting set generator makes it possible to derandomize any one-sided-error $\mathfrak{C}$-randomized algorithm, by simply trying all possible $s(n)$-bit seeds z and using $G(z)$ as a source of randomness. A stronger notion called a *pseudorandom generator* (PRG) enables us to derandomize two-sided-error randomized algorithms.

1.1 Meta-Computational View of PRG Constructions

A standard approach for constructing pseudorandom generators is to use the *hardness versus randomness* framework developed in, e.g., [72, 8, 53, 7, 38, 40, 63, 44]. One of the landmark results of Impagliazzo and Wigderson [40] states that if there exists a function in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ that is not computable by a circuit of size $2^{\alpha n}$ for some constant $\alpha > 0$, then there exists a logarithmic-seed-length pseudorandom generator secure against linear-size circuits (and, in particular, $\mathsf{P} = \mathsf{BPP}$ follows). In general, such a result is proved by using a *black-box pseudorandom generator construction* $G^{(\cdot)}$ that converts any hard function $f \notin \mathsf{SIZE}(2^{o(n)})$ to a pseudorandom generator $G^f \colon \{0,1\}^{O(n)} \to \{0,1\}^{2^{\alpha n}}$ secure against circuits of size $2^{\alpha n}$, where $\alpha > 0$ is some constant.

The underlying theme of this paper is to view black-box PRG constructions from a *meta-computational* perspective. Usually, f is regarded as a *fixed* hard function such as $f \notin \mathsf{SIZE}(2^{o(n)})$. Instead, here we regard f as an *input* to some "non-disjoint" promise problem, and regard a black-box PRG construction $G^{(\cdot)}$ as a reduction that proves the security of some (universal) hitting set generator based on the hardness of the "non-disjoint" promise problem. This new perspective can be applied to arbitrary black-box PRG constructions, and it gives rise to a "non-disjoint" promise problem associated with the black-box PRG construction. For example, the pseudorandom generator construction of [40] induces the E vs $\mathsf{SIZE}(2^{o(n)})$ problem, which is the problem of distinguishing whether $f \in \mathsf{E}/O(n)$ or $f \notin \mathsf{SIZE}(2^{o(n)})$, given the truth table of a function f.

There are two types of a pseudorandom generator. One is a *cryptographic* PRG, which is computable in polynomial time in its seed length. This notion is useful for building secure cryptographic primitives. We present in Section 1.2 "non-disjoint" promise problems whose hardness gives rise to a cryptographic hitting set generator. In particular, finding a non-disjoint witness of the promise problem implies the average-case hardness of PH, which provides a new approach for establishing the equivalence between the worst-case and average-case complexity of PH. The other is a *complexity-theoretic* PRG, which is allowed to be computed in time exponential in its seed length. This notion is sufficient for the purpose of derandomizing randomized algorithms. In Section 1.3, we generalize the hardness versus randomness framework by using the meta-computational view of black-box PRG constructions, and establish the equivalence between circuit lower bounds for "non-disjoint" promise problems and the existence of hitting set generators. Sections 1.2 and 1.3 can be read independently.

1.2 Worst-Case versus Average-Case Complexity of PH

Understanding average-case complexity is a fundamental question in complexity theory. Average-case hardness of NP is a prerequisite for building secure cryptographic primitives, such as one-way functions and cryptographic pseudorandom generators. Indeed, it is not hard to see that if there exists a polynomial-time-computable hitting set generator G, then checking whether a given string is in the image of G is a problem in NP that is hard on average (in the errorless sense). In this section, we present a new approach for proving the average-case hardness of NP, by implicitly constructing a cryptographic hitting set generator.

A fundamental open question in the theory of average-case complexity, pioneered by [48], is to establish the equivalence between the worst-case and average-case complexity of NP.

▶ **Open Question 1.** *Does* $\mathsf{P} \neq \mathsf{NP}$ *imply* $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$*?*

Here $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ is an average-case analogue of $\mathsf{NP} \neq \mathsf{P}$. Open Question 1 asks whether the worst-case hardness of NP implies that NP is hard on random instances generated efficiently. The reader is referred to the survey of Bogdanov and Trevisan [9] for background on average-case complexity.

For large enough complexity classes such as PSPACE and EXP, there is a general technique for converting any worst-case hard function f to some two-sided-error average-case hard function $\mathrm{Enc}(f)$ based on error-correcting codes [63, 66]. Here, the encoded function $\mathrm{Enc}(f)$ is computable in PSPACE given oracle access to f; thus, the worst-case and average-case complexity of such large complexity classes are known to be equivalent. Viola [71] showed limits of such an approach: $\mathrm{Enc}(f)$ cannot be computed in the polynomial-time hierarchy PH^f; thus, the proof technique of using error-correcting codes is not sufficient to resolve Open Question 1 as well as the following weaker open question:

▶ **Open Question 2.** *Does* $\mathsf{PH} \neq \mathsf{P}$ *(or, equivalently,* $\mathsf{P} \neq \mathsf{NP}$*) imply* $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$*?*

Note that Open Question 2 is an easier question than Open Question 1, since $\mathsf{PH} = \mathsf{P}$ is known to be equivalent to $\mathsf{NP} = \mathsf{P}$.

There are significant obstacles for resolving Open Questions 1 and 2. One is the relativization barrier due to Impagliazzo [39]. Another is the limits of black-box reductions due to Feigenbaum and Fortnow [19] and Bogdanov and Trevisan [9].

Recently, a non-black-box worst-case to average-case reduction that is not subject to the latter barrier was presented in [31]. The reduction shows that solving the problem GapMINKT of approximating polynomial-time-bounded Kolmogorov complexity in the worst-case sense can be reduced to solving MINKT on average. For an integer $t \in \mathbb{N}$ and an oracle A, a *t-time-bounded A-oracle Kolmogorov complexity* $\mathrm{K}^{t,A}(x)$ of a finite string x is defined as the shortest length of a program that prints x in t steps with oracle access to A (see Section 2 for a precise definition). The promise problem $\mathrm{GapMINKT} = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ asks for approximating $\mathrm{K}^t(x)$ within an additive error of $\widetilde{O}(\sqrt{\mathrm{K}^t(x)})$, and is formally defined as follows: Π_{YES} consists of $(x, 1^s, 1^t)$ such that $\mathrm{K}^t(x) \leq s$; and Π_{NO} consists of $(x, 1^s, 1^t)$ such that $\mathrm{K}^{\mathsf{poly}(|x|,t)}(x) > s + \widetilde{O}(\sqrt{s})$.

The result of [31] can be seen as providing an approach for establishing the equivalence between worst-case and average-case complexity of NP; indeed, proving NP-hardness of GapMINKT is sufficient for resolving Open Question 1. However, the approximation error $\widetilde{O}(\sqrt{s})$ caused by the reduction of [31] is not optimal, which makes the question of proving NP-hardness of GapMINKT potentially harder.

1.2.1 Gap(K^A vs K)

We herein introduce the following promise problem.

▶ **Definition 3.** *For an oracle A and an approximation quality $\tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, the problem* $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ *is defined as the following promise problem* (Π_{YES}, Π_{NO})*.*

$$\Pi_{YES} := \{\, (x, 1^s, 1^t) \mid \mathrm{K}^{t,A}(x) \leq s \,\},$$
$$\Pi_{NO} := \{\, (x, 1^s, 1^t) \mid \mathrm{K}^{\tau(|x|,t)}(x) > s + \log \tau(|x|, t) \,\}.$$

By default, we assume that τ is some polynomial and write $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$ *if there exists some polynomial τ such that* $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$*.*

In this paper, we prove

▶ **Theorem 4.** *Let A be any oracle. If* $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$*, then* $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$*.*

An immediate corollary of Theorem 4 is an improvement of the reduction of [31], by setting $A := \varnothing$. In particular, in order to resolve Open Question 1, it suffices to prove NP-hardness of approximating $\mathrm{K}^t(x)$ within an additive error of $\log \tau(|x|, t)$ given $(x, 1^t)$ as input, for any polynomial τ. A key insight for reducing the approximation error is that there are two main sources of the approximation error in the reduction of [31]: One comes from fixing a random coin flip sequence, which we remove by using the pseudorandom generator construction of Buhrman, Fortnow, and Pavan [11] under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. The other comes from the advice complexity of a black-box pseudorandom generator construction, which we reduce by using a "k-wise direct product generator" whose advice complexity is small [32].

More surprisingly, the promise problem is conjectured to be *non-disjoint* for $A := \mathsf{SAT}$. That is, it is conjectured to be *impossible* for any algorithm to solve $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ – no matter how long the algorithm is allowed to run. Nevertheless, Theorem 4 shows that under the assumption that PH is easy on average, there exists a *polynomial-time algorithm* for solving $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$. Taking its contrapositive, this means that, in order to resolve $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$, it suffices to prove a super-polynomial time lower bound for solving $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$, whose time complexity is conjectured to be *infinity* (in the sense that there exists *no algorithm* that can compute the promise problem).

We now clarify why $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is conjectured to be non-disjoint. Under the plausible assumption that $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, it is not hard to see that there are infinitely many strings x such that x is simultaneously a YES and NO instance; here, the string x is defined as the truth table of the characteristic function of $L \in \mathsf{E}^{\mathsf{NP}} \setminus \mathsf{E}/O(n)$ (see Proposition 24).

Another corollary of Theorem 4 is that under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, any string x that can be compressed with SAT oracle in polynomial time can be also compressed without any oracle. Formally:

▶ **Corollary 5.** *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$*, then there exists a polynomial* τ *such that*

$$\mathrm{K}^{\tau(|x|,t)}(x) \leq \mathrm{K}^{t,\mathsf{SAT}}(x) + \log \tau(|x|,t)$$

for any $x \in \{0,1\}^*$ *and* $t \in \mathbb{N}$.

Proof. Under the assumption, $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ can be solved by *some algorithm*. Thus $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ problem must be disjoint, from which the result follows immediately. ◀

Corollary 5 provides a new approach for resolving Open Question 2. In order to prove $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$, it suffices to find a string x that can be compressed with SAT oracle but cannot be compressed without SAT oracle. In fact, it suffices to find such a string x under the stronger assumption that $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. This is because Pavan, Santhanam, and Vinodchandran [57] proved $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ if Open Question 2 is negative.

More importantly, Theorem 4 suggests a more reasonable approach to Open Question 2. Note that finding a string x compressible with SAT oracle but incompressible without any oracle corresponds to proving the non-disjointness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$; this amounts to proving the time complexity of solving $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is *infinity*. Theorem 4 suggests that it suffices to prove that a *polynomial-time* algorithm cannot find a difference between compressible strings under SAT oracle and incompressible strings without any oracle, under the worst-case hardness assumption of NP. In particular, it suffices to prove NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$.

▶ **Corollary 6** (A new approach for Open Question 2)**.** *Suppose that the* $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ *problem is "*NP*-hard under randomized reductions"*[1] *in the sense that*

$$\mathsf{NP} \not\subseteq \mathsf{BPP} \implies \mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K}) \notin \mathsf{P}.$$

Then, Open Question 2 is positive; that is,

$$\mathsf{DistPH} \not\subseteq \mathsf{AvgP} \iff \mathsf{PH} \neq \mathsf{P}.$$

[1] Here we use the weak notion of "NP-hardness" in order to strengthen the result. Corollary 6 remains true even if one interprets NP-hardness as a randomized reduction from NP.

In a typical proof of NP-hardness of a disjoint promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$, one needs to carefully design a reduction R from SAT to Π that "preserves" a structure of SAT; i.e., any formula $\varphi \in \mathsf{SAT}$ is mapped to $R(\varphi) \in \Pi_{\text{YES}}$ and any formula $\varphi \in \mathsf{UNSAT}$ is mapped to $R(\varphi) \in \Pi_{\text{NO}}$. The task of proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) is potentially much easier, because it suffices to find a reduction R that may not preserve a structure of SAT; in principle, $R(\varphi)$ can be a fixed input x that is in the intersection of YES and NO instances of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) . It is worth mentioning that proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) is easier than proving NP-hardness of GapMINKT since GapMINKT is reducible to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) via an identity map.

1.2.2 Non-NP-Hardness Results Do Not Apply

A line of work presented evidence that NP-hardness of MINKT is not likely to be established under deterministic reductions (e.g., [45, 51, 36, 35]). For example, it is not hard to see that the proof technique of Murray and Williams [51] (who proved a similar result for MCSP) can be extended to the case of GapMINKT.

▶ **Theorem 7** (Essentially in [51]; cf. [32])**.** *If* GapMINKT *is* NP*-hard under many-one deterministic reductions, then* EXP $\neq$ ZPP*.*

This result suggests that establishing NP-hardness of GapMINKT under deterministic reductions is a challenging task. In contrast, we observe that a similar "non-NP-hardness" result cannot be applied to a non-disjoint promise problem.

▶ **Proposition 8.** *Assume that* NP*-hardness of* $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) *under many-one reductions implies* EXP $\neq$ ZPP*. Then,* EXP $\neq$ ZPP *holds unconditionally.*

The reason is that $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) is well defined only if EXP $\neq$ ZPP. More formally:

Proof. There are two cases. Either $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) is disjoint or not disjoint. In the former case, by Proposition 24, we have $\mathsf{E}^{\mathsf{NP}} = \mathsf{E}$; thus $\mathsf{EXP} = \mathsf{EXP}^{\mathsf{NP}} = \mathsf{ZPEXP} \neq \mathsf{ZPP}$. In the latter case, there exists a string x that is simultaneously a YES and NO instance. A reduction that always maps to x defines a many-one reduction from any problem to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K); thus, EXP $\neq$ ZPP follows from the assumption. ◀

In light of Proposition 8, we leave as an interesting open question whether there is any barrier explaining the difficulty of proving NP-hardness of the non-disjoint promise problem. We mention that $\mathrm{GapMINKT}^{\mathsf{SAT}}$, which is equivalent to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K}^{\mathsf{SAT}})$, is known to be DistNP-hard [32]. In particular, since $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K}^{\mathsf{SAT}})$ is reducible to $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K) via an identity map, the latter is also DistNP-hard. Therefore, in order to present a barrier for proving NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs K), one must exploit a property that holds for NP but does not hold for DistNP (unless the notion of reducibility is strong).

1.2.3 $\mathrm{Gap}(F$ vs $F^{-1})$: Meta-Computational View of HILL's PRG

We also propose another approach towards Open Question 1, by introducing a promise problem which asks for distinguishing whether a given function is computable by small circuits, or cannot be inverted by small circuits. Specifically, for an approximation quality τ, we define the promise problem $\mathrm{Gap}_\tau(F$ vs $F^{-1})$ as follows. Given a size parameter $s \in \mathbb{N}$ and an integer $n \in \mathbb{N}$ and random access to a function $F\colon \{0,1\}^n \to \{0,1\}^n$, the task is to distinguish the following two cases:

Yes: F is computable by a circuit of size s.

No: F cannot be inverted on average by any F-oracle circuit of size $\tau(n, s)$.

We show that "NP-hardness" of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ for every polynomial τ is enough for resolving Open Question 1. More specifically, we prove

▶ **Theorem 9.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a polynomial* τ *and a* coRP*-type randomized algorithm that solves* $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ *on input* (n, s) *in time* $\mathsf{poly}(n, s)$*. In particular, Open Question 1 is true if* $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ *is "*NP*-hard" for every polynomial* τ *in the following sense:* $\mathsf{NP} \subseteq \mathsf{BPP}$ *follows from the assumption that* $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ *admits a* coRP*-type algorithm.*

This is proved by viewing the black-box PRG construction based on any one-way function, which is given by Håstad, Impagliazzo, Levin, and Luby [29], from the meta-computational perspective.

It is easy to observe that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is non-disjoint under the existence of a one-way function, which is one of the most standard cryptographic primitives. Thus, it is widely believed that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is *impossible* to solve. Nevertheless, NP-hardness of $\mathrm{Gap}(F \text{ vs } F^{-1})$ is sufficient for resolving Open Question 1.

1.3 Meta-computational Circuit Lower-bound Problems; MCLPs

We now turn our attention to complexity-theoretic hitting set generators. A standard approach for constructing complexity-theoretic pseudorandom generators is to use the hardness versus randomness framework, which reduces the task of constructing a pseudorandom generator to the task of finding an explicit hard function, such as $f \in \mathsf{E} \setminus \mathsf{SIZE}(2^{o(n)})$.

It is, however, a widely accepted fact that proving a circuit size lower bound for an explicit function is extremely hard. Here by an *explicit* function, we mean that a function is computable in $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$. It is an open question whether there exists an exponential-time-computable function $f \in \mathsf{E}$ that cannot be computed by any circuit of size $4n$ (cf. [20]). On the other hand, a simple counting argument shows that most functions $f\colon \{0,1\}^n \to \{0,1\}$ cannot be computed by circuits of size $2^{\alpha n}$ for any constant $\alpha < 1$.

Why is it so difficult to prove a circuit lower bound for an explicit function? We propose to view this question from a *meta-computational* perspective. The fact that it is difficult for *human beings* to show that an explicit function cannot be computed by small circuits suggests that it should be also difficult for *algorithms* to analyze a circuit lower bound. Our results indicate that if we can make this intuition formal, then we get breakthrough results in complexity theory.

Specifically, we herein introduce a family of new computational problems, which we call *Meta-computational Circuit Lower-bound Problems* (MCLPs). These problems ask for distinguishing the truth table of explicit functions from hard functions. For example, we propose the following promise problem:

The E vs $\mathsf{SIZE}(2^{o(n)})$ Problem (informal)

Given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$, distinguish whether $f \in \mathsf{E}/O(n)$ or $f \notin \mathsf{SIZE}(2^{o(n)})$.

Before defining the problem formally, let us first observe that the E vs $\mathsf{SIZE}(2^{o(n)})$ problem is closely related to the open question of whether $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$. Indeed, it is not hard to show that $\mathsf{E}/O(n) \not\subseteq \mathsf{SIZE}(2^{o(n)})$ if and only if $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$ by regarding an advice string as a part of input. Therefore, the E vs $\mathsf{SIZE}(2^{o(n)})$ problem is non-disjoint under the standard circuit lower bound assumption that $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$.

We now define the problem formally. According to the standard notion of advice [42], the complexity class $\mathsf{E}/O(n)$ is defined as a subset of functions $f\colon \{0,1\}^* \to \{0,1\}$ that are defined on all the strings of any length. Thus, "$f \in \mathsf{E}/O(n)$" does not make sense for a

function $f\colon \{0,1\}^n \to \{0,1\}$. Instead, we interpret advice by using the notion of Levin's resource-bounded Kolmogorov complexity [47] so that the notion of advice is meaningful for a finite function $f\colon \{0,1\}^n \to \{0,1\}$. For a string $x \in \{0,1\}^*$, let $\mathrm{Kt}(x)$ denote the Kt-complexity of a string x, which is defined as the minimum of $|M| + \log t$ over all the programs M that output x in time t; here, $|M|$ denotes the description length of M. The E vs $\mathsf{SIZE}(2^{o(n)})$ problem is formally defined as follows.

▶ **Definition 10.** *For any functions $t, s\colon \mathbb{N} \to \mathbb{N}$, let $(\Pi_{YES}(t(n)), \Pi_{NO}(s(n)))$ denote the promise problem defined as*

$$\Pi^t_{YES} := \{\, f \in \{0,1\}^{2^n} \mid \mathrm{Kt}(f) \le \log t(n), n \in \mathbb{N} \,\},$$
$$\Pi^s_{NO} := \{\, f \in \{0,1\}^{2^n} \mid \mathrm{size}(f) > s(n), n \in \mathbb{N} \,\}.$$

Here, we identity a function $f\colon \{0,1\}^n \to \{0,1\}$ with its truth table representation $f \in \{0,1\}^{2^n}$, and $\mathrm{size}(f)$ denotes the minimum size of a circuit that computes f.

The E vs $\mathsf{SIZE}(2^{o(n)})$ problem is defined as the family $\{(\Pi_{YES}(2^{cn}), \Pi_{NO}(2^{\alpha n}))\}_{c,\alpha>0}$ of the promise problems. A family $\{\Pi\}$ of problems is said to be solved *by a class $\mathfrak{C}$ and denoted by $\{\Pi\} \in \mathfrak{C}$ if every problem in the family is solved by some algorithm in $\mathfrak{C}$.*

The idea behind Definition 10 is that the complexity class $\mathsf{E}/O(n)$ can be characterized as the class of the functions $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ such that, for some constant c, for all large $n \in \mathbb{N}$, $\mathrm{Kt}(f_n) \le cn$ holds. Indeed, $f \in \mathsf{E}/O(n)$ means that the truth table of f_n can be described by a Turing machine of description length $O(n)$ in time $2^{O(n)}$ for all large n. The relationship between complexity classes with advice and resource-bounded Kolmogorov complexity will be explained in detail in Section 4.2, where we interpret "$\mathsf{DTIME}(t(n))/a(n)$" as a subset of functions $f\colon \{0,1\}^n \to \{0,1\}$.

1.3.1 Meta-Computational View of the Hardness vs Randomness Framework

We show that a nearly-linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit size lower bound for solving the E vs $\mathsf{SIZE}(2^{o(n)})$ problem exactly characterizes the existence of a hitting set generator secure against $\mathsf{AC}^0 \circ \mathsf{XOR}$.

▶ **Theorem 11.** *The following (Items 1 to 4) are equivalent.*

1. *There exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuits.*
2. *For all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size $N^{1+o(1)}$ that computes the E vs $\mathsf{SIZE}(2^{o(n)})$ problem, where $N = 2^n$ denotes the input length.*

The condition can be equivalently stated without referring to the "non-disjoint" promise problem. Let $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ denote the family of the promise problems $\mathrm{MKtP}[c \log N, N^\alpha]$ for constants $c, \alpha > 0$, where, for functions $s, t\colon \mathbb{N} \to \mathbb{N}$, $\mathrm{MKtP}[s(N), t(N)]$ denotes the promise problem of distinguishing strings x such that $\mathrm{Kt}(x) \le s(|x|)$ and strings x such that $\mathrm{Kt}(x) > t(|x|)$. Then, the following are equivalent as well.

3. *For all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size $N^{1+o(1)}$ that computes $\mathrm{MKtP}[O(\log N), N^{o(1)}]$.*
4. *For any constant $k \in \mathbb{N}$, for all large $N \in \mathbb{N}$, there exists no $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit of size N^k that computes $\mathrm{MKtP}[O(\log N), N^{o(1)}]$.*

Observe that Item 1 of Theorem 11 implies a strongly exponential AC^0 circuit lower bound for E, which also implies that $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$ (see, e.g., [3, 55, 26]). These are long-standing open questions with the state of the art being Håstad's lower bounds [28]. Theorem 11

shows that, in order to improve the state-of-the-art lower bound, it is sufficient to prove a *nearly-linear* $\mathsf{AC}^0 \circ \mathsf{XOR}$ lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. In contrast, the minimum circuit for computing the E vs $\mathsf{SIZE}(2^{o(n)})$ problem is *infinity* under the standard circuit lower bound assumption that $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$.

It is instructive to compare our results with the hardness versus randomness framework. In order to obtain a hitting set generator in the latter framework, we need to find an explicit function that is hard for small circuits to compute. In our framework, finding an explicit hard function corresponds to proving that the minimum circuit size for computing MCLPs is *infinity* (or, in other words, proving that there exists no circuit of *any size* that computes MCLPs[2]). Our results significantly weaken the assumption needed to obtain a hitting set generator: It suffices to show that a nearly-linear circuit cannot find the difference between an explicit function and a hard function.

Our results can be also stated based on the case analysis. There are two cases. (1) When the circuit lower bound that $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$ holds, the work of [40] already implies the existence of a pseudorandom generator. (2) Even if the circuit lower bound does fail, Theorem 11 shows that a very modest lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ problem (which is a disjoint promise problem under the assumption) implies the existence of a hitting set generator. In either case, we obtain a hitting set generator. Our results generalize the hardness versus randomness framework in this sense.

Previously, based on the hardness versus randomness framework, it is known that $\mathsf{E} \not\subseteq \mathfrak{C}$ is equivalent to the existence of a pseudorandom generator secure against $\mathfrak{C}$ for a sufficiently large class $\mathfrak{C}$ (see, e.g., [22]). However, in the previous approach, one needs to transform a worst-case $\mathfrak{C}$-circuit lower bound to an average-case $\mathfrak{C}$-circuit lower bound; thus $\mathfrak{C}$ needs to be a sufficiently large so that it can perform local decoding, which requires the majority gate [61]. For any circuit class $\mathfrak{C}$ smaller than TC^0, it was not clear whether the existence of a hitting set generator secure against $\mathfrak{C}$ is equivalent to some worst-case $\mathfrak{C}$-circuit lower bound. Theorem 11 establishes the first equivalence for the circuit class $\mathfrak{C} = \mathsf{AC}^0 \circ \mathsf{XOR}$, which is smaller than TC^0 [59].

Our results can be stated without the non-standard notion of promise problem, as in Items 3 and 4 of Theorem 11. Indeed, any promise problem in the family $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ asks for approximating the Kt-complexity of a given string, and it is always a disjoint promise problem. In our terminology, $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ is equivalent to the E vs $\mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$ problem. Since $\mathsf{SIZE}(2^{o(n)}) \subseteq \mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$, one can observe that the E vs $\mathsf{DTIME}(2^{2^{o(n)}})/2^{o(n)}$ problem is reducible to the E vs $\mathsf{SIZE}(2^{o(n)})$ problem via an identity map, which explains the implication from Item 3 to Item 2 in Theorem 11.

We mention in passing that it is not hard to prove an AC^0 lower bound for $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ (i.e., without the bottom XOR gates) by using the pseudorandom restriction method as in [34, 17, 14]. (See Appendix B for a proof.)

▶ **Proposition 12.** *For any constants $\alpha < 1, k, d \in \mathbb{N}$, there exists a constant c such that*

$$\mathrm{MKtP}[c \log N, N^{\alpha}] \notin \mathsf{i.o.AC}^0_d(N^k).$$

[2] This should be compared with the fact that any *disjoint* promise problem can be computed by a circuit of size $O(2^n/n)$ on inputs of length n.

For any classes $\mathfrak{C}, \mathfrak{D}$ of functions, one can define the $\mathfrak{C}$ vs $\mathfrak{D}$ problem. A particularly interesting problem is the E vs $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ problem, where $\widetilde{\mathfrak{D}}(s; \delta)$ denotes the class of functions that can be computed by a $\mathfrak{D}$-circuit of size s on at least a $(1-\delta)$ fraction of inputs. We prove that, if nearly-linear-size AC^0 circuits cannot distinguish an explicit function from a function that cannot be approximated by small AC^0 circuits, then a logarithmic-seed-length hitting set generator can be obtained. (Moreover, the converse direction is easy to prove.)

▶ **Theorem 13.** *The following are equivalent.*

1. *The* E vs $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ *problem cannot be computed by* AC^0 *circuits of size* $N^{1+o(1)}$ *for all large* $N = 2^n$.
2. *There exists a hitting set generator* $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ *computable in time* $n^{O(1)}$ *and secure against linear-size* AC^0 *circuits.*
3. $\mathrm{MKtP}[O(\log N), N-1] \notin \mathsf{i.o.AC}^0(N^{1+\beta})$ *for some constant* $\beta > 0$.
4. $\mathrm{MKtP}[O(\log N), N-1] \notin \mathsf{i.o.AC}^0(N^k)$ *for any constant* k.

An interesting aspect of Theorem 13 is its self-referential nature; intuitively, Item 1 means that AC^0 circuits cannot analyze AC^0 circuits itself. Note that self-reference is crucial for proving, e.g., time hierarchy theorems for uniform computational models. Theorem 13 provides an analogue in a non-uniform circuit model.

Why do we consider "non-disjoint" promise problems, despite the fact that Theorems 11 and 13 can be stated by using only the standard notions?[3] First, Theorem 11 is obtained by viewing (a variant of) the black-box PRG construction of Impagliazzo and Wigderson [40] from a meta-computational perspective; thus, it is natural to state Theorem 11 as a connection between the existence of a hitting set generator and a lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. Second, an identity map reduces $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ to the E vs $\mathsf{SIZE}(2^{o(n)})$ problem, and thus it is easier to prove a lower bound for the latter problem. Third, the known worst-case-and-average-case equivalence between $\mathsf{E} \subseteq \mathsf{SIZE}(2^{o(n)})$ and $\mathsf{E} \subseteq \widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ [63] can be naturally regarded as a reduction from the E vs $\mathsf{SIZE}(2^{o(n)})$ problem to the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ problem. Indeed, Theorem 13 is proved by viewing the Nisan–Wigderson pseudorandom generator from a meta-computational perspective, and then Theorem 13 is translated into Theorem 11 by using the worst-case-and-average-case equivalence.

We also present a potential approach for resolving the $\mathsf{RL} = \mathsf{L}$ question. Here, RL is the complexity class of languages that can be solved by a one-sided-error randomized $O(\log n)$-space Turing machine that reads its random tape *only once*. A canonical approach for proving $\mathsf{RL} = \mathsf{L}$ is to construct a log-space-computable hitting set generator of seed length $O(\log n)$ secure against $O(n)$-size read-once branching programs. A state-of-the-art result is the pseudorandom generator of seed length $O(\log^2 n)$ given by Nisan [52] for read-once (known-order) oblivious branching programs, and the pseudorandom generator of seed length $O(\log^3 n)$ given by Forbes and Kelley [21] for read-once unknown-order oblivious branching programs.[4]

[3] We also mention that the non-disjointness itself can provide new consequences, such as Corollaries 5 and 16.

[4] In the area of unconditional derandomization of space-bounded randomized algorithms, it is common to assume that a branching program is oblivious and reads the input in the fixed order. Here, we do not assume these properties.

We show that a hitting set generator of seed length $O(\log n)$ can be constructed if nearly-linear-size read-once co-nondeterministic branching programs cannot distinguish linear-space-computable functions from hard functions.

▶**Theorem 62.** *Suppose that there exist some constants $\alpha, \beta > 0$ such that the* $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{O(\alpha n)}; 2^{-\alpha n})$ *problem cannot be computed by read-once co-nondeterministic branching programs of size $N^{1+\beta}$ for all large input lengths $N = 2^n \in \mathbb{N}$. Then there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in $O(\log n)$ space and secure against linear-size read-once branching programs (and, in particular,* $\mathsf{RL} = \mathsf{L}$ *follows).*

Theorem 62 can be compared with the result of Klivans and van Melkebeek [44], which shows the existence of a pseudorandom generator secure against branching programs under the assumption that $\mathsf{DSPACE}(n)$ requires a circuit of size $2^{\Omega(n)}$. Under the same assumption, by using a worst-case-to-average-case reduction for $\mathsf{DSPACE}(n)$, it can be shown that the $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)$ problem is non-disjoint for any sufficiently small $\delta \geq 0$ (cf. Proposition 43). In this case, the minimum size of a co-nondeterministic branching program for computing the MCLP is *infinity*; thus, Theorem 62 generalizes the result of [44].

It should be noted that limits of the computational power of read-once non-deterministic branching programs are well understood. For example, Borodin, Razborov, and Smolensky [10] presented an explicit function that cannot be computed by any read-k-times nondeterministic branching program of size $2^{o(n)}$ for any constant k. Theorem 62 shows that, in order to resolve $\mathsf{RL} = \mathsf{L}$, it suffices to similarly analyze the read-once co-nondeterministic branching program size lower bound for computing the MCLP. This approach could be useful; by using the Nechiporuck method, it can be shown that neither nondeterministic nor co-nondeterministic branching programs of size $o(N^{1.5}/\log N)$ can compute MKtP [16], which is a much more general lower bound than read-k-times nondeterministic branching programs.

We also mention that a partial converse of Theorem 62 is easy to prove: If there exists a log-space-computable hitting set generator secure against linear-size read-once nondeterministic branching programs, then the $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)$ problem cannot be computed by a read-once co-nondeterministic branching programs of size N^k, where $N = 2^n$ and k is an arbitrary constant. More generally, any results showing the existence of a hitting set generator secure against $\mathfrak{C}$ must entail a $\mathsf{co}\mathfrak{C}$-lower bound for MCLPs (cf. Proposition 54).

1.3.2 Non-trivial Derandomization and Lower Bounds for MKtP

Our proof techniques can be also applied to *uniform* algorithms. We consider the question of whether one-sided-error uniform algorithms can be non-trivially derandomized in time $2^{n-\omega(\sqrt{n}\log n)}$. We say that an algorithm A is a *derandomization algorithm for* $\mathsf{DTIME}(t(n))$ if, for any machine M running in time $t(n)$, A takes 1^n and a description of M as input and outputs $y \in \{0,1\}^n$ such that $M(y) = 1$ for infinitely many $n \in \mathbb{N}$ such that $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$. Unlike the standard notion of derandomization algorithm for non-uniform computational models, the description length of M is at most a constant; thus, our notion of derandomization algorithm is essentially equivalent to the existence of a hitting set generator secure against $\mathsf{DTIME}(t(n))$. Applying our proof techniques to this setting, we establish the following equivalence between the existence of a derandomization algorithm for uniform algorithms and a lower bound for approximating Kt complexity.

▶ **Theorem 14.** *For any constant $0 < \epsilon < 1$, the following are equivalent:*

1. *There exists a derandomization algorithm for* $\mathsf{DTIME}(2^{O(\sqrt{N}\log N)})$ *that runs in time* $2^{N-\omega(\sqrt{N}\log N)}$.
2. $\mathrm{MKtP}[N - \omega(\sqrt{N}\log N), N-1] \notin \mathsf{DTIME}(2^{O(\sqrt{N}\log N)})$.
3. $\mathrm{MKtP}[N^\epsilon, N^\epsilon + \omega(\sqrt{N^\epsilon}\log N)] \notin \mathsf{DTIME}(2^{O(\sqrt{N^\epsilon}\log N)})$.

Usually, the time complexity is measured with respect to the input size. Our result, however, suggests that the time complexity of $\mathrm{MKtP}[s(N), s(N) + \widetilde{\omega}(\sqrt{N})]$ is well captured by the size parameter $s(N)$ rather than the input size N: Indeed, Theorem 14 implies that

$$\mathrm{MKtP}[N^\epsilon, N^\epsilon + \omega(\sqrt{N^\epsilon} \log N)] \in \mathsf{DTIME}(2^{O(\sqrt{N^\epsilon} \log N)})$$

is equivalent to

$$\mathrm{MKtP}[N^\delta, N^\delta + \omega(\sqrt{N^\delta} \log N)] \in \mathsf{DTIME}(2^{O(\sqrt{N^\delta} \log N)})$$

for any $0 < \epsilon, \delta < 1$.

Theorem 14 highlights the importance of a lower bound for MKtP. In fact, it is a long-standing open question whether $\mathrm{MKtP} \notin \mathsf{P}$, despite the fact that MKtP is an EXP-complete problem under non-uniform reductions [2]. Towards resolving the open question, we show that some promise problem can be solved in coRP under the assumption that $\mathrm{MKtP} \in \mathsf{P}$.

▶ **Theorem 15.** *Assume that* $\mathrm{MKtP} \in \mathsf{P}$. *Then, there exists a* coRP*-algorithm that solves the* Kt vs K^t *problem, which is defined as follows: Given a string* $x \in \{0,1\}^*$ *of length* n *and a parameter* $s \in \mathbb{N}$*, distinguish whether* $\mathrm{Kt}(x) \leq s$ *or* $\mathrm{K}^t(x) \geq s + O(\sqrt{s} \log n + \log^2 n)$*, where* $t = \mathsf{poly}(n)$.

Using the disjointness of the Kt vs K^t problem and setting $s := \mathrm{Kt}(x)$, we obtain

▶ **Corollary 16.** *If* $\mathrm{MKtP} \in \mathsf{P}$*, then* $\mathrm{K}^t(x) \leq \mathrm{Kt}(x) + O(\sqrt{\mathrm{Kt}(x)} \log n + \log^2 n)$ *for any* $x \in \{0,1\}^n$ *and any* $t \geq \mathsf{poly}(n)$.

Since $\mathrm{Kt}(x) \leq \mathrm{K}^{\mathsf{poly}(n)}(x) + O(\log n)$ holds unconditionally, Corollary 16 shows that $\mathrm{Kt}(x)$ and $\mathrm{K}^{\mathsf{poly}(n)}(x)$ are close to each other under the assumption that $\mathrm{MKtP} \in \mathsf{P}$. We mention that the problem of computing $\mathrm{K}^{t(n)}(x)$ given $x \in \{0,1\}^n$ cannot be computed in polynomial time when $t(n) = n^{\omega(1)}$ [32].

1.3.3 Related Work: Minimum Circuit Size Problem

The definitions of MCLPs are inspired by the *Minimum Circuit Size Problem* (MCSP). While the history of MCSP is said to date back to as early as 1950s [64], its importance was not widely recognized until Kabanets and Cai [41] named the problem as MCSP and investigated it based on the natural proof framework of Razborov and Rudich [60]. The task of MCSP is to decide whether there exists a size-s circuit that computes f, given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$ and a size parameter $s \in \mathbb{N}$. It turned out that MCSP is one of the central computational problems in relation to wide research areas of complexity theory, including circuit lower bounds [60], learning theory [13], and cryptography [60, 2, 4, 31].

Over the last twenty years of the study of MCSP, it has been recognized that MCSP lacks one desirable mathematical property – *monotonicity* with respect to an underlying computational model. MCSP can be defined for any circuit classes $\mathfrak{C}$; for example, $\mathfrak{C}$-MCSP stands for a version of MCSP where the task is to find the minimum $\mathfrak{C}$-circuit size; MCSP^A stands for the minimum A-oracle circuit size problem. We are tempted to conjecture that, as a computational model becomes stronger, the corresponding minimization problem becomes harder; e.g., MCSP^A should be harder than MCSP for any oracle A. However, this is not the case – Hirahara and Watanabe [35] showed that there exists an oracle A such that $\mathrm{MCSP} \not\leq_T^p \mathrm{MCSP}^A$ unless $\mathrm{MCSP} \in \mathsf{P}$. Moreover, DNF-MCSP [49, 3] and $(\mathrm{DNF} \circ \mathrm{XOR})$-MCSP [33] are known to be NP-complete, whereas NP-completeness of MCSP is a long-standing open question.

Why does the monotonicity of MCSP fail? $\mathfrak{C}$-MCSP can be regarded as a special case of the $\mathfrak{C}$ vs $\mathfrak{D}$ problem where $\mathfrak{C} = \mathfrak{D}$. It is easy to observe that the $\mathfrak{C}$ vs $\mathfrak{D}$ problem is reducible to the $\mathfrak{C}'$ vs $\mathfrak{D}'$ problem via an identity map if $\mathfrak{C} \subseteq \mathfrak{C}'$ and $\mathfrak{D} \supseteq \mathfrak{D}'$; thus, MCLPs have monotonicity properties in this sense. In contrast, the monotonicity property of MCLPs fails when $\mathfrak{C} = \mathfrak{D} \subseteq \mathfrak{C}' = \mathfrak{D}'$, which corresponds to the case of MCSP.

In an attempt to remedy the monotonicity issue, Hirahara and Santhanam [34] observed that average-case complexity of MCSP is monotone increasing. Carmosino, Impagliazzo, Kabanets, and Kolokolova [13] implicitly showed that the complexity of MCSP is monotone increasing under non-black-box reductions.

In contrast, MCLPs incorporate the monotonicity property in the definition itself, which makes a mathematical theory cleaner. For example, recall that it can be shown that $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$ if and only if $\mathsf{E} \not\subseteq \widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ by using error-correcting codes [63]. Viewing this equivalence from a meta-computational perspective, it can be interpreted as an efficient reduction from the E vs $\mathsf{SIZE}(2^{o(n)})$ problem to the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ problem (cf. Theorem 59). A similar reduction was proved for MCSP under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$ [14]; finding such a reduction for MCSP requires certain creativity, whereas working with the definition of MCLPs makes it trivial to find the reduction.

1.3.4 Related Work: Hardness Magnification

A recent line of work [55, 54, 50, 15, 14] exhibit surprising phenomena, which are termed as "hardness magnification phenomena." Oliveira, Santhanam, and Pich [55, 54] showed that very weak lower bounds for MCSP and related problems are sufficient for resolving long-standing open questions about circuit lower bounds, such as $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$. A surprising aspect of hardness magnification phenomena is that, as argued in [6, 55], the argument does not seem to be subject to the natural proof barrier of Razborov and Rudich [60], which is one of the major obstacles of complexity theory. Our results provide an interpretation of hardness magnification phenomena from the perspective of a construction of a hitting set generator.

It is worthwhile to point out that our reductions have very similar structures to hardness magnification phenomena. For example, it was shown in [55, 54] that, for a parameter $s = s(N)$, the problem $\mathrm{MKtP}[s, s + O(\log N)]$ can be solved by an $\mathsf{AND}_{O(N)} \circ D_{\mathsf{poly}(s)} \circ \mathsf{XOR}$ circuit, where $D_{\mathsf{poly}(s)}$ is an oracle gate computable in EXP that takes an input of length $\mathsf{poly}(s)$. This reduction shows that $\mathsf{EXP} \subseteq \mathsf{P/poly}$ implies $\mathrm{MKtP}[s, s + O(\log N)] \in \mathsf{SIZE}(N \cdot \mathsf{poly}(s))$. Taking its contrapositive, it can be interpreted as a hardness magnification phenomenon: for $s(N) \ll N$, a (seemingly) weak lower bound $\mathrm{MKtP}[s, s + O(\log N)] \notin \mathsf{SIZE}(N \cdot \mathsf{poly}(s))$ can be "hardness magnified" to a strong circuit lower bound $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.

Theorem 61 has exactly the same structure with the reduction mentioned above. Given a circuit D that avoids a hitting set generator, we construct a nearly-linear-size $\mathsf{AND}_{O(N)} \circ D \circ \mathsf{XOR}$ circuit that computes the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. Thus, our reductions are as efficient as the reductions presented in the line of work of hardness magnification phenomena.

More importantly, our results significantly strengthen the consequences of hardness magnification: Not only circuit lower bounds, but also hitting set generators can be obtained. This is especially significant for the case of read-once branching programs. Since there is already an exponential size lower bound for read-once branching programs [10], it does not make sense to try to hardness-magnify a lower bound for read-once branching programs. In contrast, our results (Theorem 62) indicate that a nearly-linear lower bound for co-nondeterministic read-once branching programs is enough for resolving $\mathsf{RL} = \mathsf{L}$.

An intriguing question about hardness magnification is this: By using hardness magnification phenomena, can we prove any new consequences, such as circuit lower bounds or derandomization? Theorem 62 adds a new computational model, i.e., co-nondeterministic read-once branching programs, for exploring this question.

Chen, Hirahara, Oliveira, Pich, Rajgopal, and Santhanam [14] proposed a barrier for the question, termed as a "locality barrier." Briefly speaking, the idea there is to regard hardness magnification phenomena as a (black-box) reduction to oracles with small fan-in, and then to show that most circuit lower bound proofs can be extended to rule out such a reduction; thus, such a circuit lower bound proof technique cannot be combined with hardness magnification phenomena. A salient feature of our reductions is that our reductions are *non-black-box* in the sense that we exploit the efficiency of oracles; the non-black-box property appears in the definition of NO instances of the $\mathfrak{C}$ vs $\mathfrak{D}$ problem. Therefore, our results provide a potential approach for bypassing the locality barrier: Try to develop a circuit lower bound proof technique that crucially exploits the structure of the NO instances of the $\mathfrak{C}$ vs $\mathfrak{D}$ problem. The existing circuit lower bound proof techniques for MCSP and related problems fail to exploit such a structure.

1.4 Proof Techniques: Meta-Computational View of PRG Constructions

All of our results are given by a *single principle* – that views any black-box pseudorandom generator construction from a meta-computational perspective. The differences among our theorems simply originate from the fact that we use a different black-box pseudorandom generator construction. The underlying principle is this:

> Any black-box construction of a pseudorandom generator G^f based on a hard function $f \notin \mathcal{R}$ gives rise to a non-black-box security reduction for a hitting set generator based on the hardness of a non-disjoint promise problem (e.g., the E vs $\mathcal{R}$ problem).

For the purpose of exposition, we take a specific PRG construction of Impagliazzo and Wigderson [40], and explain the connection between the PRG construction and the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. The theorem of [40] states that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{o(n)})$ implies the existence of a pseudorandom generator. The PRG construction is a *black-box pseudorandom generator construction* G^f based on a hard function $f \notin \mathsf{SIZE}(2^{o(n)})$ in the following sense.

Explicitness. $G^f(z)$ is computable in polynomial time given the truth table of f and a seed z.

Security. If there exists a function D that distinguishes the output distribution of $G^f(\text{-})$ from the uniform distribution, then $f \in \mathsf{SIZE}^D(2^{o(n)})$.

Here, by "black-box", we mean that the security of the PRG is proved by a (black-box) reduction, i.e., the security reduction works for *every* function D. In contrast, we say that a reduction is non-black-box if the reduction may not be correct when an oracle is inefficient. This is in the same spirit with the non-black-box reduction of [31], which overcomes the black-box reduction barrier of Bogdanov and Trevisan [9]. We explain below how a black-box PRG construction gives rise to a *non-black-box* security reduction of a hitting set generator.

The goal is to construct some secure hitting set generator $H = \{H_m \colon \{0,1\}^{O(\log m)} \to \{0,1\}^m\}_{m \in \mathbb{N}}$. As a choice of H, we simply take a "universal" hitting set generator: Let U be a universal Turing machine, i.e., a machine that simulates every Turing machine efficiently. Then we define $H_m(z)$ to be the output of U on input z if U halts in $2^{|z|}$ steps, where $z \in \{0,1\}^{O(\log m)}$. The choice of H is universal, in the sense that the existence of some exponential-time computable HSG implies that H is also secure.

The strategy for proving the security of a hitting set generator H is to regard f as an input of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem, and to view the black-box PRG construction G^f as a (non-black-box) reduction from the E vs $\mathsf{SIZE}(2^{o(n)})$ problem to the task of avoiding H.

We claim that H is a hitting set generator secure against linear-size circuits, under the assumption that the E vs $\mathsf{SIZE}(2^{o(n)})$ problem cannot be solved by small circuits. To this end, we present a reduction from the task of solving the E vs $\mathsf{SIZE}(2^{o(n)})$ problem to the task of "avoiding" H. That is, if H is not secure, then there exists a linear-size circuit D that *avoids* H, i.e., every image of H is rejected by D whereas D accepts at least a half of all inputs. A randomized reduction R for solving the E vs $\mathsf{SIZE}(2^{o(n)})$ problem is extremely simple:

A randomized algorithm R for solving the E vs $\mathsf{SIZE}(2^{o(n)})$ problem with D oracle

Given f as an input, pick a random seed z of G^f, and accept if and only if $D(G^f(z)) = 0$.

The correctness of the reduction R can be proved as follows. Assume that f is a YES instance of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem; in other words, this means that $\mathrm{Kt}(f) \leq O(\log |f|) = O(n)$. Since G^f is efficiently computable, it follows that $\mathrm{Kt}(G^f(z)) \leq O(n)$ for every seed $z \in \{0,1\}^{O(n)}$. By using the property of the universal hitting set generator H_m and choosing m large enough, it can be observed that $G^f(z)$ is in the image of H_m; thus $G^f(z)$ is rejected by D.

Conversely, we prove that any NO instance of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem is rejected by the algorithm R with high probability. Intuitively, this is because of the fact that if f is a hard function, then D cannot distinguish $G^f(\text{-})$ from the uniform distribution. More formally, we claim the contrapositive. Assume that D rejects $G^f(z)$ with high probability, say, at least $\frac{3}{4}$. This means that

$$\Pr_z[D(G^f(z)) = 1] \leq \frac{1}{4}.$$

On the other hand, since D avoids H, we have $\Pr_w[D(w) = 1] \geq \frac{1}{2}$. Therefore, D distinguishes the distribution of G^f from the uniform distribution with advantage $\frac{1}{4}$; by the black-box security proof of G^f, we obtain that $f \in \mathsf{SIZE}^D(2^{o(n)})$. Since D is a linear-size circuit, we conclude that $f \in \mathsf{SIZE}(2^{o(n)})$, which means that f is not a NO instance of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. Note here that we rely on the efficiency of D, which makes the security proof of the HSG H non-black-box.

We conclude that there exists a randomized circuit for computing the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. By using a standard trick of Adleman [1], the randomness of the circuit can be fixed, and obtain a deterministic circuit that computes the E vs $\mathsf{SIZE}(2^{o(n)})$ problem.

Note that the proof above shows a generic connection between a black-box pseudorandom generator construction and a "non-disjoint" promise problem. The efficiency of the security reduction depends on the choice of a black-box PRG construction. In the rest of this paper, we present some of the instantiations of the proof ideas above based on several specific constructions of PRGs; however, we emphasize that our reductions are not limited to those specific instantiations, and new black-box PRG constructions can lead to a more efficient reduction and a "non-disjoint" promise problem that is easy to analyze.

1.5 Perspective: Meta-Computational View of Complexity Theory

More broadly, we propose to view complexity theory from a meta-computational perspective.

In order to explain the view, it is helpful to regard an algorithm that tries to solve MCLPs as a malicious prover that tries to falsify a circuit lower bound. To be more specific, consider the E vs $\mathsf{SIZE}(2^{o(n)})$ problem. As we explained earlier, the existence of any algorithm (of

any complexity) that solves the E vs $\mathsf{SIZE}(2^{o(n)})$ problem implies that $\mathsf{E} \subseteq \mathsf{SIZE}(2^{o(n)})$, and moreover the converse direction is also true. In this sense, any algorithm that solves an MCLP can be regarded as an adversary that falsifies circuit lower bounds such as $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$.

Complexity theory can be regarded as a game between us (i.e., complexity theorists, who try to prove circuit lower bounds) and provers (i.e., algorithms that solve MCLPs). We lose the game if some prover can solve MCLPs (and hence circuit lower bounds fail). We win the game if we find an explicit function whose circuit complexity is high. This is equivalent to finding a witness for the non-disjointness of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem, and thus it is equivalent to showing that there exists no prover that can solve the MCLP. In other words, prior to our work, we implicitly tried to fight against *every prover without any restriction on efficiency*.

What we showed in this work is that we do not have to fight against every non-efficient prover. Instead, in order to obtain a circuit lower bound (which is implied by the existence of a hitting set generator), it suffices to show that no *efficient* algorithms such as nearly-linear-size $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuits can find the difference between explicit functions and hard functions. In principle, it should be easier to prove a nearly-linear circuit size lower bound for some problem when we believe that the problem does not admit any algorithm (because of the non-disjointness). While we have not found any existing method for proving such a lower bound that is sufficient for breakthrough results, we believe that this is simply because of the fact that MCLPs were not investigated explicitly. We leave as an important open question to develop a proof technique to analyze MCLPs.

2 Preliminaries

2.1 Notation

For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we denote by $\mathtt{tt}(f)$ the truth table of f, i.e., the concatenation of $f(x)$ for all $x \in \{0,1\}^n$ in the lexicographical order. Conversely, for a string $y \in \{0,1\}^N$, the function $\mathtt{fn}(y)\colon \{0,1\}^{\lceil \log N \rceil} \to \{0,1\}$ is defined as $\mathtt{fn}(y)(i) :=$ (the ith bit of y) if $i \leq N$ and $\mathtt{fn}(y)(i) := 0$ otherwise, where $i \in [2^{\lceil \log N \rceil}]$ is identified with a binary representation in $\{0,1\}^{\lceil \log N \rceil}$. For a string $x \in \{0,1\}^*$, we denote by $\mathrm{size}(x)$ the minimum circuit size for computing the Boolean function $\mathtt{fn}(x)\colon \{0,1\}^{\lceil \log |x| \rceil} \to \{0,1\}$. We often identify a string x with its function version $\mathtt{fn}(x)$. For a parameter δ, we denote by $\widetilde{\mathrm{size}}(x;\delta)$ the minimum size of a circuit C such that $\mathtt{fn}(x)(y) = C(y)$ on at least a $(1-\delta)$ fraction of inputs y.

For a function $f\colon \{0,1\}^n \to \{0,1\}$ and an integer $k \in \mathbb{N}$, we denote by $f^k\colon (\{0,1\}^n)^k \to \{0,1\}^k$ the direct product of f. We denote by $f^{\oplus k}\colon (\{0,1\}^n)^k \to \{0,1\}$ the function $\oplus_k \circ f^k$, where $\oplus_k$ is the parity function on k-bit inputs.

2.2 Pseudorandomness

Let $G\colon \{0,1\}^d \to \{0,1\}^m$ and $D\colon \{0,1\}^m \to \{0,1\}$ be functions. For an $\epsilon > 0$, we say that D *ϵ-distinguishes* the output distribution of $G(\text{-})$ from the uniform distribution if

$$\Pr_{z\sim\{0,1\}^d}[D(G(z)) = 1] - \Pr_{w\sim\{0,1\}^m}[D(w) = 1] \geq \epsilon$$

In this case, we refer D as a *ϵ-distinguisher* for G. Conversely, G is said to *ϵ-fool* D if D is not an ϵ-distinguisher for G. Similarly, we say that D *ϵ-avoids* G if $\Pr_{w\sim\{0,1\}^m}[D(w) = 1] \geq \epsilon$ and $D(G(z)) = 0$ for every $z \in \{0,1\}^d$. By default, we assume that $\epsilon := \frac{1}{2}$.

For a circuit class $\mathfrak{C}$ and functions $s\colon \mathbb{N} \to \mathbb{N}$ and $\epsilon\colon \mathbb{N} \to [0,1]$, a family of functions $G = \{G : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ is said to be a *pseudorandom generator* that ϵ-fools $\mathfrak{C}$ if, for all large $n \in \mathbb{N}$ and any circuit $C \in \mathfrak{C}$ of n inputs, G $\epsilon(n)$-fools C. We say that G is a *hitting set generator* secure against $\mathfrak{C}$ if for all large $n \in \mathbb{N}$, there is no circuit $D \in \mathfrak{C}$ on n inputs that avoids G_n.

2.3 Circuits

We measure circuit size by the number of gates (except for the input gates). For a circuit type $\mathfrak{C}$ and $s \in \mathbb{N}$ and $\delta \in [0,1]$, we denote by $\widetilde{\mathfrak{C}}(s;\delta)$ the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a circuit of size s such that $\Pr_{x\sim\{0,1\}^n}[f(x) = C(x)] \geq 1-\delta$. We define $\mathfrak{C}(s) := \widetilde{\mathfrak{C}}(s;0)$. For the standard circuit class, we use the notation $\widetilde{\mathsf{SIZE}}(s;\delta)$ and $\mathsf{SIZE}(s)$.

2.4 Time-Bounded Kolmogorov Complexity

We fix an efficient universal Turing machine U. Time-bounded Kolmogorov complexity is defined as follows.

▶ **Definition 17** (Time-bounded Kolmogorov Complexity). *The t-time-bounded A-oracle Kolmogorov complexity of a string $x \in \{0,1\}^*$ is defined as*

$$\mathrm{K}^{t,A}(x) := \{\, |d| \mid U^A \text{ outputs } x \text{ in } t \text{ steps on input } d \,\},$$

where A is an oracle and $t \in \mathbb{N}$.

3 Meta-Computational View of Cryptographic PRG Constructions

In this section, we provide a meta-computational view of cryptographic pseudorandom generator constructions.

3.1 $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$

We present a proof of the following result.

▶ **Reminder of Theorem 4.** *Let A be any oracle. If $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, then $\mathrm{Gap}(\mathrm{K}^A \text{ vs } \mathrm{K}) \in \mathsf{P}$.*

At the core of the proof of Theorem 4 is to use a black-box PRG construction whose advice complexity is small. Following [32], we observe that a k-wise direct product generator, which is one of the simplest constructions of pseudorandom generators, has small advice complexity.

▶ **Theorem 18** (Direct Product Generator [32]). *For any parameters $k, \ell \in \mathbb{N}$ and $\epsilon > 0$, there exist an oracle algorithm $\mathrm{DP}_k^{(\cdot)}\colon \{0,1\}^d \to \{0,1\}^{d+k}$ that takes an oracle $f\colon \{0,1\}^\ell \to \{0,1\}$ and a reconstruction algorithm Rec such that, for any $f\colon \{0,1\}^\ell \to \{0,1\}$ and any ϵ-distinguisher $D\colon \{0,1\}^{d+k} \to \{0,1\}$ for DP_k^f, there exists an advice function $A\colon \{0,1\}^r \to \{0,1\}^a$ such that*

$$\Pr_{s\sim\{0,1\}^r}\left[\forall z \in \{0,1\}^\ell,\ \mathrm{Rec}^D(z;s,A(s)) = f(z)\right] \geq \frac{\epsilon}{2k},$$

where A is computable by a D-oracle circuit of size $\mathsf{poly}(k/\epsilon, 2^\ell)$, the seed length d is at most $O((\ell + \log(k/\epsilon)) \cdot k)$, the advice complexity a is at most $k + O(\log(k/\epsilon))$, the randomness complexity r is at most $O(d)$, and Rec is computable in time $\mathsf{poly}(k/\epsilon, 2^\ell)$.

For the proof of Theorem 18, we make use of the following list-decodable error-correcting code, which can be constructed by concatenating a Reed-Solomon code with an Hadamard code.

▶ **Lemma 19** (List-Decodable Error-Correcting Code; cf. [62, 46])**.** *There exists a function* Enc *such that:*

1. $\mathrm{Enc}(x; N, \epsilon)$ *outputs a string of length* $\widehat{N} = \mathsf{poly}(N, 1/\epsilon)$ *for any* $x \in \{0,1\}^N$*, and is computable in time* $\mathsf{poly}(N, 1/\epsilon)$*.*
2. *There exists a deterministic algorithm* $\mathrm{Dec}(\text{-}; N, \epsilon)$ *running in time* $\mathsf{poly}(N, 1/\epsilon)$ *such that, given any* $r \in \{0,1\}^{\widehat{N}}$*, outputs a list of all the strings* $x \in \{0,1\}^N$ *such that* $\mathsf{Dist}(r, \mathrm{Enc}(x; N, \epsilon)) \leq \frac{1}{2} - \epsilon$*, and the size of the list is at most* $\mathsf{poly}(1/\epsilon)$*.*

Proof Sketch of Theorem 18. We describe the pseudorandom generator construction DP_k^f, which we call a *k-wise direct product generator*. Let Enc denote the error-correcting code of Lemma 19, and let $\hat{f}\colon \{0,1\}^{\widehat{\ell}} \to \{0,1\}$ be the function specified by the truth table $\mathrm{Enc}(f; 2^\ell, \epsilon' := \epsilon/2k) \in \{0,1\}^{2^{\widehat{\ell}}}$, where $\widehat{\ell} = O(\ell + \log(k/\epsilon))$. The pseudorandom generator construction $G_k^f\colon \{0,1\}^{\widehat{\ell}k} \to \{0,1\}^{\widehat{\ell}k+k}$ is defined as

$$G_k^f(z^1, \cdots, z^k) := (z^1, \cdots, z^k, \widehat{f}(z^1), \cdots, \widehat{f}(z^k))$$

for $(z^1, \cdots, z^k) \in \left(\{0,1\}^{\widehat{\ell}}\right)^k$, and $d := \widehat{\ell}k$.

Since the security proof of DP_k^f can be proved by using a standard hybrid argument (see, e.g, [53, 69]), we only provide a proof sketch. Assume that D satisfies

$$\Pr_{\bar{z}}\left[D(z^1, \cdots, z^k, \widehat{f}(z^1), \cdots, \widehat{f}(z^k)) = 1\right] - \Pr_{\bar{z}, b}\left[D(z^1, \cdots, z^k, b_1, \cdots, b_k) = 1\right] \geq \epsilon.$$

For any $i \in \{0, \cdots, k\}$, define the ith hybrid distribution H_i as the distribution of

$$(z^1, \cdots, z^k, \widehat{f}(z^1), \cdots, \widehat{f}(z^i), b_{i+1}, \cdots, b_k),$$

where $\bar{z} = (z^1, \cdots, z^k) \sim \left(\{0,1\}^{\widehat{\ell}}\right)^k$ and $b_{i+1}, \cdots, b_k \sim \{0,1\}$. By this definition, H_0 is identically distributed with the uniform distribution, and H_k is an identical distribution with $\mathrm{DP}_k^f(x^1, \cdots, x^k)$. Therefore,

$$\mathop{\mathbb{E}}_{\substack{i \sim [k] \\ \bar{x}, b}}\left[D(H_i) - D(H_{i-1})\right] \geq \frac{\epsilon}{k}.$$

By an averaging argument, we obtain

$$\Pr_{\substack{i \sim [k], b \\ z^1, \cdots, z^{i-1}, z^{i+1}, \cdots, z^k}}\left[\mathop{\mathbb{E}}_{z^i \sim \{0,1\}^{\widehat{\ell}}}[D(H_i) - D(H_{i-1})] \geq \frac{\epsilon}{2k}\right] \geq \frac{\epsilon}{2k}. \tag{1}$$

Consider the following deterministic algorithm $\mathrm{Rec}_0^D(z; s, A_0(s))$: The coin flip sequence s is regarded as $i \sim [k]$, $z^1, \cdots, z^{i-1}, z^{i+1}, \cdots, z^k \sim \{0,1\}^{\widehat{\ell}}$, and $b \sim \{0,1\}^k$. We set $z^i := z$ and $A_0(s) := (\widehat{f}(z^1), \cdots, \widehat{f}(z^{i-1}), b_i, \cdots, b_k)$. Then, the output of Rec_0^D is defined as $D(\bar{z}, A_0(s)) \oplus 1 \oplus b_i$.

By a standard calculation, it follows from Equation (1) that

$$\Pr_z\left[\mathrm{Rec}_0^D(z, s, A_0(s)) = \widehat{f}(z)\right] \geq \frac{1}{2} + \frac{\epsilon}{2k}$$

with probability at least $\epsilon/2k$ over the random choice of $s = (i, z^{[k]\setminus\{i\}}, b)$. By evaluating $\mathrm{Rec}_0^D(z, s, A_0(s))$ for every $z \in \{0,1\}^{\widehat{\ell}}$, we obtain a string $f' \in \{0,1\}^{\widehat{2^\ell}}$ that encodes a function that agrees with $\widehat{f}$ on at least a $(1/2 + \epsilon/2k)$-fraction of inputs.

The final reconstruction algorithm $\mathrm{Rec}^D(z; s, A(s))$ runs the decoding algorithm $\mathrm{Dec}(f'; 2^\ell, \epsilon/2k)$ of Lemma 19, and obtains a list of strings $f_1, \cdots, f_L$. We define the advice function A as $A(s) := (A_0(s), j)$, where $j \in [L]$ is an index such that f_j coincides with the truth table of f. The algorithm $\mathrm{Rec}^D(z; s, A(s))$ outputs the zth position of f_j.

The advice complexity is at most $|A_0(\text{-})| + \log L = k + O(\log(k/\epsilon))$. Moreover, it is easy to observe that the advice function $A(s)$ can be computed in time $\mathsf{poly}(k/\epsilon, 2^\ell)$, given s as input and oracle access to f and D. By hard-wiring f into a circuit, the advice function A can be computed by a D-oracle circuit of size $\mathsf{poly}(k/\epsilon, 2^\ell)$. ◀

One of important ingredients of the proof for Theorem 4 is a pseudorandom generator constructed by Buhrman, Fortnow, and Pavan [11].

▶ **Lemma 20** (Buhrman, Fortnow, and Pavan [11]). *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a constant* c *and a pseudorandom generator* $G = \{G_n : \{0,1\}^{c \log n} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ *that* $(1/n)$*-fools size-*n *circuits.*

Another ingredient is the fact that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$ implies that a dense subset of A-oracle time-bounded Kolmogorov-random strings can be rejected in polynomial time.

▶ **Lemma 21** ([31]). *Assume that* $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$*. Then, there exists a polynomial-time algorithm* M *such that*

1. $M(x, 1^t) = 1$ *for every* x *such that* $\mathrm{K}^{t,A}(x) < |x| - 1$*, and*
2. $\Pr_{x \sim \{0,1\}^n}[M(x, 1^t) = 0] \geq \frac{1}{4}$ *for every* $n \in \mathbb{N}$ *and every* $t \in \mathbb{N}$*.*

Now we are ready to prove Theorem 4.

Proof of Theorem 4. Under the assumption that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$, we have the secure pseudorandom generator $G = \{G_m : \{0,1\}^{c_0 \log m} \to \{0,1\}^m\}_{m \in \mathbb{N}}$ of Lemma 20. In particular, Promise-BPP = Promise-P; thus it suffices to present a randomized polynomial-time algorithm M_1 for computing $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ for some polynomial τ.

Fix any input $(x, 1^s, 1^t)$, where $x \in \{0,1\}^*$ is a string of length $n \in \mathbb{N}$ and $s, t \in \mathbb{N}$. Take the k-wise direct product generator $\mathrm{DP}_k^{\mathtt{fn}(x)} : \{0,1\}^d \to \{0,1\}^{d+k}$ of Theorem 18, where k is some parameter chosen later and $\epsilon := \frac{1}{8}$. Let M_0 be the polynomial-time algorithm M of Lemma 21. Let τ_0 be some polynomial chosen later.

The randomized algorithm M_1 operates as follows. On input $(x, 1^s, 1^t)$, M_1 samples a string $\bar{z} \sim \{0,1\}^d$ uniformly at random. Then, M_1 simulates M_0 on input $(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z}), 1^{t'})$, where $t' := \tau_0(n, t)$, and accepts if and only if M_0 accepts. (That is, $M_1(x, 1^s, 1^t)$ is defined to be $M_0(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z}), 1^{t'})$ for a random $\bar{z} \sim \{0,1\}^d$.)

We claim the correctness of the algorithm M_1 below.

▷ Claim 22.

1. If $\mathrm{K}^{t,A}(x) \leq s$, then $M_1(x, 1^s, 1^t)$ accepts with probability 1.
2. If $\mathrm{K}^{\tau(n,t)}(x) > s + \log \tau(n, t)$, then $M_1(x, 1^s, 1^t)$ rejects with probability at least $\frac{1}{8}$.

We claim the first item. Fix any $\bar{z} \in \{0,1\}^d$. Since the output $\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z})$ of the direct product generator can be described by $n, k \in \mathbb{N}$, the seed $\bar{z} \in \{0,1\}^d$ and the program for describing x of size $\mathrm{K}^{t,A}(x)$ in time $t' = \tau_0(n, t)$, where τ_0 is some polynomial, it holds that

$$\mathrm{K}^{t',A}(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z})) \leq d + \mathrm{K}^{t,A}(x) + c_1 \log n, \tag{2}$$

for some constant c_1. We set $k := s + c_1 \log n + 2$. Note that under the assumption that $\mathrm{K}^{t,A}(x) \leq s$, Equation (2) is less than $d+k-1$. Therefore, by Lemma 21, $M_0(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z})) = 1$ for every $\bar{z} \in \{0,1\}^d$; thus M_1 accepts.

We claim the second item, by proving its contrapositive. Assume that $M_1(x, 1^s, 1^t)$ rejects with probability less than $\frac{1}{8}$. This means that

$$\Pr_{\bar{z}\sim\{0,1\}^d}\left[M_0(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z}), 1^t) = 0\right] < \frac{1}{8}.$$

On the other hand, by Lemma 21, we also have

$$\Pr_{w\sim\{0,1\}^{d+k}}\left[M_0(w, 1^t) = 0\right] \geq \frac{1}{4}.$$

Therefore,

$$\Pr_{w\sim\{0,1\}^{d+k}}\left[M_0(w, 1^t) = 0\right] - \Pr_{\bar{z}\sim\{0,1\}^d}\left[M_0(\mathrm{DP}_k^{\mathtt{fn}(x)}(\bar{z}), 1^t) = 0\right] \geq \frac{1}{8}.$$

By the property of the reconstruction algorithm Rec of Theorem 18, there exists an advice function $A'\colon \{0,1\}^r \to \{0,1\}^a$ such that

$$\Pr_{\rho\sim\{0,1\}^r}\left[\forall z \in \{0,1\}^{\lceil \log n\rceil},\ \mathrm{Rec}^{\neg M_0(\text{-},1^t)}(z; \rho, A'(\rho)) = \mathtt{fn}(x)(z)\right] \geq \frac{1}{16k}, \tag{3}$$

where the advice complexity a is at most $k + O(\log(k/\epsilon)) = s + O(\log(nk/\epsilon))$. Now we derandomize the random choice of ρ of Equation (3) by using the secure pseudorandom generator G. That is, we argue that ρ can be replaced with $G(\rho_0)$ for some short ρ_0, which enables us to obtain a short description for x. To this end, we define a statistical test $T\colon \{0,1\}^r \to \{0,1\}$ that checks the condition of Equation (3) as follows:

$$T(\rho) = 1 \iff \forall z \in \{0,1\}^{\lceil \log n\rceil},\ \mathrm{Rec}^{\neg M_0(\text{-},1^t)}(z; \rho, A'(\rho)) = \mathtt{fn}(x)(z),$$

for each $\rho \in \{0,1\}^r$.

We claim that T can be computed by a small circuit. Indeed, by Theorem 18, the advice function A' can be computed by a $M_0(\text{-}, 1^t)$-oracle circuit of size $\mathsf{poly}(k, n)$, and Rec can be computed in time $\mathsf{poly}(k, n) \leq \mathsf{poly}(n)$ with oracle access to $M_0(\text{-}, 1^t)$.[5] The oracle $M_0(\text{-}, 1^t)$ can be simulated by a circuit of size $\mathsf{poly}(d + k, t) \leq \mathsf{poly}(n, t)$. Overall, T is computable by a circuit of size $m := \mathsf{poly}(n, t)$; here we take m large enough so that $m \geq r$ and $m > 16k$.

Now we replace the random bits $\rho \in \{0,1\}^r$ with the first r bits of the pseudorandom sequence $G_m(\rho_0)$. By Equation (3), we have $\Pr_{\rho\sim\{0,1\}^r}[T(\rho) = 1] \geq \frac{1}{16k}$. It follows from the property of G_m that

$$\Pr_{\rho_0\sim\{0,1\}^{c_0 \log m}}[T(G(\rho_0)) = 1] > 0.$$

In particular, there exists a seed $\rho_0 \in \{0,1\}^{c_0 \log m}$ such that $T(G(\rho_0)) = 1$.

We are ready to present the algorithm for describing x. In order to describe x, it takes as a description $n, t, m \in \mathbb{N}$, the seed $\rho_0 \in \{0,1\}^{c_0 \log m}$, and the advice string $\alpha := A'(G(\rho_0)) \in \{0,1\}^a$. Since $T(G(\rho_0)) = 1$, the string x can be obtained by concatenating the output of $\mathrm{Rec}^{\neg M_0(\text{-},1^t)}(z; G(\rho_0), \alpha)$ for all $z \in [n]$. The running time of this procedure can be bounded by $\tau_1(n, t)$ for some polynomial τ_1. Therefore,

$$\mathrm{K}^{\tau_1(n,t)}(x) \leq a + c_0 \log m + O(\log nt) \leq s + O(\log nt).$$

[5] We may assume without loss of generality that $k := s + O(\log n) = O(n)$, as otherwise we trivially have $\mathrm{K}^{t,A}(x) \leq s$.

In particular, by choosing a polynomial τ large enough, we have

$$\mathrm{K}^{\tau(n,t)}(x) \leq \mathrm{K}^{\tau_1(n,t)}(x) \leq s + \log \tau(n,t).$$

This completes the proof of Claim 22.

Since M_1 is a one-sided-error algorithm, the success probability can be amplified by repeating the computation of M_1 for independent random coin flips. We thus conclude that $\mathrm{Gap}_\tau(\mathrm{K}^A \text{ vs } \mathrm{K})$ is in Promise-$\mathsf{BPP} =$ Promise-P. ◀

Let $\Sigma_k\mathsf{SAT}$ denote the canonical Σ_k^{p}-complete problem. By using the disjointness of $\mathrm{Gap}(\mathrm{K}^{\Sigma_k\mathsf{SAT}} \text{ vs } \mathrm{K})$, we immediately obtain the following.

▶ **Corollary 23.** *If* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$*, then for any constant* $k \in \mathbb{N}$*, there exists a polynomial* τ *such that* $\mathrm{K}^{\tau(|x|,t)}(x) \leq \mathrm{K}^{t,\Sigma_k\mathsf{SAT}}(x) + \log \tau(|x|,t)$ *for any* $x \in \{0,1\}^*$ *and* $t \in \mathbb{N}$*.*

Proof. Let $A := \Sigma_k\mathsf{SAT}$. By Theorem 4, under the assumption that $\mathsf{DistNP}^A \subseteq \mathsf{DistPH} \subseteq \mathsf{AvgP}$, there exists an algorithm M such that $M(x, 1^s, 1^t) = 1$ for every x such that $\mathrm{K}^{t,A}(x) \leq s$ and $M(x, 1^s, 1^t) = 0$ for every x such that $\mathrm{K}^{\tau(|x|,t)}(x) > s + c \log |x|$, for any $s \in \mathbb{N}$ and $t \in \mathbb{N}$. In particular, the set of Yes and that of No instances are disjoint, as otherwise we have $0 = M(x, 1^s, 1^t) = 1$ for some instance $(x, 1^s, 1^t)$, which is a contradiction. For any $x \in \{0,1\}^*$ and $t \in \mathbb{N}$, define $s := \mathrm{K}^{t,A}(x)$; then we obtain that $\mathrm{K}^{\mathsf{poly}(|x|,t)}(x) \leq s + \log \tau(|x|,t)$ by the disjointness. ◀

An important corollary is that NP-hardness of $\mathrm{Gap}(\mathrm{K}^{\Sigma_k\mathsf{SAT}} \text{ vs } \mathrm{K})$ is sufficient for proving an equivalence between worst-case and average-case complexity of PH.

▶ **Restatement of Corollary 6.** *Assume that* $\mathrm{Gap}(\mathrm{K}^{\Sigma_k\mathsf{SAT}} \text{ vs } \mathrm{K})$ *is "*NP*-hard" for some* $k \in \mathbb{N}$ *in the sense that*

$$\mathsf{NP} \not\subseteq \mathsf{BPP} \implies \mathrm{Gap}(\mathrm{K}^{\Sigma_k\mathsf{SAT}} \text{ vs } \mathrm{K}) \notin \mathsf{P}.$$

Then,

$$\mathsf{DistPH} \not\subseteq \mathsf{AvgP} \iff \mathsf{PH} \neq \mathsf{P}.$$

Proof. It is obvious that $\mathsf{PH} = \mathsf{P}$ implies that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. Thus we prove the converse direction. Assume that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. By Theorem 4, we obtain $\mathrm{Gap}(\mathrm{K}^{\Sigma_k\mathsf{SAT}} \text{ vs } \mathrm{K}) \in \mathsf{P}$ for any constant $k \in \mathbb{N}$. By the assumption, we have $\mathsf{NP} \subseteq \mathsf{BPP}$; moreover, by Lemma 20, we also have $\mathsf{BPP} = \mathsf{P}$. Therefore, it follows that $\mathsf{NP} = \mathsf{P}$, which is equivalent to $\mathsf{PH} = \mathsf{P}$. ◀

Under the plausible assumption that $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$, we observe that $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}} \text{ vs } \mathrm{K})$ is non-disjoint.

▶ **Proposition 24.** *If* $\mathsf{E}^{\mathsf{NP}} \neq \mathsf{E}$*, then, for some polynomial* τ_0 *and any polynomial* τ*, there are infinitely many strings* x *such that* $\mathrm{K}^{t,\mathsf{SAT}}(x) = O(\log |x|)$ *and* $\mathrm{K}^{\tau(|x|)}(x) > \log \tau(|x|)$*, where* $t := \tau_0(|x|)$*.*

Proof. By [12], the assumption is equivalent to $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{E}/O(n)$. Take a language $L \in \mathsf{E}^{\mathsf{NP}} \setminus \mathsf{E}/O(n)$. For each $n \in \mathbb{N}$, define x_n to be the truth table of length 2^n that encodes the characteristic function of $L \cap \{0,1\}^n$. Since x_n can be described in $\tau_0(|x_n|) = \mathsf{poly}(|x_n|)$ time given $n \in \mathbb{N}$ and oracle access to SAT, we have $\mathrm{K}^{t,\mathsf{SAT}}(x) = O(\log |x_n|)$. On the other hand, if $\mathrm{K}^{\tau(|x_n|)}(x_n) \leq \log \tau(|x_n|)$ for all large $n \in \mathbb{N}$, there exists an advice string of length $\log \tau(|x_n|) = O(n)$ that makes it possible to compute $L \cap \{0,1\}^n$ in time $\mathsf{poly}(\tau(|x_n|)) = 2^{O(n)}$, which contradicts $L \notin \mathsf{E}/O(n)$. ◀

Finally, we observe that the complexity of $\mathrm{Gap}(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ is closely related to the complexity of MINKT.

▶ **Proposition 25.** *For any polynomial τ, the following hold.*

- $\mathrm{Gap}_\tau(\mathrm{K}$ vs $\mathrm{K})$ *is reducible to* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ *via an identity map.*
- *If* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ *is disjoint, then* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$ *is reducible to* MINKT*; in particular,* $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K}) \in \mathsf{NP}$.

Proof. The first item is obvious because $\mathrm{K}^{t,\mathsf{SAT}}(x) \leq \mathrm{K}^t(x)$ for any $t \in \mathbb{N}$ and $x \in \{0,1\}^*$. For the second item, let $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ denote $\mathrm{Gap}_\tau(\mathrm{K}^{\mathsf{SAT}}$ vs $\mathrm{K})$. Since $(\overline{\Pi_{\mathrm{NO}}}, \Pi_{\mathrm{NO}})$ is a problem of checking whether $\mathrm{K}^{\tau(|x|,t)}(x) \leq s + \log \tau(|x|,t)$ given $(x, 1^s, 1^t)$ as input, it is reducible to MINKT. In particular, $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathsf{NP}$ holds as well under the assumption that it is disjoint. ◀

3.2 $\mathrm{Gap}(F$ vs $F^{-1})$: PRG Construction from One-Way Functions

Håstad, Impagliazzo, Levin, and Luby [29] showed that a cryptographic pseudorandom generator can be constructed from any one-way function. In this section, we view the black-box PRG construction from a meta-computational perspective, which leads us to the promise problem $\mathrm{Gap}(F$ vs $F^{-1})$, which is a problem of asking whether a given function f is computable by a small circuit or f is hard to invert by any small circuit. Here we assume that the function $f\colon \{0,1\}^n \to \{0,1\}^n$ is given as oracle, and we focus on an algorithm that runs in time $\mathsf{poly}(n)$. In other words, we consider a sublinear-time algorithm that is given random access to the truth table of f.

▶ **Definition 26.** *For a function $\tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\mathrm{Gap}_\tau(F$ vs $F^{-1})$ is a promise problem (Π_{YES}, Π_{NO}) defined as follows. The input consists of a size parameter $s \in \mathbb{N}$, an integer $n \in \mathbb{N}$, and black-box access to a function $f\colon \{0,1\}^n \to \{0,1\}^n$.*

- *The set Π_{YES} consists of inputs (n, s, f) such that* $\mathrm{size}(f) \leq s$.
- *The set Π_{NO} consists of inputs (n, s, f) such that, for any oracle circuit C of size $\tau(n, s)$,*

$$\Pr_{x \sim \{0,1\}^n}\left[C^f(f(x)) \in f^{-1}(f(x))\right] < \frac{1}{2}.$$

▶ **Theorem 27.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a polynomial τ and a* coRP*-type randomized algorithm M that solves $\mathrm{Gap}_\tau(F$ vs $F^{-1}) = (\Pi_{YES}, \Pi_{NO})$ on input (n, s) in time* $\mathsf{poly}(n, s)$*. That is, M is a randomized oracle algorithm such that*

1. $\Pr_M[M^f(n, s) = 1] = 1$ *for every* $(n, s, f) \in \Pi_{YES}$*,*
2. $\Pr_M[M^f(n, s) = 0] \geq \frac{1}{2}$ *for every* $(n, s, f) \in \Pi_{NO}$*, and*
3. $M^f(n, s)$ *runs in time* $\mathsf{poly}(n, s)$*.*

For the proof, we make use of the following black-box construction of a pseudorandom generator based on any one-way function.

▶ **Lemma 28** (A black-box PRG Construction from Any OWF [29])**.** *There exists a polynomial $d = d(n)$ such that, for a parameter $m \in \mathbb{N}$, there exists a polynomial-time oracle algorithm $G_m^{(\cdot)}\colon \{0,1\}^{d(n)} \to \{0,1\}^m$ that takes a function $f\colon \{0,1\}^n \to \{0,1\}^n$ and there exists an oracle algorithm R such that, for any function $D\colon \{0,1\}^m \to \{0,1\}$, if $m > d$ and*

$$\Pr_{z \sim \{0,1\}^n}\left[D(G^f(z)) = 1\right] - \Pr_{w \sim \{0,1\}^m}\left[D(w) = 1\right] \geq \frac{1}{8},$$

then

$$\Pr_{x,R}[R^{f,D}(f(x)) \in f^{-1}(f(x))] \geq \frac{1}{2}.$$

The running time of $G_m^{(\cdot)}$ and R is at most $\mathsf{poly}(n, m)$*.*

Proof Sketch. Since a weakly one-way function exists if and only if a strongly one-way function exists ([72]), it suffices to present a reduction that inverts f with probability at least $1/\mathsf{poly}(n,m)$. (To be more specific, we first amplify the hardness of f by taking a direct product $f^t(x_1,\cdots,x_t) := (f(x_1),\cdots,f(x_t))$, where t is some appropriately chosen parameter, and then apply the HILL construction to f^t described below.)

We invoke the pseudorandom generator construction G^f_{HILL} of Håstad, Impagliazzo, Levin, and Luby [29] based on f. They presented a security reduction R such that if there exists a function D that distinguishes G^f_{HILL} from the uniform distribution, then an oracle algorithm $R^{f,D}(f(x))$ can compute an element in $f^{-1}(f(x))$ with probability at least $1/\mathsf{poly}(n,m)$ over the choice of $x \sim \{0,1\}^n$ and the internal randomness of R. ◀

Proof of Theorem 27. Let $G^{(\cdot)}$ be the black-box pseudorandom generator construction of Lemma 28, and let R be the security reduction of Lemma 28.

Under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, by Lemma 21, there exists a polynomial-time algorithm M such that $M(x,1^t) = 1$ for every x such that $\mathrm{K}^t(x) < |x| - 1$, and $\Pr_{x\sim\{0,1\}^n}[M(x,1^t)=0] \geq \frac{1}{4}$ for every $n \in \mathbb{N}$ and every $t \in \mathbb{N}$.

The algorithm M' for computing $\mathrm{Gap}(F \text{ vs } F^{-1})$ is defined as follows. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ and $s \in \mathbb{N}$ be inputs. Define $m := p(n,s)$ for some polynomial p chosen later. Pick a random $z \in \{0,1\}^n$. Then M' accepts if and only if $M(G^f_m(z),1^t)$ accepts for a sufficiently large $t = \mathsf{poly}(n,s)$.

We claim the correctness of M' below.

▷ Claim 29.

1. M' accepts any f such that $\mathrm{size}(f) \leq s$ with probability 1.
2. M' rejects any f such that $\Pr_x\left[C^f(f(x)) \in f^{-1}(f(x))\right] < \frac{1}{2}$ with probability at least $\frac{1}{8}$.

We claim that any f such that $\mathrm{size}(f) \leq s$ is accepted by the algorithm M'. Indeed, since f is computable by some circuit of size s and G^f_m is polynomial-time-computable, the output of the generator $G^f_m(z)$ can be described by using the description of the circuit of size s, the seed z of length $d(n)$, and $n,m \in \mathbb{N}$ in polynomial time; thus, for a sufficiently large $t = \mathsf{poly}(n,s)$,

$$\mathrm{K}^t(G^f_m(z)) \leq d(n) + \widetilde{O}(s) + O(\log n).$$

Choosing $m = p(n,s)$ large enough, this is bounded by $m-2$. Thus M' accepts.

Conversely, suppose that the algorithm M' accepts with probability at least $\frac{7}{8}$. This means that

$$\Pr_{z\sim\{0,1\}^n}\left[M(G^f_m(z),1^t)=1\right] \geq \frac{7}{8}.$$

On the other hand, by Lemma 21, we have

$$\Pr_{w\sim\{0,1\}^m}\left[M(w,1^t)=1\right] \leq \frac{3}{4}.$$

Therefore, $M(\text{-},1^t)$ is a distinguisher for G^f_m. It follows from the property of the security reduction R that

$$\Pr_{x,R}\left[R^{f,M(\text{-},1^t)}(f(x)) \in f^{-1}(f(x))\right] \geq \frac{1}{2}.$$

By fixing the internal randomness of R and simulating the polynomial-time algorithm $R^{f,M(\text{-},1^t)}$ by a polynomial-size circuit C^f, we conclude that f can be inverted by the oracle circuit C^f of size $\mathsf{poly}(n,s) =: \tau(n,s)$. Thus f is not a NO instance of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$. ◀

▶ Remark 30. In fact, the assumption that DistNP $\subseteq$ AvgP of Theorem 27 can be weakened to the assumption that there exists a P-natural property useful against $\mathsf{SIZE}(2^{o(n)})$ (which is essentially equivalent to an errorless heuristic algorithm for MCSP [34, 31]). Indeed, as in [2, 60, 5], the pseudorandom function generator construction of [25] can be used to construct a black-box pseudorandom generator G^f based on a one-way function f that satisfies $\mathrm{size}(G^f(z)) \leq \mathsf{poly}(|z|, \log |G^f(z)|)$ for any seed $z \in \{0,1\}^d$; such a pseudorandom generator G^f can be distinguished from the uniform distribution by using the natural property.

We now explain that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is conjectured to be non-disjoint. An *auxiliary-input one-way function* (AIOWF) $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x \in \{0,1\}^*}$ is a polynomial-time-computable function such that, for some infinite set I, for any non-uniform polynomial-time algorithm A, $\Pr_y\left[A(x, f_x(y)) \in f_x^{-1}(f_x(y))\right] < 1/n^{\omega(1)}$ for all large $n \in \mathbb{N}$ and any $x \in I$. This is a weaker cryptographic primitive than a one-way function (i.e., the existence of a one-way function implies that of an auxiliary-input one-way function). Ostrovsky [56] showed that non-triviality of SZK implies the existence of an auxiliary-input one-way function. (see also [68]). We observe that the existence of an auxiliary-input one-way function implies the non-disjointness of $\mathrm{Gap}(F \text{ vs } F^{-1})$.

▶ **Proposition 31.** *If there exists an auxiliary-input one-way function $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x \in \{0,1\}^*}$, then, for any polynomial τ, $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ is non-disjoint.*

Proof. Take an infinite set $I \subseteq \{0,1\}^*$ that is hard for polynomial-size circuits to invert $\{f_x\}_{x \in I}$. Since f is polynomial-time-computable, $\mathrm{size}(f_x) \leq n^c$ for some constant c, where $n = |x|$. We set the size parameter $s := n^c$. On the other hand, by the property of AIOWF, for any circuit A of size $\tau(n, s)$, it holds that

$$\Pr_y\left[A(x, f_x(y)) \in f_x^{-1}(f_x(y))\right] < \frac{1}{2},$$

for a sufficiently large $x \in I$. This means that f_x is a YES and NO instance of $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$. ◀

An immediate corollary of Theorem 27 and Proposition 31 is that the existence of AIOWF implies DistNP $\not\subseteq$ AvgP (which is already shown in [31]). An interesting open question is to prove "NP-hardness" of $\mathrm{Gap}(F \text{ vs } F^{-1})$, which has the following important consequence:

▶ **Corollary 32.** *If, for any polynomial τ, it is "NP-hard" to solve $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ in time $\mathsf{poly}(n, s)$, then the worst-case and average-case complexity of NP is equivalent in the sense that P $\neq$ NP iff DistNP $\not\subseteq$ AvgP.*

Proof. The assumption that $\mathrm{Gap}(F \text{ vs } F^{-1})$ is "NP-hard" means that, for any polynomial τ, if there exists a coRP-type algorithm that solves $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ on input (n, s) in time $\mathsf{poly}(n, s)$, then NP $\subseteq$ BPP. If DistNP $\subseteq$ AvgP, then Theorem 27 implies that there exists some polynomial τ such that $\mathrm{Gap}_\tau(F \text{ vs } F^{-1})$ can be solved by a coRP-type algorithm in time $\mathsf{poly}(n, s)$. By the assumption, we obtain NP $\subseteq$ BPP $=$ P, where the last equality is from Lemma 20. ◀

4 Meta-Computational View of Complexity-Theoretic PRG Constructions

We now turn our attention to a meta-computational view of *complexity-theoretic* PRG constructions.

4.1 Universal Hitting Set Generators and Kolmogorov Complexity

We review the notion of KS- and Kt-complexity and present definitions of universal hitting set generators. The notion of KS-complexity was introduced in [2] as a space-bounded analogue of Kt-complexity. Here we slightly modify the definition and add an additive term of $+2$ so that a result about a universal hitting set generator becomes cleaner.

▶ **Definition 33.** *For a string $x \in \{0,1\}^*$, the* KS *complexity of x is defined as*

$$\mathrm{KS}(x) := \min\{\, |d| + s + 2 \mid U^d(i) \text{ outputs } x_i \text{ in space } s \text{ for every } i \in [|x|+1] \,\}.$$

Here x_i denotes the ith bit of x for $i \in [|x|]$, and $x_{|x|+1} := \bot$.

We consider the following universal hitting set generator.

▶ **Definition 34** (Universal Space-Bounded Hitting Set Generator)**.** *For a function $s\colon \mathbb{N} \to \mathbb{N}$, define $\mathrm{HS}^s = \{\mathrm{HS}^s_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ as the function computed by the following algorithm: If the input is of the form $1^t 0 d 0 1^a$ for some $t, a \in \mathbb{N}$ and $d \in \{0,1\}^*$, HS^s_n simulates $U^d(i)$ using at most t space, for every $i \in [n+1]$. If every simulation succeeds and $U^d(i) \in \{0,1\}$ for $i \in [n]$ and $U^d(n+1) = \bot$, then output $U^d(1)\cdots U^d(n)$. Otherwise, output 0^n.*

The hitting set generator HS is universal in the sense that, if there exists a hitting set generator G of seed length $s(n)$ that is computable in $s(n)$ space, then $\mathrm{HS}^{O(s)}_n$ is also a hitting set generator. This observation immediately follows from the following property.

▶ **Proposition 35** (Universality of HS)**.** *For every function $s\colon \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, the image of HS^s_n contains every string $x \in \{0,1\}^n$ such that $\mathrm{KS}(x) \le s(n)$. Moreover, HS^s_n can be computed in $O(s(n) + \log n)$ space.*

Proof. Consider any string x of length n such that $\mathrm{KS}(x) \le s(n)$. By the definition of KS complexity, there exists a description $d \in \{0,1\}^*$ such that $U^d(i)$ outputs x_i using at most t space for every $i \in [n+1]$, where $|d| + t + 2 \le s(n)$. By the definition of HS^s_n, we have $\mathrm{HS}^s_n(1^t 0 d 0 1^a) = x$ for $a := s(n) - t - |d| - 2 \ge 0$. ◀

We recall Levin's resource-bounded Kolmogorov complexity and define a time-bounded version of a universal hitting set generator.

▶ **Definition 36** ([47])**.** *For a string $x \in \{0,1\}^*$, Levin's* Kt *complexity of x is defined as*

$$\mathrm{Kt}(x) := \min\{\, |d| + t + 2 \mid U^d(i) \text{ outputs } x_i \text{ in time } 2^t \text{ for every } i \in [|x|+1] \,\}.$$

▶ **Definition 37** (Universal Time-Bounded Hitting Set Generator)**.** *For a function $s\colon \mathbb{N} \to \mathbb{N}$, define $\mathrm{Ht}^s = \{\mathrm{Ht}^s_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ as the function computed by the following algorithm: If the input is of the form $1^t 0 d 0 1^a$ for some $t, a \in \mathbb{N}$ and $d \in \{0,1\}^*$, Ht^s_n simulates $U^d(i)$ for 2^t time, for every $i \in [n+1]$. If every simulation succeeds and $U^d(i) \in \{0,1\}$ for $i \in [n]$ and $U^d(n+1) = \bot$, then output $U^d(1)\cdots U^d(n)$. Otherwise, output 0^n.*

▶ **Proposition 38** (Universality of Ht)**.** *For every function $s\colon \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, the image of Ht^s_n contains every string $x \in \{0,1\}^n$ such that $\mathrm{Kt}(x) \le s(n)$. Moreover, the range of Ht^s_n can be enumerated in time $\widetilde{O}(2^{s(n)+\log n})$.*

4.2 Advice and Resource-Bounded Kolmogorov Complexity

In order to define meta-computational circuit lower-bound problems, we modify the standard notion of advice. Usually, a complexity class with advice such as $\mathsf{E}/O(n)$ is defined as a subset of functions $f\colon \{0,1\}^* \to \{0,1\}$ that are defined on all the strings of any length. Here, for any $n \in \mathbb{N}$, we define "$\mathsf{DTIME}(2^{O(n)})/^n O(n)$" as a subset of functions $f\colon \{0,1\}^n \to \{0,1\}$, where the superscript n in "$/^n$" is appended in order to emphasize that it depends on n.

▶ **Definition 39.** *For any integers $t, a, n \in \mathbb{N}$, we denote by $\mathsf{DTIME}(t)/^n a$ the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a Turing machine M whose description length is a and that outputs $f(x)$ on input $x \in \{0,1\}^n$ in time t. Similarly, let $\mathsf{DSPACE}(t)/^n a$ denote the class of functions $f\colon \{0,1\}^n \to \{0,1\}$ such that there exists a Turing machine M whose description length is a and that outputs $f(x)$ on input $x \in \{0,1\}^n$ in* space t.

This definition is slightly different from the standard notion of complexity classes with advice, but these are essentially the same. In order to clarify the difference, for functions $t, a\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{DTIME}(t)/^{\mathrm{KL}} a$ denote the complexity class $\mathsf{DTIME}(t)$ with a-bit advice strings in the standard sense of Karp and Lipton [42].[6] That is, a function $f\colon \{0,1\}^* \to \{0,1\}$ is in $\mathsf{DTIME}(t)/^{\mathrm{KL}} a$ if and only if there exists a Turing machine M such that, for any $n \in \mathbb{N}$, there exists an advice string $\alpha_n \in \{0,1\}^{a(n)}$ such that M outputs $f(x)$ on input (x, α_n) in time $t(n)$ for every $x \in \{0,1\}^n$. Then, the advice "$/^n$" and the Karp–Lipton advice "$/^{\mathrm{KL}}$" are equivalent in the following sense.

▶ **Fact 40.** *For any functions $t, a\colon \mathbb{N} \to \mathbb{N}$ and any family of functions $f = \{f_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ (which is identified with a function $f\colon \{0,1\}^* \to \{0,1\}$), the following are equivalent.*

1. *There exists a constant c such that $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$, where $t'(n) := t(n)^c + c$ and $a'(n) := c \cdot a(n) + c$.*
2. *There exists a constant c such that, for any $n \in \mathbb{N}$, $f_n \in \mathsf{DTIME}(t(n)^c + c)/^n c \cdot a(n) + c$.*

Proof Sketch. If $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$, then there exists a machine M that takes an advice string α_n on inputs of length n. For each $n \in \mathbb{N}$, define M_n to be the machine that, on input x, simulates M on input (x, α_n); the description length of M_n is at most $O(|\alpha_n|)$, and thus $f_n \in \mathsf{DTIME}(t(n)^{O(1)})/^n O(a(n))$. Conversely, if, for any $n \in \mathbb{N}$, there exists a Turing machine M_n whose description length is $O(a(n))$, then a universal Turing machine U witnesses $f \in \mathsf{DTIME}(t')/^{\mathrm{KL}} a'$. ◀

The advice "$/^n$" is equivalent to Kt-complexity up to a constant factor in the following sense.

▶ **Fact 41.** *For any function $t\colon \mathbb{N} \to \mathbb{N}$ such that $t(n) \geq n$ and for any family of functions $f = \{f_n\colon \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$, the following are equivalent.*

1. *$f_n \in \mathsf{DTIME}(t(n)^{O(1)})/^n O(\log t(n))$ for all large $n \in \mathbb{N}$.*
2. *$\mathrm{Kt}(f_n) = O(\log t(n))$ for all large $n \in \mathbb{N}$.*

4.3 Meta-computational Circuit Lower-bound Problems (MCLPs)

We define promise problems of distinguishing the truth table of explicit functions (e.g., computable in $\mathsf{DTIME}(2^{cn})/^n cn$) from the truth table of hard functions (e.g., that cannot be computed in $\mathsf{SIZE}(2^{\epsilon n})$). We call these Meta-computational Circuit Lower-bound Problems (MCLPs).

[6] It is also common to use the notation $\mathsf{DTIME}(t(n))/^{\mathrm{KL}} a(n)$, where n is an indeterminate.

▶ **Definition 42** (Meta-computational Circuit Lower-bound Problems; MCLPs)**.** *Let $\mathcal{E}, \mathcal{D}$ be families of functions.* The $\mathcal{E}$ vs $\mathcal{D}$ problem *is defined as the following promise problem* (Π_{YES}, Π_{NO}).

$$\Pi_{YES} := \{\, \mathtt{tt}(f) \mid f \in \mathcal{E} \,\},$$
$$\Pi_{NO} := \{\, \mathtt{tt}(f) \mid f \notin \mathcal{D} \,\}.$$

We will mainly consider a non-uniform computational model for computing the $\mathcal{E}$ vs $\mathcal{D}$ problem; for $N \in \mathbb{N}$, we denote by $(\mathcal{E} \text{ vs } \mathcal{D})_N$ the problem restricted to the input length of N.

We denote by $(\mathsf{E} \text{ vs } \mathcal{D})$ a family of problems $\{\mathsf{E}_c \text{ vs } \mathcal{D}\}_{c\in\mathbb{N}}$, where

$$\mathsf{E}_c := \bigcup_{n\in\mathbb{N}} \mathsf{DTIME}(2^{cn})/^n cn.$$

For a circuit class $\mathfrak{C}$, we say that $(\mathsf{E} \text{ vs } \mathcal{D}) \in \mathsf{i.o.}\mathfrak{C}(s(N))$ if, for every constant c, there exists a family of $\mathfrak{C}$-circuits $\{C_N\}_{N\in\mathbb{N}}$ of size $s(N)$ such that C_N solves the promise problem $(\mathsf{E}_c \text{ vs } \mathcal{D})_N$ for infinitely many N. We also denote by $\left(\mathsf{E} \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ a family of problems $\{\mathsf{E}_c \text{ vs } \mathsf{SIZE}(2^{\alpha n})\}_{c\in\mathbb{N},\alpha>0}$.

Similarly, $\left(\mathsf{DSPACE}(n) \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ denotes the family

$$\left\{ \bigcup_{n\in\mathbb{N}} \mathsf{DSPACE}(cn)/^n cn \text{ vs } \mathsf{SIZE}(2^{\alpha n}) \right\}_{c\in\mathbb{N},\alpha>0}.$$

The definition of the E vs $\mathsf{SIZE}(2^{o(n)})$ problem given here is slightly different from Definition 10 given earlier. In Definition 10, we defined the E vs $\mathsf{SIZE}(2^{o(n)})$ problem by using the notion of Kt-complexity; however, these definitions are essentially equivalent in light of Fact 41.

We justify the notation of $\left(\mathsf{DSPACE}(n) \text{ vs } \mathsf{SIZE}(2^{o(n)})\right)$ below. One can observe that the open question of whether $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}(2^{o(n)})$ is closely related to the $\mathsf{DSPACE}(n)$ vs $\mathsf{SIZE}(2^{o(n)})$ problem.

▶ **Proposition 43.** *The following are equivalent.*

1. $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant $\epsilon > 0$.*
2. *No circuit can solve the* $\mathsf{DSPACE}(n)$ vs $\mathsf{SIZE}(2^{o(n)})$ *problem for all large $n \in \mathbb{N}$.*
3. *No circuit can solve the* $\mathsf{DSPACE}(n)$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ *problem for all large $n \in \mathbb{N}$.*
4. *There exist some constants $c \in \mathbb{N}, \alpha > 0$ such that, for all large $n \in \mathbb{N}$, there exists a function $f\colon \{0,1\}^n \to \{0,1\}$ such that $f \in \mathsf{DSPACE}(cn)/^n cn$ and $f \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$.*

Proof. First, observe that Item 3 is equivalent to Item 4 by the definition. We claim that Item 1 implies Item 4. By using a locally-decodable error-correcting code, it can be shown that Item 1 is equivalent to $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.}\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$ for some constant $\alpha > 0$ (cf. [63, 44]). Take a problem $L \in \mathsf{DSPACE}(n) \setminus \mathsf{i.o.}\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$. For each $n \in \mathbb{N}$, let $f_n\colon \{0,1\}^n \to \{0,1\}$ be the characteristic function of $L \cap \{0,1\}^n$. Then, there exists a constant c such that, for all large $n \in \mathbb{N}$, $f_n \in \mathsf{DSPACE}(cn)/^n cn$ and $f_n \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$, which completes the proof of Item 4. It is immediate from the definition that Item 4 implies Item 2.

We claim that Item 2 implies Item 1. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ be a family of functions such that $f_n \in \mathsf{DSPACE}(cn)/^n cn$ and $f_n \notin \mathsf{SIZE}(2^{\alpha n})$ for all large $n \in \mathbb{N}$. In particular, for all large $n \in \mathbb{N}$, there exists some machine M_n of description length cn that computes $f_n(x)$ on input $x \in \{0,1\}^n$ in space cn. Let $L \subseteq \{0,1\}^*$ be a language such that $(x, M) \in L$ if and only if the description length of a Turing machine M is $c|x|$ and M

accepts x in space $c|x|$. It is clear that $L \in \mathsf{DSPACE}(n)$. Moreover, for all large $n \in \mathbb{N}$, the characteristic function of $\{ x \in \{0,1\}^n \mid (x, M_n) \in L \}$ is equal to f_n; thus, it requires circuits of size $2^{\alpha n}$. Therefore, $L \notin \mathsf{i.o.SIZE}(2^{\alpha n/(c+1)})$. ◀

It is also possible to define MCLPs whose non-disjointness characterizes other circuit lower bounds. For example, the $\mathsf{EXP/poly}$ vs $\mathsf{SIZE}(2^{o(n)})$ problem defined as

$$\left\{ \bigcup_{n\in\mathbb{N}} \mathsf{DTIME}(2^{n^c})/^n n^c \text{ vs } \mathsf{SIZE}(2^{\alpha n}) \right\}_{c\in\mathbb{N},\alpha>0}$$

is non-disjoint if and only if $\mathsf{EXP/poly} \not\subseteq \mathsf{SIZE}(2^{o(n)})$.

4.4 MCLPs from HSG Constructions

We formalize the notion of black-box PRG and HSG construction and then we present a general connection between black-box PRG and HSG constructions and meta-computational circuit lower bound problems.

▶ **Definition 44** (Black-Box PRG and HSG Construction). *Let $G^{(\cdot)}\colon \{0,1\}^d \to \{0,1\}^m$ be an oracle algorithm that expects an oracle of the form $f\colon \{0,1\}^\ell \to \{0,1\}$. Let $\mathfrak{C}$ be a circuit class and $\mathcal{R}^{(\cdot)}$ be an oracle circuit class. The algorithm G is referred to as a* black-box $\mathfrak{C}$-pseudorandom generator (resp. $\mathfrak{C}$-hitting set generator) construction with $\mathcal{R}$-reconstruction *and error parameter ϵ if the following hold.*

$\mathfrak{C}$-Construction *For any seed $z \in \{0,1\}^d$, there exists a $\mathfrak{C}$-circuit that takes $\mathtt{tt}(f)$ as input and outputs $G^f(z)$.*

$\mathcal{R}$-Reconstruction *For any function $f\colon \{0,1\}^\ell \to \{0,1\}$ and any function $D\colon \{0,1\}^m \to \{0,1\}$, if D is an ϵ-distinguisher for G^f (resp. D ϵ-avoids G^f), then $f \in \mathcal{R}^D$.*

We now present a generic connection between MCLPs and HSG constructions.

▶ **Theorem 45.** *Let $G^{(\cdot)}\colon \{0,1\}^s \to \{0,1\}^m$ be a black-box $\mathfrak{C}$-construction with $\mathcal{R}$-reconstruction that takes a function $f\colon \{0,1\}^n \to \{0,1\}$. Define $N := 2^n$. Let $H\colon \{0,1\}^d \to \{0,1\}^m$ be a function such that $G^f(z) \in \mathrm{Im}(H)$ for every $f \in \mathcal{E}$ and every $z \in \{0,1\}^d$. Let $D\colon \{0,1\}^m \to \{0,1\}$ be any function that ϵ-avoids H. Then, the following hold.*

1. *If G^f is a HSG construction with error parameter ϵ, then $(\mathcal{E} \text{ vs } \mathcal{R}^D)_N \in \mathsf{AND}_{2^s} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$.*
2. *If G^f is a PRG construction with error parameter $\epsilon/2$, then $(\mathcal{E} \text{ vs } \mathcal{R}^D)_N \in \mathsf{AND}_{O(N/\epsilon)} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$.*

Proof. Let $G^{(\cdot)}$ be a HSG construction with error parameter ϵ. We first present a randomized circuit for solving $(\mathcal{E} \text{ vs } \mathcal{R}^D)$ on inputs of length N. Let $f\colon \{0,1\}^n \to \{0,1\}$ denote an input. Consider a circuit D_1 such that $D_1(f;z) := D\big(G^f(z)\big)$, where z is an auxiliary input (that will be regarded as non-deterministic bits or random bits). We claim that D_1 can solve $(\mathcal{E} \text{ vs } \mathcal{D})$ co-nondeterministically.

▷ Claim 46.

1. If $f \in \mathcal{E}$, then $D_1(f;z) = 0$ for every $z \in \{0,1\}^s$.
2. If $f \notin \mathcal{R}^D$, then $D_1(f;z) = 1$ for some $z \in \{0,1\}^s$.

Suppose that f is a YES instance of $(\mathcal{E} \text{ vs } \mathcal{D})$, that is, $f \in \mathcal{E}$. By the assumption, we have $G^f(z) \in \mathrm{Im}(H)$. Therefore, by the property of D, we obtain $D_1(f;z) = D\big(G^f(z)\big) = 0$.

Conversely, suppose that $D\big(G^f(z)\big) = 0$ for every $z \in \{0,1\}^s$. This means that D ϵ-avoids G^f. By the reconstruction property of G^f, we obtain $f \in \mathcal{R}^D$. Taking its contrapositive, it follows that if $f \notin \mathcal{R}^D$ then $D_1(f;z) = D\big(G^f(z)\big) = 1$ for some $z \in \{0,1\}^s$. This completes the proof of Claim 46.

Now consider a circuit D_2 defined as $D_2(f) := \bigwedge_{z\in\{0,1\}^s} \neg D_1(f;z)$. Then, it follows from Claim 46 that the circuit D_2 solves ($\mathcal{E}$ vs $\mathcal{R}^D$) on inputs of length N.

We move on to the case when $G^{(\text{-})}$ is a PRG construction. In this case, we claim that the second item of Claim 46 can be strengthened to the following: If $f \notin \mathcal{R}^D$, then $\Pr_{z\sim\{0,1\}^s}[D_1(f;z) = 1] \geq \frac{\epsilon}{2}$. We prove the contrapositive of this claim. Assume that $\Pr_{z\sim\{0,1\}^s}[D_1(f;z) = 1] < \frac{\epsilon}{2}$. Since D ϵ-avoids H, we have $\Pr_{w\sim\{0,1\}^m}[D(w) = 1] \geq \epsilon$. Therefore, D $\frac{\epsilon}{2}$-distinguishes $G^f(\text{-})$ from the uniform distribution; by the reconstruction of $G^{(\text{-})}$, we obtain $f \in \mathcal{R}^D$, as desired.

Therefore, $D_1(\text{-};z)$ is a one-sided-error randomized circuit that computes ($\mathcal{E}$ vs $\mathcal{R}^D$) with probability at least $\epsilon/2$. Now define a randomized circuit D_2' such that $D_2'(f) := \bigwedge_{i=1}^k \neg D_1(f;z_i)$, where $z_1, \cdots, z_k \sim \{0,1\}^s$ are chosen independently and k is a parameter chosen later. Then, it is easy to see that $D_2'(f) = 1$ for any $f \in \mathcal{E}$; on the other hand, for any $f \notin \mathcal{R}^D$, $D_2'(f) = 1$ with probability at most $(1-\epsilon/2)^k$, which is less than 2^{-N} by choosing $k = O(N/\epsilon)$ large enough. By using a union bound, one can hardwire random bits in D_2' as in Adleman's trick [1], and obtain a deterministic $\mathsf{AND} \circ \mathsf{NOT} \circ D \circ \mathfrak{C}$ circuit that computes ($\mathcal{E}$ vs $\mathcal{R}^D$). ◀

4.5 The Nisan–Wigderson Generator

The pseudorandom generator construction of Nisan and Wigderson [53] is particularly efficient. Indeed, each output bit of the Nisan–Wigderson generator depends on only 1 bit of the truth table of a candidate hard function.

▶ **Theorem 47** (Nisan–Wigderson Pseudorandom Generator Construction). *For every constant $\gamma > 0$ and any $\ell, m \in \mathbb{N}$, there exists a $\mathfrak{C}$-pseudorandom generator construction $G^{(\text{-})}\colon \{0,1\}^d \to \{0,1\}^m$ with $\mathcal{R}$-reconstruction and error parameter ϵ that takes a function $f\colon \{0,1\}^\ell \to \{0,1\}$ such that*

- *$d = O(\ell)$, $m \leq 2^\ell$,*
- *$\mathfrak{C} = \mathsf{NC}^0_1$, and*
- *$\mathcal{R}^D = D \circ \mathsf{AC}^0_2(m \cdot 2^{\gamma\ell}; \frac{1}{2} - \frac{\epsilon}{m})$.*

Moreover, the output $G^f(z)$ of the PRG can be computed in space $O(\ell)$ given a function f and a seed z as input.

We first recall the construction of the Nisan–Wigderson generator. In order to have a space-efficient algorithm for computing the Nisan–Wigderson generator, we use the following construction of a combinatorial design.

▶ **Lemma 48** (Klivans and van Melkebeek [44], Viola [71]). *For every constant $\gamma > 0$, for any $\ell \in \mathbb{N}$, there exist ℓ-sized subsets $S_1, \cdots, S_{2^\ell}$ of $[d]$ for some $d = O(\ell)$ such that*

1. *$|S_i \cap S_j| \leq \gamma \cdot \ell$ for every distinct $i, j \in [2^\ell]$, and*
2. *$S_1, \cdots, S_{2^\ell}$ can be constructed in space $O(\ell)$.*

A nearly disjoint generator ND is defined based on the design.

▶ **Definition 49.** *For a string $z \in \{0,1\}^d$ and a subset $S \subseteq [d]$ of indices, z_S denotes the string obtained by concatenating the ith bit z_i of z for every $i \in S$. For ℓ-sized subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ of $[d]$, define a* nearly disjoint generator ND$\colon \{0,1\}^d \to (\{0,1\}^\ell)^m$ *as* $\mathrm{ND}(z) := (z_{S_1}, \cdots, z_{S_m})$ *for every $z \in \{0,1\}^d$.*

Using the nearly disjoint generator, the Nisan–Wigderson generator is defined as follows.

▶ **Definition 50** (Nisan–Wigderson generator [53]). *For a Boolean function $f\colon \{0,1\}^\ell \to \{0,1\}$ and for ℓ-sized subsets $\mathcal{S} = \{S_1, \cdots, S_m\}$ of $[d]$, the Nisan–Wigderson generator $\mathrm{NW}^f\colon \{0,1\}^d \to \{0,1\}^m$ is defined as*

$$\mathrm{NW}^f(z) := f^m \circ \mathrm{ND}(z) = f(z_{S_1}) \cdots f(z_{S_m}).$$

It was shown in [53] that the pseudorandom generator construction is secure in the following sense.

▶ **Lemma 51** (Security of the Nisan–Wigderson Generator [53]). *For any functions $T\colon \{0,1\}^m \to \{0,1\}$ and $f\colon \{0,1\}^\ell \to \{0,1\}$, for any $\epsilon > 0$, if*

$$\Pr_{w\sim\{0,1\}^m}[T(w) = 1] - \Pr_{z\sim\{0,1\}^d}\left[T(\mathrm{NW}^f(z)) = 1\right] \geq \epsilon,$$

then there exists a one-query depth-2 oracle circuit $C \in \mathsf{AC}^0_2$ of size $O(m \cdot 2^{\gamma\ell})$ such that

$$\Pr_{x\sim\{0,1\}^\ell}\left[C^T(x) = f(x)\right] \geq \frac{1}{2} + \frac{\epsilon}{m}.$$

Proof of Theorem 47. We use the Nisan–Wigderson generator $\mathrm{NW}^{(\cdot)}$ defined in Definition 50 (i.e., we define $G^{(\cdot)} := \mathrm{NW}^{(\cdot)}$). $\mathrm{NW}^{(\cdot)}$ is an NC^0_1-PRG construction because each bit of $\mathrm{NW}^f(z)$ for any fixed seed z is equal to one bit of the truth table of f. The $\mathcal{R}$-reconstruction property of $\mathrm{NW}^{(\cdot)}$ follows from Lemma 51. ◀

4.6 Meta-Computational View of the Nisan–Wigderson Generator

Using the Nisan–Wigderson generator construction, we present a general connection between the existence of hitting set generators and lower bounds for meta-computational circuit lower-bound problems. We consider any family of circuits that is closed under taking projections in the following sense.

▶ **Definition 52.** *A class $\mathfrak{C}$ of circuits is said to be* closed under taking projections *if, for any $s \in \mathbb{N}$, for every size-s circuit $C \in \mathfrak{C}$ of n inputs, a circuit C' defined as $C'(x_1, \cdots, x_n) = C(x_{\sigma(1)}, \cdots, x_{\sigma(n)})$ for some function $\sigma\colon [n] \to [n]$ can be simulated by a size-s $\mathfrak{C}$-circuit.*

▶ **Theorem 53.** *Let $\mathfrak{C}$ be any circuit class that is closed under taking projections. Suppose that $(\mathsf{E}$ vs $\widetilde{\mathfrak{C} \circ \mathsf{AC}^0_2}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n}))$ $\notin$ i.o.$\mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathfrak{C}$ circuits.*

Proof. We prove the contrapositive. Assume that, for every function $s(m) = O(\log m)$, there exists a linear-size $\mathfrak{C}$ circuit D that avoids the universal hitting set generator Ht^s_m for infinitely many $m \in \mathbb{N}$. Given arbitrary constants $c, \alpha, \beta > 0$, we will choose a small constant $\gamma > 0$, and define $s(m) := c' \log m/\gamma$ for some large constant c'. Then using D that avoids Ht^s_m, we present a $\mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C}$-circuit of size $N^{1+\beta}$ that solves the E_c vs $\widetilde{\mathfrak{C} \circ \mathsf{AC}^0_2}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$ problem.

Let $f\colon \{0,1\}^n \to \{0,1\}$ denote the input of the MCLP. We use the Nisan–Wigderson generator construction $\mathrm{NW}^f\colon \{0,1\}^d \to \{0,1\}^m$ of Theorem 47, where $d = O(n)$, $m = 2^{\gamma n}$, and $\epsilon := \frac{1}{4}$. By Theorem 47, this is a black-box NC^0_1-PRG construction with $\widetilde{\mathsf{AC}^0_2}(2^{2\gamma n}; \frac{1}{2} - \frac{1}{4m})$ reconstruction.

In order to apply Theorem 45, we claim that $\mathrm{NW}^f(z) \in \mathrm{Im}(\mathrm{Ht}^s_m)$ for every $f \in \mathsf{E}_c$ and every $z \in \{0,1\}^d$. By Theorem 47, the output $\mathrm{NW}^f(z)$ can be described by using a seed z and a description for x in time $N^{O(1)}$, and thus

$$\mathrm{Kt}(\mathrm{NW}^f(z)) \leq |z| + \mathrm{Kt}(f) + O(\log N) = O(n).$$

In particular, for a large enough constant c', we have

$$\mathrm{Kt}(\mathrm{NW}^f(z)) \leq c'n = s(m),$$

By the universality of Ht^s_m (Proposition 38) we obtain that $\mathrm{NW}^f(z) \in \mathrm{Im}(\mathrm{Ht}^s_m)$.

By applying Theorem 45, we have $(\mathsf{E}_c \text{ vs } D \circ \widetilde{\mathsf{AC}^0_2}(2^{2\gamma n}; \frac{1}{2} - \frac{1}{4m}))_N \in \mathsf{AND}_{O(N)} \circ \mathsf{NOT} \circ D \circ \mathsf{NC}^0_1$. Since D is a $\mathfrak{C}$-circuit of size $m = 2^{\gamma n}$, it follows that $(\mathsf{E}_c \text{ vs } \mathfrak{C} \circ \widetilde{\mathsf{AC}^0_2}(2^{O(\gamma n)}; \frac{1}{2} - \frac{1}{4m}))_N \in \mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C}(O(N^{1+\gamma}))$. Note that this holds for infinitely many $N = m^{1/\gamma}$. By choosing γ small enough depending on α, β, the result follows. ◀

We observe that the converse direction also holds. In particular, for any circuit class $\mathfrak{C} \subseteq \mathsf{P/poly}$ such that $\mathfrak{C}$ is closed under taking projections and $\mathsf{AND} \circ \mathsf{NOT} \circ \mathfrak{C} = \mathfrak{C}$, our reductions in fact establish the equivalence between the existence of a hitting set generator secure against $\mathfrak{C}$ and a $\mathfrak{C}$-lower bound for the E vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ problem.

▶ **Proposition 54** (Converse of Theorem 53). *Let $\mathfrak{C}$ be any circuit complexity class. Suppose that there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathfrak{C}$ circuits. Then, for any constant $k \in \mathbb{N}$, for all sufficiently large $N \in \mathbb{N}$, no $\mathsf{NOT} \circ \mathfrak{C}$ circuit of size N^k can solve $\left(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})\right)_N$ for $\alpha := 1/4$ nor $\mathrm{MKtP}[O(\log N), N-1]$ on input length N.*

Proof. We first observe that $\mathrm{MKtP}[O(\log N), N-1]$ is reducible to $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n}))$ via an identity map: Take any function $f \in \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})$. Since the truth table of f can be described by a circuit of size N^α and $\log \binom{N}{\leq N/2 - N^{1-\alpha}}$ bits of information in time $N^{O(1)}$, the Kt-complexity of $\mathtt{tt}(f)$ is at most

$$\widetilde{O}(N^\alpha) + N \cdot \mathrm{H}_2(1/2 - N^{-\alpha}) + O(\log N) \leq N - \Omega(N^{1-2\alpha}),$$

which is much smaller than $N - 1$. Therefore, it suffices to prove the result only for $\mathrm{MKtP}[O(\log N), N-1]$.

We prove the contrapositive. Suppose that, for some constant $k \in \mathbb{N}$, for any constant c, for infinitely many $N \in \mathbb{N}$, there exists a $\mathsf{NOT} \circ \mathfrak{C}$ circuit of size N^k that solves the promise problem $\mathrm{MKtP}[c \log N, N-1]$. Consider any family of functions $G = \{G_m : \{0,1\}^{d \log m} \to \{0,1\}^m\}_{m \in \mathbb{N}}$ computable in time m^d for a constant d. Let $c := 4kd$, and take a $\mathsf{NOT} \circ \mathfrak{C}$ circuit $\neg C$ of size N^k that solves $\mathrm{MKtP}[c \log N, N-1]$ on inputs of length N.

We regard C as a circuit that takes $m := N^k$ input bits by ignoring $m - N$ input bits, and in what follows we claim that the linear-size circuit C avoids G_m. For a string $w \in \{0,1\}^m$, denote by $w\restriction_N$ the first N bits of w.

Let $z \in \{0,1\}^{d \log m}$ be any seed of G. Since $G(z)\restriction_N$ can be described by $N \in \mathbb{N}$ and $z \in \{0,1\}^{d \log m}$ in time m^d, its Kt complexity is

$$\mathrm{Kt}(G(z)\restriction_N) \leq \log N + |z| + d \log m + o(\log m) \leq 4kd \log N,$$

which means that $G(z)\restriction_N$ is a YES instance of $\mathrm{MKtP}[c \log N, N-1]$ and thus $G(z)$ is rejected by C.

Now consider a string $w \sim \{0,1\}^m$ chosen uniformly at random. By a standard counting argument, $\mathrm{Kt}(w{\restriction_N}) \geq N-1$ with probability at least $\frac{1}{2}$; thus C accepts at least a half of all inputs. Therefore, the function G is not secure against C. ◀

Applying Theorem 53 to depth-d circuits $\mathfrak{C} := \mathsf{AC}^0_d$, we obtain the following.

▶ **Corollary 55.** *Let d be a constant. Suppose that $(\mathsf{E} \text{ vs } \widetilde{\mathsf{AC}^0_{d+2}}(2^{\alpha n}; \frac{1}{2} - 2^{-\alpha n})) \notin \mathsf{i.o.AC}^0_{d+1}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.*

This means that, in order to obtain a nearly optimal hitting set generator for AC^0_d, it suffices to prove that nearly-linear-size AC^0_{d+1} circuits cannot distinguish the truth tables of functions in $\mathsf{E}/O(n)$ from the truth tables of functions that cannot be approximated by AC^0_{d+2} circuits.

We present a proof of Theorem 13.

▶ **Restatement of Theorem 13.** *The following are equivalent.*

1. *For any constants d, d', there exists a constant $\beta > 0$ such that*

$$(\mathsf{E} \text{ vs } \widetilde{\mathsf{AC}^0_{d'}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})) \notin \mathsf{i.o.AC}^0_d(N^{1+\beta}).$$

2. *For any constant d, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.*
3. *For any constant d, there exist constants $c, \beta > 0$ such that*

$$\mathrm{MKtP}[c \log N, N-1] \notin \mathsf{i.o.AC}^0_d(N^{1+\beta}).$$

4. *For any constants d, k, there exists a constant $c > 0$ such that*

$$\mathrm{MKtP}[c \log N, N-1] \notin \mathsf{i.o.AC}^0_d(N^k).$$

Proof of Theorem 13. Item 1 $\implies$ Item 2 follows from Corollary 55. Item 2 $\implies$ Item 4 follows from Proposition 54. Item 4 $\implies$ Item 3 is obvious. Item 3 $\implies$ Item 1 holds because the truth table of any function in $\widetilde{\mathsf{AC}^0}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)})$ has Kt complexity less than $N-1$. ◀

4.7 Hardness Amplification and MCLPs

The main advantage of studying MCLPs is that hardness amplification can be naturally regarded as a reduction between two different MCLPs. In this section, we present such reductions.

Impagliazzo and Wigderson [40] gave a derandomized hardness amplification theorem. We use the following generalized version of their result.

▶ **Theorem 56** (Derandomized hardness amplification)**.** *Let $\gamma > 0$ be an arbitrary constant. There exists a hardness amplification procedure* Amp *that takes a function $f\colon \{0,1\}^n \to \{0,1\}$ and parameters $\delta, \epsilon > 0$, and returns a Boolean function $\mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$ satisfying the following:*

1. *If $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; 1/2 - \epsilon) \leq s$, then $\widetilde{\mathrm{size}}(f; \delta) \leq s \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$.*
2. *For any fixed $y \in \{0,1\}^{O(n+\log(1/\epsilon))}$, there exist strings $v_1, \cdots, v_k \in \{0,1\}^n$ for some $k = O(\log(1/\epsilon)/\delta)$ such that $\mathrm{Amp}^f_{\epsilon,\delta}(y) = f(v_1) \oplus \cdots \oplus f(v_k)$. Moreover, if y is distributed uniformly at random, for each $i \in [k]$, v_i is distributed uniformly at random.*
3. *$\mathrm{Amp}^f_{\epsilon,\delta}(y)$ can be computed in $O(n + \log(1/\delta) + \log(1/\epsilon))$ space, given f, y, ϵ and δ as an input.*

Theorem 56 slightly differs from derandomized hardness amplification theorems of [40, 30] in that we are also interested in the case when the hardness parameter δ is $o(1)$, which is not required in a standard application of derandomized hardness amplification theorems. We defer a proof of Theorem 56 to Appendix A.

Using Theorem 56, we give a reduction among different MCLPs.

▶ **Theorem 57.** *For any constants $\alpha, \delta > 0$, for all sufficiently small $\beta > 0$, there exists a $\mathsf{XOR}_{O(\beta \log N)}$-computable reduction from $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$ to $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$.*

Proof. The reduction is to simply take the hardness amplification procedure Amp of Theorem 56. Specifically, given the truth table of a function $f\colon \{0,1\}^n \to \{0,1\}$, the reduction maps f to the truth table of $\mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{n'} \to \{0,1\}$, where $\epsilon := 2^{-\beta n}$ and $n' = O(n)$. By the second item of Theorem 56, each output of the reduction is computable by a XOR of k bits, where $k = O(\log(1/\epsilon)/\delta) = O(\beta n)$.

We claim the correctness of the reduction. Suppose that $\mathrm{Kt}(f) \leq O(\log N)$. Then, by the third item of Theorem 56, we obtain that $\mathrm{Kt}(\mathrm{Amp}^f_{\epsilon,\delta}) \leq O(\log N)$.

Conversely, suppose that $f \notin \widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)$; that is, $\widetilde{\mathrm{size}}(f; \delta) > 2^{\alpha n} > 2^{\beta n} \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$, where the last inequality holds by choosing $\beta > 0$ small enough. By the first item of Theorem 56, we obtain that $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; \frac{1}{2} - 2^{-\beta n}) > 2^{\beta n}$, which implies that $\mathrm{Amp}^f_{\epsilon,\delta} \notin \widetilde{\mathsf{SIZE}}(2^{\beta' n'}; \frac{1}{2} - 2^{-\beta' n'})$ for some constant $\beta' > 0$. ◀

Applying the reduction of Theorem 57 to Corollary 55, we obtain the following.

▶ **Corollary 58.** *Let $d \in \mathbb{N}, \delta > 0$ be constants. Suppose that $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)) \notin \mathsf{i.o.AC}^0_{d+2}(N^{1+\beta})$ for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size AC^0_d circuits.*

Proof. We prove the contrapositive. Under the assumption that no hitting set generator is secure against AC^0_d, it follows from Corollary 55 that $(\mathsf{E}$ vs $\widetilde{\mathsf{AC}^0_{d+2}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n})) \in \mathsf{i.o.AC}^0_{d+1}(N^{1+\gamma})$ for any constants $\beta, \gamma > 0$. Our goal is to prove that $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta)) \in \mathsf{i.o.AC}^0_{d+2}(N^{1+\eta})$ for any constants $\alpha, \eta > 0$.

Fix any constants $\alpha, \eta > 0$. By Theorem 57, there exists a $\mathsf{XOR}_{O(\beta n)}$-computable reduction from $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$ to $(\mathsf{E}$ vs $\widetilde{\mathsf{AC}^0_{d+2}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$, for all sufficiently small $\beta > 0$. The latter problem can be solved by an $\mathsf{AC}^0_{d+1}(N^{1+\gamma})$. Thus we obtain an $\mathsf{AC}^0_{d+1}(N^{1+\gamma}) \circ \mathsf{XOR}_{O(\beta n)}$ circuit that computes the former problem. Since $\mathsf{XOR}_{O(\beta n)}$ can be computed by a depth-2 circuit of size $N^{O(\beta)}$, by merging one bottom layer of the AC^0_{d+1} circuit, we obtain a circuit in $\mathsf{AC}^0_{d+2}(N^{1+\gamma+O(\beta)})$ that computes $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\alpha n}; \delta))$. The result follows by choosing $\beta, \gamma > 0$ small enough depending on $\eta > 0$. ◀

Note that AC^0 circuits are not capable of computing XOR gates of large fan-in. If a computational model can compute XOR gates, it is possible to compute a locally-decodable error-correcting code. Specifically, we provide an efficient reduction from $(\mathsf{E}$ vs $\mathsf{SIZE}(2^{o(n)}))$ to $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{o(n)}; \frac{1}{2} - 2^{-o(n)}))$ that is computable by a single layer of XOR gates.

▶ **Theorem 59** (Reductions by Error-Correcting Codes)**.** *For any constants $\alpha, \gamma > 0$, for all sufficiently small $\beta > 0$, there exists a reduction from $(\mathsf{E}$ vs $\mathsf{SIZE}(2^{\alpha n}))$ to $(\mathsf{E}$ vs $\widetilde{\mathsf{SIZE}}(2^{\beta n}; \frac{1}{2} - 2^{-\beta n}))$ that is computable by one layer of $O(N^{1+\gamma})$ XOR gates.*

▶ **Lemma 60** (cf. [63, 69, 73]). *For any small constant $\beta > 0$, for all large $n \in \mathbb{N}$, there exists an error-correcting code* Enc^f *that encodes a function* $f\colon \{0,1\}^n \to \{0,1\}$ *as a function* $\mathrm{Enc}^f\colon \{0,1\}^{(1+O(\sqrt{\beta}))n} \to \{0,1\}$ *satisfying following:*

1. $\mathrm{size}(f) \leq \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta}n)}$.
2. *For any fixed* $y \in \{0,1\}^{(1+O(\sqrt{\beta}))n}$, $\mathrm{Enc}^f(y)$ *can be computed by an* XOR *of some bits of the truth table of* f.
3. Enc^f *can be computed in time* $2^{O(n)}$.

Proof Sketch. We use a Reed-Muller code concatenated with a Hadamard code. The crux is that the length of a codeword of can be made small because the query complexity of local-list-decoding algorithms is allowed to be quite large.

Let $f\colon \{0,1\}^n \to \{0,1\}$. Let $\mathbb{F}_q$ be a finite field, where $q = 2^k$ for some k. Pick $H \subseteq \mathbb{F}$, and encode any element of $\{0,1\}^n$ as an element of H^t by taking an injection η from $\{0,1\}^n$ to H^t, where t is some large constant chosen later. Let the size $|H|$ of H be $2^{n/t}$. Any $f\colon \{0,1\}^n \to \{0,1\}$ can be encoded as a unique low-degree extension $\widehat{f}\colon \mathbb{F}_q^t \to \mathbb{F}_q$ such that $\widehat{f}$ and $f \circ \eta$ agree on H^q. The total degree of $\widehat{f}$ is at most $d := t|H| = t2^{n/t}$. We will set $q = t2^{n/t+O(\beta n)}$.

Then, each alphabet $\widehat{f}(x)$ in the Reed-Muller code is encoded with a Hadamard code. Namely, $\mathrm{Enc}^f(x,y) := \langle \widehat{f}(x), y\rangle$, where $x \in \mathbb{F}_q^t$, $y \in \mathbb{F}_2^k$ and $\langle \text{-}, \text{-}\rangle$ denotes the inner product function over $\mathbb{F}_2$. The length of the truth table of Enc^f is at most $q^{t+1} = O(t^{t+1}2^{(n/t+O(\beta n))(t+1)})$. By choosing $t := 1/\sqrt{\beta}$, this is bounded by $2^{(1+O(\sqrt{\beta}))\cdot n}$.

Sudan, Trevisan, and Vadhan [63] gave a local list-decoding algorithm for the code Enc^f running in time $\mathsf{poly}(t,q)$ that can handle a $\left(\frac{1}{2} - (d/q)^{\Omega(1)}\right)$-fraction of errors, which is more than $\frac{1}{2} - 2^{-\beta n}$ by choosing $q := t2^{n/t+O(\beta n)}$ large enough. Given a circuit that approximates Enc^f on a $\frac{1}{2} - 2^{-\beta n}$ fraction of inputs, one can apply the local list-decoding algorithm to obtain a circuit that computes f on every input; thus we have $\mathrm{size}(f) \leq \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta})n}$. ◀

Proof of Theorem 59. We apply the error-correcting code Enc of Theorem 59. Specifically, given $f\colon \{0,1\}^n \to \{0,1\}$ as input, we map f to $\mathrm{Enc}^f\colon \{0,1\}^{n'} \to \{0,1\}$. Since the length of $\mathtt{tt}(\mathrm{Enc}^f)$ is $2^{n'} = N^{1+O(\sqrt{\beta})}$ and each bit is computable by a XOR of some bits of $\mathtt{tt}(f)$, the reduction is computable by one layer of $O(N^{1+\gamma})$ XOR gates for all sufficiently small $\beta > 0$.

We claim the correctness of the reduction. Suppose that $\mathrm{Kt}(f) = O(\log N)$; then, by the third item of Lemma 60, we have $\mathrm{Kt}(\mathrm{Enc}^f) = O(\log N)$.

Now suppose that $f \notin \mathsf{SIZE}(2^{\alpha n})$. By Lemma 60, we have $2^{\alpha n} < \mathrm{size}(f) \leq \widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) \cdot 2^{O(\sqrt{\beta}n)}$. Therefore, we obtain that $\widetilde{\mathrm{size}}(\mathrm{Enc}^f; \frac{1}{2} - 2^{-\beta n}) > 2^{(\alpha - O(\sqrt{\beta}))n} \geq 2^{\beta n}$, where the last inequality holds by choosing $\beta > 0$ small enough. Therefore, $\mathrm{Enc}^f \notin \widetilde{\mathsf{SIZE}}(2^{\beta' n'}; \frac{1}{2} - 2^{-\beta' n'})$ holds for some constant $\beta' > 0$. ◀

Applying the reduction, we obtain the following.

▶ **Theorem 61.** *Let d be a constant. Suppose that* $(\mathsf{E}$ vs $\mathsf{SIZE}(2^{\alpha n})) \notin \mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR}(N^{1+\beta})$ *for some constants $\alpha, \beta > 0$. Then, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size* $\mathsf{AC}^0_d \circ \mathsf{XOR}$ *circuits.*

Proof. We prove the contrapositive. Let $\alpha, \beta > 0$ be arbitrary constants. Assuming that no hitting set generator is secure against $\mathsf{AC}^0_d \circ \mathsf{XOR}$, by Theorem 53, we have $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha_0 n}; \frac{1}{2} - 2^{-\alpha_0 n})) \in \mathsf{i.o.AC}^0_{d+1} \circ \mathsf{XOR}(N^{1+\beta_0})$ for any constants $\alpha_0, \beta_0 > 0$. By Theorem 59, $(\mathsf{E} \text{ vs } \mathsf{SIZE}(2^{\alpha n}))$ is reducible to $(\mathsf{E} \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha_0 n}; \frac{1}{2} - 2^{-\alpha_0 n}))$ by one layer of $O(N^{1+\gamma})$ XOR gates for any constant $\gamma > 0$ and any small enough constant $\alpha_0 > 0$. Therefore, we obtain a circuit in $\mathsf{AC}^0_{d+1} \circ \mathsf{XOR} \circ \mathsf{XOR}((N^{1+\gamma})^{1+\beta_0})$ that computes $(\mathsf{E} \text{ vs } \mathsf{SIZE}(2^{\alpha n}))$. By merging the bottom two XOR layers, the circuit can be written as an $\mathsf{AC}^0_{d+1} \circ \mathsf{XOR}$ circuit of size $N^{1+\gamma+\beta_0+\gamma\beta_0}$.[7] The result follows by choosing positive constants $\gamma, \beta_0 \ll \beta$ small enough. ◀

We are now ready to complete a proof of Theorem 11, which establishes the equivalence between the existence of a hitting set generator secure against $\mathsf{AC}^0 \circ \mathsf{XOR}$ and the $\mathsf{AC}^0 \circ \mathsf{XOR}$ circuit lower bound for the E vs $\mathsf{SIZE}(2^{o(n)})$ problem.

▶ **Restatement of Theorem 11.** *The following are equivalent.*

1. *For any constant d, there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in time $n^{O(1)}$ and secure against linear-size $\mathsf{AC}^0_d \circ \mathsf{XOR}$ circuits.*
2. *For any constant d, for some constant $\beta > 0$, $(\mathsf{E} \text{ vs } \mathsf{SIZE}(2^{o(n)})) \notin \mathsf{i.o.AC}^0_d(N^{1+\beta})$.*
3. *For any constant d, for some constant $\beta > 0$, $\mathrm{MKtP}[O(\log N), N^{o(1)}] \notin \mathsf{i.o.AC}^0_d \circ \mathsf{XOR}(N^{1+\beta})$.*
4. *For any constants $d, k \in \mathbb{N}$, $\mathrm{MKtP}[O(\log N), N^{o(1)}] \notin \mathsf{i.o.AC}^0_d \circ \mathsf{XOR}(N^k)$.*

Proof of Theorem 11. The implications from Item 4 to Item 3 and from Item 3 to Item 2 are trivial. The implication from Item 2 to Item 1 immediately follows from Theorem 61. The implication from Item 1 to Item 4 is a standard approach for showing a lower bound for MKtP, and follows from Proposition 54. ◀

4.8 KS Complexity and Read-Once Branching Program

We now turn our attention to KS complexity. This amounts to considering a hitting set generator that is computable in a limited amount of space. Applying our proof ideas to the case of read-once branching programs, we provide a potential approach for resolving $\mathsf{RL} = \mathsf{L}$.

▶ **Theorem 62.** *There exists a universal constant $\rho > 0$ satisfying the following. Suppose that, for some constants $\alpha, \beta > 0$, $(\mathsf{DSPACE}(n) \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; 2^{-\rho\alpha n}))$ cannot be computed by a read-once co-nondeterministic branching program of size $N^{1+\beta}$ for all large input length $N \in \mathbb{N}$. Then there exists a hitting set generator $G = \{G_n : \{0,1\}^{O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ computable in $O(\log n)$ space and secure against linear-size read-once branching programs.*

Since the class of read-once branching programs is not closed under taking several reductions presented so far, we provide a self-contained proof below.

Proof. We prove the contrapositive of Theorem 62 for the universal hitting set generator HS. Assume that, for every function $s(m) = O(\log m)$, there exists a linear-size read-once branching program D that avoids HS^s_m for infinitely many $m \in \mathbb{N}$. Given arbitrary constants $c, \alpha, \beta > 0$, we will choose a small constant $\gamma > 0$, and define $s(m) := c' \log m/\gamma$ for some large constant c'. Then using D that avoids HS^s_m, we present a coRP-type randomized read-once branching program that solves $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) := (\mathsf{DSPACE}(cn)/^n cn \text{ vs } \widetilde{\mathsf{SIZE}}(2^{\alpha n}; 2^{-\rho\alpha n}))$ on inputs of length $N = 2^n$, for some sufficiently large $N := m^{1/\gamma}$. Here, a *randomized* branching program means a probability distribution on branching programs.

[7] Recall that we count the number of gates except for input gates.

For simplicity, we first explain a construction of a branching program that may not be read-once. A randomized branching program D_0 is defined as follows. Given an input $f\colon \{0,1\}^n \to \{0,1\}$, set $\epsilon := 1/4m$. Define $\widetilde{f} := \mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(\log N)} \to \{0,1\}$. Let ℓ denote the input length of $\widetilde{f}$, and let $d = O(\ell) = O(n)$ denote the seed length of the Nisan–Wigderson generator instantiated with ℓ-sized m subsets (Definition 50). Pick $z \sim \{0,1\}^d$ and output $\neg D(\mathrm{NW}^{\widetilde{f}}(z))$. This is a randomized branching program because for each fixed z, each bit of $\mathrm{NW}^{\widetilde{f}}(z)$ is equal to $\widetilde{f}(z_S)$ for some subset S, and hence by Item 2 of Theorem 56 it is some linear combination of at most k bits of $\mathtt{tt}(f)$, which can be computed by a read-once width-2 branching program of size $O(k)$. Thus $\neg D(\mathrm{NW}^{\widetilde{f}}(z))$ can be implemented as a branching program of size $O(m \cdot k)$, by replacing each node of D by the read-once width-2 branching programs that compute some linear combinations of $\mathtt{tt}(f)$.

We claim the correctness of the randomized branching program D_0.

▷ Claim 63.

1. D_0 accepts every $f \in \Pi_{\mathrm{YES}}$ with probability 1.
2. For every $f \in \Pi_{\mathrm{NO}}$, the probability that D_0 rejects f is at least $\frac{1}{4}$.
3. The size of D_0 is at most N^β.

Take any YES instance $f \in \Pi_{\mathrm{YES}}$. We observe that the output $\mathrm{NW}^{\widetilde{f}}(z)$ of the generator has small KS complexity. Indeed,

$$\mathrm{KS}(\mathrm{NW}^{\widetilde{f}}(z)) \le |z| + \mathrm{KS}(\widetilde{f}) + O(\log N) \le \mathrm{KS}(f) + O(\log N),$$

where we used Lemma 48 and Item 3 of Theorem 56. In particular, for a large enough constant c', we have

$$\mathrm{KS}(f) \le c' \log N = s(m),$$

and thus $D(\mathrm{NW}^{\widetilde{f}}(z)) = 0$ by Proposition 35 and the assumption that D avoids HS^s_m. This means that the algorithm D_0 accepts for every choice of z.

Conversely, suppose that the algorithm D_0 accepts some input f with probability at least $\frac{3}{4}$. We claim that $f \notin \Pi_{\mathrm{NO}}$. The assumption means that $\Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 0] \ge \frac{3}{4}$, which is equivalent to saying that $\Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 1] \le \frac{1}{4}$. On the other hand, since D avoids HS^s_m, we have $\Pr_w[D(w) = 1] \ge \frac{1}{2}$. In particular, we obtain

$$\Pr_w[D(w) = 1] - \Pr_z[D(\mathrm{NW}^{\widetilde{f}}(z)) = 1] \ge \frac{1}{4}.$$

By the security proof of the Nisan–Wigderson generator (Lemma 51), there exists a one-query oracle circuit C of size $O(m \cdot 2^{\gamma\ell})$ such that

$$\Pr_{y \sim \{0,1\}^\ell}[C^D(y) = \widetilde{f}(y)] \ge \frac{1}{2} + \frac{1}{4m}.$$

Now replacing the oracle gate of C with a circuit that simulates D, we obtain a circuit of size $m^{O(1)} + m \cdot N^{O(\gamma)}$. By the property of $\mathrm{Amp}_{\epsilon,\delta}$ (Item 1 of Theorem 56), we obtain another circuit C' of size $m^{O(1)} \cdot N^{O(\gamma)} \cdot (1/\delta)^{O(1)}$ such that

$$\Pr_{x \sim \{0,1\}^n}[C'(x) = f(x)] \ge 1 - \delta.$$

Choosing $\delta := N^{-\gamma}$, we obtain that $\widetilde{\mathrm{size}}(f; N^{-\gamma}) \le N^{O(\gamma)}$, where $\gamma > 0$ is an arbitrary small constant. This completes the proof of the second item of Claim 63, by choosing γ small enough so that $O(\gamma) < \alpha$.

Recall that the size of D_0 is $O(m \cdot k)$. Here k is the parameter from Theorem 56 and $k = O(\log(1/\epsilon)/\delta) = N^{O(\gamma)}$. Thus the size of D_0 is at most $N^{O(\gamma)} \leq N^{\beta}$, by choosing γ small enough so that $O(\gamma) \leq \beta$. This completes the proof of Claim 63.

Now we modify the construction of D_0 in order to obtain a *read-once* branching program D_1. The branching program $D_0(\mathrm{NW}^{\widetilde{f}}(z))$ may read some x-th bit of the input $\mathtt{tt}(f)$ twice only if there exists a pair of distinct indices (i, j) such that the ith bit of $\mathrm{NW}^{\widetilde{f}}(z)$ and the jth bit of $\mathrm{NW}^{\widetilde{f}}(z)$ are linear combinations of $\mathtt{tt}(f)$ that contain $f(x)$. We say that a coin flip z is *bad* if this happens. To ensure that a coin flip is bad with small probability, we take a pairwise independent generator $G_2 \colon \{0,1\}^{O(\ell)} \to (\{0,1\}^{\ell})^m$, and define a modified Nisan–Wigderson generator $\mathrm{NW}'^{\widetilde{f}} := \widetilde{f}^m \circ (\mathrm{ND} \oplus G_2)$, where $\mathrm{ND} \oplus G_2$ denotes the function such that $\mathrm{ND} \oplus G_2(u, v) := \mathrm{ND}(u) \oplus G_2(v)$. Using this modified construction, the read-once branching program D_1 is defined as $\neg D(\mathrm{NW}'^{\widetilde{f}}(z))$ if z is not bad, and otherwise defined as a trivial branching program that outputs 1 always. (Note here that since we deal with a non-uniform computation, one does not need to check the badness of z by using a branching program.) By the definition, it is obvious that D_1 is read-once; hence it remains to claim that D_1 satisfies the promise of $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$.

As in the case of D_0, for $f \in \Pi_{\mathrm{YES}}$, it can be seen that $D(\mathrm{NW}'^{\widetilde{f}}(z)) = 0$ for every z, and hence D_1 always accepts. (We note that the KS complexity increases by an additive term of the input length of G_2, which is $O(\log N)$.) Conversely, we claim that if D_1 accepts with probability at least $\frac{7}{8}$, then $f \notin \Pi_{\mathrm{NO}}$. Assume that $\Pr_z[D_1 \text{ accepts}] \geq \frac{7}{8}$. We claim that the probability that z is bad is small: Fix any distinct indices $(i, j) \in [m]^2$. Recall that by Item 2 of Theorem 56 for each fixed $y \in \{0,1\}^{\ell}$, there exist inputs $v_1^i, \cdots, v_k^i$ of f such that $\widetilde{f}(y)$ can be written as a linear combination of $f(v_1^i), \cdots, f(v_k^i)$, where $k = O(\log(1/\epsilon)/\delta)$. Fix any indices $i', j' \in [k]$. Then the probability that $v_{i'}^i = v_{j'}^j$ is at most $1/N$ because of the pairwise independence of G_2 and each $v_{i'}^i$ is uniformly distributed. By the union bound, the probability that z is bad is bounded above by $(km)^2 \cdot 1/N \leq N^{O(\gamma)-1} \leq \frac{1}{8}$, for a sufficiently small $\gamma > 0$ and a large $N \in \mathbb{N}$. Thus we obtain

$$\Pr_z[D(\mathrm{NW}'^{\widetilde{f}}(z)) = 0] \geq \Pr[D_1 \text{ accepts}] - \Pr[z \text{ is bad}] \geq \frac{3}{4}.$$

The rest of a proof of the correctness is essentially the same with the case of D_0, observing that the security proof of the Nisan–Wigderson generator also works for the modified version NW'.

Finally, we convert the randomized read-once branching program D_1 of size N^{β} into a co-nondeterministic read-once branching program of size $N^{1+\beta}$. This can be done by using the standard Adleman's trick [1]: Specifically, the success probability of D_1 can be amplified to $1 - 2^{-N}$ by taking AND of $O(N)$ independent copies of D_1. By the union bound, there exists a good coin flip sequence such that AND of $O(N)$ copies of D_1 solves $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ on every input of length N. Hard-wiring such a coin flip sequence and simulating the AND gate by using a co-nondeterministic computation, we obtain a co-nondeterministic read-once branching program of size $O(N^{1+\beta})$. ◀

5 Non-trivial Derandomization and MKtP

In this section, we provide a characterization of non-trivial derandomization for uniform algorithms by a lower bound for MKtP. We start with a formal definition of non-trivial derandomization for uniform algorithms.

▶ **Definition 64** (Non-trivial derandomization). *An algorithm A is said to be a* derandomization algorithm for $\mathsf{DTIME}(t(n))$ that runs in time $s(n)$ *if the following hold. The algorithm takes 1^n and a description of a machine M and outputs an n-bit string $A(1^n, M) \in \{0,1\}^n$ in time $s(n)$. For any machine M running in time $t(n)$ on inputs of length $n \in \mathbb{N}$, there exist infinitely many $n \in \mathbb{N}$ such that, if $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$, then $M(A(1^n, M)) = 1$.*

In the following, for a function $s\colon \mathbb{N} \to \mathbb{N}$, we denote by R^s_{Kt} the set $\{\, x \in \{0,1\}^* \mid \mathrm{Kt}(x) \geq s(|x|) \,\}$ of Kt-random strings with threshold s. We say that a set $R \subseteq \{0,1\}^*$ is dense if $\Pr_{x\sim\{0,1\}^n}[x \in R] \geq \frac{1}{2}$ for all large $n \in \mathbb{N}$.

▶ **Proposition 65.** *The following are equivalent for any time-constructible functions t, s such that $t(n), s(n) \geq n$.*

1. *There exists a derandomization algorithm for $\mathsf{DTIME}(t(n))$ that runs in time $2^{s(n)+O(\log t(n))}$.*
2. *Any dense subset of $R^{s(n)+O(\log t(n))}_{\mathrm{Kt}}$ cannot be accepted by any $t(n)$-time algorithm.*

Proof. (Item 1 $\implies$ Item 2) Let A be a derandomization algorithm for $\mathsf{DTIME}(t(n))$. Since A runs in time $2^{s(n)+O(\log t(n))}$ on input $(1^n, M)$, the Kt-complexity of $A(1^n, M)$ is at most $s(n) + O(\log t(n)) + |M|$.

Let M be any $t(n)$-time algorithm such that $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$ for all large $n \in \mathbb{N}$. By the property of A, we have $M(A(1^n, M)) = 1$ for infinitely many n. This means that M accepts the string $A(1^n, M)$ that has Kt-complexity at most $s(n) + O(\log t(n))$; therefore, M does not accept a dense subset of Kt-random strings.

(Item 2 $\implies$ Item 1) Let A be the algorithm that takes $(1^n, M)$, enumerates all the strings x whose Kt-complexity is at most $s(n) + O(\log t(n))$, and, for each string x with small Kt-complexity, simulates M on input x; if M accepts x in time $t(n)$, output x and halt. The running time of A is clearly at most $2^{s(n)+O(\log t(n))}$. To prove the correctness, assume towards a contradiction that some algorithm M runs in time $t(n)$ and for all large n, $\Pr_{x\sim\{0,1\}^n}[M(x) = 1] \geq \frac{1}{2}$ and $M(A(1^n, M)) = 0$. The latter condition means that A cannot find a string x that is accepted by M; thus, M rejects all the strings x whose Kt-complexity is at most $s(n) + O(\log t(n))$. Therefore, M accepts a dense subset of Kt-random strings, which is a contradiction. ◀

▶ **Theorem 66.** *Let $t, s\colon \mathbb{N} \to \mathbb{N}$ be time-constructible functions such that t is non-decreasing and $\omega(\log^2 n) \leq s(n) \leq n$ and $\mathsf{poly}(n) \leq t(n)$ for all large n, where poly is some universal polynomial.*

For any constant $c > 0$, there exists a constant c' such that, if there is a $t(n)$-time algorithm that accepts a dense subset of $R^{n-c\sqrt{n}\log n}_{\mathrm{Kt}}$, then the following promise problem can be solved in $\mathsf{DTIME}(O(t(2s(n)) \cdot 2^{\sqrt{s(n)}\log n})) \cap \mathsf{coRTIME}(O(t(2s(n))))$.

Yes: *strings x such that $\mathrm{Kt}(x) \leq s$, where $s := s(|x|)$.*

No: *strings x such that $\mathrm{K}^{t'}(x) > s + c'\sqrt{s}\log n$, where $s := s(|x|)$ and $t' = t(2s) \cdot \mathsf{poly}(|x|)$.*

▶ **Lemma 67** (cf. [31]). *For any $d, m \leq n, \epsilon > 0$, there exists a black-box pseudorandom generator construction $G^x\colon \{0,1\}^d \to \{0,1\}^m$ that takes a string $x \in \{0,1\}^n$ and satisfies the following.*

1. *For any ϵ-distinguisher $T\colon \{0,1\}^m \to \{0,1\}$ for G^x, it holds that*

$$\mathrm{K}^{t,T}(x) \leq \exp(\ell^2/d) \cdot m + d + O(\log(n/\epsilon)),$$

   where $\ell = O(\log n)$ and $t = \mathsf{poly}(n, 1/\epsilon)$.
2. *$G^x(z)$ is computable in time $\mathsf{poly}(n, 1/\epsilon)$.*

Proof Sketch. The pseudorandom generator construction G^x is defined as $G^x(z) := \mathrm{NW}^{\mathrm{Enc}(x)}(z)$ for any $z \in \{0,1\}^d$, where NW is the Nisan–Wigderson generator [53] that is instantiated with the weak design of [58] and the function whose truth table is $\mathrm{Enc}(x)$ as a candidate hard function. ◀

Proof of Theorem 66. Fix $n \in \mathbb{N}$ and an input $x \in \{0,1\}^n$. For simplicity, let $s := s(n)$. Define $d := \sqrt{s}\log n$. Since $s(n) = \omega(\log^2 n)$, we have $4cd \leq s$ for a sufficiently large n. Set $m := s + 4cd \leq 2s$. Let D be the $t(n)$-time algorithm that accepts a dense subset of $R_{\mathrm{Kt}}^{n-c\sqrt{n}\log n}$.

We claim that there exists a coRP-type algorithm A that uses d random bits and solves the promise problem. The algorithm A operates as follows. Pick a random seed $z \in \{0,1\}^d$, and accept if and only if $D(G^x(z)) = 0$, where the output length of G^x is set to be m. This runs in time $t(m) + \mathsf{poly}(n) \leq O(t(2s))$ and can be derandomized in time $O(t(2s)) \cdot 2^d$ by exhaustively trying all the random bits.

It remains to prove the correctness of the algorithm A. Assume that $\mathrm{Kt}(x) \leq s$. Then, since $G^x(z)$ is computable in polynomial time, for any $z \in \{0,1\}^d$, we have

$$\mathrm{Kt}(G^x(z)) \leq d + s + O(\log n) \leq s + 2d \leq s + 4cd - 2cd < m - c\sqrt{m}\log n,$$

where, in the last inequality, we used the fact that $\sqrt{m}\log n \leq \sqrt{2s}\log n < 2d$. Therefore, $G^x(z) \notin R_{\mathrm{Kt}}^{m-c\sqrt{m}\log m}$, which is rejected by D; hence, A accepts.

Conversely, assume that the probability that A accepts is at least $\frac{3}{4}$. This means that

$$\Pr_{z\sim\{0,1\}^d}[D(G^x(z)) = 0] \geq \frac{3}{4}.$$

Since we have

$$\Pr_{w\sim\{0,1\}^m}[D(w) = 0] \leq \frac{1}{2},$$

D is a distinguisher for G^x. By the security of Lemma 67, we obtain that, for $t' := \mathsf{poly}(n) \cdot t(m)$,

$$\begin{aligned}\mathrm{K}^{t'}(x) &\leq \exp(\ell^2/d) \cdot m + O(\log n)\\ &\leq (1 + 2\ell^2/d) \cdot (s + 4cd) + O(\log n)\\ &\leq s + O(\sqrt{s}\log n),\end{aligned}$$

which can be bounded above by $s + c'\sqrt{s}\log n$ by choosing a large enough constant c'. Thus, x is not a No instance of the promise problem. Taking the contrapositive, we conclude that any No instance x is rejected by A with probability at least $\frac{1}{4}$. ◀

In the special case when $t(n) = n^{O(1)}$, Theorem 66 implies that the Kt vs K^t problem can be solved in coRP.

Proof Sketch of Theorem 15. We claim that the Kt vs K^t problem can be solved in coRP under the assumption that $\mathrm{MKtP} \in \mathsf{P}$. We use the same proof of Theorem 66 for $t(n) = n^{O(1)}$ when the parameter $s = \omega(\log^2 n)$; when $s = O(\log^2 n)$, we set $d := \ell^2$ and $m := O(d)$ as in [31]. ◀

▶ **Restatement of Theorem 14.** *For any constant $0 < \epsilon < 1$, the following are equivalent.*

1. *For some c_1, for any large c_2, there exists no derandomization algorithm for $\mathsf{DTIME}(2^{c_1\sqrt{n}\log n})$ that runs in time $2^{n-c_2\sqrt{n}\log n}$.*
2. *For some c_1, for any large c_2, there exists an algorithm running in time $2^{c_1\sqrt{n}\log n}$ that accepts a dense subset of $R_{\mathrm{Kt}}^{n-c_2\sqrt{n}\log n}$.*
3. *For some c_1, for any large c_2, $\mathrm{MKtP}[n - c_2\sqrt{n}\log n, n-1] \in \mathsf{DTIME}(2^{c_1\sqrt{n}\log n})$.*
4. *For some c_1, for any large c_2, $\mathrm{MKtP}[n^\epsilon, n^\epsilon + c_2\sqrt{n^\epsilon}\log n] \in \mathsf{DTIME}(2^{c_1\sqrt{n^\epsilon}\log n})$.*

Proof. (Item 1 $\iff$ Item 2) This equivalence follows from Proposition 65.

(Item 2 $\implies$ Item 4) We apply Theorem 66 for $t(n) := 2^{c_1\sqrt{n}\log n}$ and $s(n) := n^\epsilon$. Then we obtain a $\mathsf{DTIME}(O(t(2s)) \cdot 2^{\sqrt{s}\log n})$-algorithm A that distinguishes YES instances x such that $\mathrm{Kt}(x) \le s$ from NO instances x such that $\mathrm{K}^{t'}(x) > s + c_2'\sqrt{s}\log n$, where $t' = t(2s)\cdot\mathsf{poly}(n)$ and c_2' is some constant. We choose a constant c_1' large enough so that $O(t(2s))\cdot 2^{\sqrt{s}\log n} \le 2^{c_1'\sqrt{s}\log n}$, which bounds from above the running time of the algorithm A.

We claim that $\mathrm{MKtP}[s, s + c_2''\sqrt{s}\log n]$ is also solved by A for any sufficiently large c_2''. Take any string x such that $\mathrm{Kt}(x) \ge s + c_2''\sqrt{s}\log n$. Since $\mathrm{Kt}(x) \le \mathrm{K}^{t'}(x) + O(\log t') \le \mathrm{K}^{t'}(x) + O(c_1\sqrt{s}\log n)$, it follows that $\mathrm{K}^{t'}(x) \ge s + c_2''\sqrt{s}\log n - O(c_1\sqrt{s}\log n) > s + c_2'\sqrt{s}\log n$, where the last inequality holds for any sufficiently large c_2''; thus, x is rejected by A.

(Item 4 $\implies$ Item 3) By the assumption, there exists a constant c_1 such that, for all large c_2, there exists an algorithm A that solves $\mathrm{MKtP}[n^\epsilon, n^\epsilon + c_2\sqrt{n^\epsilon}\log n]$ in time $2^{c_1\sqrt{n^\epsilon}\log n}$. Define $c_1' := c_1/\epsilon$. For all large c_2', we construct an algorithm B that solves $\mathrm{MKtP}[n - c_2'\sqrt{n}\log n, n-1]$ in time $2^{c_1'\sqrt{n}\log n}$. The algorithm B takes an input x of length n and simulates A on input $x10^{m-n-1}$ for $m := (n - 2c_2\sqrt{n}\log n)^{1/\epsilon}$. The running time of B is at most $2^{c_1\sqrt{m}\log m} \le 2^{c_1/\epsilon\cdot\sqrt{m}\log n} = 2^{c_1'\sqrt{m}\log n}$.

It remains to prove the correctness of B. Take any $x \in \{0,1\}^n$ such that $\mathrm{Kt}(x) \le n - c_2'\sqrt{n}\log n$. Then we have $\mathrm{Kt}(x10^{m-n-1}) \le n - c_2'\sqrt{n}\log n + O(\log n) \le m^\epsilon$ for any large c_2; thus B accepts x. Conversely, take any x such that $\mathrm{Kt}(x) \ge n-1$. Then we have $\mathrm{Kt}(x10^{m-n-1}) \ge n - O(\log n) = m^\epsilon + 2c_2\sqrt{n}\log n - O(\log n) \ge m^\epsilon + c_2\sqrt{m^\epsilon}\log n$; thus, B rejects.

(Item 3 $\implies$ Item 2) This immediately follows from the fact that the complement of $\mathrm{MKtP}[n - c_2\sqrt{n}\log n, n-1]$ is a dense subset of $R_{\mathrm{Kt}}^{n-c_2\sqrt{n}\log n+1}$. ◀

References

1. Leonard M. Adleman. Two Theorems on Random Polynomial Time. In *FOCS*, pages 75–83, 1978. doi:10.1109/SFCS.1978.37.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. doi:10.1137/050628994.
3. Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008. doi:10.1137/060664537.
4. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *MFCS*, pages 54:1–54:14, 2017. doi:10.4230/LIPIcs.MFCS.2017.54.
5. Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. *TOCT*, 11(4):27:1–27:27, 2019. doi:10.1145/3349616.
6. Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010. doi:10.1145/1706591.1706594.

7 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3:307–318, 1993. doi:10.1007/BF01275486.

8 Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984. doi:10.1137/0213053.

9 Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006. doi:10.1137/S0097539705446974.

10 Allan Borodin, Alexander A. Razborov, and Roman Smolensky. On Lower Bounds for Read-K-Times Branching Programs. *Computational Complexity*, 3:1–18, 1993. doi:10.1007/BF01200404.

11 Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some Results on Derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005. doi:10.1007/s00224-004-1194-y.

12 Harry Buhrman and Steven Homer. Superpolynomial Circuits, Almost Sparse Oracles and the Exponential Hierarchy. In *FSTTCS*, pages 116–127, 1992. doi:10.1007/3-540-56287-7_99.

13 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *CCC*, pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.

14 Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond Natural Proofs: Hardness Magnification and Locality. In *ITCS*, pages 70:1–70:48, 2020. doi:10.4230/LIPIcs.ITCS.2020.70.

15 Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness Magnification for all Sparse NP Languages. In *FOCS*, pages 1240–1255, 2019. doi:10.1109/FOCS.2019.00077.

16 Mahdi Cheraghchi, Shuichi Hirahara, Dimitrios Myrisiotis, and Yuichi Yoshida. One-Tape Turing Machine and Branching Program Lower Bounds for MCSP. manuscript, 2020.

17 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit Lower Bounds for MCSP from Local Pseudorandom Generators. In *ICALP*, pages 39:1–39:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.39.

18 Shimon Even, Alan L. Selman, and Yacov Yacobi. The Complexity of Promise Problems with Applications to Public-Key Cryptography. *Information and Control*, 61(2):159–173, 1984. doi:10.1016/S0019-9958(84)80056-X.

19 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.

20 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A Better-Than-3n Lower Bound for the Circuit Complexity of an Explicit Function. In *FOCS*, pages 89–98, 2016. doi:10.1109/FOCS.2016.19.

21 Michael A. Forbes and Zander Kelley. Pseudorandom Generators for Read-Once Branching Programs, in Any Order. In *FOCS*, pages 946–955, 2018. doi:10.1109/FOCS.2018.00093.

22 Lance Fortnow. Comparing Notions of Full Derandomization. In *CCC*, pages 28–34, 2001. doi:10.1109/CCC.2001.933869.

23 Oded Goldreich. On Promise Problems: A Survey. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 254–290, 2006. doi:10.1007/11685654_12.

24 Oded Goldreich. A Sample of Samplers: A Computational Perspective on Sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 302–332. Springer, 2011. doi:10.1007/978-3-642-22670-0_24.

25 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

26 Oded Goldreich and Avi Wigderson. On the Size of Depth-Three Boolean Circuits for Computing Multilinear Functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:43, 2013. URL: http://eccc.hpi-web.de/report/2013/043.

27 Oded Goldreich and Avi Wigderson. On derandomizing algorithms that err extremely rarely. In *STOC*, pages 109–118, 2014. doi:10.1145/2591796.2591808.

28 Johan Håstad. *Computational limitations of small-depth circuits*. PhD thesis, MIT, 1986.

29 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

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

31 Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *FOCS*, pages 247–258, 2018.

32 Shuichi Hirahara. Unexpected Hardness Results for Kolmogorov Complexity Under Uniform Reductions. *To appear in STOC'20*, 2020.

33 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits. In *CCC*, pages 5:1–5:31, 2018. doi:10.4230/LIPIcs.CCC.2018.5.

34 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *CCC*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

35 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *CCC*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

36 John M. Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *FSTTCS*, pages 236–245, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

37 Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

38 Russell Impagliazzo. Hard-Core Distributions for Somewhat Hard Problems. In *FOCS*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.

39 Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *CCC*, pages 104–114, 2011. doi:10.1109/CCC.2011.34.

40 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *STOC*, pages 220–229, 1997. doi:10.1145/258533.258590.

41 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *STOC*, pages 73–79, 2000. doi:10.1145/335305.335314.

42 Richard M. Karp and Richard J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–209, 1982.

43 Adam R. Klivans and Rocco A. Servedio. Boosting and Hard-Core Set Construction. *Machine Learning*, 51(3):217–238, 2003. doi:10.1023/A:1022949332276.

44 Adam R. Klivans and Dieter van Melkebeek. Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

45 Ker-I Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM J. Comput.*, 20(5):962–986, 1991. doi:10.1137/0220059.

46 Ravi Kumar and D. Sivakumar. Proofs, Codes, and Polynomial-Time Reducibilities. In *CCC*, pages 46–53, 1999. doi:10.1109/CCC.1999.766261.

47 Leonid A. Levin. Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control*, 61(1):15–37, 1984. doi:10.1016/S0019-9958(84)80060-1.

48 Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.

49 William J Masek. Some NP-complete set covering problems. *Unpublished manuscript*, 1979.

50 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *STOC*, pages 1215–1225, 2019. doi:10.1145/3313276.3316396.

51 Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017. doi:10.4086/toc.2017.v013a004.

52 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

53 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

54 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness Magnification near State-Of-The-Art Lower Bounds. In *CCC*, pages 27:1–27:29, 2019. doi:10.4230/LIPIcs.CCC.2019.27.

55 Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *FOCS*, pages 65–76, 2018.

56 Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 133–138, 1991. doi:10.1109/SCT.1991.160253.

57 Aduri Pavan, Rahul Santhanam, and N. V. Vinodchandran. Some Results on Average-Case Hardness Within the Polynomial Hierarchy. In *FSTTCS*, pages 188–199, 2006. doi:10.1007/11944836_19.

58 Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002. doi:10.1006/jcss.2002.1824.

59 Alexander Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. doi:10.1007/BF01137685.

60 Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

61 Ronen Shaltiel and Emanuele Viola. Hardness Amplification Proofs Require Majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.

62 Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997. doi:10.1006/jcom.1997.0439.

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

64 Boris A. Trakhtenbrot. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. doi:10.1109/MAHC.1984.10036.

65 Luca Trevisan. List-Decoding Using The XOR Lemma. In *FOCS*, pages 126–135, 2003. doi:10.1109/SFCS.2003.1238187.

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

67 Luca Trevisan and Tongke Xue. A Derandomized Switching Lemma and an Improved Derandomization of AC0. In *CCC*, pages 242–247, 2013. doi:10.1109/CCC.2013.32.

68 Salil P. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006. doi:10.1137/S0097539705447207.

69 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

70 Leslie G. Valiant and Vijay V. Vazirani. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

71 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005. doi:10.1007/s00037-004-0187-1.

72 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *FOCS*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

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

A Derandomized Hardness Amplification Theorem

We review a simplified proof of derandomized hardness amplification given by Healy, Vadhan and Viola [30], and observe that the parameter shown in Theorem 56 can be achieved by slightly modifying the construction.

▶**Theorem 56** (Derandomized hardness amplification). *Let $\gamma > 0$ be an arbitrary constant. There exists a hardness amplification procedure* Amp *that takes a function $f\colon \{0,1\}^n \to \{0,1\}$ and parameters $\delta, \epsilon > 0$, and returns a Boolean function $\mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$ satisfying the following:*

1. *If $\widetilde{\mathrm{size}}(\mathrm{Amp}^f_{\epsilon,\delta}; 1/2-\epsilon) \le s$, then $\widetilde{\mathrm{size}}(f;\delta) \le s \cdot \mathsf{poly}(1/\epsilon, 1/\delta)$.*
2. *For any fixed $y \in \{0,1\}^{O(n+\log(1/\epsilon))}$, there exist strings $v_1, \cdots, v_k \in \{0,1\}^n$ for some $k = O(\log(1/\epsilon)/\delta)$ such that $\mathrm{Amp}^f_{\epsilon,\delta}(y) = f(v_1) \oplus \cdots \oplus f(v_k)$. Moreover, if y is distributed uniformly at random, for each $i \in [k]$, v_i is distributed uniformly at random.*
3. *$\mathrm{Amp}^f_{\epsilon,\delta}(y)$ can be computed in $O(n + \log(1/\delta) + \log(1/\epsilon))$ space, given f, y, ϵ and δ as an input.*

First, we explain the construction of our hardness amplification procedure. The construction is a XOR of the nearly disjoint generator ND and a hitter Hit (cf. [24]). In fact, we need a slightly generalized version of a hitter, for which we show that the same construction of [24] suffices. We note that in the previous works [40, 30], an expander walk was used instead of Hit; this does not give us a nearly optimal construction of a hitter when δ is not a constant.

▶ **Lemma 68.** *There exists a* "hitter" Hit *such that, given parameters $n \in \mathbb{N}, \epsilon, \delta > 0$, a function $\mathrm{Hit}_{n,\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to (\{0,1\}^n)^{k_{n,\epsilon,\delta}}$ takes a seed of length $O(n + \log(1/\epsilon))$ and outputs a list L of $k_{n,\epsilon,\delta} = O(\log(1/\epsilon)/\delta)$ strings of length n such that, for any subsets $H_1, \cdots, H_{k_{n,\epsilon,\delta}} \subseteq \{0,1\}^n$ of size $\ge \delta 2^n$, with probability at least $1-\epsilon$, there exists an index $i \in [k_{n,\epsilon,\delta}]$ such that the ith string in the list L is in H_i. Moreover, $\mathrm{KS}(L) \le O(n + \log(1/\epsilon))$.*

Proof Sketch. We observe that the construction of [24, Appendix C] satisfies the requirement of Lemma 68. The construction is as follows: First, we take a pairwise independent generator $G_2\colon \{0,1\}^{O(n)} \to (\{0,1\}^n)^{O(1/\delta)}$. By Chebyshev's inequality, with probability at least $\frac{1}{2}$ over the choice of a seed $r \in \{0,1\}^{O(n)}$, G_2 hits some $H_i, \cdots, H_{i+O(1/\delta)}$, where i is an arbitrary index. Now we take an explicit construction of a constant-degree expander on $2^{O(n)}$ vertices, and generate a random walk $v_1, \cdots, v_\ell$ of length $\ell = O(\log(1/\epsilon))$ over $\{0,1\}^{O(n)}$, which takes random bits of length $O(n + \log(1/\epsilon))$. The output of Hit is defined as the concatenation of $G_2(v_1), \cdots, G_2(v_\ell)$. The correctness follows by using the Expander Random Walk Theorem [24, Theorem A.4]. ◀

▶ **Definition 69.** *Let f be a function $f\colon \{0,1\}^n \to \{0,1\}$, and $\epsilon, \delta > 0$ be arbitrary parameters. Let $k := k_{n,\epsilon,\delta}$. Let $\mathrm{ND}\colon \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be the nearly disjoint generator (Definition 49) defined with the design of Lemma 48. We define a generator*

$$\mathrm{IW}_{n,\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to (\{0,1\}^n)^k$$

as

$$\mathrm{IW}_{n,\epsilon,\delta}(x,y) := \mathrm{ND}(x) \oplus \mathrm{Hit}(y).$$

Then we define a hardness amplified version

$$\mathrm{Amp}^f_{\epsilon,\delta}\colon \{0,1\}^{O(n+\log(1/\epsilon))} \to \{0,1\}$$

of f as the function $\mathrm{Amp}^f_{\epsilon,\delta} := f^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}$.

We proceed to a proof of Theorem 56. Recall several notions from [30]. For a subset $H \subseteq \{0,1\}^n$ (which is supposed to be a hard-core set) and a function $f\colon \{0,1\}^n \to \{0,1\}$, we consider a *probabilistic function* $f_H\colon \{0,1\}^n \to \{0,1\}$, i.e., a distribution over a Boolean function, defined as follows: For each $x \in H$, the value $f_H(x)$ is defined as a random bit chosen uniformly at random and independently; For each $x \in \{0,1\}^n \setminus H$, the value $f_H(x)$ is defined as $f(x)$. Assuming that H is indeed a hard-core set for f, the distributions $(x, f(x))$ and $(x, f_H(x))$ where $x \sim \{0,1\}^n$ are computationally indistinguishable. Indeed:

▶ **Proposition 70** (cf. [65]). *Let $H \subseteq \{0,1\}^n$ be an arbitrary subset, $f\colon \{0,1\}^n \to \{0,1\}$ be an arbitrary function, and $\epsilon > 0$ be an arbitrary parameter. If*

$$\Pr_{x\sim\{0,1\}^n}[A(x, f(x)) = 1] - \Pr_{x\sim\{0,1\}^n}[A(x, f_H(x)) = 1] \geq \epsilon,$$

for some function $A\colon \{0,1\}^{n+1} \to \{0,1\}$, then there exists a one-query oracle circuit of size $O(1)$ such that

$$\Pr_{x\sim H}[C^A(x) = f(x)] \geq \frac{1}{2} + \frac{\epsilon}{\delta}.$$

For a probabilistic function $h\colon \{0,1\}^n \to \{0,1\}$, the expected bias of h is defined as

$$\mathrm{ExpBias}[h] := \mathop{\mathbb{E}}_{x\sim\{0,1\}^n}[\mathrm{Bias}(h(x))],$$

where $\mathrm{Bias}(h(x))$ is defined as $|\Pr_h[h(x) = 0] - \Pr_h[h(x) = 1]|$. It was shown in [30] that the hardness of $f^{\oplus k} \circ G(z)$ is essentially characterized by the expected bias of $\mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}]$, using a property of the nearly disjoint generator ND.

▶ **Lemma 71** ([30, Lemma 5.2 and Lemma 5.12]). *Let g be an arbitrary probabilistic function, and $\epsilon > 0$ be arbitrary parameters. Suppose that there exists a function A such that*

$$\Pr_z[A(z) = f^{\oplus k} \circ \mathrm{IW}(z)] \geq \frac{1}{2} + \frac{1}{2}\mathrm{ExpBias}[g^{\oplus k} \circ \mathrm{IW}] + \frac{\epsilon}{2}$$

Then there exists a one-query oracle circuit C of size $O(k \cdot 2^{\gamma n})$ such that

$$\mathop{\mathbb{E}}_x[C^A(x, f(x)) - C^A(x, g(x))] \geq \frac{\epsilon}{2k},$$

where $\gamma > 0$ is an arbitrary constant of Lemma 48.

We will apply Lemma 71 for $g := f_H$. By using a property of the hitter, we show that the expected bias of f_H is small whenever a hard-core set H is large enough.

▶ **Lemma 72.** *Let $\epsilon, \delta > 0$ be arbitrary parameters, and $H \subseteq \{0,1\}^n$ be a subset of size at least $\delta 2^n$. Let $k := k_{n,\epsilon,\delta}$. Then $\mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}] \leq \epsilon$*

Proof. The idea is that when a hitter hits a hard-core set H, the expected bias becomes 0. More specifically, recall that $\mathrm{IW}(x, y)$ is defined as $\mathrm{ND}(x) \oplus \mathrm{Hit}_{n,\epsilon,\delta}(y)$. Fix any x, and define H_i as a shifted version of H: namely, $H_i := \mathrm{ND}(x)_i \oplus H := \{\, \mathrm{ND}(x)_i \oplus h \mid h \in H \,\}$ for every $i \in [k]$. We apply Lemma 68 for $H_1, \cdots, H_k$. Since $|H_i| = |H| \geq \delta 2^n$ for every $i \in [k]$, by the property of the hitter, there exists some index $i \in [k]$ such that $\mathrm{Hit}_{n,\epsilon,\delta}(y)_i \in H_i = \mathrm{ND}(x)_i \oplus H$ with probability at least $1 - \epsilon$ over the choice of y. In this case, $\mathrm{IW}_{n,\epsilon,\delta}(x, y)_i = \mathrm{ND}(x)_i \oplus \mathrm{Hit}_{n,\epsilon,\delta}(y)_i \in H$, and hence the bias of $f_H(\mathrm{IW}_{n,\epsilon,\delta}(x, y))$ is exactly 0. Therefore,

$$\begin{aligned}
&\mathrm{ExpBias}[f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}] \\
&= \Pr_{x,y}\left[\mathrm{Bias}(f_H^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}(x, y))\right] \\
&\leq \Pr_{x,y}\left[\, \mathrm{IW}_{n,\epsilon,\delta}(x, y)_i \notin H \text{ for every } i \in [k] \,\right] \;\leq\; \epsilon
\end{aligned}$$

◀

Now we combine Proposition 70 and Lemmas 71 and 72 and obtain the following:

▶ **Corollary 73.** *Let $n \in \mathbb{N}$ and $\epsilon, \delta > 0$. Let $H \subseteq \{0,1\}^n$ be an arbitrary subset of size at least $\delta 2^n$. If some function A satisfies*

$$\Pr_z[A(z) = f^{\oplus k} \circ \mathrm{IW}_{n,\epsilon,\delta}(z)] \geq \frac{1}{2} + \epsilon$$

then there exists a one-query oracle circuit C of size $O(k \cdot 2^{\gamma n})$ such that,

$$\Pr_{x \sim H}[C^A(x) = f(x)] \geq \frac{1}{2} + \frac{\epsilon}{2\delta k}.$$

At this point, we make use of Impagliazzo's hard-core lemma [37]. Equivalently, one can view it as a boosting algorithm (cf. [43]).

▶ **Lemma 74** (cf. [37, 43])**.** *Under the condition of Corollary 73, there exists an oracle circuit C of size $O(k \cdot 2^{\gamma n}) \cdot \mathsf{poly}(k/\epsilon)$ such that*

$$\Pr_{x \sim \{0,1\}^n}[C^A(x) = f(x)] \geq 1 - \delta.$$

Now we take any circuit A of size s such that

$$\Pr_z[A(z) = \mathrm{Amp}^f_{\epsilon,\delta}(z)] \geq \frac{1}{2} + \epsilon.$$

By using Corollary 73 and Lemma 74 and replacing each oracle gate by a circuit A, we obtain a circuit C of size $O(k \cdot 2^{\gamma n}) \cdot \mathsf{poly}(k/\epsilon) \cdot s$ such that

$$\Pr_{x \sim \{0,1\}^n}[C(x) = f(x)] \geq 1 - \delta.$$

This completes the proof of Theorem 56.

B AC^0 Lower Bounds for MKtP

In this section, we prove that there exists no constant-depth fixed-polynomial-size circuit that computes $\mathrm{MKtP}[O(\log N), N^{o(1)}]$ (Proposition 12). Previously, [14] used the pseudorandom restriction of Trevisan and Xue [67] to obtain AC^0 lower bounds for $\mathrm{MKtP}[\mathrm{polylog} N]$. Here we make use of a polynomial-time-computable pseudorandom restriction that shrinks AC^0 circuits, which enables us to prove a lower bound for $\mathrm{MKtP}[O(\log N)]$. The following lemma is proved in the context of quantified derandomization.

▶ **Lemma 75** (Goldreich and Wigderson [27])**.** *For any constants $\alpha < 1$ and $d \in \mathbb{N}$ and any polynomials p, q, there exists a constant k and a polynomial-time algorithm of $O(\log n)$ randomness complexity that produces restrictions on n variables such that the following conditions hold:*

1. *The number of undetermined variables in each restriction is at least $2n^\alpha$.*
2. *For any n-input circuit of depth d and size at most $p(n)$, with probability at least $1 - 1/q(n)$, the corresponding restricted circuit depends on at most k variables.*

Proof of Proposition 12. Fix any polynomial p and a depth d. We claim that, for some constant $c > 0$, there exists no depth-d circuit of size $p(n)$ that computes $\mathrm{MKtP}[c \log n, n^\alpha]$ for all large n. Assume, towards a contradiction, that there exists a depth-d circuit C of size $p(n)$ that computes $\mathrm{MKtP}[c \log n, n^\alpha]$. We use Lemma 75 for $q \equiv 2$. Then, there exists a

polynomial-time algorithm that produces a pseudorandom restriction ρ that shrinks C to a circuit of size $k = O(1)$ with probability at least $\frac{1}{2}$. Fix one pseudorandom restriction ρ such that the restricted circuit $C\restriction_\rho$ is a constant-size circuit.

We claim that C does not compute $\mathrm{MKtP}[c \log n, n^\alpha]$. Let σ be the restriction that fixes k variables on which $C\restriction_\rho$ depend to 0. Consider $0^n \circ \rho = 0^n \circ \sigma \circ \rho$, i.e., the string obtained by fixing undetermined variables in ρ to 0. Since ρ is generated by a polynomial-time algorithm, there exists a constant c such that $\mathrm{Kt}(0^n \circ \rho) \leq c \log n$. Thus, $0^n \circ \sigma \circ \rho$ is a YES instance of $\mathrm{MKtP}[c \log n, n^\alpha]$. Since $C\restriction_{\sigma\circ\rho}$ is a constant circuit, we have $1 = C\restriction_{\sigma\circ\rho}(0^n) = C\restriction_{\sigma\circ\rho}(x)$ for any $x \in \{0,1\}^n$. However, for a random $x \in \{0,1\}^n$, by a simple counting argument, it holds that $\mathrm{Kt}(x \circ \sigma \circ \rho) \geq 2n^\alpha - k - 1 > n^\alpha$ with high probability. Therefore, $x \circ \sigma \circ \rho$ is a NO instance of $\mathrm{MKtP}[c \log n, n^\alpha]$ for some x, which is a contradiction. ◀

Algebraic Branching Programs, Border Complexity, and Tangent Spaces

Markus Bläser
Department of Computer Science, Saarland University,
Saarland Informatics Campus, Saarbrücken, Germany

Christian Ikenmeyer
University of Liverpool, UK

Meena Mahajan
The Institute of Mathematical Sciences, HBNI, Chennai, India

Anurag Pandey
Max Planck Institute for Informatics,
Saarland Informatics Campus, Saarbrücken, Germany

Nitin Saurabh
Technion-IIT, Haifa, Israel

Abstract

Nisan showed in 1991 that the width of a smallest noncommutative single-(source,sink) algebraic branching program (ABP) to compute a noncommutative polynomial is given by the ranks of specific matrices. This means that the set of noncommutative polynomials with ABP width complexity at most k is Zariski-closed, an important property in geometric complexity theory. It follows that approximations cannot help to reduce the required ABP width.

It was mentioned by Forbes that this result would probably break when going from single-(source,sink) ABPs to trace ABPs. We prove that this is correct. Moreover, we study the commutative monotone setting and prove a result similar to Nisan, but concerning the analytic closure. We observe the same behavior here: The set of polynomials with ABP width complexity at most k is closed for single-(source,sink) ABPs and not closed for trace ABPs. The proofs reveal an intriguing connection between tangent spaces and the vector space of flows on the ABP. We close with additional observations on VQP and the closure of VNP which allows us to establish a separation between the two classes.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Complexity classes

Keywords and phrases Algebraic Branching Programs, Border Complexity, Tangent Spaces, Lower Bounds, Geometric Complexity Theory, Flows, VQP, VNP

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.21

Funding *Christian Ikenmeyer*: Part of this research was done when CI was at the Max Planck Institute for Software Systems, Saarbrücken, Germany. CI was supported by DFG grant IK 116/2-1.
Anurag Pandey: Supported by the Chair of Raimund Seidel, Department of Computer Science, Saarland University, Saarbrücken, Germany.
Nitin Saurabh: Part of this work was done when the author was at the Max Planck Institute for Informatics, Saarbrücken, Germany.

Acknowledgements We thank Michael Forbes for illuminating discussions and for telling us about his (correct) intuition concerning Nisan's result. We thank the Simons Institute for the Theory of Computing (Berkeley), Schloss Dagstuhl - Leibniz-Zentrum für Informatik (Dagstuhl), and the International Centre for Theoretical Sciences (Bengaluru), for hosting us during several phases of this research.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 21; pp. 21:1–21:24

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

1 Introduction and Results

Algebraic branching programs (ABPs) are an elegant model of computation that is widely studied in algebraic complexity theory (see e.g. [4, 40, 30, 31, 1, 3, 25, 27, 15, 17]) and is a focus of study in geometric complexity theory [28, 18, 19]. An ABP is a layered directed graph with $d+1$ layers of vertices (edges only go from layers i to $i+1$) such that the first and last layer have exactly the same number of vertices. Each vertex in the first layer has exactly one so-called *corresponding* vertex in the last layer. One interesting classical case is when the first and last layer have exactly one vertex, which is usually studied in theoretical computer science. We call this the *single-(source,sink) model*. Among algebraic geometers working on ABPs it is common to not impose restrictions on the number of vertices in the first and last layer [28, 18, 29]. We call this the *trace model*. Every edge in an ABP is labeled by a homogeneous linear form. If we denote by $\ell(e)$ the homogeneous linear form of edge e, then we say that the ABP computes $\sum_p \prod_{e\in p} \ell(e)$, where the sum is over all paths that start in the first layer and end in the last layer at the vertex corresponding to the start vertex.

The *width* of an ABP is the number of vertices in its largest layer. We denote by $\mathsf{w}(f)$ the minimal width required to compute f in the trace model and we call $\mathsf{w}(f)$ the *trace ABP width complexity* of f. We denote by $\mathsf{w}_1(f)$ the minimal width required to compute f in the single-(source,sink) model and we call $\mathsf{w}_1(f)$ the *single-(source,sink) ABP width complexity* of f.

The complexity class VBP is defined as the set of sequences of polynomials (f_m) for which the sequence $\mathsf{w}(f_m)$ is polynomially bounded. Let $\mathrm{per}_m := \sum_{\pi\in\mathfrak{S}_m} \prod_{i=1}^m x_{i,\pi(i)}$ be the permanent polynomial. Valiant's famous VBP $\neq$ VNP conjecture can concisely be stated as "The sequence of natural numbers $\big(\mathsf{w}(\mathrm{per}_m)\big)_m$ is not polynomially bounded." Alternatively, this is phrased with w_1 or other polynomially related complexity measures in a completely analogous way. In geometric complexity theory (GCT), one searches for lower bounds on algebraic complexity measures over $\mathbb{C}$ such as w and w_1 for explicit polynomials such as the permanent. All lower bounds methods in GCT and most lower bounds methods in algebraic complexity theory are *continuous*, which means that if f_ε is a curve of polynomials with $\lim_{\varepsilon\to 0} f_\varepsilon = f$ (coefficient-wise limit) and $\mathsf{w}(f_\varepsilon) \leq w$, then these methods cannot be used to prove $\mathsf{w}(f) > w$. This is usually phrased in terms of so-called *border complexity* (see e.g. [14, 28]): The *border trace ABP width complexity* $\underline{\mathsf{w}}(f)$ is the smallest w such that f can be approximated arbitrarily closely by polynomials f_ε with $\mathsf{w}(f_\varepsilon) \leq w$. Analogously, we define the *border single-(source,sink) ABP width complexity* $\underline{\mathsf{w}_1}(f)$ as the smallest w such that f can be approximated arbitrarily closely by polynomials f_ε with $\mathsf{w}_1(f_\varepsilon) \leq w$. We define $\overline{\mathrm{VBP}}$ as the set of sequences of polynomials such that $(\underline{\mathsf{w}}(f_m))$ is polynomially bounded. Clearly VBP $\subseteq \overline{\mathrm{VBP}}$. Mulmuley and Sohoni [32, 33, 14] (see also [12, 10] for a related conjecture) conjectured a strengthening of Valiant's conjecture, namely that VNP $\not\subseteq \overline{\mathrm{VBP}}$. In principle it could be that $\underline{\mathsf{w}}(f) < \mathsf{w}(f)$; the gap could even be superpolynomial, which would mean that VBP $\subsetneq \overline{\mathrm{VBP}}$. If VBP $= \overline{\mathrm{VBP}}$, then Valiant's conjecture is the same as the Mulmuley-Sohoni conjecture, which would mean that if VBP $\neq$ VNP, then continuous lower bounds methods exist that show this separation.

Border complexity is an old area of study in algebraic geometry. In theoretical computer science it was introduced by Bini et al. [6], where [5] proves that in the study of fast matrix multiplication, the gap between complexity and border complexity is not too large. The study of the gap between complexity and border complexity of algebraic complexity measures in general has started recently [21, 9, 26] as an approach to understand if strong algebraic complexity lower bounds can be obtained from continuous methods.

In this paper we study two very different settings of ABPs: The noncommutative and the monotone setting. To capture commutative, noncommutative, and monotone computation, let R be a graded semiring with homogeneous components R_d. In our case the settings for R_d are

- $R_d = \mathbb{F}[x_1, \ldots, x_m]_d$ the set of homogeneous degree d polynomials in m variables over a field $\mathbb{F}$,
- $R_d = \mathbb{F}\langle x_1, \ldots, x_m\rangle_d$ the set of homogeneous degree d polynomials in m noncommuting variables over a field $\mathbb{F}$,
- $R_d = \mathbb{R}_+[x_1, \ldots, x_m]_d$ the set of homogeneous degree d polynomials in m variables with nonnegative coefficients.

As it is common in the theoretical computer science literature, we call elements of R_d *polynomials*. Note that $\mathbb{F}\langle x_1, \ldots, x_m\rangle_d$ is naturally isomorphic to the d-th tensor power of $\mathbb{F}^m$, so *tensor* would be the better name. We hope that no confusion arises when in the later sections (where we use concepts from multilinear algebra) we use the tensor language. In the *homogeneous setting*, all ABP edge labels are in R_1, and hence the polynomial that is computed is in R_d. In the *affine setting*, all ABP edge labels are in $R_0 + R_1$, and hence the polynomial that is computed is in $\bigoplus_{d' \le d} R_{d'}$.

Noncommutative ABPs

Let $R_d = \mathbb{F}\langle x_1, \ldots, x_m\rangle_d$ and consider the homogeneous setting. We write ncw instead of w and ncw_1 instead of w_1 to highlight that we are in the noncommutative setting. Nisan [35] proved:

▶ **Theorem 1.** *Let M_i denote the $m^i \times m^{d-i}$ matrix whose entry at position $((k_1, \ldots, k_i), (k_{i+1}, \ldots, k_d))$ is the coefficient of the monomial $x_{k_1} x_{k_2} \cdots x_{k_d}$ in f. Then every single-(source,sink) ABP computing f has at least $\mathsf{rk}(M_i)$ many vertices in layer i. Conversely, there exists a single-(source,sink) ABP computing f with exactly $\mathsf{rk}(M_i)$ many vertices in layer i.*

Nisan used this formulation to prove strong complexity lower bounds for the noncommutative determinant and permanent. Forbes [16] remarked that Theorem 1 implies that for fixed w

$$\text{the set } \{f \mid \mathsf{ncw}_1(f) \le w\} \text{ is Zariski-closed}^1 \tag{1}$$

and hence that

$$\underline{\mathsf{ncw}_1}(f) = \mathsf{ncw}_1(f) \text{ for all } f. \tag{2}$$

Proving a similar result (even up to polynomial blowups) in the commutative setting would be spectacular: It would imply VBP $= \overline{\text{VBP}}$ and hence that Valiant's conjecture is the same as the Mulmuley-Sohoni conjecture. By a general principle, for all standard algebraic complexity measures, over $\mathbb{C}$ we have that the Zariski-closure of a set of polynomials of complexity at most w equals the Euclidean closure [34, §2.C].

[1] We identify each m-variate homogeneous degree d polynomial with its coefficient vector. There is a standard topology on the vector space of these coefficient vectors that we call the Euclidean topology. The Zariski-closure of a set X of vectors is the smallest set of vectors that contains X and that is the common zero set of a finite set of polynomials in the coordinate variables, see e.g. [7, Ch. 4] for the commutative case.

Forbes mentioned that he believes that Nisan's proof cannot be lifted to the trace model. In this paper we prove that Forbes is correct, by constructing a polynomial f_0 with

$$\underline{\mathsf{ncw}}(f_0) < \mathsf{ncw}(f_0). \tag{3}$$

The proof is given in Sections 5–8. It is a surprisingly subtle application of differential geometry (inspired by [24]) and interprets tangent spaces to certain varieties as vector spaces of flows on an ABP digraph.

The gap between $\underline{\mathsf{ncw}}(f)$ and $\mathsf{ncw}(f)$ can never be very large though:

$$\underline{\mathsf{ncw}}(f) \leq \mathsf{ncw}(f) \leq \mathsf{ncw}_1(f) \overset{(2)}{=} \underline{\mathsf{ncw}_1}(f) \leq \big(\underline{\mathsf{ncw}}(f)\big)^2 \text{ for all } f. \tag{4}$$

It is worth noting that for our separating polynomial f_0, the gap is even less; $\underline{\mathsf{ncw}}(f_0) < \mathsf{ncw}(f_0) \leq 2\underline{\mathsf{ncw}}(f_0)$. This is the first algebraic model of computation where complexity and border complexity differ, but their gap is known to be polynomially bounded! For most models of computation almost nothing is known about the gap between complexity and border complexity. For commutative width 2 affine ABPs the gap is even as large as between computable and non-computable [9]!

Monotone ABPs

Let $R_d = \mathbb{R}_+[x_1, \ldots, x_m]_d$ and consider the affine or homogeneous setting.

Since $\mathbb{R}$ is not algebraically closed, we switch to a more algebraic definition of approximation. Let $\mathbb{R}[\varepsilon, \varepsilon^{-1}]_+$ denote the ring of Laurent polynomials that are nonnegative for all sufficiently small $\varepsilon > 0$. Clearly, elements from $\mathbb{R}[\varepsilon, \varepsilon^{-1}]_+$ can have a pole at $\varepsilon = 0$ of arbitrarily high order. We define $\underline{\mathsf{mw}}(f)$ to be the smallest w such that there exists a polynomial f' over the ring $\mathbb{R}[\varepsilon, \varepsilon^{-1}]_+$ such that

- there exists a width w ABP over $\mathbb{R}[\varepsilon, \varepsilon^{-1}]_+$ that computes f',
- no coefficient in f' contains an ε with negative exponent, and setting ε to 0 in f' yields f, i.e., $f'_{\varepsilon=0} = f$.

We prove a result that is comparable to (2), but uses a very different proof technique:

$$\underline{\mathsf{mw}_1}(f) = \mathsf{mw}_1(f) \text{ for all } f. \tag{5}$$

In terms of complexity classes, this implies

$$\mathrm{MVBP} = \overline{\mathrm{MVBP}}^{\mathbb{R}}.$$

Our proof also works if the ABP is not layered and the labels are affine.

Intuitively, in this monotone setting, one would think that approximations do not help, because there cannot be cancellations. But quite surprisingly the same construction as in (3) can be used to find f_0 such that

$$\underline{\mathsf{mw}}(f_0) < \mathsf{mw}(f_0). \tag{6}$$

[2] Given a trace ABP Γ computing f and a pair of corresponding start and end vertices, we can extract a single-(source,sink) ABP by deleting all other start and end vertices. If we do this for each pair of start and end vertices, and if we then idenfity all start vertices to a single start vertex, and also all end vertex to a single end vertex, then we obtain a single-(source,sink) ABP computing f. The width has grown by a factor of w, where w is the number of start vertices in Γ.

By the same reasoning as in (4), we obtain

$$\underline{\mathsf{mw}}(f) \leq \mathsf{mw}(f) \leq \left(\underline{\mathsf{mw}}(f)\right)^2 \text{ for all } f. \tag{7}$$

This gives a natural monotone model of computation where approximations speed up the computation. Again, the gap is polynomially bounded!

Separating VQP from $\overline{\text{VNP}}$

Bürgisser in his monograph [11] defined the complexity class VQP as the class of polynomials with quasi-polynomially bounded straight-line programs, and established its relation to the classes VP and VNP (see Section 9 for definitions). He showed that the determinant polynomial is VQP-complete with respect to the so-called *qp*-projections (see [11], Corollary 2.29). He strengthened Valiant's hypothesis of VNP $\not\subseteq$ VP to VNP $\not\subseteq$ VQP and called it *Valiant's extended hypothesis* (see [11], section 2.5). He further showed that VP is strictly contained in VQP as one would intuitively expect (see [11], section 8.2). Finally, he also showed that VQP is not contained in VNP (see [11], Proposition 8.5 and Corollary 8.9). In this article, we observe that his proof is stronger and actually shows that VQP is not contained in $\overline{\text{VNP}}$ either, where $\overline{\text{VNP}}$ is the closure of the complexity class VNP in the sense as mentioned above.

Structure of the paper

In Section 4 we prove (5). Sections 5 to 8 are dedicated to proving (3) and (6) via a new connection between tangent spaces and flow vector spaces. In Section 9, we discuss the separation between VQP and $\overline{\text{VNP}}$.

2 Related work

Grenet [20] showed that $\mathsf{mw}(\text{per}_m) \leq \binom{m}{\lceil m/2 \rceil}$ by an explicit construction of a monotone single-(source,sink) ABP. Even though the construction is monotone, its size is optimal for $m = 3$ [2] (for 4 this is already unknown). The noncommutative version of this setting has been studied in [17]. [42] recently showed that the monotone circuit classes MVP and MVNP are different. We refer the reader to [42] and [38] and the references therein to get more information about monotone algebraic models of computation and their long history.

Hüttenhain and Lairez [24] present a method that can be used to show that a complexity measure and its border variant are not the same. They used it to prove that an explicit polynomial has border determinantal complexity 3, but higher determinantal complexity. We use their ideas as a starting point in Section 5 and the later sections.

3 Preliminaries

For a homogeneous degree d ABP Γ, we denote by V the set of vertices of Γ and by V^i the set of vertices in layer i, $1 \leq i \leq d+1$. We choose an explicit bijection between the sets V^1 and V^{d+1}, so that each vertex v in V^1 has exactly one *corresponding* vertex $\mathsf{corr}(v)$ in V^{d+1}. We denote by E^i the set of edges from V^i to V^{i+1}. Let E denote the union of all E^i.

There is a classical interpretation in terms of iterated matrix multiplication: Fix some arbitrary ordering of the vertices within each layer, such that the i-th vertex in V^1 corresponds to the i-th vertex in V^{d+1}. For $1 \leq k \leq d$ let M_k be the $|V^k| \times |V^{k+1}|$ matrix whose entry

at position (i,j) in M_k is the label from the i-th vertex in V^k to the j-th vertex in V^{k+1}. Then Γ computes the trace

$$\sum_{\substack{1\le k_1\le |V^1|\\ 1\le k_2\le |V^2|\\ \vdots \\ 1\le k_d\le |V^d|}} (M_1)_{k_1,k_2}(M_2)_{k_2,k_3}\cdots(M_{d-1})_{k_{d-1},k_d}(M_d)_{k_d,k_1} = \mathsf{tr}\big(M_1M_2\cdots M_d\big). \tag{8}$$

Hence the name *trace model*. In the single-(source,sink) model, the trace is taken of a 1×1 matrix.

4 Monotone commutative single-(source,sink) ABPs are closed

For fixed $w\in\mathbb{N}$ we study

$$\text{the set } \{f\in\mathbb{R}_+[x_1,\ldots,x_n]_d \mid \mathsf{mw}_1(f)\le w\}. \tag{9}$$

We first start with the simple observation that it is *not* Zariski-closed.

▶ **Proposition 2.** *$\{f\in\mathbb{R}_+[x_1,\ldots,x_n]_d \mid \mathsf{mw}_1(f)\le w\}$ is not Zariski-closed.*

Proof. Note that a homogeneous degree d single-(source,sink) width w ABP has $2w+w^2(d-2)$ many edges. The label on each edge is a linear form in n variables, so such an ABP is determined by $N := n(2w+w^2(d-2))$ many parameters. Let $F:\mathbb{C}^N\to\mathbb{C}[x_1,\ldots,x_n]_d$ be the map that maps these parameters to the polynomial computed by the ABP. Every coordinate function of F is given by polynomials in N variables, so F is Zariski-continuous. Therefore

$$\overline{F((\mathbb{R}_+)^N)} = \overline{F(\overline{(\mathbb{R}_+)^N})} = \overline{F(\mathbb{C}^N)} \supseteq F(\mathbb{C}^N) \supsetneq F((\mathbb{R}_+)^N),$$

where the overline means the Zariski-closure. We remark that we did not use any special feature of the model of computation other than the fact that it is defined over $\mathbb{R}$. ◀

Recall that an ABP has $d+1$ layers of vertices. If an ABP has w_i many vertices in layer i, $1\le i\le d$, we say the ABP has *format* $w=(w_1,w_2,\ldots,w_d)$. We further recall that $w_{d+1}=w_1$. The following theorem is our closure result, which proves (5) and hence $\mathrm{MVBP}=\overline{\mathrm{MVBP}}^{\mathbb{R}}$.

▶ **Theorem 3.** *Given a polynomial f over $\mathbb{R}$ and given a format w single-(source,sink) ABP with affine linear labels over $\mathbb{R}[\varepsilon,\varepsilon^{-1}]_+$ computing f_ε such that $\lim_{\varepsilon\to 0} f_\varepsilon = f$. Then there exists a format w monotone single-(source,sink) ABP that computes f.*

Proof. The proof is constructive and done by a two-step process. In the first step (which is fairly standard and works in many computational models) we move all the ε with negative exponents to edges adjacent to the source. The second step then uses the monotonicity.

Given Γ with affine linear labels over $\mathbb{R}[\varepsilon,\varepsilon^{-1}]_+$ we repeat the following process until all labels that contain an ε with a negative exponent are incident to the source vertex.

- Let e be an edge whose label contains ε with a negative exponent $-i<0$. Moreover, assume that e is not incident to the source vertex. Let v be the start vertex of e. We rescale all edges outgoing of v with ε^i and we rescale all edges incoming to v with ε^{-i}.

If we always choose the edge with the highest layer, then it is easy to see that this process terminates. Since every path from the source to the sink that goes through a vertex v must use exactly one edge that goes into v and exactly one edge that comes out of v, throughout the process the value of Γ does not change. We finish this first phase by taking the highest negative power i among all labels of edges that are incident to the source and then rescale all these edges with ε^i. The resulting ABP Γ^i computes $\varepsilon^i f_\epsilon$ and no label contains an ε with negative exponent. We now start phase 2 that transforms Γ^i into Γ^{i-1} that computes $\varepsilon^{i-1} f_\epsilon$ without introducing negative exponents of ε. We repeat phase 2 until we reach Γ^0 in which we safely set ε to 0. Throughout the whole process we do not change the structure of the ABP and only rescale edge labels with powers of ε, which preserves monotonicity, so the proof is finished. It remains to show how Γ^i can be transformed into Γ^{i-1}. An edge whose label is divisible by ε is called an *ε-edge*. Consider the set Δ of vertices that are reachable from the source using only non ε-edges in Γ^i. The crucial insight is that since Γ^i is monotone and computes a polynomial that is divisible by ε, we know that every path in Γ^i from the source to the sink uses an ε-edge. Therefore Δ cannot contain the sink. We call a vertex in Δ whose outdegree is zero a *leaf* vertex. We repeat the following procedure until the source is the only leaf vertex:

- Let v be a non-source leaf vertex in Δ. We rescale all edges outgoing of v with ε^{-1} and we rescale all edges incoming to v with ε.

It is easy to see that this process terminates with the source being the only leaf vertex. Since the source is a leaf vertex, all edges incident to the source are ε-edges. We divide all their labels by ε to obtain Γ^{i-1}. ◀

5 Explicit construction of f_0 with higher complexity than border complexity

Fix some $d \geq 3$. In this section for every $m \geq 2$ we construct f_0 such that

$$m = \underline{\mathsf{ncw}}(f_0) < \mathsf{ncw}(f_0). \tag{10}$$

A completely analogous construction can be used to find f_0 with $\underline{\mathsf{w}}(f_0) < \mathsf{w}(f_0)$ and with $\underline{\mathsf{mw}}(f_0) < \mathsf{mw}(f_0)$. For the sake of simplicity, we carry out only the proof for (10).

Recall that in a format w ABP we have $w_{d+1} = w_1$. In each layer i we enumerate the vertices $V^i = \{v_1^i, \ldots, v_{w_i}^i\}$ and we assume without loss of generality that the correspondence bijection between V^{d+1} and V^1 is the identity on the indices j of v_j^1, i.e., the jth vertex in V^1 corresponds to the jth vertex in V^{d+1}.

Fix an ABP format $w = (w_1, w_2, \ldots, w_d)$ such that for all i, $w_i \geq 2$. Let Γ_{com} denote the directed acyclic graph underlying an ABP of format w. An edge can be described by the triple (a, b, i), where $1 \leq i \leq d$, $1 \leq a \leq w_i$ and $1 \leq b \leq w_{i+1}$. Consider the following labeling of the edges with triple-indexed variables: $\ell_{\mathrm{com}}((a, b, i)) = x_{(a,b)}^{(i)}$. Define f_{com} to be the polynomial computed by Γ_{com} with edge labels ℓ_{com}.

We now construct f_0 as follows. Let d be odd (the case when d is even works analogously). Since in each layer we enumerated the vertices, we can now assign to each vertex its parity: even or odd. We call an edge between two even or two odd vertices *parity preserving*, while we call the other edges *parity changing*. Let us consider the following labeling of Γ_{com}: We set $\ell_0((a, b, i)) := x_{(a,b)}^{(i)}$ if (a, b, i) is parity changing (i.e., $a \not\equiv b \pmod 2$) and set the label $\ell_0((a, b, i)) := \varepsilon x_{(a,b)}^{(i)}$ otherwise, where $\varepsilon \in \mathbb{C}$. Let f'_ε be the polynomial computed by Γ_{com} with edge labels ℓ_0 and set $f_\varepsilon := \frac{1}{\varepsilon} f'_\varepsilon$ for $\varepsilon \neq 0$. We define $f_0 := \lim_{\varepsilon \to 0} f_\varepsilon$ (convergence follows from the construction, because d is odd). By definition, for all $\varepsilon \neq 0$, f_ε can be computed by a format w ABP. However, we will now prove that this property fails for the limit point f_0.

▶ **Theorem 4.** *Fix an ABP format $w = (w_1, w_2, \ldots, w_d)$ such that for all i, $w_i \geq 2$. Let f_0 be defined as above. Then, f_0* cannot *be computed by an ABP of format w.*

Note that for a format where $m = w_1 = \cdots = w_d$, this gives the f_0 which was desired in (10). (Note, however, that f_0 can be computed by an ABP of width $2m$ as follows. Construct an ABP Γ' that has, for each vertex $v \in \Gamma_{\text{com}}$, vertices v' and v''. For each parity changing edge $(a, b) \in \Gamma_{\text{com}}$ with label ℓ_0, add edges (a', b') and (a'', b'') with the same label ℓ_0. For each parity preserving edge $(a, b) \in \Gamma_{\text{com}}$ with label ℓ_0, add edge (a', b'') with label $(\frac{1}{\varepsilon})\ell_0$. For corresponding vertices u, v in Γ_{com}, let v'' be the corresponding vertex for u' and v' be the corresponding vertex for u'' in Γ'. All paths between corresponding vertices in this ABP use exactly one parity preserving edge of Γ_{com}, and so this ABP computes f_0.)

The proof of Theorem 4 works as follows. Let $\mathsf{G} := \mathsf{GL}_{w_1 w_2} \times \mathsf{GL}_{w_2 w_3} \times \cdots \times \mathsf{GL}_{w_d w_{d+1}}$. Let $\mathsf{End} := \overline{G}$ denote its Euclidean closure, i.e., tuples of matrices in which one or several matrices can be singular.

We consider noncommutative homogeneous polynomials in the variables $x^{(i)}_{(a,b)}$ such that the i-th variable in each monomial is $x^{(i)}_{(a,b)}$ for some $a \in [w_i]$ and $b \in [w_{i+1}]$. The vector space of these polynomials is isomorphic to $W := \mathbb{C}^{w_1 w_2} \otimes \mathbb{C}^{w_2 w_3} \otimes \cdots \otimes \mathbb{C}^{w_d w_{d+1}}$ and the monoid End (and thus also the group G) acts on this space in the canonical way. The set

$$\{f \in W \mid f \text{ can be computed by a format } w \text{ ABP}\}$$

is precisely the orbit $\mathsf{End} f_{\text{com}}$. We follow the overall proof strategy in [24]. The monoid orbit $\mathsf{End} f_{\text{com}}$ decomposes into two disjoint orbits:

$$\mathsf{End} f_{\text{com}} = \mathsf{G} f_{\text{com}} \cup (\mathsf{End} \setminus \mathsf{G}) f_{\text{com}}.$$

Our goal is to show two things independently:
1. $f_0 \notin (\mathsf{End} \setminus \mathsf{G}) f_{\text{com}}$, and
2. $f_0 \notin \mathsf{G} f_{\text{com}}$,

which finishes the proof of Theorem 4.

All elements in $(\mathsf{End} \setminus \mathsf{G}) f_{\text{com}}$ are *not concise*, a term that we define in Section 6, where we also prove that f_0 is concise. Therefore $f_0 \notin (\mathsf{End} \setminus \mathsf{G}) f_{\text{com}}$.

All elements in $\mathsf{G} f_{\text{com}}$ have *full orbit dimension*, a term that we define in Section 7 and we prove that f_0 does *not* have full orbit dimension in Section 8. This finishes the proof of Theorem 4.

6 Conciseness

In this section we show that $f_0 \notin (\mathsf{End} \setminus \mathsf{G}) f_{\text{com}}$. To do so we use a notion called *conciseness*. Informally, it captures whether a polynomial depends on all variables independent of a change of basis, or a tensor cannot be embedded into a tensor product of smaller spaces.

Given a tensor f in $\mathbb{C}^{m_1} \otimes \mathbb{C}^{m_2} \otimes \cdots \otimes \mathbb{C}^{m_d}$, we associate the following matrices with f. For $j \in [d]$, define a matrix M^j_f of dimension $m_j \times (\prod_{i \in [d] \setminus \{j\}} m_i)$ with rows labeled by the standard basis of $\mathbb{C}^{m_j}$, and columns by elements in the Cartesian product $\{\text{standard basis of } \mathbb{C}^{m_1}\} \times \cdots \times \{\text{standard basis of } \mathbb{C}^{m_{j-1}}\} \times \{\text{standard basis of } \mathbb{C}^{m_{j+1}}\} \times \cdots \times \{\text{standard basis of } \mathbb{C}^{m_d}\}$. We write the tensor f in the standard basis

$$f = \sum_{\substack{1 \leq i_1 \leq m_1 \\ 1 \leq i_2 \leq m_2 \\ \vdots \\ 1 \leq i_d \leq m_d}} \alpha_{i_1, \ldots, i_d} e_{i_1} \otimes \cdots \otimes e_{i_d}$$

and associate to it the matrix M^j_f whose entry at position $((i_j), (i_1, i_2, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d))$ is $\alpha_{i_1, \ldots, i_d}$.

▶ **Definition 5.** *We say that a tensor f in $\mathbb{C}^{m_1} \otimes \mathbb{C}^{m_2} \otimes \cdots \otimes \mathbb{C}^{m_d}$ is* concise *if and only if for all $j \in [d]$, M_f^j has full rank.* [3]

As a warm-up exercise we now show that f_{com} is concise.

▶ **Proposition 6.** *f_{com} is concise.*

Proof. We know that $f_{\mathrm{com}} \in W$. Let us consider the matrix $M^j_{f_{\mathrm{com}}}$ for some $j \in [d]$. To establish that $M^j_{f_{\mathrm{com}}}$ has full rank, it suffices to show that rows are linearly independent. In order to show that, we argue that every row is non-zero and every column has at most one non-zero entry. In other words, rows are supported on disjoint sets of columns.

A row of $M^j_{f_{\mathrm{com}}}$ is labeled by an edge in the j-th layer of the ABP Γ_{com}. Recall that only paths that start at a vertex in V^1 and end at the corresponding vertex in V^{d+1} contribute to the computation in Γ_{com}. We call such paths *valid paths*. An entry in $M^j_{f_{\mathrm{com}}}$ is non-zero iff the corresponding row and column labels form a valid path in Γ_{com}. Thus, it is easily seen that a row is non-zero iff there is a valid path in Γ_{com} that *passes through* the edge given by the row label. By the structure of Γ_{com}, in particular that every layer is a complete bipartite graph, we observe that passing through every edge there is some valid path. Hence, we obtain that every row is non-zero.

The second claim now follows from the observation that fixing $d-1$ edges either defines a unique dth edge so that these d edges form a valid path, or for these $d-1$ edges there is no such dth edge. ◀

As mentioned in Section 5, to establish $f_0 \notin (\mathsf{End} \setminus \mathsf{G}) f_{\mathrm{com}}$ we will show that f_0 is concise while any element in $(\mathsf{End} \setminus \mathsf{G}) f_{\mathrm{com}}$ is not.

▶ **Lemma 7.** *f_0 is concise.*

Proof. Analogous to the proof of Proposition 6, we again show that every row of $M^j_{f_0}$ is non-zero and every column of it has at most one non-zero entry. That is, rows of $M^j_{f_0}$ are supported on disjoint sets of columns.

From the construction of f_0 it is seen that a path in Γ_{com} contributes to the computation of f_0 iff it is a valid path that comprises of *exactly one* parity preserving edge. The second claim of every column having at most one non-zero entry now follows for the same reason as in the proof of Proposition 6.

Before proving the first claim, we recall two assumptions in the construction of f_0. The first is that the format $w = (w_1, w_2, \ldots, w_d)$ is such that $w_i \geq 2$ for all $i \in [d]$ and the second is that d is odd. To argue that a row is non-zero it suffices to show that a valid path comprising of only one parity preserving edge passes through the edge given by the row level. Let us consider an arbitrary edge e in Γ_{com}. We have two cases to consider depending on whether it is parity *preserving* or *changing*.

Case 1. Suppose e is parity preserving and it belongs to a layer $j \in [d]$. The number of layers on the left of e is $j-1$ and on the right is $d-j$. Since d is odd, these numbers are either both even or both odd. We now argue for the case when they are even (the odd case is analogous). Choose a vertex v in V^1 that has the same parity (different in the odd

[3] When f is viewed as a set-multilinear polynomial (see [36, Section 1.4]), this condition translates to the linear independence of the partial derivatives of f. In particular, M_f^j is testing if the partial derivatives of f with respect to the j-th block of variables are all linearly independent. This partial derivatives based criterion for testing if a polynomial depends on all the variables, independent of a change of basis, is pretty standard: see, for instance, [23, Corollary 5.1.4].

case) as one of the end points of e. (Such a choice exists because $w_1 \geq 2$.) We now claim that there exists a valid path starting at v that passes through e and contains exactly one parity preserving edge. Since e is parity preserving, all edges in the claimed path must be parity changing. We observe that e can be easily extended in both directions using parity changing edges such that the path ends at $\mathsf{corr}(v)$. The existence of parity changing edges at each layer uses the assumption that $w_i \geq 2$.

Case 2. Otherwise $e = (a, b)$ is parity changing. Again as before there are two cases based on whether both $j-1$ and $d-j$ are even or odd. Consider the case when they are even (the odd case being analogous). We first assume that $j \neq d$. Choose a vertex v in V^1 that has the same parity as a. We now construct a valid path from v to $\mathsf{corr}(v)$ that passes through e and contains exactly one parity preserving edge. It is easily seen that there exists a path from v to a using only parity changing edges. We choose a parity preserving outgoing edge incident to b. We call its endpoint v_1. Since v_1 and v have different parities, we can connect v_1 to $\mathsf{corr}(v)$ in V^{d+1} using only parity changing edges. Thus we obtain the following valid path $v \to \cdots \to a \to b \to v_1 \to \cdots \to \mathsf{corr}(v)$ passing through exactly one parity preserving edge (b, v_1). In the case that $j = d$, choose an incoming parity preserving edge incident on a instead of an outgoing edge on b. ◀

▶ Remark 8. We note that if the format $w = (w_1, \ldots, w_d)$ defining f_0 is such that for some $j \in [d]$, $w_j = 1$, then f_0 is *not* concise. This can be seen as follows.

Let $w_j = 1$, and let v denote the unique vertex in V^j. Let e be the edge $e = (1, 1, j)$. If $j < d$, let e' be the edge $e' = (1, 1, j+1)$, otherwise let e' be the edge $e' = (1, 1, j-1)$. Both e, e' are parity preserving edges. By construction, every valid path using e' must also use e. Hence the corresponding row in the matrix $M_{f_0}^{j+1}$ if $j < d$, and in $M_{f_0}^{j-1}$ otherwise, is zero. Therefore f_0 is not concise.

This is an interesting observation, because this is the point where our proof fails for single-(source,sink) ABPs, and this is expected, because Nisan [35] had shown that the set of polynomials computed by such ABPs of format w is a closed set.

▶ **Lemma 9.** *Let $f \in (\mathsf{End} \setminus \mathsf{G}) f_{\mathrm{com}}$. Then f is not concise.*

Proof. This statement is true in very high generality. In our specific case a proof goes as follows. If $f \in (\mathsf{End} \setminus \mathsf{G}) f_{\mathrm{com}}$, then $f = g f_{\mathrm{com}}$ for some $g \in \mathsf{End} \setminus \mathsf{G}$. Let $g = (g_1, \ldots, g_d)$, where $g_i \in \mathbb{C}^{w_i w_{i+1} \times w_i w_{i+1}}$. Since $g \notin \mathsf{G}$, at least one of the g_i must be singular. The crucial property is $M^i_{g f_{\mathrm{com}}} = g_i M^i_{f_{\mathrm{com}}}$, which finishes the proof. ◀

7 Orbit dimension, tangent spaces, and flows

In this section we introduce tangent spaces and study their dimensions. We especially study them in the context of $\mathsf{G} f_{\mathrm{com}}$, and $\mathsf{G} f_0$.

The *orbit dimension* of a tensor $f \in \mathbb{C}^{w_1 w_2} \otimes \mathbb{C}^{w_2 w_3} \otimes \cdots \otimes \mathbb{C}^{w_d w_{d+1}}$ is the dimension of the orbit $\mathsf{G} f$ as an affine variety. It can be determined as the dimension of the tangent space T_f of the action of G at f, which is a vector space defined as follows. Let $\mathfrak{g} := \mathbb{C}^{w_1 w_2 \times w_1 w_2} \times \cdots \times \mathbb{C}^{w_d w_{d+1} \times w_d w_{d+1}}$. For $A \in \mathfrak{g}$ we define the *Lie algebra action* $Af := \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} ((\mathrm{id} + \varepsilon A) f - f)$, where $\mathrm{id} \in \mathsf{G}$ is the identity element. We define the vector space

$$T_f := \mathfrak{g} f = \{Af \mid A \in \mathfrak{g}\}.$$

▷ Claim 10. The dimension $\dim T_h$ is the same for all $h \in \mathsf{G} f$.

Proof. Since the action of G is linear, for all $g \in \mathsf{G}$ and $A \in \mathfrak{g}$ we have

$$\begin{aligned} A(gf) &= \lim_{\varepsilon\to 0} \tfrac{1}{\varepsilon}\left((\mathrm{id}+\varepsilon A)(gf) - gf\right) = \lim_{\varepsilon\to 0} \tfrac{1}{\varepsilon}\left(gg^{-1}(\mathrm{id}+\varepsilon A)gf - gf\right) \\ &= g \lim_{\varepsilon\to 0} \tfrac{1}{\varepsilon}\left((\mathrm{id}+\varepsilon(g^{-1}Ag))f - f\right) = g((g^{-1}Ag)f) \end{aligned}$$

Since $A \mapsto g^{-1}Ag$ is a bijection on $\mathfrak{g}$, it follows that $T_{gf} = gT_f$. Hence the claim follows. ◁

In the following we will use Claim 10 to argue $f_0 \notin \mathsf{G}f_{\mathrm{com}}$ by showing that $\dim T_{f_{\mathrm{com}}}$ and $\dim T_{f_0}$ are different.

Let $e, e' \in E^i$ and let $A^{(i)}_{e,e'} \in \mathfrak{g}$ denote the matrix tuple where the i-th matrix has a 1 at position (e, e') and all other entries (also in all other matrices) are 0. Since these matrices form a basis of $\mathfrak{g}$, it follows that

$$\mathfrak{g}f = \mathrm{linspan}\{A^{(i)}_{e,e'}f\}.$$

For a tensor f we define the *support* of f as the set of monomials (i.e., standard basis tensors) for which f has nonzero coefficient. For a linear subspace $V \subseteq \mathbb{C}^{w_1w_2} \otimes \mathbb{C}^{w_2w_3} \otimes \cdots \otimes \mathbb{C}^{w_dw_{d+1}}$ we define the *support* of V as the union of the supports of all $f \in V$.

We write $e \cap e' = \emptyset$ to indicate that two edges e and e' do not share any vertex. We write $|e \cap e'| = 1$ if they share exactly one vertex. We observe that for $f \in \{f_{\mathrm{com}}, f_0\}$ the vector space T_f decomposes into a direct sum of three vector spaces,

$$\begin{aligned} \mathfrak{g}_2 &:= \mathrm{linspan}\{A^{(i)}_{e,e'} \mid 1 \le i \le d, 1 \le e, e' \le w_iw_{i+1}, e \cap e' = \emptyset\} \\ \mathfrak{g}_1 &:= \mathrm{linspan}\{A^{(i)}_{e,e'} \mid 1 \le i \le d, 1 \le e, e' \le w_iw_{i+1}, |e \cap e'| = 1\} \\ \mathfrak{g}_0 &:= \mathrm{linspan}\{A^{(i)}_{e,e} \mid 1 \le i \le d, 1 \le e \le w_iw_{i+1}\}. \\ \mathfrak{g} &= \mathfrak{g}_0 \oplus \mathfrak{g}_1 \oplus \mathfrak{g}_2 \\ T_f &= \mathfrak{g}_0 f \oplus \mathfrak{g}_1 f \oplus \mathfrak{g}_2 f \end{aligned}$$

The last direct sum decomposition follows from the fact that $\mathfrak{g}_0 f$, $\mathfrak{g}_1 f$, and $\mathfrak{g}_2 f$ have pairwise disjoint supports.

We show in this section that $\dim \mathfrak{g}_2 f_{\mathrm{com}} = \dim \mathfrak{g}_2 f_0$, and that $\dim \mathfrak{g}_1 f_{\mathrm{com}} = \dim \mathfrak{g}_1 f_0$. In Section 8 we show that $\dim \mathfrak{g}_0 f_{\mathrm{com}} > \dim \mathfrak{g}_0 f_0$, which then implies $f_0 \notin \mathsf{G}f_{\mathrm{com}}$ by Claim 10. In fact, Theorem 13 gives the exact dimension of $\mathfrak{g}_0 f_{\mathrm{com}}$ by proving that $\mathfrak{g}_0 f_{\mathrm{com}}$ is isomorphic to the vector space of flows on the ABP digraph when identifying vertices in V^1 with their corresponding vertices in V^{d+1}. Theorem 14 establishes an additional equation based on the vertex parities that shows that $\mathfrak{g}_0 f_0$ is strictly lower dimensional than $\mathfrak{g}_0 f_{\mathrm{com}}$.

We start with Lemma 11, which shows that $\dim \mathfrak{g}_2 f_{\mathrm{com}}$ and $\dim \mathfrak{g}_2 f_0$ have *full* dimension.

▶ **Lemma 11.** *Let $f \in \{f_{\mathrm{com}}, f_0\}$. The space $\mathfrak{g}_2 f$ has full dimension. That is, its dimension equals $\sum_{i=1}^{d} w_iw_{i+1}(w_i - 1)(w_{i+1} - 1)$.*

Proof. Suppose $f = f_{\mathrm{com}}$. The other case being analogous, we only argue this case.

We analyze the monomials that appear in the different $A^{(i)}_{e,e'}f_{\mathrm{com}}$ and argue that a monomial that appears in some $A^{(i)}_{e,e'}f_{\mathrm{com}}$ can only appear in that specific $A^{(i)}_{e,e'}f_{\mathrm{com}}$. Indeed, each monomial corresponds to a valid path in which one edge e in layer i is changed to e'. Since e and e' share no vertex, from this edge sequence we can reconstruct i, e, and e' uniquely: e' is the edge that does not have any vertex in common with the rest of the edge sequence, i is its layer, and e is the unique edge that we can replace e' by in order to form a valid path. We conclude that the $A^{(i)}_{e,e'}f_{\mathrm{com}}$ have disjoint support and the lemma follows. ◀

To establish that $\dim \mathfrak{g}_1 f_{\mathrm{com}} = \dim \mathfrak{g}_1 f_0$, we introduce some notation.

For a connected directed graph $G = (V, E)$ we define a *flow* to be a labeling of the edge set E by complex numbers such that at every vertex the sum of the labels of the incoming edges equals the sum of the labels of the outgoing edges. It is easily seen that the set of flows forms a vector space F. We have

$$\dim F = |E| - |V| + 1, \tag{11}$$

see e.g. Theorem 20.7 in [8].

Recall that E^i denotes the set of edges from V^i to V^{i+1}. Let $\mathscr{X} := E^1 \times \cdots \times E^d$ denote the direct product of the sets of edge lists. Each directed path of length d from layer 1 to $d+1$ is an element of $\mathscr{X}$, but $\mathscr{X}$ contains other edge sets as well. Define $E_i := \mathbb{C}^{E^i}$. Consider the following map φ from $\mathscr{X}$ to $E_1 \otimes \cdots \otimes E_d$,

$$\varphi(e_1, \ldots, e_d) = x_{e_1} \otimes \cdots \otimes x_{e_d} \in E_1 \otimes \cdots \otimes E_d$$

where (x_j) is the standard basis of E_i. Note φ is a bijection between $\mathscr{X}$ and the standard basis of $E_1 \otimes \cdots \otimes E_d$.

An edge set in $\mathscr{X}$ is called a *valid path* if it forms a path that starts and ends at corresponding vertices (see Sec. 1). Let $\mathscr{P} \subseteq \mathscr{X}$ denote the set of valid paths.

▶ **Proposition 12.** $\dim \mathfrak{g}_1 f_{\mathrm{com}} = \dim \mathfrak{g}_1 f_0 = \sum_{i=1}^{d} (w_{i-1} + w_{i+1} - 1)(w_i - 1)w_i$, *where* $w_0 := w_d$.

Proof. The proof works almost analogously for f_{com} and f_0, so we treat only the more natural case f_{com}. We show that $\mathfrak{g}_1 f_{\mathrm{com}}$ is isomorphic to a direct sum of vector spaces of flows on very simple digraphs. Fix $1 \le i \le d$. Fix distinct $1 \le a, b \le w_i$. For distinct edges $e, e' \in E^i$, let $\mathscr{P}_{e,e'} \subseteq \mathscr{X}$ be the set of edge sets containing e' that are not valid paths, but that become valid paths by removing e' and adding e. Let $\mathscr{P}^i_{a,b} \subseteq \mathscr{X}$ be the set of edge sets that are not valid paths, but that become valid paths by switching the end point of the $(i-1)$-th edge to v^i_b and that also become valid paths by switching the start point of the i-th edge to v^i_a (if $i - 1 = 0$, then interpret $i - 1 := d$). Pictorially, this means that elements in $\mathscr{P}^i_{a,b}$ are almost valid paths, but there is a discontinuity at layer i, where the path jumps from vertex v^i_a to vertex v^i_b. We have

$$A^{(i)}_{e,e'} f_{\mathrm{com}} = \sum_{p \in \mathscr{P}_{e,e'}} \varphi(p).$$

The vectors $\{A^{(i)}_{e,e'} f_{\mathrm{com}} \mid 1 \le i \le d, e, e' \in E^i, |e \cap e'| = 1\}$ are not linearly independent, because for $a \neq b$ we have

$$\sum_{\substack{e \text{ and } e' \text{ have the same start point} \\ e' \text{ ends at the } a\text{-th vertex} \\ e \text{ ends at the } b\text{-th vertex}}} A^{(i-1)}_{e,e'} f_{\mathrm{com}} = \sum_{p \in \mathscr{P}^i_{a,b}} \varphi(p) = \sum_{\substack{h \text{ and } h' \text{ have the same end point} \\ h \text{ starts at the } a\text{-th vertex} \\ h' \text{ starts at the } b\text{-th vertex}}} A^{(i)}_{h,h'} f_{\mathrm{com}}. \tag{12}$$

Define

$$T_{a,b,i} := \operatorname{linspan}\left\{ A^{(i-1)}_{e,e'} f_{\mathrm{com}} \;\middle|\; \begin{array}{c} e \text{ and } e' \text{ have the same start point} \\ e' \text{ ends at the } a\text{-th vertex} \\ e \text{ ends at the } b\text{-th vertex} \end{array} \right\} + \operatorname{linspan}\left\{ A^{(i)}_{h,h'} f_{\mathrm{com}} \;\middle|\; \begin{array}{c} h \text{ and } h' \text{ have the same end point} \\ h \text{ starts at the } a\text{-th vertex} \\ h' \text{ starts at the } b\text{-th vertex} \end{array} \right\}.$$

The support of $T_{a,b,i}$ and $T_{\tilde{a},\tilde{b},\tilde{i}}$ are disjoint, provided $(a,b,i) \neq (\tilde{a},\tilde{b},\tilde{i})$. Hence

$$\mathfrak{g}_1 f_{\mathrm{com}} = \bigoplus_{\substack{1\leq i\leq d\\ 1\leq a,b\leq w_i\\ a\neq b}} T_{a,b,i}$$

It remains to prove that the dimension of $T_{a,b,i}$ is $w_{i-1}+w_{i+1}-1$, because then

$$\dim \mathfrak{g}_1 f_{\mathrm{com}} = \sum_{\substack{1\leq i\leq d\\ 1\leq a,b\leq w_i\\ a\neq b}} (w_{i-1}+w_{i+1}-1) = \sum_{i=1}^{d} (w_{i-1}+w_{i+1}-1)(w_i-1)w_i.$$

Note that $T_{a,b,i}$ is defined as the linear span of $w_{i-1}+w_{i+1}$ many vectors, but (12) shows that these are not linearly independent. We prove that (12) is the only equality by showing that $T_{a,b,i}$ is isomorphic to a flow vector space. We define a multigraph with two vertices: $\odot$ and $\circledast$. We have w_{i+1} many edges from $\odot$ to $\circledast$, and we have w_{i-1} many edges from $\circledast$ to $\odot$. We denote by $\circledast \xrightarrow{k} \odot$ the k-th edge from $\circledast$ to $\odot$. Let $F_{a,b,i}$ denote the vector space of flows on this graph. Its dimension is $w_{i-1}+w_{i+1}-1$, see (11). We define $\varrho : E^1 \otimes \cdots \otimes E^d \to F_{a,b,i}$ on rank 1 tensors via

$$\varrho(x_{e_1}\otimes\cdots\otimes x_{e_d})(\circledast \xrightarrow{k} \odot) = \begin{cases} 1 & \text{if } e_{i-1} \text{ starts at } k \text{ in layer } i-1 \text{ and ends at } a \text{ in layer } i,\\ 0 & \text{otherwise.}\end{cases}$$

$$\varrho(x_{e_1}\otimes\cdots\otimes x_{e_d})(\odot \xrightarrow{l} \circledast) = \begin{cases} 1 & \text{if } e_i \text{ starts at } b \text{ in layer } i \text{ and ends at } l \text{ in layer } i+1,\\ 0 & \text{otherwise.}\end{cases}$$

Using (12) it is readily verified that ϱ maps $T_{a,b,i}$ to $F_{a,b,i}$. It remains to show that $\varrho : T_{a,b,i} \to F_{a,b,i}$ is surjective. Let $\alpha := |\mathscr{P}^i_{a,b}|$. We observe that

$$\begin{aligned}
\varrho(A^{(i-1)}_{e,e'} f_{\mathrm{com}})(\circledast \xrightarrow{k} \odot) &= \begin{cases} \alpha/w_{i-1} & \text{if } e \text{ and } e' \text{ both start at the } k\text{-th vertex}\\ 0 & \text{if } e \text{ and } e' \text{ both start at the same vertex, but not at the } k\text{-th}\end{cases}\\
\varrho(A^{(i-1)}_{e,e'} f_{\mathrm{com}})(\odot \xrightarrow{l} \circledast) &= \alpha/(w_{i-1}w_{i+1})\\
\varrho(A^{(i)}_{h,h'} f_{\mathrm{com}})(\odot \xrightarrow{l} \circledast) &= \begin{cases} \alpha/w_{i+1} & \text{if } h \text{ and } h' \text{ both end at the } l\text{-th vertex}\\ 0 & \text{if } h \text{ and } h' \text{ both end at the same vertex, but not at the } l\text{-th}\end{cases}\\
\varrho(A^{(i)}_{h,h'} f_{\mathrm{com}})(\circledast \xrightarrow{k} \odot) &= \alpha/(w_{i-1}w_{i+1})
\end{aligned}$$

Let $\Xi := \sum A^{(i-1)}_{e,e'} f_{\mathrm{com}}$. Then $\forall k : \varrho(\Xi)(\circledast \xrightarrow{k} \odot) = \alpha/w_{i-1}$ and $\forall l : \varrho(\Xi)(\odot \xrightarrow{l} \circledast) = \alpha$. Therefore, for e, e' starting at the k_0-th vertex and h, h' ending at the l_0-th vertex we have that

$$\varrho\Big(w_{i-1}w_{i+1}\varrho(A^{(i-1)}_{e,e'} f_{\mathrm{com}}) + w_{i-1}w_{i+1}\varrho(A^{i}_{h,h'} f_{\mathrm{com}}) - \Xi\Big)$$

is nonzero only on exactly two edges: $\circledast \xrightarrow{k_0} \odot$ and $\odot \xrightarrow{l_0} \circledast$. Cycles form a generating set of the vector space $F_{a,b,i}$, which finishes the proof of the surjectivity of ϱ. ◀

8 Flows on ABPs

We now proceed to the analysis of $\mathfrak{g}_0 f_{\mathrm{com}}$ and $\mathfrak{g}_0 f_0$. The connection to flow vector spaces will be even more prevalent than in Proposition 12. The main result of this section is $\dim \mathfrak{g}_0 f_{\mathrm{com}} > \dim \mathfrak{g}_0 f_0$ (Theorems 13 and 14), which implies that f_{com} and f_0 have different orbit dimensions. We thereby conclude that $f_0 \notin \mathsf{G} f_{\mathrm{com}}$.

To each edge e we assign its *path tensor* $\psi(e)$ by summing tensors over all valid paths passing through e,

$$\psi(e) := \sum_{p \in \mathscr{P} \text{ with } e \in p} \varphi(p) \in E_1 \otimes \cdots \otimes E_d.$$

By linear continuation this gives a linear map $\psi : \mathbb{C}^E \to E_1 \otimes \cdots \otimes E_d$.

Observe that $\psi(e) = A^{(i)}_{e,e} f_{\mathrm{com}}$. Let $\mathscr{T}$ denote the linear span of all $\psi(e)$, $e \in E$. In other words, $\mathscr{T} = \mathfrak{g}_0 f_{\mathrm{com}}$.

Let $\mathscr{P}' \subseteq \mathscr{P} \subseteq \mathscr{X}$ be the set of valid paths that contain exactly one parity preserving edge. To each edge e we assign its *parity path tensor* $\psi'(e)$ by summing tensors over paths in $\mathscr{P}'$,

$$\psi'(e) := \sum_{p \in \mathscr{P}' \text{ with } e \in p} \varphi(p) \in E_1 \otimes \cdots \otimes E_d.$$

By linear continuation this gives a linear map $\psi' : \mathbb{C}^E \to E_1 \otimes \cdots \otimes E_d$. Observe that $\psi'(e) = A^{(i)}_{e,e} f_0$. Let $\mathscr{T}'$ denote the linear span of all $\psi'(e)$, $e \in E$. In other words, $\mathscr{T}' = \mathfrak{g}_0 f_0$.

We will establish the following bounds on the dimensions of $\mathscr{T}$ and $\mathscr{T}'$.

▶ **Theorem 13.** $\dim \mathscr{T} = |E| - \sum_{i=1}^{d} w_i + 1$.

▶ **Theorem 14.** $\dim \mathscr{T}' \leq |E| - \sum_{i=1}^{d} w_i$.

The rest of this section is dedicated to the proofs of Theorem 13 and Theorem 14 by showing that $\mathscr{T}$ is isomorphic to the vector space of flows "on the ABP", while the parity constraints lead to a smaller dimension of $\mathscr{T}'$.

From an ABP Γ we construct a digraph $\tilde{\Gamma}$ by identifying corresponding vertices from the first and the last layer in V and calling the resulting vertex set $\tilde{V}$. Note $|\tilde{V}| = \sum_{i=1}^{d} w_i$. The directed graphs Γ and $\tilde{\Gamma}$ have the same edge set. The resulting directed graph is called $\tilde{\Gamma} = (\tilde{V}, E)$. Let F denote the vector space of flows on $\tilde{\Gamma}$. Note that by (11) we have $\dim F = |E| - |\tilde{V}| + 1$. All directed cycles in $\tilde{\Gamma}$ have a length that is a multiple of d. In particular, all cycles of length exactly d are in one-to-one correspondence with valid paths in Γ_{com}. For an edge $e \in E$, let $\chi(e) \in \mathbb{C}^E$ denote the characteristic function of e, i.e., the function whose value is 1 on e and 0 everywhere else.

We now prove Theorem 13 by establishing a matching upper (Lemma 15) and lower bound (Lemma 16) of $|E| - |\tilde{V}| + 1 = \dim F$ on $\dim \mathscr{T}$.

The upper bound

▶ **Lemma 15.** $\dim \mathscr{T} \leq |E| - |\tilde{V}| + 1$.

Proof. For $v \in \tilde{V}$, let $\mathsf{in}(v) \subseteq E$ denote the set of incoming edges incident to v and $\mathsf{out}(v) \subseteq E$ denote the set of outgoing edges incident to v. For each $v \in \tilde{V}$, define the row vector

$$r_v = \sum_{e \in \mathsf{in}(v)} \chi(e) - \sum_{e \in \mathsf{out}(v)} \chi(e).$$

These vectors are the rows of the signed incidence matrix of $\tilde{\Gamma}$, and since $\tilde{\Gamma}$ is connected, they span a space of dimension $|\tilde{V}| - 1$ ([8, Ex. 1.5.6]). Now observe that for all $v \in \tilde{V}$,

$$\sum_{e \in \mathsf{in}(v)} \psi(e) = \sum_{e \in \mathsf{out}(v)} \psi(e).$$

Since ψ is linear, this is equivalent to

$$\psi\left(\sum_{e \in \mathsf{in}(v)} \chi(e) - \sum_{e \in \mathsf{out}(v)} \chi(e) \right) = 0.$$

Hence each r_v is in the kernel of ψ, and hence $\dim \ker \psi \geq |\tilde{V}| - 1$. Using (11), we obtain $\dim \mathscr{T} = \dim \operatorname{im} \psi = |E| - \dim \ker \psi \leq |E| - |\tilde{V}| + 1 = \dim F$. ◀

The lower bound

To obtain the lower bound, we define a linear map $\varrho : E_1 \otimes \cdots \otimes E_d \to \mathbb{C}^E$ such that the image of ϱ restricted to $\mathscr{T}$ equals F. This will imply that $\dim \mathscr{T} \geq \dim F$, thereby achieving the required lower bound.

We define the linear map ϱ on standard basis elements $x_{e_1} \otimes \cdots \otimes x_{e_d}$ as follows,

$$\varrho(x_{e_1} \otimes \cdots \otimes x_{e_d}) := \chi(e_1) + \cdots + \chi(e_d),$$

and then extend it to the domain $E_1 \otimes \cdots \otimes E_d$ via linear continuation.

▶ **Lemma 16.** *Let $\varrho|_{\mathscr{T}}$ denote the restriction of ϱ to the linear subspace $\mathscr{T}$. Then,* $\operatorname{im} \varrho|_{\mathscr{T}} = F$. *In particular,* $\dim \mathscr{T} \geq \dim F = |E| - |\tilde{V}| + 1$.

Proof. To prove equality it suffices to show $\operatorname{im} \varrho|_{\mathscr{T}} \subseteq F$ and $F \subseteq \operatorname{im} \varrho|_{\mathscr{T}}$.

The first containment is easy to see. For an edge e, consider the image of $\psi(e)$ under the map ϱ,

$$\varrho(\psi(e)) = \sum_{e \in p \in \mathscr{P}} \sum_{e' \in p} \chi(e').$$

Observe that for a path $p \in \mathscr{P}$, $\sum_{e' \in p} \chi(e')$ is a flow on $\tilde{\Gamma}$ and hence it belongs to F. Thus, we have $\varrho(\psi(e)) \in F$. Since $\mathscr{T}$ is spanned by $\psi(e)$, for $e \in E$, we obtain that $\operatorname{im} \varrho|_{\mathscr{T}} \subseteq F$.

To establish the second containment it suffices to show that the image of $\mathscr{T}$ under the map ϱ contains a basis of F. We identify a specific basis for F in Claim 17 and prove that it is contained in $\operatorname{im} \varrho|_{\mathscr{T}}$ in Claim 18 to complete the argument. ◀

We identify directed cycles with their characteristic flows, i.e., flows that have value 1 on the cycle's edges and 0 everywhere else. We also identify directed cycles that use edges in any direction with their characteristic flow: the characteristic flow is defined to take the value 1 on an edge e if e is traversed in the direction of e, and value -1 on e if e is traversed against its direction.

From the theory of flows we know that for every (undirected) spanning tree T of $\tilde{\Gamma}$, the vector space $F \in \mathbb{C}^E$ has a basis given by the characteristic flows of cycles that only use edges from T and exactly one additional edge (for example, see Theorem 20.8 in [8]). Thus, the cycle flows corresponding to the elements not in the spanning tree form a basis of F.

▷ Claim 17. F is spanned by the set of directed cycles in $\tilde{\Gamma}$ of length exactly d.

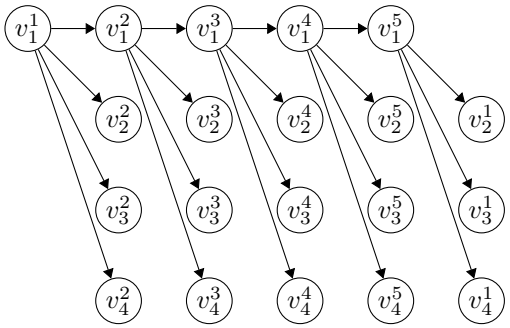


Figure 1 The spanning tree construction for width 4 and $d = 5$.

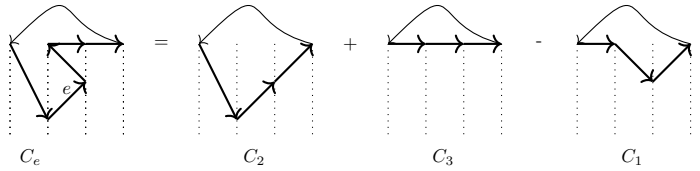


Figure 2 Decomposing a cycle of length $d+2$ as a linear combination of cycles of length d. The figure is an illustration when $d = 3$. The dotted layers in each cycle from the left are V^3, V^1, V^2, and V^3 again.

Proof. We construct a spanning tree τ as follows, which will be a tree whose edges are all directed away from its root. Informally, the tree is given by the following subgraph, we make the first vertex in V^1 as root, and include all the outgoing edges incident to it. We then move to the first vertex in V^2 and include all the outgoing edges incident to it. We continue in this way until we reach V^d. Upon reaching the first vertex in V^d we include all but one outgoing edges incident to it. The one that is an incoming edge to the root is not included. Figure 1 illustrates the construction. We now formally define this.

Let $v_1^i \in V^i$ denote the first vertex in the layer i, $1 \le i \le d$. Further recall $\mathsf{in}(v) \subseteq E$ and $\mathsf{out}(v) \subseteq E$ denote the set of incoming and outgoing edges, respectively, incident to v. Define the edge set

$$\tau := \left(\bigcup_{i=1}^{d} \mathsf{out}(v_1^i) \right) \setminus \{(v_1^d, v_1^1)\},$$

which is a spanning tree in $\tilde{\Gamma}$. We know that every edge not in the tree when added to the tree gives a unique undirected cycle. We now show that the characteristic flows of these undirected cycles can be expressed as a linear combination of the characteristic flows of directed cycles of length d. For $e \in E \setminus \tau$, let c_e denote the characteristic flow of the unique undirected cycle that uses e in its correct direction and only edges of τ. We argue depending on which layer the edge e belongs to.

- Suppose $e \in E^1 \setminus \tau$.
  - If e is incident to v_1^2, the first vertex in V^2, then the inclusion of e creates a directed cycle of length d. Hence, c_e equals the characteristic flow of this directed cycle.
  - Otherwise, the inclusion of e creates an undirected cycle of length $d+2$. If $e = (v_{j_1}^1, v_{j_2}^2)$ for some $j_1 \in [2, w_1]$ and $j_2 \in [2, w_2]$, then the cycle c_e is given as follows:

    $$v_1^d - v_{j_1}^1 - v_{j_2}^2 - v_1^1 - v_1^2 - \cdots - v_1^{d-1} - v_1^d.$$

Consider the following two directed cycles:

$$C_1 : v_1^1 - v_{j_2}^2 - \cdots - v_1^d - v_1^1 \text{ and}$$
$$C_2 : v_{j_1}^1 - v_{j_2}^2 - \cdots - v_1^d - v_{j_1}^1,$$

such that the part $v_{j_2}^2 - \cdots - v_1^d$ between $v_{j_2}^2$ and v_1^d in the two cycles is the same. Let us denote the characteristic flow of a cycle C by $\chi(C)$. We now observe that $\chi(C_2) - \chi(C_1)$ equals the characteristic flow of the undirected cycle $v_{j_1}^1 - v_{j_2}^2 - v_1^1 - v_1^d - v_{j_1}^1$. This is because the common part in C_1 and C_2 cancels out. To $\chi(C_2) - \chi(C_1)$ we add the characteristic flow of the directed cycle,

$$C_3 : v_1^1 - v_1^2 - v_1^3 - \cdots - v_1^{d-1} - v_1^d - v_1^1.$$

It is now easily seen that $\chi(C_2) - \chi(C_1) + \chi(C_3)$ equals the characteristic flow of the cycle c_e (see Figure 2 for an illustration).

- Suppose $e \in E^d \setminus \tau$.
  - If e is incident to v_1^1, the first vertex in V^1, then as before the inclusion of e creates a directed cycle of length d. Hence, c_e equals the characteristic flow of this directed cycle.
  - Otherwise, the inclusion of e creates an undirected cycle of length 4. If $e = (v_{j_1}^d, v_{j_2}^1)$ for some $j_1 \in [2, w_d]$ and $j_2 \in [2, w_1]$, then the cycle c_e is given as follows:

    $$v_{j_1}^d - v_{j_2}^1 - v_1^d - v_1^{d-1} - v_{j_1}^d.$$

    Consider the following two directed cycles:

    $$C_4 : v_{j_2}^1 - \cdots - v_1^{d-1} - v_1^d - v_{j_2}^1 \text{ and}$$
    $$C_5 : v_{j_2}^1 - \cdots - v_1^{d-1} - v_{j_1}^d - v_{j_2}^1,$$

    such that the part $v_{j_2}^1 - \cdots - v_1^{d-1}$ between $v_{j_2}^1$ and v_1^{d-1} in the two cycles is the same. We now claim that $\chi(C_5) - \chi(C_4)$ equals the characteristic flow of c_e. This is because the common part in C_4 and C_5 cancels out.
- Otherwise $e \in E^i \setminus \tau$ for some $i \in \{2, \ldots, d-1\}$. In such a case inclusion of e creates an undirected cycle of length 4. We can again argue exactly like in the previous case, and so we omit the argument here. ◁

We now prove that the generating set given by the directed cycles of length d is contained in the image of $\mathscr{T}$ under the map ϱ.

▷ Claim 18. $\mathrm{im}(\varrho|_{\mathscr{T}})$ contains the characteristic flow of each directed cycle of length d.

Proof. Let $\{e_1, e_2, \ldots, e_d\} \subseteq E$ be a directed cycle of length d, where each e_i points from a vertex in V^i to a vertex in V^{i+1}. Let $\{e_i^{(j)}\}$ denote the set of edges that start at the same vertex as e_i, but for which $e_i^{(j)} \neq e_i$. Thus $|\{e_i^{(j)}\}| = |V^{i+1}| - 1$. Let

$$\bar{\psi}(e) := \frac{1}{|\{p \in \mathscr{P} \text{ with } e \in p\}|} \psi(e),$$

so that $\varrho(\bar{\psi}(e))$ is a flow with value 1 on the edge e. It is instructive to have a look at the left side of Figure 3, where $\varrho(\bar{\psi}(e_1))$ is depicted. Subtracting $\frac{1}{w_3} \sum_{j=1}^{w_3 - 1} \varrho(\bar{\psi}(e_2^{(j)}))$ and adding $\frac{w_3 - 1}{w_3} \varrho(\bar{\psi}(e_2))$ reduces the support significantly and brings us one step closer to the cycle, see the right side of Figure 3. We iterate this process until only the cycle is left. Formally:

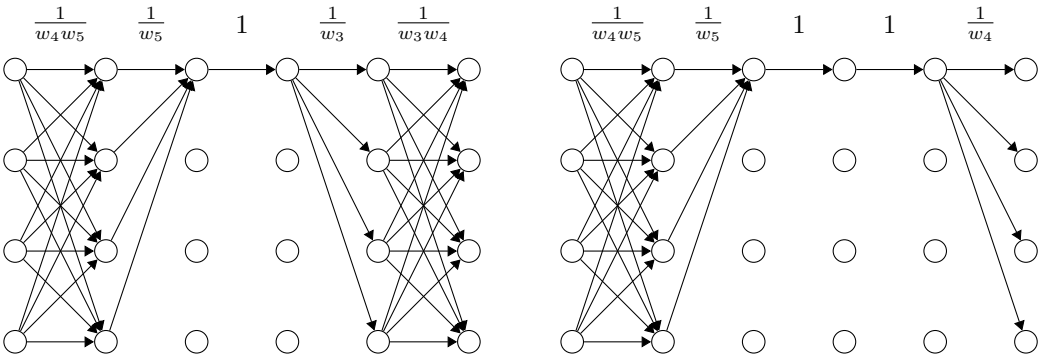


Figure 3 On the left: $\varrho(\bar{\psi}(e_1))$. On the right: $\varrho(\bar{\psi}(e_1)) - \frac{1}{w_3}\sum_{j=1}^{w_3-1}\varrho(\bar{\psi}(e_2^{(j)})) + \frac{w_3-1}{w_3}\varrho(\bar{\psi}(e_2))$. This is the case $d = 5$ and format $(4, 4, 4, 4, 4)$. Edges that are not drawn carry 0 flow. All edges in the same layer carry either 0 flow or the value that is depicted above the edge layer. For the purposes of illustation, e_1 is the top edge in the *center*. Here we assume that each e_i points from the first vertex in V^i to the first vertex in V^{i+1}.

$$\begin{aligned}\chi(e_1,\ldots,e_d) &= \varrho(\bar{\psi}(e_1))\\ &\quad + \tfrac{w_3-1}{w_3}\varrho(\bar{\psi}(e_2)) - \tfrac{1}{w_3}\sum_{j=1}^{w_3-1}\varrho(\bar{\psi}(e_2^{(j)}))\\ &\quad + \cdots\\ &\quad + \tfrac{w_d-1}{w_d}\varrho(\bar{\psi}(e_{d-1})) - \tfrac{1}{w_d}\sum_{j=1}^{w_d-1}\varrho(\bar{\psi}(e_{d-1}^{(j)})).\end{aligned}$$ ◁

The stronger upper bound via parities

We now proceed to upper bound $\dim \mathscr{T}'$ (Theorem 14). The proof is analogous to the proof of Lemma 15.

▶ **Theorem 19** (Restatement of Theorem 14). $\dim \mathscr{T}' \leq |E| - |\tilde{V}|$.

Proof. As in the proof of Lemma 15, for $v \in \tilde{V}$, we have

$$\sum_{e\in\mathsf{in}(v)}\psi'(e) = \sum_{e\in\mathsf{out}(v)}\psi'(e).$$

Furthermore, we have the following additional constraint on ψ',

$$(d-1)\sum_{e \text{ parity preserving}}\psi'(e) = \sum_{e \text{ parity changing}}\psi'(e).$$

By the linearity of ψ', we have

$$\psi'\left((d-1)\sum_{e \text{ parity preserving}}\chi(e) - \sum_{e \text{ parity changing}}\chi(e)\right) = 0.$$

Therefore, the kernel of ψ' is spanned by the vectors $(\sum_{e\in\mathsf{in}(v)}\chi(e) - \sum_{e\in\mathsf{out}(v)}\chi(e))$, for $v \in \tilde{V}$, and an additional vector $((d-1)\sum_{e \text{ parity preserving}}\chi(e) - \sum_{e \text{ parity changing}}\chi(e))$.

We now claim that the new vector is linearly independent from the earlier set of vectors. We prove the claim by constructing a vector in $\mathbb{C}^E$ that is orthogonal to the earlier set of vectors but is non-orthogonal to the additional vector. One such vector is given by the characteristic flow of the directed cycle $v_1^1 - v_1^2 - v_1^3 - \cdots - v_1^{d-1} - v_1^d - v_1^1$.

Thus, it follows that $\dim \ker \psi' \geq |\tilde{V}|$, and hence $\dim \mathscr{T}' \leq |E| - |\tilde{V}|$. ◀

9 VQP versus $\overline{\text{VNP}}$

In this section, we compare the complexity classes VQP and $\overline{\text{VNP}}$. Valiant in his seminal paper [41] defined the complexity classes that are now called as VP and VNP, and the central question of algebraic complexity is to understand whether the two complexity classes are indeed different as sets (Valiant's hypothesis). Bürgisser [11] defined the complexity class VQP and related it to the complexity classes VP and VNP. We proceed to define the above three classes for establishing the context. For an exhaustive treatment of the classes, we refer the readers to Bürgisser's monograph [11] from where we are lifting the definitions. We first need to define so-called p-families.

▶ **Definition 20.** *A sequence $f = (f_n)$ of multivariate polynomials over a field k is called a p-family (over k) iff the number of variables as well as the degree of f_n are bounded by polynomial functions in n.*

We now need to define the model of computation and the notion of complexity in order to define the complexity classes of interest.

▶ **Definition 21.** *A* straight-line program Γ *(expecting m inputs) represents a sequence $(\Gamma_1, \ldots, \Gamma_r)$ of instructions $\Gamma_\rho = (\omega_\rho; i_\rho, j_\rho)$ with operation symbols $\omega_\rho \in \{+, -, *\}$ and the address i_ρ, j_ρ which are integers satisfying $-m < i_\rho, j_\rho < \rho$. We call r the size of Γ.*

So, essentially, in a straight-line program, we either perform addition or subtraction or multiplication on the inputs or the previously computed elements. The size of the straight-line program naturally induces a size complexity measure on polynomials as follows:

▶ **Definition 22.** *The* complexity *$L(f)$ of a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is the minimal size of a straight-line program computing f from variables x_i and constants in $\mathbb{F}$.*

We are now all set to define the above discussed complexity classes.

▶ **Definition 23.** *A p-family $f = (f_n)$ is said to be p-computable iff the complexity $L(f_n)$ is a polynomially bounded function of n. $\text{VP}_\mathbb{F}$ consists of all p-computable families over the field $\mathbb{F}$.*

▶ **Definition 24.** *A p-family $f = (f_n)$ is said to be* p-definable *iff there exists a* p-computable *family $g = (g_n)$, $g_n \in \mathbb{F}[x_1, \ldots, x_{u(n)}]$, such that for all n*

$$f_n(x_1, \ldots, x_{v(n)}) = \sum_{e \in \{0,1\}^{u(n)-v(n)}} g_n(x_1, \ldots, x_{v(n)}, e_{v(n)+1}, \ldots, e_{u(n)}).$$

The set of p-definable families over $\mathbb{F}$ forms the complexity class $\text{VNP}_\mathbb{F}$.

▶ **Definition 25.** *A p-family $f = (f_n)$ is said to be qp-*computable *iff the complexity $L(f_n)$ is a quasi-polynomially bounded function of n. The complexity class $\text{VQP}_\mathbb{F}$ consists of all qp-computable families over $\mathbb{F}$.*

In the above three definitions, if the underlying field is clear from the context, we can drop the subscript $\mathbb{F}$ and simply represent the classes as VP, VNP and VQP respectively. In what follows, the underlying field is always assumed to be $\mathbb{Q}$, the field of rational numbers.

In [11], Bürgisser showed the completeness of the determinant polynomial for VQP under *qp*-projections and strengthened Valiant's hypothesis of VNP $\not\subseteq$ VP to VNP $\not\subseteq$ VQP and called it *Valiant's extended hypothesis* (see [11], Section 2.5). He also established that VP $\subsetneq$ VQP and went on to show that VQP $\not\subseteq$ VNP (see [11], Proposition 8.5 and Corollary 8.9). The main observation of this section is that his proof is stronger and is sufficient to conclude that VQP is not contained in the closure of VNP either, where the closure is in the sense as mentioned in Section 1.

In fact, Bürgisser in his monograph [11] also gives a set of conditions which if the coefficients of a polynomial sequence satisfies, then that polynomial sequence cannot be in VNP [11, Theorem 8.1]. His theorem and the proof is inspired by Heintz and Sieveking [22]. The second observation of this section is that this proof is even stronger and actually those conditions are sufficient to show that the given polynomial sequence is not contained in $\overline{\text{VNP}}$ either.

We now discuss both the observations.

9.1 VQP $\not\subseteq \overline{\text{VNP}}$

We first show that there is a $\log n$ variate polynomial of degree $(n-1)\log n$ which is in VQP but not in $\overline{\text{VNP}}$. In this exposition, for the sake of better readability, we do not present the Bürgisser's statements in full generality since it is not essential for the theorem that we want to show here. Moreover, the less general version that we present here contains all the ideas for the theorem statements and their proofs.

▶ **Theorem 26.** *Let* $N_n := \{0, \ldots, n-1\}^{\log n}$ *and* $f_n := \sum_{\mu \in N_n} 2^{2^{j(\mu)}} X_1^{\mu_1} \cdots X_{\log n}^{\mu_{\log n}}$, *where* $j(\mu) := \sum_{j=1}^{\log n} \mu_j n^{j-1}$. *Then* $f_n \in$ VQP, *but* $f_n \notin \overline{\text{VNP}}$, *and hence* VQP $\not\subseteq \overline{\text{VNP}}$.

The theorem consists of two parts. The containment in VQP follows immediately from the fact that the total number of monomials in f_n is $n^{\log n}$. For the other part, we closely follow Bürgisser's lower bound proof [11, Proposition 8.5] against VNP here, making transparent the fact that the proof works also against $\overline{\text{VNP}}$. His proof techniques were borrowed from Strassen ([39]). The idea is to use the universal representation for polynomial sequences in VNP, so that we get a hold on how the coefficients of the polynomials look like. Using that, we establish polynomials H_n that vanish on all the polynomial sequences in VNP (in other words, H_n is in the vanishing ideal of sequences in VNP), but do not vanish on f_n (because the growth rate of its coefficients is too high), hence giving the separation. Since the vanishing ideal of a set characterizes its closure, we get the stronger separation, i.e., f_n does not belong to the closure of VNP, namely, $\overline{\text{VNP}}$.

Proof of Theorem 26. As stated above, the proof works in three stages: first, assuming the contrary and writing f_n using the universal representation for the polynomial sequences in VNP, then giving polynomials H_n of special forms in the vanishing ideal of polynomial sequences in VNP, and finally showing that H_n cannot vanish on our sequence f_n, hence arriving at a contradiction.

Assuming $(f_n) \in$ VNP implies the existence of a family $(g_n) \in$ VP, with $L(g_n)$ bounded by a polynomial $r(n)$, and a polynomial $u(n)$ such that

$$f_n(X_1, \ldots, X_{\log n}) = \sum_{e \in \{0,1\}^{u(n)-\log n}} g_n(X_1, \ldots, X_{\log n}, e_{\log n+1}, \ldots, e_{u(n)}).$$

Next, we use the universal representation theorem (see [39], [37]) as stated in Bürgisser's monograph ([11], Proposition 8.3; for a proof see [13], Proposition 9.11) for size $r(n)$ straight-line program to get that there exist polynomials $G_\nu^{(n)} \in \mathbb{Z}[Y_1, \ldots, Y_{q(n)}]$, with $q(n)$ being a polynomial in n (more precisely, it is a polynomial in $r(n)$ and $u(n)$) which for $|\nu| \leq \deg g_n = n^{O(1)}$, guarantee that $\deg G_\nu = n^{O(1)}, \log \mathsf{wt}(G_\nu)^{(n)} = 2^{n^{O(1)}}$, and also guarantee the existence of some $\zeta \in \overline{\mathbb{Q}}^{q(n)}$, such that

$$g_n = \sum_\nu G_\nu^{(n)}(\zeta) X_1^{\nu_1} \cdots X_{u(n)}^{\nu_{u(n)}},$$

where weight of a polynomial f, $\mathsf{wt}(f)$ refers to the sum of the absolute values of its coefficients.

Now, taking exponential sum yields that

$$f_n = \sum_{\mu \in N_n} F_\mu^{(n)}(\zeta) X_1^{\mu_1} \cdots X_{\log n}^{\mu_{\log n}},$$

where the polynomials $F_\mu^{(n)}$ are obtained as a sum of at most $2^{u(n)}$ polynomials $G_\nu^{(n)}$. Thus, we now have a good hold on $F_\mu^{(n)}$, i.e. $\deg F_\mu^{(n)} \leq \alpha(n)$ and $\log \mathsf{wt}(F_\mu^{(n)}) \leq 2^{\beta(n)}$, where both $\alpha(n)$ and $\beta(n)$ are polynomially bounded functions of n.

Thus, for f_n to be in VNP, the coefficients of f_n should be in the image of the polynomial map $F_\mu^n : \overline{\mathbb{Q}}^{q(n)} \to \overline{\mathbb{Q}}^{n^{\log n}}$. In other words, we must have some $\zeta \in \overline{\mathbb{Q}}^{q(n)}$, such that for all $\mu \in N_n$, we have $F_\mu^n(\zeta) = 2^{2^{j(\mu)}}$, where $j(\mu) := \sum_{j=1}^{\log n} \mu_j n^{j-1}$. Since F_μ^n takes all the values from 2^{2^0} to $2^{2^{n^{\log n}-1}}$, we have a subset of indices $\tilde{N}_n \subseteq N_n$ of size $s(n) := \lfloor |N_n|/n \rfloor = \lfloor n^{\log n}/n \rfloor$, such that for $\sigma \in \{0, 1, \ldots, s(n)-1\}$ and a bijection $\delta : \{0, 1, \ldots, s(n)-1\} \to \tilde{N}_n$ with $\sigma \mapsto \delta(\sigma)$, we have $F_{\delta(\sigma)}^n = 2^{2^{\sigma n+1}}$.

Now we can apply Lemma 9.28 from [13] which asserts that there will be polynomials of low height (ht) (the maximum of the absolute value of the coefficients) on which these coefficients shall vanish. More precisely, there exists non-zero forms $H_n \in \mathbb{Z}[Y_\mu \mid \mu \in \tilde{N}_n]$ with $\mathrm{ht}(H_n) \leq 3$, $\deg H_n \leq D(n)$, and such that $H_n(F_\mu^n \mid \mu \in N_n) = 0$, given that $D(n)^{s(n)-q(n)-2} > \alpha(n)^{q(n)} s(n)^{s(n)} 2^{\beta(n)}$.

It can be seen that $D(n) = 2^n - 1$ satisfies the above inequality, since $\alpha(n), \beta(n)$ and $q(n)$ are polynomially bounded and 2^n grows much faster than $s(n) = \lfloor n^{\log n}/n \rfloor$. This allows us to write $H_n = \sum_e \lambda_e \prod_{\mu \in \tilde{N}_n} Y_\mu^{e_\mu}$, where the absolute values of λ_e are bounded by 3. Since H_n vanishes on the subset of coefficients of f_n, i.e it vanishes on $F_{\delta(\sigma)}^n = 2^{2^{\sigma n+1}}$ with $\sigma \in \{0, 1, \ldots, s(n)-1\}$, we have

$$0 = H_n(F_\mu^n \mid \mu \in \tilde{N}_n) = \sum_e \lambda_e \prod_{\sigma=0}^{s(n)-1} 2^{e_{\delta(\sigma)} 2^{\sigma n+1}} = \sum_e \lambda_e \cdot 4^{\sum_\sigma e_{\delta(\sigma)} (2^n)^\sigma}.$$

The last sum is essentially a 4-adic integer, since firstly, $|\lambda_e| \leq 3$, and secondly, all the exponents of 4, that is, $\sum_\sigma e_{\delta(\sigma)} (2^n)^\sigma$ are all distinct, as they can be seen as 2^n-adic representation since $e_{\delta(\sigma)} < 2^n$. Thus λ_e has to be zero for all e. Hence H_n must be identically zero, which is a contradiction. ◀

9.2 A criterion for non-membership in $\overline{\mathrm{VNP}}$

In this section, we discuss a criterion Bürgisser presented in his monograph [11] based on a proof due to Heintz and Sieveking which gives a set of conditions that puts a p-family out of VNP. We observe that those conditions if satisfied, in fact, put a given p-family out of $\overline{\mathrm{VNP}}$ as well.

▶ **Theorem 27.** *Let (p_n) be a sequence of polynomials over $\overline{\mathbb{Q}}$ and let $N(n)$ denote the degree of the field extension generated by the coefficients of p_n over $\mathbb{Q}$. Further suppose the following holds:*

1. *The map $n \mapsto \lceil \log N(n) \rceil$ is not p-bounded.*
2. *For all n, there is a system G_n of rational polynomials of degree at most $D(n)$ with finite zeroset, containing the coefficient system of f_n, and such that $n \mapsto \lceil \log D(n) \rceil$ is p-bounded.*

Then the family $(p_n) \notin \overline{\mathrm{VNP}}$.

Thus the above theorem shows that certain p-families with algebraic coefficients of high degree are not contained in $\overline{\mathrm{VNP}}$. We now give a simple example from [11] to illustrate the theorem.

▶ **Example 28.** Consider the following multivariate family defined as

$$p_n = \sum_{e \in \{0,1\}^n \setminus 0} \sqrt{p_{j(e)}} X^e,$$

where $j(e) = \sum_{s=1}^{n} e_s 2^{s-1}$ and p_j refers to the j-th prime number. Then using the above Theorem 27, we can conclude that $p_n \notin \overline{\mathrm{VNP}}$. This is because the degree of field extension $N(n) = [\mathbb{Q}(\sqrt{p_j} \mid 1 \leq j \leq 2^n) : \mathbb{Q}] = 2^{2^n - 1}$ (see for example [13], Lemma 9.20), hence condition 1 above is satisfied. Condition 2 is also satisfied because the coefficients are the roots of the system $G_n = \{Z_j^2 - p_j \mid 1 \leq j < 2^n\}$, with $D(n) = 2$.

For a proof of the theorem, we refer the readers to [11, Theorem 8.1]. We point out that the proof in its original form already works. In his proof, he wanted to conclude that $f_n \notin \mathrm{VNP}$. However, along the way, he arrives at a contradiction to the assertion that f_n is contained in the Zariski-closure of VNP, which is exactly what is now known as $\overline{\mathrm{VNP}}$. During the time of the original proof, the complexity class $\overline{\mathrm{VNP}}$ was not defined.

References

1 Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *Comput. Complex.*, 25(1):217–253, March 2016. doi:10.1007/s00037-015-0114-7.

2 Jarod Alper, Tristram Bogart, and Mauricio Velasco. A lower bound for the determinantal complexity of a hypersurface. *Foundations of Computational Mathematics*, pages 1–8, 2015.

3 Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. In *Proceedings of the 31st Conference on Computational Complexity*, CCC '16, Dagstuhl, DEU, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

4 Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992. doi:10.1137/0221006.

5 D. Bini. Relations between exact and approximate bilinear algorithms. applications. *CALCOLO*, 17(1):87–97, January 1980. doi:10.1007/BF02575865.

6 Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Inf. Process. Lett.*, 8(5):234–235, 1979. doi:10.1016/0020-0190(79)90113-3.

7 Markus Bläser and Christian Ikenmeyer. Introduction to geometric complexity theory, 2018. , version from July 25. URL: http://pcwww.liv.ac.uk/~iken/teaching_sb/summer17/introtogct/gct.pdf.

8 J.A. Bondy and U.S.R Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 2008.

9 Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *J. ACM*, 65(5), August 2018. doi:10.1145/3209663.

10 Peter Bürgisser. Erratum to the complexity of factors of multivariate polynomials. URL: https://www.math.tu-berlin.de/fileadmin/i26_fg-buergisser/erratum_factors.pdf.

11 Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

12 Peter Bürgisser. The complexity of factors of multivariate polynomials. *Found. Comput. Math.*, 4(4):369–396, 2004. doi:10.1007/s10208-002-0059-5.

13 Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 2013.

14 Peter Bürgisser, J.M. Landsberg, Laurent Manivel, and Jerzy Weyman. An overview of mathematical issues arising in the Geometric complexity theory approach to VP v.s. VNP. *SIAM J. Comput.*, 40(4):1179–1209, 2011.

15 Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. A quadratic lower bound for algebraic branching programs. *CoRR*, abs/1911.11793, 2019. To appear in the proceedings of Conference on Computational Complexity, CCC'20. arXiv:1911.11793.

16 Michael Forbes. Some concrete questions on the Border Complexity of polynomials, 2016. Talk at the Workshop on Algebraic Complexity Theory (WACT) 2016 in Tel Aviv. URL: https://www.youtube.com/watch?v=1HMogQIHT6Q.

17 Hervé Fournier, Guillaume Malod, Maud Szusterman, and Sébastien Tavenas. Nonnegative rank measures and monotone algebraic branching programs. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 150 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 15, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019.

18 Fulvio Gesmundo. Geometric aspects of iterated matrix multiplication. *Journal of Algebra*, 461:42–64, 2016.

19 Fulvio Gesmundo, Christian Ikenmeyer, and Greta Panova. Geometric complexity theory and matrix powering. *Differential Geom. Appl.*, 55:106–127, 2017. doi:10.1016/j.difgeo.2017.07.001.

20 Bruno Grenet. An upper bound for the permanent versus determinant problem, 2011. Manuscript. URL: http://www.lirmm.fr/~grenet/publis/Gre11.pdf.

21 Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In *43rd International Colloquium on Automata, Languages, and Programming*, volume 55 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 34, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

22 Joos Heintz and Malte Sieveking. Lower bounds for polynomials with algebraic coefficients. *Theoretical Computer Science*, 11(3):321–330, 1980.

23 Jesko Hüttenhain. *Geometric Complexity Theory and Orbit Closures of Homogeneous Forms*. PhD thesis, Technische Universität Berlin, July 2017.

24 Jesko Hüttenhain and Pierre Lairez. The boundary of the orbit of the 3-by-3 determinant polynomial. *Comptes Rendus Mathematique*, 354(9):931–935, 2016.

25 Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of full rank algebraic branching programs. *ACM Trans. Comput. Theory*, 11(1), November 2018. doi:10.1145/3282427.

26 Mrinal Kumar. On top fan-in vs formal degree for depth-3 arithmetic circuits, 2018. URL: https://eccc.weizmann.ac.il/report/2018/068/.

27 Mrinal Kumar. A quadratic lower bound for homogeneous algebraic branching programs. *computational complexity*, 28(3):409–435, 2019. doi:10.1007/s00037-019-00186-3.

28 J. M. Landsberg. Geometric complexity theory: an introduction for geometers. *ANNALI DELL'UNIVERSITA' DI FERRARA*, 61(1):65–117, May 2015.

29 J. M. Landsberg. *Geometry and Complexity Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2017. doi:10.1017/9781108183192.

30 Meena Mahajan and V. Vinay. Determinant: combinatorics, algorithms, and complexity. *Chicago J. Theoret. Comput. Sci.*, pages Article 5, 26 pp. (electronic), 1997.

31 Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *Journal of Complexity*, 24(1):16–38, 2008.

32 K.D. Mulmuley and M. Sohoni. Geometric Complexity Theory. I. An approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526 (electronic), 2001.

33 K.D. Mulmuley and M. Sohoni. Geometric Complexity Theory. II. Towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.*, 38(3):1175–1206, 2008.

34 D. Mumford. *Algebraic geometry. I: Complex projective varieties*. Classics in mathematics. Springer-Verlag, Berlin, 1995. Reprint of the 1976 edition in Grundlehren der mathematischen Wissenschaften, vol. 221.

35 Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 410–418. ACM, 1991.

36 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):Art. 40, 15, 2013. doi:10.1145/2535928.

37 Claus-Peter Schnorr. Improved lower bounds on the number of multiplications/divisions which are necessary to evaluate polynomials. In *International Symposium on Mathematical Foundations of Computer Science*, pages 135–147. Springer, 1977.

38 Srikanth Srinivasan. Strongly exponential separation between monotone VP and monotone VNP, 2019. arXiv:1903.01630.

39 Volker Strassen. Polynomials with rational coefficients which are hard to compute. *SIAM Journal on Computing*, 3(2):128–149, 1974.

40 Seinosuke Toda. Classes of Arithmetic Circuits Capturing the Complexity of the Determinant. *IEICE TRANS. INF. & SYST.*, E75-D(1):116–124, 1992.

41 L. G. Valiant. Completeness classes in algebra. In *Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing (Atlanta, Ga., 1979)*, pages 249–261. ACM, New York, 1979.

42 Amir Yehudayoff. Separating monotone VP and VNP. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 425–429, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316311.

NP-Hardness of Circuit Minimization for Multi-Output Functions

Rahul Ilango
Massachusetts Institute of Technology, Cambridge, MA, US
rilango@mit.edu

Bruno Loff
University of Porto and INESC-Tec, Portugal
bruno.loff@gmail.com

Igor C. Oliveira
University of Warwick, Coventry, UK
igor.oliveira@warwick.ac.uk

Abstract

Can we design efficient algorithms for finding fast algorithms? This question is captured by various circuit minimization problems, and algorithms for the corresponding tasks have significant practical applications. Following the work of Cook and Levin in the early 1970s, a central question is whether minimizing the circuit size of an explicitly given function is NP-complete. While this is known to hold in restricted models such as DNFs, making progress with respect to more expressive classes of circuits has been elusive.

In this work, we establish the first NP-hardness result for circuit minimization of total functions in the setting of general (unrestricted) Boolean circuits. More precisely, we show that computing the minimum circuit size of a given multi-output Boolean function $f\colon \{0,1\}^n \to \{0,1\}^m$ is NP-hard under many-one polynomial-time randomized reductions. Our argument builds on a simpler NP-hardness proof for the circuit minimization problem for (single-output) Boolean functions under an extended set of generators.

Complementing these results, we investigate the computational hardness of minimizing communication. We establish that several variants of this problem are NP-hard under deterministic reductions. In particular, unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time computable function can approximate the deterministic two-party communication complexity of a partial Boolean function up to a polynomial. This has consequences for the class of structural results that one might hope to show about the communication complexity of partial functions.

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

Keywords and phrases MCSP, circuit minimization, communication complexity, Boolean circuit

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.22

Related Version This paper first appeared as an ECCC report at `https://eccc.weizmann.ac.il/report/2020/021/`.

Funding *Rahul Ilango*: This work was supported in part by an Akamai Presidential Fellowship.
Bruno Loff: The author was the recipient of FCT postdoctoral grant number SFRH/BPD/116010/2016. This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.
Igor C. Oliveira: This work was supported in part by a Royal Society University Research Fellowship.

Acknowledgements Igor C. Oliveira would like to thank Ján Pich and Rahul Santhanam for discussions on the complexity of circuit minimization for partial Boolean functions. Bruno Loff would like to thank Eric Allender for posing a question that inspired some of the results in this work, and the Higher School of Economics for inviting him to the conference "Randomness, Information, Complexity", in honor of Alexander Shen and Nikolay Vereshchagin's 60th birthday, where said question

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 22; pp. 22:1–22:36

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

was asked. Rahul Ilango would like to thank Eric Allender, Marco Carmosino, Russell Impagliazzo, Michael Saks, Rahul Santhanam, and Ryan Williams for their encouragement, suggestions, and helpful discussions.

1 Introduction

The Minimum Circuit Size Problem (MCSP) asks for the size (number of gates) of the smallest Boolean circuit that computes a given Boolean function $f\colon \{0,1\}^m \to \{0,1\}$, where f is represented as a string of length $n = 2^m$. Researchers have investigated this problem and its variants from several angles since the early stages of complexity theory (see [75] for some historical perspective). In particular, over the last two decades there has been a significant interest in understanding the computational hardness of circuit minimization. This is motivated in part by the discovery of connections between this problem and a variety of areas, including complexity theory [43], learning theory [17], cryptography and circuit complexity [70], and proof complexity (see e.g. [51, Part VIII]). In addition, Boolean circuit minimization is of high practical relevance, and a number of textbooks and monographs have been written about heuristics and other applied aspects of this problem (cf. [59, 73, 29]).[1]

Despite considerable efforts to understand the computational complexity of circuit minimization, its NP-hardness status has remained wide open. While there is strong evidence that finding optimal circuits is intractable (see Section 1.3), some researchers have suggested that circuit minimization problems such as MCSP might be NP-intermediate (that is, neither in P nor NP-complete). There is a vast literature on MCSP and this question, and we review the references more directly related to our work in Sections 1.2 and 1.3 below.

1.1 Results

We investigate the natural variant of MCSP where the input function $f\colon \{0,1\}^n \to \{0,1\}^m$ is allowed to have multiple output bits. Our main contribution is a proof that the circuit minimization problem for such multi-output functions is NP-hard with respect to randomized reductions. This is the first NP-hardness result for the circuit minimization of total functions that holds with respect to the class of general (unrestricted) Boolean circuits. Previous NP-hardness results for total functions were known when the computational model is considerably restricted, for instance with respect to DNFs [55, 58] (also known as two-level minimization; see e.g. [77]) and DNFs extended with parity gates at the bottom layer [33].

There are well-known connections between computation and communication (see e.g. [53]). In the second part of this work, we explore the complexity of minimizing communication cost with respect to deterministic protocols, a question that dates back to Yao's seminal work on communication complexity [80, Section 4, Problem E]. Among other contributions, we establish the first NP-hardness result for this model, in the setting that the input communication problem is described by a partial Boolean matrix in $\{0,1,*\}^{n\times n}$. In a remarkable paper, Kushilevitz and Weinreb [54] had previously established the intractability of this problem over total Boolean matrices, but their techniques require cryptographic assumptions. Our proof extends to a stronger hardness of approximation result, and this has interesting consequences for communication complexity.

We now describe in more detail each of our NP-hardness results and their implications.

[1] The problem is also referred to as Boolean function or Boolean algebra minimization, logic synthesis, circuit synthesis, logic minimization, circuit optimization, or multi-level minimization in different communities.

1.1.1 NP-hardness of circuit minimization

First, let us fix some notation and terminology. A Boolean circuit consists of fan-in two AND gates, fan-in two OR gates, and NOT gates. The input gates are labelled by variables $x_1, \ldots, x_n$. We measure the size of a Boolean circuit C, denoted $|C|$, using the number of AND, OR, and NOT gates in the circuit. (While this is the convention adopted here, our techniques are robust to modifications of the circuit size measure and of the gate types in the circuit.)

We now introduce the circuit minimization problems considered in our work.

Multi-output Boolean functions. In practice, one is often interested in computing Boolean functions f that have multiple output bits. Indeed, the vast majority of computations, such as addition, multiplication, encryption schemes, error-correcting codes, solutions to search problems, etc., have multiple outputs. In this case, MCSP can give a misleading picture of the circuit complexity of f. For example, if f is the problem of multiplying two $n \times n$ matrices over $\mathbb{F}_2$, then computing any specific choice of one of the n^2 output bits of f requires a circuit with $\Omega(n)$ gates, which seems to suggest a lower bound of $\Omega(n^3)$ on matrix multiplication. Of course, it is widely-known that one can beat the $O(n^3)$ time algorithm for matrix multiplication quite significantly!

This motivates the study of circuit minimization for multi-output Boolean functions. We begin by fixing our notion of multi-output computation. The *components* of a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ are the single-output functions that compute the ith output bit of f for $i \in [m]$. We say a Boolean circuit C computes a multi-output Boolean function f if for each component f_i of f, there is a gate or input wire in C that computes f_i.

▶ **Definition 1.** Multi-MCSP *is defined as follows:*

- Input. *Positive integers n, m, and s represented in unary, and a (multi-output) Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ represented by a string of length $m \cdot 2^n$.*
- Output. *The input is accepted if and only if there exists a Boolean circuit C of size at most s that computes f.*

Note that Multi-MCSP is in NP,[2] and that this problem is at least as hard as MCSP.

Partial Boolean functions. Despite the fundamental nature of MCSP, in several natural scenarios arising from practical or theoretical considerations one does not really care about the output of a Boolean function on *every* string of length n.[3] For instance, for a problem on graphs, one might be interested only in graphs that are planar. In such situations, it becomes relevant to understand the complexity of the corresponding function on a subset of inputs. This is more naturally captured by a different formulation of MCSP, where irrelevant or inessential inputs of the Boolean function are omitted. In other words, while MCSP refers to total Boolean functions, it is equally natural to consider circuit minimization over partial Boolean functions.

▶ **Definition 2.** Partial-MCSP *is defined as follows:*

- Input. *A positive integer n represented in unary, a collection $\mathcal{P}$ of pairs (x_i, b_i), where $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$, and a positive integer s.*
- Output. *The input is accepted if and only if there exists a Boolean circuit C of size at most s such that $C(x_i) = b_i$ for every pair $(x_i, b_i) \in \mathcal{P}$.*

Note that, like Multi-MCSP, Partial-MCSP is in NP and is at least at hard as MCSP.

[2] If the parameter s is large then the answer is trivial.
[3] Such inputs are associated with "don't care" values in the applied literature.

Boolean functions over an arbitrary set of generators. We also consider circuit minimization for (single-output, total) Boolean functions $f\colon \{0,1\}^m \to \{0,1\}$ under an arbitrary set of *generators*.

To explain, let V be a finite set, called the *ground set*, and $\mathcal{B} = \{B_i\}_{i\in[m]}$ a family of nonempty sets $B_i \subseteq V$ called *generators*, and let $A \subseteq V$. Then we let $D(A \,|\, \mathcal{B})$ denote the minimum number of unions, intersections, and complements that are required to construct A from the sets in $\mathcal{B}$. More precisely, the complement of a set $U \subseteq V$ is defined as $V \setminus U$, and we represent a construction of A from $\mathcal{B}$ as a sequence $B_1, \ldots, B_m, E_1, \ldots, E_\ell$ of sets in V such that $E_\ell = A$ and each set E_j is either the union or intersection of two previously generated sets, or the complement of a previously generated set. We assume for convenience that $D(B \,|\, \mathcal{B}) = 0$ if $B \in \mathcal{B}$. We refer to $D(A \,|\, \mathcal{B})$ as the *discrete complexity*[4] of A with respect to $\mathcal{B}$.

It may then be seen the minimum number of AND, OR and NOT gates needed for a Boolean circuit to compute a Boolean function $f : \{0,1\}^m \to \{0,1\}$ is exactly $D(f^{-1}(1) \mid \mathcal{B})$, where $V = \{0,1\}^m$, and $\mathcal{B} = \{x_1, \ldots, x_m\} \subseteq \{0,1\}^m$ contains the input variables $x_1 \ldots, x_m$, seen as subsets of $\{0,1\}^m$ (i.e.. x_i is the set of strings $w \in \{0,1\}^m$ such that $w_i = 1$). So computing the discrete complexity $D(A \mid \mathcal{B})$, for given A and $\mathcal{B}$, generalizes the task of computing the minimum circuit size, by allowing for the consideration of generator sets $\mathcal{B}$ other than $\{x_1, \ldots, x_m\}$.

Another possibility is to consider circuit complexity over a family $\mathcal{B}$ of generators over a ground set V other than the set of binary strings. For example, the *graph complexity* (see [42] §1.7) of a given bipartite graph $G = (U \times V, E)$, with $E \subseteq U \times V$, is $D(E \mid \mathcal{B})$, where $\mathcal{B}$ contains all product sets $A \times B$ with $A \subseteq U$ and $B \subseteq V$. So computing the discrete complexity $D(A \mid \mathcal{B})$, for given a given A and $\mathcal{B}$, also generalizes the task of computing graph complexity.

In analogy to MCSP, we now introduce the Minimum Discrete Complexity Problem (MDCP).

▶ **Definition 3.** MDCP *is defined as follows:*

- Input. *A positive integer n represented in unary describing the size of the ground set $V = [n]$, a target set $A \subseteq V$, a family $\mathcal{B}$ of nonempty subsets of V, and an integer s.*
- Output. *The input is accepted if and only if $D(A \,|\, \mathcal{B}) \leq s$.*

It is easy to see that MDCP is in NP and that it is more general than MCSP. Our hardness result for MDCP, discussed below, also holds under the assumption that the ground set V is a hypercube $\{0,1\}^m$ and that the collection $\mathcal{B}$ contains the sets generated by $x_1, \ldots, x_m$.

NP-hardness of MDCP, Partial-MCSP, and Multi-MCSP. Note that establishing the hardness of these problems is *necessary* before proving hardness of MCSP. This is because instances of MCSP can be easily converted into instances of each one them.

By adapting techniques from [38], it is not hard to show that MDCP is NP-hard under randomized reductions. Moreover, this result almost immediately implies the NP-hardness of Partial-MCSP, since there is a simple way of converting the circuit minimization of Boolean functions under an arbitrary set of generators into a problem about partial Boolean functions (see Section 3.3). We note that previous works in learning theory [30, 1] implicitly contain a

[4] This general setting was already considered in [68], see also §1.7.2 of Jukna's book [42]. By Stone's representation theorem for Boolean algebras, discrete complexity can be seen as the investigation of circuit complexity with respect to an arbitrary Boolean algebra.

substantially simpler proof that Partial-MCSP is NP-hard, even under *deterministic* reductions. Unfortunately, there is strong evidence that this simpler proof has limitations. For instance, if it could be adapted to show deterministic NP-hardness of MDCP for instances extending the hypercube, then EXP $\neq$ ZPP. This follows from an argument analogous to [62].

Proving the NP-hardness of circuit minimization for total functions seems to require a more sophisticated argument. We are able to combine the technique that we use to show NP-hardness of MDCP and Partial-MCSP with several new ideas to establish the following result.

▶ **Theorem 4.** Multi-MCSP *is* NP*-hard under many-one randomized polynomial time reductions.*

We explain the insights leading to the proof of Theorem 4 in Section 1.2. The final argument is not overly technical, though it took us considerable time to discover the right conceptual ingredients. Could it be the case that MCSP admits a randomized NP-hardness that relies on a clever modification of existing techniques? As far as we know, the existence of a "standard" randomized reduction would not imply a breakthrough such as a complexity class separation.

The hardness results mentioned above come with certain features and consequences that might be of independent interest. We discuss them next.

Search-to-decision reductions for circuit minimization. The formula satisfiability problem (SAT) admits a well-known *search-to-decision* reduction. In other words, if one can easily check if a formula is satisfiable, then it is not much harder to find a satisfiable assignment, whenever one exists. As a consequence of the NP-completeness of SAT, all NP-complete problems must have search-to-decision reductions. On the other hand, designing a search-to-decision reduction for MCSP is open. We refer to [17, 32] for recent developments in this direction which can be interpreted as weak search-to-decision reduction for MCSP.

A corollary of Theorem 4 is that the *search version* of Multi-MCSP and its *decision version* are computationally equivalent under polynomial-time randomized reductions. Inspired by this consequence, we further investigate this phenomenon, and in Section 4.3 we describe a natural *deterministic* search-to-decision reduction for Multi-MCSP. Additionally, we show in Section 3.4 that Partial-MCSP has a simple *deterministic* search-to-decision reduction. These search-to-decision reductions rely on ideas employed in our NP-hardness proofs. This suggests that establishing the NP-hardness of MCSP and obtaining a corresponding search-to-decision reduction might be closely related tasks.

Satisfiability versus Learning. It is known that the appropriate *average-case* formulation[5] of Partial-MCSP captures the complexity of learning general Boolean circuits under the uniform distribution using random examples. This follows by a combination of the ideas in [78] and [15], and for completeness we provide a proof of this equivalence in Appendix A. Consequently, we can base the hardness of learning Boolean circuits on the assumption NP $\not\subseteq$ RP if and only if the existence of an efficient algorithm for average-case Partial-MCSP implies the existence of an efficient (worst-case) algorithm for Partial-MCSP (see Appendix A for details). We refer to [14, 9] for more information about learning algorithms and average-case versus worst-case assumptions.

[5] Here one needs to *distinguish*, for a random choice of polynomially many inputs x_i, whether the labels b_i are randomly generated or are consistent with a fixed circuit of size at most s. We say that an algorithm solves Partial-MCSP on average (for a given choice of the size parameter s) if the distinguishing probability of this experiment is noticeable on every circuit of size at most s.

It seems interesting that the *worst-case* and *average-case* complexities of a natural problem connect to *satisfiability* and *learning*, respectively. Further investigating these relations might be a fruitful research direction.

1.1.2 NP-hardness of communication minimization

Let $f\colon [n] \times [n] \to \{0,1,*\}$ be a partial Boolean function. The two-party communication problem of computing f is defined as follows. Alice is given $x \in [n]$, Bob is given $y \in [n]$, and they are promised that $f(x,y)$ is defined. Their goal is to exchange the minimum number of bits in order to compute $f(x,y)$. We refer to Section 5.1 for definitions, and to [52] for more information about communication complexity in general.

Note that many communication problems of interest are captured by partial Boolean functions, such as Gap-Hamming-Distance (see e.g. [18]) and Unique-Disjointness (cf. [28]).

We are primarily interested in the computational hardness of estimating the communication cost of optimal deterministic protocols for a given function $f\colon [n] \times [n] \to \{0,1,*\}$. This function will be naturally represented by an $n \times n$ matrix $M \in \{0,1,*\}^{n \times n}$, so that $M[x,y] = f(x,y) \in \{0,1\}$ if $f(x,y)$ is defined over the input pair (x,y), and $M[x,y] = *$ otherwise. In order to state our main result in the context of communication complexity, we introduce the Minimum Communication Complexity Problem for partial Boolean functions.

▶ **Definition 5.** Partial-MCCP *is defined as follows:*

- Input. *A positive integer n represented in unary, a matrix $M \in \{0,1,*\}^{n \times n}$ representing a partial Boolean function $f : [n] \times [n] \to \{0,1,*\}$, and a positive integer s.*
- Output. *The input is accepted if and only if there exists a two-party deterministic protocol for computing f whose communication cost is at most s.*

Note that Partial-MCCP is in NP, as the full communication matrix is represented as part of the input, and any non-trivial protocol for f can be described by a string of length polynomial in n.[6]

We prove that computing communication complexity and several related measures is NP-hard under deterministic reductions.

▶ **Theorem 6.** Partial-MCCP *is* NP*-hard under many-one deterministic polynomial time reductions. Furthermore, analogous results hold with respect to leaf complexity, partition number, cover number, and the smallest number of nodes in a* DAG*-like protocol of a partial Boolean function.*

Our hardness results are actually significantly stronger: we show that all these complexity measures are *hard to approximate* in the context of partial Boolean functions. The NP-hardness of approximating communication cost will be discussed in more detail below. The remaining four measures – leaf complexity, partition number, cover number, and the smallest number of nodes in a DAG-like protocol – are NP-hard to approximate up to a factor of $n^{1-\varepsilon}$ (for any fixed $\varepsilon > 0$), which is essentially optimal.

We note that in the setting of NP-hardness results for MCSP with respect to restricted classes of circuits, such as DNF [4] and DNF-XOR [33], a successful strategy has been to first establish hardness of a variant of the problem where the input is the full truth table of a given *partial* function. This is then followed by a reduction to the case of total functions. We leave as an open problem whether Partial-MCCP can be reduced to minimizing communication complexity of total matrices.

[6] In contrast to this definition, note that the NP-hardness result for circuit minimization of partial Boolean functions refers to functions succinctly described by a list of its $\{0,1\}$-valued entries.

Hardness of approximating communication cost and its consequences. Our complexity-theoretic results have implications for the theory of communication complexity. In order to make the discussion more concrete, we first consider an example.

The *log-rank conjecture* of Lovász and Saks [56] states that the deterministic communication complexity of a total Boolean function $f : [n] \times [n] \to \{0,1\}$, denoted $\mathsf{D}(f)$, is characterized up to a polynomial by the logarithm of the rank (over $\mathbb{R}$) of the corresponding communication matrix M_f. It is a basic, well-known fact that $\mathsf{D}(f) \geq \log(\mathrm{rk}\, M_f)$ (cf. [53]). If we do not allow for a super-constant additive error term, the log-rank conjecture says that there is a universal constant $c > 0$ such that $\frac{1}{c}\mathsf{D}(f)^{1/c} - c \leq \log(\mathrm{rk}\, M_f)$ for every total function f.

In the context of our work, the significance of this conjecture is that, if true, it would provide an algorithm (compute the rank and take the logarithm) for approximating the deterministic communication complexity of a given *total* Boolean function. This algorithm runs in time polynomial in the communication matrix, which means that computing such an approximation of $\mathsf{D}(f)$ is not NP-complete, unless $\mathsf{P} = \mathsf{NP}$. While the status of the log-rank conjecture remains unclear,[7] there may be other algebraic or analytic quantities that approximately capture communication cost.

Similar considerations can be made about the communication complexity of *partial* Boolean functions. More generally, we would like it if there were *some* polynomial-time computable function r that would estimate the communication complexity up to a polynomial, in the sense that for some constant $c > 0$ and for every large enough n, any partial function $f\colon [n] \times [n] \to \{0,1,*\}$ satisfies

$$\frac{1}{c} \cdot \mathsf{D}(f)^{1/c} - c \;\leq\; r(M_f) \;\leq\; c \cdot \mathsf{D}(f)^c + c.$$

However, we are able to prove a strong negative result in this direction. We establish that there is no function r as we just described, under the assumption that $\mathsf{P} \neq \mathsf{NP}$. This result is a consequence of the techniques behind the proof of Theorem 6, which also imply certain hardness of approximating results for communication complexity. In more detail, we prove that it is NP-hard to approximate $\mathsf{D}(f)$ up to a sub-exponential function of $\mathsf{D}(f)$. (This result makes sense because $\mathsf{D}(f)$ might be a constant independent of n.) Additionally, we prove that it is NP-hard to estimate $\mathsf{D}(f)$ with an additive error term of $(1 - \Omega(1)) \log n$, or within a fixed but arbitrary constant factor. We refer to Section 5.3 for the precise statements.

1.2 Techniques

1.2.1 Circuit complexity

The proof of Theorem 4 builds on insights from several works on the complexity of circuit minimization, including the references [4], [62] [33], and [38].

In [4], the authors provide a new proof that DNF-MCSP is NP-hard, i.e., the variant of MCSP where the circuit complexity of the input function is measured with respect to DNF size. Their proof employs a deterministic reduction from a set cover problem. More precisely, in the r-Bounded Set Cover Problem (cf. [26]), we are given a collection $\mathcal{S}$ of subsets of $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$, and the goal is to cover $[n]$ with the minimum number of such sets. We are also promised that each set $S \in \mathcal{S}$ has size at most r. The argument of [4] relies on the NP-hardness of solving this problem on certain structured inputs.

[7] In a recent paper, Chattopadhyay et al. [19] (see also [74, 8]) showed that an analogous conjecture for randomized communication complexity is false.

Extending the techniques of [4], a more recent work [33] established that (DNF-XOR)-MCSP is NP-hard under deterministic reductions. Crucial for the argument of [33] to go through is the stronger result proved by [76] showing that r-Bounded Set Cover is NP-hard even to *approximate* (the proof in [76] relies on ideas from [23]). In particular, for any constant-factor approximation parameter α, there exists a parameter r independent of n such that computing an α-approximation of the optimal cover size in r-Bounded Set Cover is NP-hard. Intuitively, this hardness of approximation result provides more flexibility when implementing a reduction from a cover problem to a circuit minimization problem.

Note that the results discussed above rely on the weakness of the circuit classes (DNF and DNF-XOR) to establish the NP-hardness of the corresponding circuit minimization problems. In particular, structural properties of these low-depth circuits are explored in crucial ways. On the other hand, hardly anything is known about the limitations of *unrestricted* Boolean circuits. For instance, while exponential lower bounds are known against DNF-XOR circuits [21], the strongest known explicit lower bounds against general Boolean circuits are of the form cn for a small constant c (cf. [40, 25]). Perhaps this explains in part why many researchers have been pessimistic about the possibility of extending such techniques to show NP-hardness of circuit minimization for more expressive classes of Boolean circuits.

Moreover, some results (see [62], [36], and [38, Appendix B]) strongly suggest that designing *deterministic* reductions for circuit minimization problems that refer to general circuits might be a challenging task. Formally, [62] proved that if MCSP is NP-hard under polynomial-time deterministic many-one reductions, then $\mathsf{EXP} \neq \mathsf{ZPP}$. In other words, it is not possible to design a deterministic reduction showing NP-hardness of MCSP without a breakthrough in complexity theory. While it is not immediately clear to us if this connection applies to problems such as Multi-MCSP, given these results it is more natural to focus on *randomized* reductions.

In sharp contrast to previous works, which have considered the NP-hardness of circuit minimization for *restricted* circuit classes, [38] has recently established the NP-hardness of MCSP for *unrestricted circuits with oracle gates*. In more detail, let MOCSP (Minimum Oracle Circuit Size Problem) be the problem where we are given a parameter s and total functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g_1, \ldots, g_t\colon \{0,1\}^m \to \{0,1\}$, and the goal is to decide if f can be computed by a circuit with at most s AND, OR, NOT, and ORACLE gates, where each ORACLE gate can compute any one of the functions g_i. Perhaps surprisingly, [38] was able to exploit the presence of arbitrary oracle gates to show that MOCSP is NP-hard under randomized reductions.

While computations with oracles might behave very differently than normal computations,[8] this result gave us some optimism, and it was the starting point of our investigation. Our techniques build on the reduction of [38], which relies in part on ideas from [4] and [33].

Similarly to [33] and [38], we will also employ the NP-hardness of approximating r-Bounded Set Cover. Our proofs are technically not very involved, and we focus here on some key conceptual ideas. We refer to Sections 3 and 4 for details.

First, we sketch the proof that MDCP is NP-hard. Building on this argument, we discuss the NP-hardness of Multi-MCSP under many-one polynomial-time randomized reductions (Theorem 4).

[8] For instance, it is well known that there exist oracles A and B such that $\mathsf{P}^A \neq \mathsf{NP}^A$ and $\mathsf{P}^B = \mathsf{NP}^B$, respectively [10].

Hardness of circuit minimization under arbitrary generators (MDCP). We are given a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of sets $S_i \subseteq [n]$, where each set S_i has size at most r. The goal is to compute from $\mathcal{S}$ and $[n]$ an input instance of MDCP whose complexity approximates the cover complexity of $[n]$ with respect to $\mathcal{S}$.

One can view the cover complexity of $[n]$ with respect to $\mathcal{S}$ as measuring the complexity of "generating" the set $[n]$ from the sets in $\mathcal{S}$ using only union operations. In more detail, let $\mathsf{cover}([n], \mathcal{S})$ denote the smallest possible number of sets in $\mathcal{S}$ required to cover $[n]$. It is easy to see that, if $\mathsf{cover}([n], \mathcal{S}) \leq \ell$, then $[n]$ can be generated from the sets in $\mathcal{S}$ using at most ℓ fan-in-two union operations. Similarly, it is not hard to see that if $[n]$ can be generated using ℓ fan-in-two union operations when starting from sets in $\mathcal{S}$, then a trivial upper bound is that $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$. In other words, the minimum number of fan-in-two union operations necessary to generate $[n]$ from sets in $\mathcal{S}$ gives a 2-approximation for $\mathsf{cover}([n], \mathcal{S})$.

The discussion above shows that $\mathsf{cover}([n], \mathcal{S}) = \Theta(D_{\cup}([n] \,|\, \mathcal{S}))$, where $D_{\mathcal{O}}(A \,|\, \mathcal{B})$ denotes the minimum number of $\mathcal{O}$-operations sufficient to generate A from $\mathcal{B}$ when only set operations in $\mathcal{O}$ are allowed. It is not hard to see that intersections are not helpful when generating the entire "ground set" $[n]$. More precisely, one can easily argue that $\mathsf{cover}([n], \mathcal{S}) = \Theta(D_{\{\cup,\cap\}}([n] \,|\, \mathcal{S}))$ by simply replacing intersections by unions. This simple argument and the hardness of approximating set cover can be used to show that, under such a generalization of circuit complexity and when allowing only monotone operations (unions and intersections), it is NP-hard to compute circuit complexity.

We consider next the case of non-monotone operations, which are also present in discrete complexity. Note that when negations (complementations) are allowed, the argument sketched above completely breaks down: by taking the complement of a single set in $\mathcal{S}$, one might be able to cover most of $[n]$. In order to handle this issue, a new ingredient seems necessary. Instead of translating the cover problem given by $([n], \mathcal{S})$ into a direct instance $D([n] \,|\, \mathcal{S})$, we employ a more involved construction based on an idea from [38]. (Intuitively, the construction inoculates the power of negations.) In more detail, we map each element $i \in [n]$ to a block B_i of size n^2 inside the larger ground set $V = [n^3]$. This induces a partition of V into $B_1, \ldots, B_n$. A set $S_j \in \mathcal{S} = \{S_1, \ldots, S_t\}$ is mapped to the union of the blocks B_i with $i \in S_j$. We now consider a certain set $A \subseteq [n^3]$ with nice properties, and use the previously described map to create an instance $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$, where each W_{S_j} is the intersection of A with the union of the sets B_i with $i \in S_j$. (By construction, a cover of $[n]$ by sets in $\mathcal{S}$ provides a way to write any set A as a union of its corresponding sets W_{S_j}.) For a *random set* A (and for this reason we can only get a randomized reduction), we are able to show that with high probability the discrete complexity $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$ *approximates* the cover complexity $\mathsf{cover}([n], \mathcal{S})$. Roughly speaking, this is true because a random set A is so complex that taking negations of the sets W_{S_j} does not really help to considerably reduce its complexity, and the best way to generate A is essentially to use a cover of $[n]$ by sets in $\mathcal{S}$ as a recipe. This allows us to prove that computing discrete complexity is NP-hard under randomized reductions.

Hardness of circuit minimization for multi-output Boolean functions. Next, we try to adapt the NP-hardness result for discrete complexity to Multi-MCSP. Loosely summarizing, the discrete complexity reduction works as follows (continuing with the notation from before):

(1) Randomly convert a set cover problem $([n], \mathcal{S})$ into an essentially equivalent set cover problem $(A, W_{S_1}, \ldots, W_{S_t})$ on a larger ground set in a way that (with high probability) the randomness inoculates against there being way of building A from $W_{S_1}, \ldots, W_{S_t}$ that is significantly better than the naive method of unioning over an optimal set cover.

(2) Compute how hard it is to build A from $W_{S_1}, \ldots, W_{S_t}$ by outputting $D(A \,|\, W_{S_1}, \ldots, W_{S_t})$.

We translate each of these two steps into a Multi-MCSP version separately. For the first step, we translate the sets $(A, W_{S_1}, \ldots, W_{S_t})$ into the truth tables $(T, T_{S_1}, \ldots, T_{S_t})$ corresponding to the sets' characteristic functions, and then replace the notion of building a set from other sets by the notion of computing a function on some input given the values of other functions on that input. In detail, the Multi-MCSP version of Step (1) becomes

(1') Randomly convert a set cover problem $([n], \mathcal{S})$ into an essentially equivalent set cover problem $(A, W_{S_1}, \ldots, W_{S_t})$ on a larger ground set in a way that (w.h.p.) the randomness inoculates against the existence of a circuit C satisfying $T(x) = C(x, T_{S_1}(x), \ldots, T_{S_t}(x))$ that is significantly smaller than the naive method of computing $\bigvee_{S \in \mathcal{S}_0} T_S(x)$ where $\mathcal{S}_0 \subseteq \mathcal{S}$ is an optimal cover.

The main technical challenge for the Multi-MCSP reduction comes from the lack of a simple translation for Step (2). To some degree, this is because there is a "type mismatch" between the problems of computing discrete complexity, where you have a notion of computing "from," and the problem of Multi-MCSP, where there is no notion of computing "from," only a notion of computing "in addition to." Perhaps the closest Multi-MCSP analogue to how hard it is to compute T "from" $T_{S_1}, \ldots, T_{S_t}$ is the quantity

$$\Delta \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet T_{S_1} \bullet \cdots \bullet T_{S_t}) - \mathsf{CC}(T_{S_1} \bullet \cdots \bullet T_{S_t})$$

where the notation $f_1 \bullet \ldots \bullet f_k$ denotes the multi-output function whose components are the functions $f_1, \ldots, f_k$. Informally, the quantity Δ corresponds to how much harder is it to compute T along with $T_{S_1}, \ldots, T_{S_t}$ than it is without T.[9]

However, this quantity does not really compute what we want it to compute. For example, if there is an optimal circuit for computing $T_{S_1} \bullet \cdots \bullet T_{S_t}$ that also computes T at some gate (which might be possible), then $\Delta = 0$. But a solution to a non-trivial set cover problem is never zero! One might hope that we could use randomness again to inoculate against these possibilities, but we have not yet figured out how to do so.

Our key idea for overcoming this barrier is to add additional output functions in order to force $T_{S_1}, \ldots, T_{S_t}$ to be computed in a way such that (with high probability) none of the gates used for computing $T_{S_1}, \ldots, T_{S_t}$ compute T or even help very much in computing T. These new outputs will correspond to the functions we want computed "along the way" to computing $T_{S_1}, \ldots, T_{S_t}$.

The actual implementation of this idea is rather subtle, but here is an, admittedly sketchy, outline. Let D be the circuit that computes $T_{S_1} \bullet \cdots \bullet T_{S_t}$ by just computing each of these functions individually via their naive DNF formula. Our random choice of T can be shown to ensure that none of the functions computed by gates in D "help too much in computing T." Next, define the Evaluation Function induced by D, denoted Eval-D, to be the multi-output function whose components are all those functions which are either computed by a gate in D or an input wire in D. Finally, the Multi-MCSP version of Step (2) will be

(2') Compute how hard it is to compute T at some input from the values of $T_{S_1}, \ldots, T_{S_t}$ at that input by outputting $\Delta' \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet \text{Eval-}D) - \mathsf{CC}(\text{Eval-}D)$.

Since any circuit for computing Eval-D can be converted into a circuit for $T \bullet$ Eval-D using at most t gates (since $T = T_{S_1} \vee \cdots \vee T_{S_t}$), the parameters in our reduction can be set so that the overwhelming number of gates in an optimal circuit for $T \bullet$ Eval-D are functions that are computed in D which we know do not "help too much in computing T." We can

[9] The above definition of Δ brings to mind the chain rule for Kolmogorov complexity $K(x \mid y) = K(xy) - K(y) + O(1)$, so one may think intuitively of Δ as measuring the "complexity", or "entropy", of T given $T_{S_1} \ldots T_{S_t}$.

then show that the quantity Δ' approximates how hard it is to compute T on some input given the values of $T_{S_1}, \ldots, T_{S_t}$ on that input, which in turn, by Step (1'), approximates the size of an optimal cover in $(n, \mathcal{S})$.

1.2.2 Communication complexity

The intractability of computing deterministic communication complexity is known under certain cryptographic assumptions [54]. However, it is unclear how to exploit the techniques in their work to prove a hardness result under a worst-case assumption (see also [54, Remark 4.4]).

While in the context of circuit minimization we have explored reductions from variants of the set cover problem, the proof of Theorem 6 relies on a reduction from graph colorability. The NP-hardness of approximating the chromatic number of a given graph G is now well established (see [57, 31, 24, 82]): it is NP-hard to approximate $\chi(G)$ up to a factor of $n^{1-\varepsilon}$, where n is the number of nodes in G, and $\varepsilon > 0$ is an arbitrary constant.

Our reduction from graph colorability to Partial-MCCP is elementary, and we describe it next. Given a graph $G = ([n], E)$, where $E \subseteq \binom{[n]}{2}$, we construct from it a partial function $f_G : [n] \times [n] \to \{0, 1, *\}$, such that the complexity of f_G under any of the measures considered in Theorem 6 will give us an approximation on $\chi(G)$. The partial function f_G is given by $f_G(i, j) = 1$ if $i = j$, $f_G(i, j) = 0$ if $\{i, j\} \in E$, and $f_G(i, j)$ is undefined otherwise. Note that the matrix $M_G \in \{0, 1, *\}^{n \times n}$ encoding f_G is easily constructed from the input graph G. This completes the description of the communication problem output by the reduction.

For the reader familiar with standard notions from communication complexity, we briefly explain why the communication complexity of f_G (denoted by $\mathsf{D}(f_G)$) provides information about $\chi(G)$. First, it is not hard to show that the 1-cover number of M_G is exactly the chromatic number of G. Using the relation between 1-cover number and deterministic communication complexity, this shows that $\mathsf{D}(f_f) \geq \chi(G)$. On the other hand, it can be shown that there is a deterministic protocol for M_G which has no more than $2 \cdot \chi(G)$ leaves in its protocol tree. Moreover, this protocol is balanced, and this provides a useful upper bound on $\mathsf{D}(f)$. Theorem 6 will then follow from these two claims.

While this reduction and the aforementioned $n^{1-\varepsilon}$-inapproximability results for graph coloring allow us to derive strong hardness of approximation results for communication measures such as leaf complexity and partition number, there is a significant loss with respect to computing $\mathsf{D}(f)$. This happens because in the worst-case $\mathsf{D}(f)$ is only logarithmically related to the other measures. As a consequence, these results are insufficient to establish the consequences described in the second part of Section 1.1.2. To achieve that, we rely on more recent results on the hardness of graph coloring for a different regime of parameters. In more detail, we make crucial use of the works of Huang [37] and Wrochna and Živný [79]. They established that, for any large enough constant k, it is NP-hard to distinguish k-colorable graphs from graphs that are not $g(k)$-colorable, where the function g is *exponential* in k. This translates to new hardness results for approximating $\mathsf{D}(f)$, and we refer to Section 5.3 for more details.

1.3 Further related work

In this section we provide additional pointers to works and research directions related to our results.

Circuit minimization of Boolean functions (MCSP). The circuit minimization problem for single-output Boolean functions with respect to a circuit class $\mathcal{C}$ (denoted by $\mathcal{C}$-MCSP) is known to be NP-complete when $\mathcal{C} \in \{\mathsf{DNF}, \mathsf{DNF\text{-}XOR}\}$. Hardness of DNF-MCSP was first established by Masek [58], with alternate proofs appearing in [22, 4]. This result has also been extended to an almost tight hardness of approximation result for DNF-MCSP (see [48]). The NP-hardness result for (DNF-XOR)-MCSP is more recent [33]. We are not aware of NP-hardness results for $\mathcal{C}$-MCSP for stronger classes. We refer to [69] for more information on circuit minimization for restricted computational models, and for pointers to several related works.

In the case of general Boolean circuits, MCSP is known to be hard for NC^1 (and for slightly stronger classes) under non-uniform AC^0 reductions [64, 27]. Moreover, it has been proved that any function in P can be approximated with noticeable advantage by AC^0 circuits containing a single oracle gate that decides MCSP [64]. Other works have established that every problem in the complexity class SZK (including graph isomorphism) is efficiently reducible to MCSP (see [2, 3]). Interestingly, it has been proved under a cryptographic assumption that a version of MCSP with a large gap between positive and negative instances is NP-intermediate [5].

Several works have shown that establishing the NP-hardness of MCSP with respect to certain classes of reductions would have significant implications to our understanding of algorithms and complexity. For some restricted notions of reduction, hardness results cannot be established even for very simple subclasses of P (see [62]). We refer to [43, 62] and subsequent papers [36, 35, 5, 7] for more information about this line of work. We discussed the influence of these works in our results in Section 1.2.

It is widely known that if solving MCSP is feasible then modern cryptography is insecure [67, 70]. The hardness of MCSP also plays a fundamental role in circuit complexity via the notion of natural proofs [70], and more recently in connection to hardness magnification [63, 60] (see the paragraph on unconditional lower bounds below). As mentioned above, MCSP is closely related to learning algorithms, and we refer to [17] and to Section A for more details.[10] The hardness of MCSP and $\mathcal{C}$-MCSP is also connected to questions in proof complexity (cf. [51, 61]). For several relations between MCSP and complexity theory, we refer to [43].

As mentioned in Section 1.2, it is known that that an extension of MCSP to circuits with oracle gates is NP-hard under randomized reductions [38]. A different formulation of MCSP with oracles has been investigated in [6, 35, 39].

Note that many results discussed above also apply to circuit minimization for partial and multi-output functions, since the corresponding problems generalize MCSP.

Partial Boolean functions (Partial-MCSP) and learning algorithms. The circuit minimization problem for partial Boolean functions with respect to DNF size was shown to be NP-hard by Levin in his seminal work [55].[11] A proof of this NP-hardness result can be found for instance in [77]. Kearns and Valiant [45] showed cryptographic hardness for (Formula)-Partial-MCSP, and a proof that Partial-MCSP is NP-complete under deterministic reductions is implicit in [30] and [1].

[10] Indeed, the opening question in our abstract is also naturally captured by investigations about the power and limitations of learning algorithms. Modern results in complexity theory and learning theory show that circuit minimization and learning are directly related tasks.

[11] This result corresponds to Problem 2 in the English translation of Levin's paper, which can be found in the appendix of [75].

The complexity of Partial-MCSP plays a crucial role in learning theory. More precisely, the search version of $\mathcal{C}$-Partial-MCSP for a concept class $\mathcal{C}$ has long been known to be hard with respect to problems in computational learning theory ([13]; see also [46, Chapter 2]). In other words, an efficient algorithm for the search version of $\mathcal{C}$-Partial-MCSP implies that $\mathcal{C}(\mathsf{poly})$ is PAC learnable in polynomial time (this is often referred to as the "Occam's Razor" principle). This has led to numerous learning algorithms, since the problem is known to be solvable by non-trivial algorithms if $\mathcal{C}$ is simple enough (for example, in the case of decision lists [71]). Other works have established NP-hardness results for slightly more complex classes $\mathcal{C}$, such as decision trees [30][12], or neural networks with a fixed topology [41, 12].

More recently, [78] proved that an efficient algorithm for $\mathcal{C}$-Partial-MCSP also implies PAC learnability. Indeed, he showed that these two tasks are equivalent when one considers a relaxation of worst-case $\mathcal{C}$-Partial-MCSP. (In Section A, we adapt his result to the case of learnability under the uniform distribution.) A certain robust variant of Partial-MCSP is also known to be tightly connected to learnability in the agnostic case (see [50] for more details).

Ko [49] considers the problem MINLT (which roughly corresponds to a variant of Partial-MCSP based on Turing machines rather than circuits) and shows that there exist oracles $\mathcal{O}$ such that $\mathsf{MINLT}^{\mathcal{O}}$ is *not* complete for $\mathsf{NP}^{\mathcal{O}}$ under polynomial-time Turing reductions.

Multi-output Boolean functions (Multi-MCSP) and circuit minimization in practice. There have been quite a few developments on the theoretical aspects of multi-output circuit minimization, and some of these works have had an impact on the practice of circuit minimization. Indeed, chip designers are interested in (and have developed many heuristics for) solving the circuit minimization problem for multi-output (partial) Boolean functions under different input representations. The problem has a long history (see e.g. Karnaugh [44] for two-level minimization and Roth and Karp [72] for multi-level minimization), and we refer to a textbook such as [29] for details.

Regarding theoretical hardness results, Boyar, Matthews, and Peralta [16] show that the multi-output minimization problem for computing linear forms (computing Ax where A is a fixed matrix and x is the input vector) in the restricted model of "linear straight-line programs" (where operations consist of taking linear combinations of inputs) is NP-hard.

Unconditional complexity lower bounds for MDCP, Partial-MCSP, and Multi-MCSP. Results from the emerging area of hardness magnification (see e.g. [63, 60]) show that weak unconditional lower bounds for MCSP and related problems against a variety of computational models can be magnified to complexity separations as strong as $\mathsf{P} \neq \mathsf{NP}$. It is worth noting that Partial-MCSP has played a crucial role in the proof of earlier results in this area, both in circuit complexity [65, Theorem 1] and in proof complexity [61, Proposition 4.14]. Motivated in part by hardness magnification, it is now known that most combinatorial circuit lower bounds established in complexity theory can be shown to hold for MCSP as well (see [27, 20] and references therein). All these results immediately imply state-of-the-art lower bounds for MDCP, Partial-MCSP, and Multi-MCSP, since the instances of MCSP easily embed into these problems.

[12] Note that this NP-hardness result for decision trees is for *partial* functions, represented by a list as in Definition 2. For total functions, the problem is solvable in polynomial time by a simple dynamic programming algorithm.

Hardness of estimating communication complexity. It has long been known that computing the non-deterministic communication complexity of a given total function is NP-hard [66]. We refer to a subsequent work [57] for an inapproximability result.

In the setting of deterministic communication complexity, which is the main focus of our results, the following was known. In [54], Kushilevitz and Weinreb proved under cryptographic assumptions that one cannot efficiently compute the communication complexity of a given total two-player function $f : [n] \times [n] \to \{0, 1\}$. In more detail, if one assumes that there are pseudorandom function generators in NC_1 which fool polynomial size distinguishers, then it is hard to estimate communication complexity with an approximation ratio of ≈ 1.1. On the other hand, if one assumes that there are pseudorandom function generators in NC secure against distinguishers of sub-exponential size, then the hardness result is improved to an approximation ratio of order $n^{1/2}$.

To our knowledge, previously to our work no result was known on the hardness of estimating deterministic communication complexity under *worst-case* assumptions.

2 Preliminaries

We let $[n]$ denote the set $\{1, \ldots, n\}$.

We will use the NP-hardness of approximating Set Cover with respect to sets of bounded size [76]. A weaker version of the result from [76] is sufficient.

We say that sets $S_1, \ldots, S_\ell$ *cover* a set T if $T \subseteq S_1 \cup \cdots \cup S_\ell$. For a collection of sets $\mathcal{S}$ and a set T, we use $\mathsf{cover}(T, \mathcal{S})$ to denote the minimum number of sets in $\mathcal{S}$ necessary to cover T.

▶ **Definition 7** (r-Bounded Set Cover Problem)**.** *For a positive integer r, the r-Bounded Set Cover Problem is defined as follows:*

- Input. *A positive integer n represented in unary, and a collection $\mathcal{S}$ of nonempty subsets of $[n]$. We are promised that $\bigcup_{S \in \mathcal{S}} S = [n]$ and that $|S| \leq r$ for each $S \in \mathcal{S}$.*
- Output. *The value* $\mathsf{cover}([n], \mathcal{S})$.

For convenience, we defined the r-Bounded Set Cover Problem as an optimization problem instead of decision problem.

▶ **Theorem 8** (Hardness of approximating r-Bounded Set Cover [76])**.** *For every constant $\alpha \geq 1$ there exists $r \in \mathbb{N}$ such that approximating the r-Bounded Set Cover Problem within a factor of α is NP-hard. More precisely, for every $L \in \mathsf{NP}$, there exists a polynomial-time algorithm that, on input x, outputs a parameter k and an instance $(1^m, \mathcal{S})$ of the r-Bounded Set Cover Problem such that if $x \in L$ then $\mathsf{cover}([m], \mathcal{S}) \leq k$, and if $x \notin L$ then $\mathsf{cover}([m], \mathcal{S}) > \alpha \cdot k$.*

3 Warm-up: NP-hardness for arbitrary generators and partial functions

We establish in this section the NP-hardness of MDCP, and as an easy corollary, provide a self-contained proof of the NP-hardness of Partial-MCSP. The argument consists of two steps: a randomized (approximate) reduction from r-Bounded Set Cover to MDCP, and a deterministic reduction from MDCP to Partial-MCSP.

3.1 Notation

Recall that we consider a generalization of Boolean circuit complexity originally proposed and investigated in a particular context by [68]. Let V be a finite set (also referred to as the *ground set*). Given a family $\mathcal{B} = \{B_i\}_{i \in [m]}$ of nonempty sets $B_i \subseteq V$ (also referred to

as the family of *generators*) and a set $A \subseteq V$, we let $D(A \mid \mathcal{B})$ denote the minimum number of unions, intersections, and complements that are required to construct A from the sets in $\mathcal{B}$. More precisely, the complement of a set $U \subseteq V$ is defined as $V \setminus U$, and we represent a construction of A from $\mathcal{B}$ as a sequence $B_1, \ldots, B_m, E_1, \ldots, E_\ell$ of sets in V such that $E_\ell = A$ and each set E_j is either the union or intersection of two previously generated sets, or the complement of a previously generated set. We assume for convenience that $D(B \mid \mathcal{B}) = 0$ if $B \in \mathcal{B}$. We refer to $D(A \mid \mathcal{B})$ as the *discrete complexity* of A with respect to $\mathcal{B}$.

3.2 A reduction from r-Bounded Set Cover to MDCP

We will use instances of MDCP with a particular structure. This makes the second reduction from MDCP to Partial-MCSP more transparent.

Let V be a ground set, and $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of nonempty subsets of V. For an element $v \in V$, we use $v^\uparrow \in \{0,1\}^n$ (the "lifted" version of v) to denote the string with the property that $v^\uparrow_i = 1$ if and only if $v \in B_i$. We say that a collection $\mathcal{B}$ is *complete* (with respect to V) if the following holds: if $a, b \in V$ and $a^\uparrow = b^\uparrow$ then $a = b$. In other words, distinct elements of V have different liftings with respect to $\mathcal{B}$. Our reduction from r-Bounded Set Cover to MDCP will always produce a family of generators that is complete.

Let r be a large enough constant, so that say 10-approximating r-Bounded Set Cover is NP-hard. Given an instance $(1^n, \mathcal{S})$ of this problem, the reduction proceed as follows. Fix $V \stackrel{\text{def}}{=} [n^3]$. Partition V into n blocks $V_1, \ldots, V_n$, where $|V_i| = n^2$ for each $i \in [n]$. Given a set $A \subseteq V$, we let $A_i \stackrel{\text{def}}{=} A \cap V_i$. Now view V as the set $\{0,1\}^{3 \log n}$, and for each $j \in [3 \log n]$, let $B_j = \{v \in V \mid v_j = 1\}$. For convenience, we let $\mathcal{F} \stackrel{\text{def}}{=} \{B_1, \ldots, B_{3 \log n}\}$. Note that any family $\mathcal{B}$ of generators that contains $\mathcal{F}$ is complete with respect to V.

Let $\mathcal{A}^{-i} \stackrel{\text{def}}{=} \{A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n\}$. We say that a set A is *critical* if, for every $i \in [n]$,

$$D(A \mid \mathcal{F} \cup \mathcal{A}^{-i}) \;>\; n \cdot \log n.$$

It is not hard to show that a uniformly random set $\boldsymbol{A} \subseteq_{1/2} V$ is typically critical.

▶ **Lemma 9.** *Let $\boldsymbol{A} \subseteq_{1/2} V$ be sampled by letting $v \in \boldsymbol{A}$ independently with probability $1/2$ for each $v \in V$. Then,*

$$\Pr_{\boldsymbol{A}}[\boldsymbol{A} \text{ is critical}] \to 1 \;\text{ as }\; n \to \infty.$$

Proof. For a fixed $i \in [n]$, we argue below that

$$\Pr_{\boldsymbol{A}}[D(\boldsymbol{A} \mid \mathcal{F} \cup \boldsymbol{\mathcal{A}}^{-i}) \leq n \cdot \log n] \;=\; o(1/n).$$

The lemma follows by a union bound.

Note that, conditioning on the choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$, the set $\boldsymbol{A}_i$ is still a uniformly distributed subset of V_i. Moreover, *after we fix* $\mathcal{F}$ and $\boldsymbol{\mathcal{A}}^{-i}$, any construction of a set $E \subseteq V$ from $\mathcal{F} \cup \boldsymbol{\mathcal{A}}^{-i}$ using s operations can be described by a binary string of length at most $O(s \cdot (\log s + \log n))$. Since $|V_i| = n^2$, the probability that the discrete complexity of $\boldsymbol{A}$ (conditioned on the choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$) given $\mathcal{F} \cup \boldsymbol{A}^{-i}$ falls below $s = n \cdot \log n$ is at most

$$\frac{2^{O(s \cdot (\log s + \log n))}}{2^{n^2}} \;=\; o(1/n).$$

Since this holds for any choice of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{i-1}, \boldsymbol{A}_{i+1}, \ldots, \boldsymbol{A}_n$, the desired probability upper bound holds. ◀

We assume from now on that we have efficiently produced a critical set A. (Our randomized reduction will always be correct on a critical set.) Note that so far we have only inspected the input 1^n from $(1^n, \mathcal{S})$. Our approximate reduction outputs a triple $(1^{n^3}, A, \mathcal{B}_\mathcal{S})$, where A is generated as above, and whose family $\mathcal{B}_\mathcal{S}$ is defined as follows. For each set $S \in \mathcal{S}$, let $W_S \stackrel{\text{def}}{=} \bigcup_{i \in S} A_i$. Now set

$$\mathcal{B}_\mathcal{S} \stackrel{\text{def}}{=} \mathcal{F} \cup \{W_S \mid S \in \mathcal{S}\}.$$

Clearly, the triple $(1^{n^3}, A, \mathcal{B}_\mathcal{S})$ can be efficiently computed from $(1^n, \mathcal{S})$.

We argue next that computing $D(A \,|\, \mathcal{B}_\mathcal{S})$ allows us to 2-approximate $\mathsf{cover}([n], \mathcal{S})$.

▶ **Lemma 10.** *If* $\mathsf{cover}([n], \mathcal{S}) = \ell$ *then* $D(A \,|\, \mathcal{B}_\mathcal{S}) \leq \ell$.

Proof. Let $S_1, \ldots, S_\ell$ be a cover of $[n]$ using sets from $\mathcal{S}$. Then, by construction of the sets W_S, we get that $A = W_{S_1} \cup \cdots \cup W_{S_\ell}$. In particular, it is possible to construct A using at most ℓ operations (unions) starting from the sets in $\mathcal{B}_\mathcal{S}$. ◀

▶ **Lemma 11.** *If* $D(A \,|\, \mathcal{B}_\mathcal{S}) = \ell$ *then* $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$.

Proof. Let $t \stackrel{\text{def}}{=} |\mathcal{S}|$, and suppose that $E_1, \ldots, E_\ell$ represent a construction of $A = E_\ell$ from sets in $\mathcal{B}_S$. For convenience, we write $B_\mathcal{S} = \mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_t}\}$. In order to prove Lemma 11, we need the following simple claim.

▷ Claim 12. $E_1, \ldots, E_\ell$ depend on at most 2ℓ sets from $\{W_{S_1}, \ldots, W_{S_t}\}$.

This claim holds simply because each set E_j depends on at most 2 other sets. One may replace 2ℓ by $\ell + 1$ by arguing more carefully, that any ℓ-node directed acyclic graph with fan-in 2 and a single sink node has at most $\ell + 1$ source nodes. But this tighter bound is unnecessary for our purpose.

Continuing with the proof of Lemma 11, and relabelling some indexes if necessary, we assume for convenience of notation that the construction of A from $\mathcal{B}_\mathcal{S}$ depends only on $\mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_{2\ell}}\}$. In other words,

$$D(A \,|\, \mathcal{F} \cup \{W_{S_1}, \ldots, W_{S_{2\ell}}\}) \leq \ell.$$

Since $W_{S_j} = \bigcup_{i \in S_j} A_i$ and $|S_j| \leq r$, we get that $D(W_{S_j} \,|\, \{A_i\}_{i \in S_j}) \leq r$. In turn, by composing constructions, we obtain that

$$D(A \,|\, \mathcal{F} \cup \{A_i\}_{i \in S_1} \cup \cdots \cup \{A_i\}_{i \in S_{2\ell}}) \;\leq\; \ell \cdot r.$$

We can assume that $r < \log n$ for a large enough n. In addition, notice that if $\ell > n$ then the statement of Lemma 11 is trivial, since by assumption $\mathcal{S}$ always covers $[n]$ and as a consequence $\mathsf{cover}([n], \mathcal{S}) \leq n$. But from $r < \log n$ and $\ell \leq n$ it follows that the expression above can be upper bounded by $n \cdot \log n$. Given that A is critical, it must be the case that the sets $S_1, \ldots, S_{2\ell}$ cover $[n]$. In other words, $\mathsf{cover}([n], \mathcal{S}) \leq 2\ell$, which completes the proof. ◀

Consequently, it immediately follows from Lemmas 9, 10, and 11 and from Theorem 8 that MDCP is NP-hard to compute under polynomial-time randomized reductions.

3.3 A reduction from MDCP to Partial-MCSP

Let $(1^n, A, \mathcal{B}, s)$ be an input to MDCP. We can assume from the properties of the previous reduction that $\mathcal{B}$ is complete. We will also assume without loss of generality that A is nonempty. We create an instance of Partial-MCSP as follows.

For every $v \in V$, where $V = [n]$, we consider the corresponding lifted vector $v^\uparrow \in \{0,1\}^k$ described above, where $\mathcal{B} = \{B_1, \ldots, B_k\}$ and each $B_i \subseteq V$ is nonempty. We let

$$V^\uparrow \stackrel{\text{def}}{=} \{v^\uparrow \mid v \in V\} \subseteq \{0,1\}^k \text{ and } A^\uparrow \stackrel{\text{def}}{=} \{a^\uparrow \mid a \in A\} \subseteq V^\uparrow.$$

Consider the partial Boolean function $f_A \colon V^\uparrow \to \{0,1\}$ defined by $f_A(x) = 1$ if and only if $x \in A^\uparrow$. The reduction outputs the tuple $(1^n, \mathcal{P}, s)$, where

$$\mathcal{P} \stackrel{\text{def}}{=} \{(x, f_A(x)) \mid x \in V^\uparrow\}.$$

It is easy to see that this tuple can be efficiently computed from the input $(1^n, A, \mathcal{B})$. In order to establish the correctness of this reduction, it suffices to prove the following lemma.

▶ **Lemma 13.** *The partial Boolean function $f_A \colon V^\uparrow \to \{0,1\}$ agrees with a Boolean circuit of size at most s if and only if $D(A \mid \mathcal{B}) \leq s$.*

Proof. The lemma is intuitively clear, since the lifting operation $\uparrow$ induces a *bijection* between $V = [n]$ and $V^\uparrow \subseteq \{0,1\}^k$. (This is the case because by assumption $\mathcal{B}$ is complete.) For completeness, we provide more details below.

Let $A \subseteq V$ be an arbitrary nonempty set, and assume that $\mathcal{B}$ is complete. If the circuit size of f_A is 0, then it must be the case that this circuit is simply x_i for some $i \in [k]$. But then $v \in A$ iff $f_A(v^\uparrow) = 1$ iff $v_i^\uparrow = 1$ iff $v \in B_i$. Equivalently, $A = B_i$. By convention, we have $D(B_i \mid \mathcal{B}) = 0$. The other direction is analogous. This establishes the base case corresponding to $s = 0$.

Suppose now that C is a Boolean circuit of size $s > 0$ that agrees with f_A over $V^\uparrow$. Replace each input variable x_i of C by the set $B_i \in \mathcal{B}$, and each Boolean operation AND, OR, and NOT in C by the corresponding set operation $\cap$, $\cup$, and complementation. We claim that this induces a construction of A from $\mathcal{B}$.

In order to see this, fix any element $v \in V$. More generally, we claim that the i-th gate of C outputs 1 on $v^\uparrow$ if and only if the i-th set E_i constructed under this transformation contains the element v. For the input gates, this follows from the discussion above. To prove this for the i-th gate g_i, assume that the result holds for $x_1, \ldots, x_k, g_1, \ldots, g_{i-1}$ (viewed as subsets of $V^\uparrow$), and consider the corresponding construction $B_1, \ldots, B_k, E_1, \ldots, E_k$ (these are subsets of V) induced by the transformation. Then it easily follows from the induction hypothesis that $g_i(v^\uparrow) = 1$ if and only if $v \in E_i$, regardless of the Boolean operation performed at g_i over the preceding gates. For instance, if $g_i = g_{i_1}$ AND g_{i_2} for $i_1, i_2 < i$, then $g_i(v^\uparrow) = 1$ iff $g_{i_1}(v^\uparrow) = 1$ and $g_{i_2}(v^\uparrow) = 1$ iff $v \in E_{i_1}$ and $v \in E_{i_2}$ iff $v \in E_{i_1} \cap E_{i_2} = E_i$.

Obtaining an upper bound on the circuit complexity of f_A from an upper bound on $D(A \mid \mathcal{B})$ can be done using the reverse transformation. This completes the proof of Lemma 13. ◀

Composing the reductions above completes the proof of NP-hardness of Partial-MCSP.

3.4 Search-to-decision reduction for Partial-MCSP

Recall that in Search-Partial-MCSP we are given $(1^n, \mathcal{P})$, where $\mathcal{P} = \{(x_i, b_i)\}_{i \in [t]}$ for some $t \in \mathbb{N}$, $x_i \in \{0,1\}^n$, and $b_i \in \{0,1\}$. The goal is to output a circuit C of minimum size that is consistent with $\mathcal{P}$. In this section, we show that this problem can be solved in deterministic

polynomial-time using an oracle $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ that solves Partial-MCSP. We assume that $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ outputs the optimal circuit size $s \in \mathbb{N}$ (a simple binary search suffices to obtain this value using oracle calls to Partial-MCSP).

We describe a recursive procedure B with access to $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}$ that solves Search-Partial-MCSP. The input to B is of the form $(1^\ell, \mathcal{Q})$, where $\mathcal{Q}$ is a collection of input pairs in $\{0,1\}^\ell \times \{0,1\}$. The initial call to B that solves Search-Partial-MCSP will be of the form $B(1^n, \mathcal{P})$. For convenience, we rely on a polynomial-time sub-routine $\mathsf{Consistent}(\mathcal{Q}, C)$ that returns true if and only if circuit C over input variables $y_1, \ldots, y_\ell$ is consistent with $\mathcal{Q}$.

Algorithm B.

Input. A pair $(1^\ell, \mathcal{Q})$.

Output. A minimum size circuit C over $y_1, \ldots, y_\ell$ that is consistent with $\mathcal{Q}$.

1. If $\mathsf{Consistent}(\mathcal{Q}, y_i)$ holds for some $i \in [\ell]$, return the circuit represented by input variable y_i.
2. Otherwise, for each operation $\star \in \{\vee, \wedge, \neg\}$, and for each appropriate choice of one or two operands from $\{y_1, \ldots, y_\ell\}$:
   - 2.1 Let $\mathcal{Q}^+$ extend each pair $(y, b) \in \mathcal{Q}$ to a pair $(y^+, b) \in \{0,1\}^{\ell+1} \times \{0,1\}$, where the new coordinate corresponds to the result of the operation. Moreover, let $D(y_1, \ldots, y_\ell)$ be a depth-1 circuit of size 1 corresponding to the same operation.
   - 2.2 If $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^{\ell+1}, \mathcal{Q}^+) < A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q})$, invoke $B(1^{\ell+1}, \mathcal{Q}^+)$. Let $C^+(y_1, \ldots, y_{\ell+1})$ be the circuit returned by this call. Return the description of $C(y_1, \ldots, y_\ell) \stackrel{\text{def}}{=} C^+(y_1, \ldots, y_\ell, D(y_1, \ldots, y_\ell))$.

We sketch next the proof that $B(1^\ell, \mathcal{Q})$ runs in polynomial time, and that it always returns a consistent circuit of minimum size. Let $s = A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q})$. The proof of correctness is by induction on s, i.e., the induction hypothesis is that the algorithm is correct on every input pair $(1^\ell, \mathcal{Q})$ whose circuit complexity is at most s.

If $s = 0$, then an input variable y_i must be consistent with $\mathcal{Q}$. In this case, algorithm B correctly returns such a circuit in step (1) above. Assume now that $s \geq 1$ and that the induction hypothesis holds for any input whose complexity is at most $s-1$. For the induction step, let $(1^\ell, \mathcal{Q})$ be an input to Search-Partial-MCSP for which $A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q}) = s$. Since $s > 0$, using any bottom layer gate in any optimal circuit for Q, it follows that there is at least one Boolean operation whose corresponding set Q^+ defined in step (2.1) will pass the test performed in step (2.2). Now consider any recursive call in step (2.2), which might not necessarily come from a bottom gate in an optimal circuit for $\mathcal{Q}$. By the induction hypothesis, a circuit C^+ consistent with the corresponding collection Q^+ and of size at most $s-1$ is returned. Clearly, the resulting circuit C obtained from C^+ and from the corresponding circuit D is consistent with Q and has size at most s. This completes the induction step, and the proof of correctness of B.

For the running time, note that on every instance $(1^\ell, \mathcal{Q})$ of B we have $s = A_{\mathsf{Partial}\text{-}\mathsf{MCSP}}(1^\ell, \mathcal{Q}) \leq O(\ell \cdot |\mathcal{Q}|)$. Consequently, at most s nested recursive calls are made. In each call, $\mathsf{Consistent}(\mathcal{Q}, y_i)$ can be computed in time $O(|\mathcal{Q}| \cdot (\ell + s))$. We also consider in the worst case all possible operations over a set of at most $\ell + s$ input coordinates, and there are at most $O(\ell + s)^2$ such operations. Consequently, B runs in time at most $O(s \cdot (|\mathcal{Q}| \cdot (\ell + s) + (\ell + s)^2)) = O(\ell \cdot |\mathcal{Q}|)^3$.

4 Main result: NP-hardness of circuit minimization for multi-output functions

4.1 Definitions

4.1.1 Multi-output Functions, Concatenations, and Truth Tables

For a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, the *components* of f are the single-output functions $f_1, \ldots, f_m$ where $f_i : \{0,1\}^n \to \{0,1\}$ is defined so $f_i(x)$ equals the ith output bit of $f(x)$.

For a multi-output Boolean function f, we let the *circuit complexity of* f, denoted $\mathsf{CC}(f)$, be the minimum size of any circuit computing f.

For strings $x, y \in \{0,1\}^\star$, we let $x \bullet y$ denote the concatenated string. We identify a multi-output Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ with the concatenated string $T_1 \bullet \cdots \bullet T_m \in \{0,1\}^{m \cdot 2^n}$ where $T_1, \cdots, T_m$ are the truth tables of the components $f_1, \ldots, f_m$ respectively.

If $f : \{0,1\}^n \to \{0,1\}^{m_1}$ and $g : \{0,1\}^n \to \{0,1\}^{m_2}$ are Boolean functions with the same number of inputs, we define the concatenated Boolean function $f \bullet g : \{0,1\}^n \to \{0,1\}^{m_1+m_2}$ given by $f \bullet g(x) = f(x) \bullet g(x)$.

We also use the $\bullet$ symbol to indicate concatenation in a similar way that $\sum$ acts for addition. For example, if $T_1, \ldots, T_m$ are truth tables of functions with the same number of inputs, then we use the notation $\bullet_{i \in [m]} T_i$ to indicate $T_1 \bullet \cdots \bullet T_m$.

4.1.2 The Evaluation Function and Multi-output Computation

Each circuit C induces a multi-output Boolean function we call the *Evaluation Function of* C, denoted $\mathsf{Eval}\text{-}C$, that computes the outputs of each of the gates in C. In more detail, for a circuit C that takes n inputs and has s gates, the evaluation function induced by C, denoted $\mathsf{Eval}\text{-}C : \{0,1\}^n \to \{0,1\}^{s+n}$, is given by $x_1 \bullet \cdots \bullet x_n \bullet g_1 \bullet \cdots \bullet g_s$ where x_i is the function computed by the ith input wire of C and g_j is the function computed by the jth gate in C (for this to be well-defined, we need to fix an ordering of the gates of C, but, for our purposes, any ordering will do).

Using the Evaluation Function, an equivalent definition of multi-output circuit computation to the one given in the introduction is that a Boolean circuit C computes a (multi-output) Boolean function f if and only if every component of f is a component of $\mathsf{Eval}\text{-}C$.

4.1.3 Windows of Truth Tables

Given a truth table T of length n and a subset $S \subseteq [n]$, we define the *S-window of T*, denoted $T_{\langle S \rangle}$, to be the truth table of length n that (informally) "sees" T on the elements of S and zeroes everywhere else. Rigorously,

$$T_{\langle S \rangle}(x) = \begin{cases} T(x) & \text{if } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

4.1.4 Canonical DNF Circuits

For each (single output) Boolean function $f : \{0,1\}^n \to \{0,1\}$, it will be useful to fix an algorithm that outputs a single "canonical" Boolean circuit for computing f.

For our purposes, many algorithms are possible, but, for concreteness, we will define the canonical circuit of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ to be the naive DNF formula for f, denoted $\mathbf{DNF}_f$, given by

$$\mathbf{DNF}_f \stackrel{\text{def}}{=} ((x_1 = y_1^1) \wedge \cdots \wedge (x_n = y_n^1)) \vee \cdots \vee ((x_1 = y_1^t) \wedge \cdots \wedge (x_n = y_n^t))$$

where

- $y^1, \ldots, y^t$ are the YES inputs of f in lexicographical order,
- $x_1, \ldots, x_n$ index the bits of the input string x,
- $y_1^j, \ldots, y_n^j$ index the bits of y^j for each $j \in [t]$,
- and for $i \in [n]$ and $j \in [t]$, $(x_i = y_i^j)$ is syntax for $\begin{cases} x_i & \text{if } y_i^j = 1, \\ \neg x_i & \text{if } y_i^j = 0. \end{cases}$

It is easy to see that $\mathbf{DNF}_f$ can be computed in polynomial-time given the truth table of f.

Moreover, reading the above definition of $\mathbf{DNF}_f$ from left to right gives a natural way of defining the kth gate in $\mathbf{DNF}_f$, whereby the kth gate corresponds to the kth gate symbol appearing in the above formula. This fact will later be useful in our analysis.

4.1.5 Lifting Sets

Our reduction will use a way to lift subsets into subsets on larger ground sets. To do this, we will first define a canonical partition of $[m]$ into n parts for $m \geq n$. Let $\mathcal{P}^{m,n} = (P_1^{m,n}, \ldots, P_n^{m,n})$ be the partition of $[m]$ into n parts given by

$$P_i^{m,n} = \{j \in [m] : j \equiv i \mod n\}.$$

From this partition, we can now lift subsets as follows. Let $n \leq m \in \mathbb{N}$. Let $S \subseteq [n]$. The *m-lift* of S, denoted S^m, is the set given by

$$S^m \stackrel{\text{def}}{=} \bigcup_{i \in S} P_i^{m,n}.$$

4.2 A reduction from r-Bounded Set Cover to Multi-MCSP

In order to show that Multi-MCSP is NP-hard, we give a probabalistic polynomial-time many-one reduction with one-sided error from a constant approximation of r-Bounded Set Cover to Multi-MCSP.

In fact, for convenience, our reduction will be from the optimization version of approximating r-Bounded Set Cover to the optimization version of Multi-MCSP (computing CC), but it will be easy to see that our reduction can be converted into the desired reduction for the corresponding decision problems.

Let r be a large enough constant, so that say 10-approximating r-Bounded Set Cover is NP-hard. Given an instance $(1^n, \mathcal{S})$ of this problem, the reduction proceeds as follows. Let $m = O(n^3)$ be the least power of two greater than n^3. Let T be a uniformly random truth table of length m. I.e., T is a binary string in $\{0,1\}^m$, representing a function from $\{0,1\}^{\log m}$ to $\{0,1\}$. Compute the truth table of

$$g \stackrel{\text{def}}{=} \bigodot_{S \in \mathcal{S}} \textsf{Eval-}\mathbf{DNF}_{T_{\langle S^m \rangle}}.$$

Let k be the number of distinct components of g that are not functions computed by an input gate, that is,

$$k \stackrel{\text{def}}{=} |\{g_i : g_i \text{ is a component of } g \text{ and } g_i \neq x_j \text{ for all } j \in [\log m]\}|.$$

The reduction then outputs [13]

$$\Delta \stackrel{\text{def}}{=} \mathsf{CC}(T \bullet g) - k.$$

First, we argue that this procedure runs in polynomial time. The only two steps that may raise concern are whether we can compute the truth table of g efficiently and whether we can compute k efficiently.

To show that the truth table of g can be computed efficiently, it suffices to show that for each $S \in \mathcal{S}$, the truth table of $\mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$ can be computed in time polynomial in n. Computing $T_{\langle S^m \rangle}$ can be done in time $O(m + |T|) = O(m)$ and outputs a truth table of length m. The canonical DNF of the truth table $T_{\langle S^m \rangle}$ can be computed in time polynomial in $|T_{\langle S^m \rangle}| = m$, and the resulting DNF has $\log m$ inputs and size at most $O(m \log m)$. Finally, computing the Evaluation Function of a circuit with $\log m$ inputs and $O(m \log m)$ gates can be done in time $O(m^3)$ by just evaluating the circuit on every input. Hence, putting these all together, computing the truth table of $\mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$ can be done in time polynomial in $m = O(n^3)$.

To see that we can compute k efficiently, realize that we have already computed the full truth table of g and that removing any components computed by one of the $\log m$ input wires along with any duplicate components takes time at most quadratic in the length of the truth table of g.

Now, we will argue for the correctness of the reduction. We will show that

$$\mathsf{cover}([n], \mathcal{S})/4 - 4 \underset{\substack{\text{(with high probability} \\ \text{using Lemma 16)}}}{\leq} \Delta \underset{\substack{\text{(unconditionally} \\ \text{using Lemma 15)}}}{\leq} \mathsf{cover}([n], \mathcal{S}),$$

and thus, with high probability Δ computes a 10-approximation of r-Bounded Set Cover when n is sufficiently large.[14] Moreover, this computation has one-sided error since the upper bound holds unconditionally.

Thus, after proving Lemmas 15 and 16, we will have shown that there is a randomized polynomial-time many-one reduction with one-sided error from 10-approximating r-Bounded Set Cover to Multi-MCSP.

Before proving Lemmas 15 and 16, we make the following observation about computing g.

▶ **Proposition 14.** *If a circuit C computes g, then there are k distinct gates in C that compute components of g. Moreover, $\mathsf{CC}(g) = k$.*

Proof. We begin by proving the first statement, which also implies the lower bound $\mathsf{CC}(g) \geq k$. Suppose C is a circuit that computes g. Then every distinct component of g has a (necessarily distinct) input wire or gate from C that computes that component. Therefore, since g has k distinct components that are not computed by an input wire, C must have at least k distinct gates computing components of g.

Next, we sketch the proof of the upper bound $\mathsf{CC}(g) \leq k$. Let C be the circuit built as follows. For each $S \in \mathcal{S}$, iterate through the gates g in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ in topological order. Let $\diamond \in \{\wedge, \vee, \neg\}$ be the gate type of g. If g computes a function that is already computed by C,

[13] For the decision problems, we can determine if there is a 10-approximate set cover of size ℓ by outputting $\mathsf{Multi\text{-}MCSP}(T \bullet g, k + \ell)$, which computes whether $\Delta \leq \ell$.

[14] Since $\mathsf{cover}([n], \mathcal{S}) \geq n/r$ (using that the sets in $\mathcal{S}$ have cardinality at most r), we will actually get that, for each $\epsilon > 0$, Δ gives a $4 + \epsilon$ approximation with high probability when n is sufficiently large.

then ignore it. Otherwise, add a $\diamond$ gate to C that takes as input(s) those gate(s) in C that compute the function(s) which are fed as inputs to g in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ (we are guaranteed to find such gates in C since we are iterating in topological order).

By construction, C computes g. (Recall that $g = \bullet_{S \in \mathcal{S}} \mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$, and our construction ensures C computes every function computed by a gate in $\mathbf{DNF}_{T_{\langle S^m \rangle}}$ for any $S \in \mathcal{S}$.) Moreover, our construction maintains that every gate in C computes a component of g that is not computed by any other gate or input wire. Thus, since g has at most k unique components not computed by input wires, C has at most k gates. ◀

One consequence of Proposition 14 is that $\Delta = \mathsf{CC}(T \bullet g) - \mathsf{CC}(g)$. With this fact, we can prove our two main lemmas.

▶ **Lemma 15.** *If* $\mathsf{cover}([n], \mathcal{S}) = \ell$*, then* $\Delta \leq \ell$*.*

Proof. Let $S_1, \ldots, S_\ell$ be a cover of $[n]$ using sets from $\mathcal{S}$. Then, by construction, we have that $T = T_{\langle S_1^m \rangle} \vee \cdots \vee T_{\langle S_\ell^m \rangle}$. Since $T_{\langle S_1^m \rangle}, \ldots, T_{\langle S_\ell^m \rangle}$ are components of g, this implies that

$$\Delta = \mathsf{CC}(T \bullet g) - \mathsf{CC}(g) \leq \ell$$

as desired. ◀

▶ **Lemma 16.** *Let* ℓ *be the largest integer such that* $\mathsf{cover}([n], \mathcal{S}) \geq 4\ell$*. Then,* $\Delta > \ell$ *with high probability.*

Proof. Our strategy will be as follows. We say that the choice of T is *bad* if, for that choice of T, $\Delta \leq \ell$. We will then upper bound the number of bad T by showing such T have short descriptions.

Fix some bad T. Then $\ell \geq \Delta = \mathsf{CC}(T \bullet g) - k$, so $\mathsf{CC}(T \bullet g) \leq \ell + k$. Since there is a circuit C computing $T \bullet g$ using at most $\ell + k$ gates and k of the gates in C must compute the unique components of g (using Proposition 14), it follows that there is a circuit D that takes $(\log(m) + k)$-inputs and has at most ℓ gates such that

$$D(x, g_1(x), \ldots, g_k(x)) = T(x)$$

for all $x \in \{0,1\}^{\log m}$ where $g_1, \ldots, g_k$ are the unique components of g. Moreover, since D has only ℓ gates of fan-in 2, it uses at most 2ℓ of the components of g in the circuit. Thus, after a possible relabeling of $g_1, \ldots, g_k$, we can assume D takes at most $(\log(m) + 2\ell)$-inputs and that

$$D(x, g_1(x), \ldots, g_{2\ell}(x)) = T(x).$$

Hence, to describe T, we just need to have a description for D as well as a description for $g_1, \ldots, g_{2\ell}$. Indeed, this will be the last step in our eventual description of T. We present the eventual description now so as to guide the reader. Our proof will subsequently proceed working through this description from bottom to top.

1. Given
   - a subset $J \subseteq [n]$ of size at most $\leq n(1 - \frac{1}{2r})$
   - for each $j \in J$ a partial truth table encoding $(j, V_j) \in [n] \times \{0,1\}^{m/n+1}$
   - r-bounded subsets $S_1, \ldots, S_{2\ell} \subseteq [n]$ whose union is J,
   - gate numbers $u_1, \ldots, u_{2\ell} \in [2m \log m]$,
   - and a circuit D of size ℓ with $(n + 2\ell)$ inputs

2. For $j \in J$, let $T_{\langle P_j^{m,n} \rangle}$ be the function whose values on $P_j^{m,n}$ in lexicographic order are given by the binary string V_j and is zero everywhere else
3. For $i \in [2\ell]$, let $T_{\langle S_i^m \rangle} = \bigvee_{j \in S_i \subseteq J} T_{\langle P_j^{m,n} \rangle}$
4. For $i \in [2\ell]$, let g_i be the function computing by the u_ith gate of $\mathbf{DNF}_{T_{\langle S_i^m \rangle}}$
5. Let $T(x) = D(x, g_1(x), \ldots, g_{2\ell}(x))$

Step 4: Describing the g_i. Since $g_1, \ldots, g_{2\ell}$ are components of g and $g = \bullet_{S \in \mathcal{S}} \mathsf{Eval}\text{-}\mathbf{DNF}_{T_{\langle S^m \rangle}}$, there exist $u_1, \ldots, u_{2\ell}$ and $S_1, \ldots, S_{2\ell}$ such that each g_i is the u_ith gate of $\mathbf{DNF}_{T_{\langle S_i^m \rangle}}$ for $i \in [2\ell]$. Moreover, each $u_i \leq 2m \log m$ by the trivial upper bound on the number of gates in a canonical DNF.

Step 3: Describing the $T_{\langle S_i^m \rangle}$. Next, we focus on encoding $T_{\langle S_i^m \rangle}$ for some i. Since $S_i^m = \bigcup_{j \in S_i} P_j^{m,n}$ (by construction of S_i^m), we have (by construction of $T_{\langle S_i^m \rangle}$) that $T_{\langle S_i^m \rangle} = \bigvee_{j \in S_i} T_{\langle P_j^{m,n} \rangle}$. Thus, to compute $T_{\langle S_i^m \rangle}$ for all $i \in [2\ell]$, it suffices to know $S_1, \ldots, S_{2\ell}$ as well as $T_{\langle P_j^{m,n} \rangle}$ for all $j \in J \stackrel{\text{def}}{=} \bigcup_{i \in [2\ell]} S_i$.

Step 2: Describing $T_{\langle P_j^{m,n} \rangle}$ for $j \in J$. The key to our encoding is to realize that $|J|$ cannot be too large, and thus, we do not need to know $T_{\langle P_j^{m,n} \rangle}$ for all $j \in [n]$. Since $\mathsf{cover}([n], \mathcal{S}) \geq 4\ell$, and J is the union of 2ℓ sets from $\mathcal{S}$, it follows that $|J| \leq n - 2\ell$. Moreover, we have that

$$n/r \leq \mathsf{cover}([n], \mathcal{S}) < 4\ell + 4$$

where the first inequality comes from the sets in $\mathcal{S}$ having cardinality at most r and the second inequality comes from the definition of ℓ. Thus,

$$|J| \leq n - 2\ell < n - \frac{n}{2r} + 2 = n(1 - \frac{1}{2r}) + 2.$$

Moreover, we can encode $T_{\langle P_j^{m,n} \rangle}$ for $j \in J$ very efficiently. To describe $T_{\langle P_j^{m,n} \rangle}$, it suffices to describe j and then give the list of values of $T_{\langle P_j^{m,n} \rangle}$ on the set $P_j^{m,n}$ in lexicographic order. Since $|P_j^{m,n}| \leq m/n + 1$ (essentially by construction), we can encode this list of values by a string $V_j \in \{0,1\}^{m/n+1}$ (where we pad this string with extra zeroes if $|P_j^{m,n}| < m/n + 1$).

Step 1: Counting the bits in the description. Now, we count the number of bits in our description of T. Describing J requires n-bits. For $j \in J$, each partial truth table encoding (j, V_j) requires at most $2 \log n + m/n + 1$ bits. Using that $|J| \leq n(1 - \frac{1}{2r})$, we get that all these encodings require at most $n(1 - \frac{1}{2r})(2 \log n + m/n + 1)$ bits. Encoding the r-bounded subsets $S_1, \ldots, S_{2\ell} \subseteq [n]$ requires at most $2n\ell \leq n^2$ bits (here we use the fact that $4\ell \leq n$ since $\mathsf{cover}([n], \mathcal{S}) = 4\ell$). Encoding the gate numbers $u_1, \ldots, u_{2\ell}$ requires at most $4\ell \log(2m \log m) = O(n \log n)$ (using that $4\ell \leq n$ and that $m = n^{O(1)}$). Finally, describing a circuit with ℓ gates and $(n + 2\ell)$ input bits where $4\ell \leq n$ requires $O(n \log n)$ bits. Putting this all together and using that $m = O(n^3)$, we get that T can be described using

$$n + (n(1 - \frac{1}{2r}) + 2)(2 \log n + m/n + 1) + n^2 + O(n \log n) = (1 - \frac{1}{2r})m + O(n^2) = (1 - \Omega(1))m$$

bits. Thus, the number of bad T is upper bounded by $2^{(1-\Omega(1))m}$, so the probability that T is bad is at most $2^{-\Omega(m)}$. ◀

4.3 Search-to-decision reduction for Multi-MCSP

A similar "bottom-up" search-to-decision reduction to the one for Partial-MCSP works for Multi-MCSP. Recall, in the Search-Multi-MCSP problem, the goal is a to output a Boolean circuit C of minimum size computing a (multi-output) Boolean function f.

We will now describe a deterministic polynomial-time procedure to solve Search-Multi-MCSP using an oracle to Multi-MCSP. In our algorithm, we will assume access to an oracle that computes CC, the exact circuit complexity of a (multi-output) Boolean function, but one can compute CC efficiently using an oracle to Multi-MCSP. Finally, our algorithm will work recursively and actually solve a slightly stronger problem.

Our algorithm makes use of the Evaluation Function Eval-C defined in Subsection 4.1, where our precise notion of multi-output computation can also be found. Additionally, we say a circuit C is a *subcircuit* of a circuit D if D can be obtained by adding gates to C.

Algorithm E.

Input. The truth table of a multi-output Boolean function f with n inputs and a circuit C with n input variables $x_1, \ldots, x_n$ and s gates $g_1, \ldots, g_s$.

Output. A minimum-sized circuit D computing f among circuits containing C as a subcircuit.

1. If C computes f, then return C.
2. Otherwise, for each operation $\star \in \{\vee, \wedge, \neg\}$, and for each appropriate choice of one or two operands from $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$:
   - 2.1 Let C^+ be the circuit obtained by adding a $\star$ gate to C with the chosen operands.
   - 2.2 If $\mathsf{CC}(\text{Eval-}C^+) > \mathsf{CC}(\text{Eval-}C)$ and $\mathsf{CC}(f \bullet \text{Eval-}C^+) = \mathsf{CC}(f \bullet \text{Eval-}C)$, then output $E(f, C^+)$.

If Algorithm E works as claimed, then it is easy to see that invoking E on a multi-output Boolean function f and an empty circuit yields the desired search-to-decision reduction.

We now sketch the proof that E runs in polynomial-time and returns the claimed output on input (f, C). We argue by induction on the quantity $s \stackrel{\text{def}}{=} \mathsf{CC}(f \bullet \text{Eval-}C) - \mathsf{CC}(\text{Eval-}C)$ (intuitively, s is the minimum number of gates that need to be added to C in order to compute f).

If $s = 0$, then it must be that C computes f. (By a similar argument to Proposition 14, $\mathsf{CC}(\text{Eval-}C)$ is exactly the number of distinct components of Eval-C not computed by input wires. It follows that any function computed by a gate in an optimal circuit for Eval-C must be a component of Eval-C and therefore be computed by C.) Thus, Algorithm E correctly returns C in step (1).

Now assume that $s \geq 1$. Let D be a circuit computing f of minimum-size among circuits containing C as a subcircuit.

▷ Claim 17. There is a gate in D whose function that computes a function h such that h is not computed in C and such that h can be computed by applying an operator $\star \in \{\vee, \wedge, \neg\}$ to operand(s) solely from the set $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$.

Proof. Imagine labeling as depth-0 all the input variables and gates in D whose functions are computed in C. Next, inductively define all other gates to have depth one more than the maximum depth of their input gates. Since D computes f and C does not, there is at least one gate in D with positive depth. Therefore there must be one gate with depth-1. Any gate with depth-1 satisfies the property that it is not computed in C and can be computed by adding a single operator to C. ◁

As a consequence of this, we get that some C^+ will pass the test in Step (2.2).

▷ Claim 18. At least one choice of operator and operand(s) in Step (2.1) will pass the test in Step (2.2).

Proof. Let h be the function guaranteed by Claim 17 that is computed by some choice of apply some operator $\star \in \{\vee, \wedge, \neg\}$ to some operand(s) solely from the set $\{x_1, \ldots, x_n, g_1, \ldots, g_s\}$, and let C^+ be the circuit obtained by adding this gate to C. Since h is not computed by C and C^+ is obtained by adding a gate to C, it follows that $\mathsf{CC}(\mathsf{Eval}\text{-}C^+) > \mathsf{CC}(\mathsf{Eval}\text{-}C)$. Next, since h is computed by D and C is a subcircuit of D, it follows that every function computed by a gate in C^+ is computed by a gate in D. Thus, we have that $\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) \leq |D|$. Combining this with the optimality of D and the fact that C is a subcircuit of C^+, we have that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C) \leq \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) \leq |D| = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C).$$

Therefore, $\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C)$, and so C^+ passes the test in Step (2.2). ◁

Now, let C^+ be any circuit that passes the test in Step (2.2). Then the quantity

$$s^+ = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) - \mathsf{CC}(\mathsf{Eval}\text{-}C^+) < \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C) - \mathsf{CC}(\mathsf{Eval}\text{-}C) = s$$

using the test conditions in Step (2.2). Thus, by the inductive hypothesis, $E(f, C^+)$ returns a circuit D that is a minimum sized circuit for f among circuits containing C^+ as a subcircuit. Since C^+ contains C as a subcircuit, it follows that D also contains C as a subcircuit. Moreover, $|D| = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^+) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C)$ (by a test condition). Hence, D is a minimum-sized circuit for f among circuits containing C as a subcircuit, as desired.

Finally, we argue that Algorithm E runs in polynomial-time on input (f, C^1). Let T be the truth table of f. Then a lower bound on the input length is $m = |T| + |C^1|$. We will show that E runs in time polynomial in m.

First, we upper bound the number of recursive calls c. Let $(f, C^1), \ldots, (f, C^c)$ denote the successive inputs to E made by the recursive calls where (f, C^1) is the original input. Using induction on the two test conditions in Step (2.2), we have that

$$\mathsf{CC}(\mathsf{Eval}\text{-}C^c) \geq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + c - 1$$

and that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^c) = \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1).$$

On the other hand, we have that

$$\mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1) \leq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + O(|T| \log |T|)$$

as witnessed by the circuit that uses trivial DNFs to compute each component of f individually and a minimum-sized circuit for $\mathsf{Eval}\text{-}C^1$ to compute $\mathsf{Eval}\text{-}C^1$. Putting these facts together, we get that

$$\begin{aligned}
\mathsf{CC}(\mathsf{Eval}\text{-}C^1) + c - 1 &\leq \mathsf{CC}(\mathsf{Eval}\text{-}C^c) \\
&\leq \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^c) \\
&= \mathsf{CC}(f \bullet \mathsf{Eval}\text{-}C^1) \\
&\leq \mathsf{CC}(\mathsf{Eval}\text{-}C^1) + O(|T| \log |T|),
\end{aligned}$$

so the number of recursive calls $c = O(|T| \log |T|) = O(m^2)$.

Next, we analyze the computation required in recursive call $i \in [c]$. Since Algorithm E adds at most one gate to C in each recursive call and $i \leq c$, we have by induction that

$$|C^i| \leq |C^1| + c - 1 = O(|C^1| + m^2) = O(m^2).$$

Therefore, the computation in Step (1) of checking whether C^i computes T can be done in time $O(|T||C^i|) = O(m^3)$ (by just evaluating C^i on all inputs). Next, trying all possible operands on all pairs of input variables and circuit gates from C^i in Step (2) takes at most $O((|C^i| + n)^2) = O(m^4)$ time. Lastly, it is easy to see that Steps (2.1) and Steps (2.2) run in $O(m)$ time. Thus, each recursive step of Algorithm E runs in time $O(m^4)$ and there are $O(m^2)$ recursive calls, so Algorithm E runs in time $O(m^6)$.

5 On the NP-hardness of communication minimization problems

5.1 Background

Hardness of graph coloring. Our NP-hardness reduction is from the chromatic number problem:

▶ **Definition 19** (Chromatic number). *A* coloring *of an undirected graph G, is a partition of the vertices such that no edge has both endpoints in the same part. The* chromatic number *of a graph G, denoted $\chi(G)$, is the smallest number of parts of a coloring of G.*

The NP-hardness of approximating the chromatic number has been established by a series of results [57, 31, 24], culminating in a paper by Zuckerman [82], where the following was proven:

▶ **Theorem 20** ([82]). *For every constant $\varepsilon > 0$ it is NP-hard to approximate $\chi(G)$ for a given n-vertex graph G, with an approximation ratio better than $n^{1-\varepsilon}$. More precisely, for every $L \in$ NP and $\varepsilon > 0$, there exists a polynomial-time algorithm that, on input x, outputs a parameter k and an n-vertex graph G such that if $x \in L$ then $\chi(G) \leq k$, and if $x \notin L$ then $\chi(G) > n^{1-\varepsilon} \cdot k$.*

In the reductions above, the parameter k is $\Theta(n^\varepsilon)$. More recent results on the hardness of approximating chromatic number allow for a different gap, where k is constant as n grows, and we wish to distinguish graphs which are k-colorable from graphs which are not $g(k)$-colorable, for a fast-growing function $g\colon \mathbb{N} \to \mathbb{N}$. An original such result with $g(k) = k^{\Omega(\log k)}$ was shown by Khot [47], which was later improved to $g(k) = 2^{\Omega(k^{1/3})}$ by Huang [37]. The latest result, by Wrochna and Živný [79], achieves $g(k) = \binom{k}{\lfloor k/2 \rfloor}$:

▶ **Theorem 21** ([79]). *Let $k \geq 4$ be a natural number. For every $L \in$ NP, there exists a polynomial-time algorithm that, on input x, outputs an graph G such that if $x \in L$ then $\chi(G) \leq k$, and if $x \notin L$ then $\chi(G) > \binom{k}{\lfloor k/2 \rfloor}$.*

Communication complexity of relations and partial functions. We will need the following definitions.

▶ **Definition 22.** *We will call a* two-player communication relation*, or simply a* relation*, to any subset $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are finite sets, such that, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, there exists at least one $z \in \mathcal{Z}$ with $(x, y, z) \in F$.*

Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, and a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we let

$$F(x, y) = \{z \in \mathcal{Z} \mid (x, y, z) \in F\} \neq \varnothing.$$

The matrix with $\mathcal{X}$-indexed rows and $\mathcal{Y}$-indexed columns, whose (x, y) entry is $F(x, y)$, is called the communication matrix *of F.*

Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a rectangle $R = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$, and an element $z \in \mathcal{Z}$, we say R is z-monochromatic for F, *if $(x, y, z) \in F$ for all $(x, y) \in R$. We say R is* monochromatic for F *if there exists a $z \in \mathcal{Z}$ with R being z-monochromatic for F.*

A partial two-player function *is a relation $f \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ such that, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, either $|f(x, y)| = 1$, in which case we say f is* defined *at (x, y), or $f(x, y) = \mathcal{Z}$, in which case we say f is* undefined *at (x, y). If f is defined at (x, y), we write $f(x, y) = z$ in place of $f(x, y) = \{z\}$, and if f is undefined at (x, y), we write $f(x, y) = *$, instead of $f(x, y) = \mathcal{Z}$.*

▶ **Definition 23** ($\mathsf{P}(F)$)**.** *Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a* partition *of F is a family Π of monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ appears in exactly one rectangle of Π. The* partition number *of a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, denoted $\mathsf{P}(F)$ is the smallest possible size of a partition of F.*

▶ **Definition 24** ($\mathsf{C}(F)$)**.** *Given a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, a* cover *of F is a family Γ of monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ appears in at least one rectangle of Γ. The* cover number *of a relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, denoted $\mathsf{C}(F)$ is the smallest possible size of a cover of F.*

In the context of Boolean-valued functions, the usual notion of *cover number* (see [42] §4.2, p. 97), and related notion of non-deterministic communication complexity, only require that the 1s of the communication matrix are covered.

▶ **Definition 25** ($\mathsf{C}_1(F)$)**.** *Given a partial two-player function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, a* 1-cover *of F is a family Γ of 1-monochromatic rectangles for F, such that every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ having $f(x, y) = 1$ appears in at least one rectangle of Γ. The* 1-cover number *of f, denoted $\mathsf{C}_1(F)$ is the smallest possible size of a 1-cover of F. We then define the* non-deterministic communication complexity *of f to be $\mathsf{N}(f) = \lceil \log \mathsf{C}_1(f) \rceil$.*

▶ **Definition 26.** *A* protocol *π over $\mathcal{X} \times \mathcal{Y}$ is a rooted tree:*

- *Each node v is associated with a rectangle $\pi^{-1}(v) = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$.*
- *Each non-leaf node v, with $\pi^{-1}(v) = A \times B$, is labeled by either (a) a partition $A = A_0 \cup A_1$ of A, in which case we say it is* Alice's node *or (b) a partition $B = B_0 \cup B_1$ of B, in which case we say it is* Bob's node.
- *The rectangle associated with the root is $\mathcal{X} \times \mathcal{Y}$.*
- *If a non-leaf node v of Alice has $\pi^{-1}(v) = A \times B$ and is labeled by a partition $A = A_0 \cup A_1$ of A, then for each $c \in \{0, 1\}$ there will be a unique child v_c of v, with $\pi^{-1}(v_c) = A_c \times B$; similarly for Bob's nodes.*

▶ **Definition 27** ($\mathsf{L}(F)$, $\mathsf{D}(F)$)**.** *Let $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a relation, and π a protocol over $\mathcal{X} \times \mathcal{Y}$. We say that π is* a protocol for F *if, for any leaf ℓ of π, the rectangle $\pi^{-1}(\ell)$ is monochromatic for F. We then let $\mathsf{L}(F)$ be the smallest possible number of leaves in a protocol for F, and we let $\mathsf{D}(F)$ be the smallest possible depth of a protocol for F.*

Note that the rectangles associated with the leaves of a protocol for F form a partition of F, any partition of F is a cover of F, and any cover of F contains a 1-cover of F, and hence we get the following:

▶ **Lemma 28.** *For any relation $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$, $\mathsf{L}(F) \geq \mathsf{P}(F) \geq \mathsf{C}(F)$, and for a partial two-player function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, $\mathsf{C}(f) \geq \mathsf{C}_1(f)$.*

▶ **Definition 29.** *A* DAG-like protocol *γ over $\mathcal{X} \times \mathcal{Y}$ is a directed acyclic graph:*

- *γ has a single source node, called the* root*, and one or more sink nodes, called* leaves*.*
- *Each node v is associated with a rectangle $\gamma^{-1}(v) = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$.*
- *Each non-leaf node v, with $\gamma^{-1}(v) = A \times B$, is labeled by either (a) a partition $A = A_0 \uplus A_1$ of A, in which case we say it is* Alice's node *or (b) a partition $B = B_0 \uplus B_1$ of B, in which case we say it is* Bob's node*.*
- *The rectangle associated with the root is $\mathcal{X} \times \mathcal{Y}$.*
- *If a non-leaf node v of Alice has $\gamma^{-1}(v) = A \times B$ and is labeled by a partition $A = A_0 \uplus A_1$ of A, then for each $c \in \{0,1\}$ there will be an edge from v to a node v_c of γ, which must be such that $A_c \times B \subseteq \gamma^{-1}(v_c)$; similarly for Bob's nodes.*

▶ **Definition 30** ($\mathsf{S}(F)$)**.** *Let $F \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ be a relation, and γ a DAG-like protocol over $\mathcal{X} \times \mathcal{Y}$. We say that γ is* a DAG-like protocol for F *if, for any leaf ℓ of γ, the rectangle $\gamma^{-1}(\ell)$ is monochromatic for F. We then let $\mathsf{S}(F)$ be the smallest possible number of nodes in a DAG-like protocol for F.*

Note that: (1) a protocol with k leaves is a DAG-like protocol with $2k-1$ nodes, (2) the rectangles associated with the leaves of a DAG-like protocol form a cover of F, and (3) any DAG with 1 source node, k sink nodes, and maximum out-degree 2 has at least $2k-1$ nodes in total. Hence:

▶ **Lemma 31.** $\mathsf{S}(F) \leq 2\mathsf{L}(F) - 1$ *and* $\mathsf{S}(F) \geq 2\mathsf{C}(F) - 1$.

5.2 A reduction from Graph Coloring to Partial-MCCP

Let us be given a graph $G = ([n], E)$ with $E \subseteq \binom{[n]}{2}$, and consider the partial two-player Boolean function $f_G : [n] \times [n] \to \{0, 1, *\}$, given by

$$f_G(i,j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } \{i,j\} \in E, \\ * & \text{if } \{i,j\} \notin E. \end{cases}$$

By $f_G(i,j) = *$ we mean that $f_G(i,j)$ is undefined, which is to say Alice and Bob may output any value when given (i,j) as input.

▶ **Theorem 32.** *We have $\mathsf{L}(f_G) \leq 2\chi(G)$ and $\mathsf{C}_1(f_G) = \chi(G)$.*

Proof. To show that $\mathsf{L}(f_G) \leq 2\chi(G)$, consider the simple protocol where Alice and Bob have agreed on a coloring of G, so Alice begins by sending the color of her vertex, and Bob replies whether the color of his vertex is the same. If the two colors match, they output 1, and otherwise they output 0.

This is a protocol for f, since a valid coloring of G will only color i and j by the same color if $i = j$, or $\{i,j\} \notin E$, in which case 1 is a valid output. If i and j are colored differently, then $i \neq j$, and so 0 is a valid output. This protocol has $2\chi(G)$ leaves.

To show that $\mathsf{C}_1(f_G) \leq \chi(G)$, we cover the diagonal by 1-monochromatic rectangles; each such rectangle corresponds to a color c, and is the smallest rectangle containing all diagonal entries (i,i) for vertices i of G that are colored c.

Now suppose we have any positive cover Γ for f. The diagonal entries must be covered by 1-monochromatic rectangles of Γ, so consider the coloring of G which colors vertex $i \in [n]$ by the 1-monochromatic rectangle which covers (i, i). By our choice of f, any two diagonal entries belonging to the same 1-monochromatic rectangle are not connected by an edge of G. And so this gives us a valid coloring of G, which implies that $\chi(G) \leq |\Gamma|$. ◀

5.3 Consequences of the reduction

We may now take Theorems 32 and 20, together with Lemmas 28 and 31, to obtain:

▶ **Corollary 33.** *For any constant $\varepsilon > 0$, it is* NP*-hard to approximate* $\mathsf{L}(f)$, $\mathsf{P}(f)$, $\mathsf{C}(f)$, $\mathsf{C}_1(f)$ *and* $\mathsf{S}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an approximation ratio better than $n^{1-\varepsilon}$.*

Observe that this hardness of approximation result with a ratio of $n^{1-\varepsilon}$ is very close to optimal, since all aforementioned measures for a partial two-player function $f : [n] \times [n] \to \{0, 1, *\}$ are upper-bounded by $2n$. Improving upon this might still be possible, and we leave it as an open problem.

Now notice that the protocol given in the proof of Theorem 20 is balanced. For this reason, its depth is logarithmic in the number of leaves. Thus, since computing $\mathsf{L}(f)$ is hard up to an approximation ratio of $n^{1-\varepsilon}$, it follows that computing the communication complexity $\mathsf{D}(f)$ for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$ is hard, even if we allow for an additive error term of $(1 - \varepsilon) \log n$. Since $\mathsf{D}(f) \leq \log n$, this implies that we cannot approximate $\mathsf{D}(f)$ with an approximation ratio better than $\approx \frac{1}{\varepsilon}$, which we can take as an arbitrarily large constant. The same reasoning applies to non-deterministic communication complexity.

▶ **Corollary 34.** *For any constant $\varepsilon > 0$, it is* NP*-hard to approximate* $\mathsf{D}(f)$ *or* $\mathsf{N}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an error term smaller than $(1 - \varepsilon) \log n$. For any constant $c > 1$, it is* NP*-hard to approximate* $\mathsf{D}(f)$ *or* $\mathsf{N}(f)$, *for a given partial function $f : [n] \times [n] \to \{0, 1, *\}$, with an approximation ratio better than c.*

Let us now prove that, assuming $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time computable approximate characterization of the communication complexity of a partial Boolean function. This is captured by a more general result, which is an immediate consequence of the reduction presented in Section 5.2 and Theorem 21.

▶ **Theorem 35.** *Let $\lambda, \eta : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be functions such that $\eta(1 + \log x) < \lambda(0.99x)$ for all sufficiently large x. Moreover, assume that these functions are non-decreasing for every large enough x. Then, if $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time computable function r, which accepts as input the communication matrix $M_f \in \{0, 1, *\}^{n \times n}$ of a partial Boolean function $f : [n] \times [n] \to \{0, 1, *\}$, and for every large enough n outputs a value $r(M_f)$ such that*

$$\lambda(\mathsf{D}(f)) \leq r(M_f) \leq \eta(\mathsf{D}(f)).$$

Proof. Suppose that functions λ, η, and r exist as described in the statement of the theorem. Let L be any language in NP, and assume that k is a sufficiently large constant. For any $x \in \{0, 1\}^n$, let G_x be the corresponding graph given by Theorem 21. Then let $f_x = f_{G_x}$ be as defined in Section 5.2 above, and let M_x be the communication matrix of f_x. If $x \in L$, we then have by Theorem 32 that $k \geq \chi(G_x) \geq \frac{1}{2}\mathsf{L}(f_x)$, and because the protocol for f_x that witnesses this fact is balanced, we get $1 + \log k \geq \mathsf{D}(f_x)$. Hence $\eta(1 + \log k) \geq \eta(\mathsf{D}(f_x)) \geq r(M_x)$. If, on the other hand, $y \notin L$, then we have $2^{0.99k} < \binom{k}{\lfloor k/2 \rfloor} < \chi(G_y) = \mathsf{C}_1(f_y) \leq \mathsf{L}(f_y)$, and

thus $0.99k < \mathsf{D}(f_y)$. Then $\lambda(0.99k) \leq \lambda(\mathsf{D}(f_y)) \leq r(M_y)$. But since $\lambda(0.99k) > \eta(1 + \log k)$ using our assumptions on these functions and on k, it follows that $r(M_y) > r(M_x)$. As a consequence, the polynomial time function r can be used to distinguish positive and negative instances of L. Since L is an arbitrary language in NP, we get that $\mathsf{P} = \mathsf{NP}$. This completes the proof. ◀

References

1 Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *Journal of Computer and System Sciences*, 74(1):16–34, 2008.

2 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.

3 Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM Journal on Computing*, 47(4):1339–1372, 2018.

4 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM Journal on Computing*, 38(1):63–84, 2008.

5 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.

6 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017.

7 Eric Allender, Rahul Ilango, and Neekon Vafa. The non-hardness of approximating circuit size. In *International Computer Science Symposium in Russia* (CSR), pages 13–24, 2019.

8 Anurag Anshu, Naresh Goud Boddu, and Dave Touchette. Quantum log-approximate-rank conjecture is also false. *arXiv*, 2018. `arXiv:1811.10525`.

9 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Symposium on Foundations of Computer Science* (FOCS), pages 211–220, 2008.

10 Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

11 Avrim Blum, Merrick L. Furst, Michael J., and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *International Cryptology Conference* (CRYPTO), pages 278–291, 1993.

12 Avrim L. Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

13 Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

14 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

15 Dan Boneh and Richard J. Lipton. Amplification of weak learning under the uniform distribution. In *Conference on Learning Theory* (COLT), pages 347–351, 1993.

16 Joan Boyar, Philip Matthews, and René Peralta. On the shortest linear straight-line program for computing linear forms. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 168–179, 2008.

17 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

18 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-Hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012. doi:10.1137/120861072.

19 Arkadev Chattopadhyay, Nikhil S Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. In *Symposium on Theory of Computing* (STOC), pages 42–53. ACM, 2019.

20 Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *Electronic Colloquium on Computational Complexity* (ECCC), 26:22, 2019.

21 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Innovations in Theoretical Computer Science* (ITCS), pages 47–58, 2016.

22 Sebastian Czort. The complexity of minimizing disjunctive normal form formulas. Master's Thesis, University of Aarhus, 1999.

23 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

24 Uriel Feige and Joe Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

25 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-3n lower bound for the circuit complexity of an explicit function. In *Symposium on Foundations of Computer Science* (FOCS), pages 89–98, 2016.

26 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

27 Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. $\mathrm{AC}^0[p]$ lower bounds against MCSP via the coin problem. *Electronic Colloquium on Computational Complexity* (ECCC), 26:18, 2019.

28 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

29 Gary D. Hachtel and Fabio Somenzi. *Logic synthesis and verification algorithms*. Springer, 2006.

30 Thomas R. Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Information and Computation*, 126(2):114–122, 1996.

31 Johan Hastad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Symposium on Foundations of Computer Science* (FOCS), pages 627–636, 1996.

32 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

33 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Computational Complexity Conference* (CCC), pages 5:1–5:31, 2018.

34 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.

35 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.

36 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.

37 Sangxia Huang. Improved hardness of approximating chromatic number. In *International Conference on Approximation Algorithms for Combinatorial Optimization Problems* (APPROX), pages 233–243, 2013.

38 Rahul Ilango. $\mathrm{AC}^0[p]$ lower bounds and NP-hardness for variants of MCSP. *Electronic Colloquium on Computational Complexity* (ECCC), 26:21, 2019.

39 Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties as Oracles. In *Computational Complexity Conference* (CCC), volume 102, pages 7:1–7:20, 2018.

40 Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 353–364, 2002.

41 J. Stephen Judd. Learning in networks is hard. In *International Conference on Neural Networks* (ICNN), volume 2, pages 685–692, 1987.

42 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer, 2012.

43 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

44 Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, 1953.

45 Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994. doi:10.1145/174644.174647.

46 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

47 Subhash Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Symposium on Foundations of Computer Science* (FOCS), pages 600–609, 2001.

48 Subhash Khot and Rishi Saket. Hardness of minimizing and learning DNF expressions. In *Symposium on Foundations of Computer Science* (FOCS), pages 231–240, 2008.

49 Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM Journal on Computing*, 20(5):962–986, 1990.

50 Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *Innovations in Theoretical Computer Science* (ITCS), pages 55:1–55:10, 2018.

51 Jan Krajícek. *Forcing with Random Variables and Proof Complexity*. Cambridge University Press, 2011.

52 Eyal Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

53 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

54 Eyal Kushilevitz and Enav Weinreb. On the complexity of communication complexity. In *Symposium on Theory of Computing* (STOC), pages 465–474, 2009.

55 Leonid Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

56 László Lovász and Michael Saks. Lattices, mobius functions and communications complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 81–90, 1988.

57 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

58 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

59 Edward J. McCluskey. *Introduction to the theory of switching circuits*. McGraw-Hill, 1965.

60 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Symposium on Theory of Computing* (STOC), 2019.

61 Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Electronic Colloquium on Computational Complexity* (ECCC), 24:144, 2017.

62 Cody D. Murray and Richard Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

63 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Conference on Computational Complexity* (CCC), 2019.

64 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

65 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018.

66 James Orlin. Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae*, 80(5):406–424, 1977.

67 Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

68 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988.

69 Netanel Raviv. Truth table minimization of computational models. *CoRR/arXiv*, abs/1306.3766, 2013. `arXiv:1306.3766`.

70 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

71 Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

72 J. Paul Roth and Richard M. Karp. Minimization over boolean graphs. *IBM Journal of Research and Development*, 6(2):227–238, 1962.

73 Christoph Scholl. *Functional decomposition with applications to FPGA synthesis*. Kluwer Academic Publishers, 2001.

74 Makrand Sinha and Ronald de Wolf. Exponential separation between quantum communication and logarithm of approximate rank. *arXiv*, 2018. `arXiv:1811.10090`.

75 Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

76 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Symposium on Theory of Computing* (STOC), pages 453–461, 2001.

77 Christopher Umans, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(7):1230–1246, 2006.

78 Salil P. Vadhan. On learning vs. refutation. In *Conference on Learning Theory* (COLT), pages 1835–1848, 2017.

79 Marcin Wrochna and Stanislav Živnỳ. Improved hardness for H-colourings of G-colourable graphs. *arXiv*, 2019. `arXiv:1907.00872`.

80 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Symposium on Theory of Computing* (STOC), pages 209–213, 1979. `doi:10.1145/800135.804414`.

81 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Symposium on Foundations of Computer Science* (FOCS), pages 80–91, 1982.

82 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Symposium on Theory of Computing* (STOC), pages 681–690, 2006.

A The connection between average-case Partial-MCSP and learning

A few years ago, [17] established an equivalence between solving MCSP on average and learning Boolean circuits under the uniform distribution using *membership queries*.[15] In this section, we describe a similar equivalence between the average-case complexity of Partial-MCSP and learning Boolean circuits under the uniform distribution using *random examples*. The result is implicit in the work of [78], and similar ideas have appeared in the literature before (see e.g. [11]). Here we simply translate these ideas to the language of circuit minimization.[16]

[15] See [34] for a discussion on the average-case complexity of MCSP and its connection to natural proofs [70].

[16] Note that [78] considers learnability in the PAC model (i.e. with respect to arbitrary distributions), while here we focus on learnability under the uniform distribution.

First, we formalize the learning model. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, an *example oracle* $\mathcal{EX}(f)$ for f is a procedure that when invoked returns a pair $(x, f(x))$ consisting of a uniformly distributed string $x \in \{0,1\}^n$ and its label $f(x)$. We say that a circuit C is ε-*close* to a function f if $\Pr_x[C(x) \neq f(x)] \leq \varepsilon$. A randomized algorithm A *learns* a class of Boolean functions $\mathcal{F}$ with *accuracy* ε and *confidence* δ if, for every $f \in \mathcal{F}$ of the form $f\colon \{0,1\}^n \to \{0,1\}$, when A is given access to an example oracle $\mathcal{EX}(f)$,

$$\Pr_{A,\,\mathcal{EX}(f)}[A^{\mathcal{EX}(f)}(1^n) \text{ outputs a circuit } C \text{ that is } \varepsilon(n)\text{-close to } f\,] \;\geq\; 1 - \delta(n).$$

The confidence parameter $\delta(n)$ can be easily boosted without a significant increase of running time. In particular, a learner that succeeds with probability at least $1/\mathsf{poly}(n)$ can be transformed into a learner that fails with probability at most $1/\mathsf{poly}(n)$ with just a polynomial overhead in the running time (cf. [46]). For this reason, the discussion below will concentrate on the accuracy parameter ε, implicitly assuming that $\delta(n) = 1/n$.

We will focus on the class $\mathcal{F} = \mathsf{SIZE}[s]$ that corresponds to Boolean functions computable by (unrestricted) Boolean circuits of size at most s, where $s\colon \mathbb{N} \to \mathbb{N}$. For simplicity, we focus on the regime where $s = \mathsf{poly}(n)$.

Next, we make precise the notion of average-case complexity of Partial-MCSP. We say that a randomized algorithm B *solves* Partial-MCSP (over dimension n) *on average* with *advantage* γ for a *size parameter* s using t *samples* if, for every $f \in \mathsf{SIZE}[s]$ of the form $f\colon \{0,1\}^n \to \{0,1\}$, we have:

$$\left| \Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] - \Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \right| \;\geq\; \gamma(n),$$

where $b \in \{0,1\}^t$ and $z^1, \ldots, z^t \in \{0,1\}^n$ are independent and uniformly random, and $t = t(n)$.

▶ **Theorem 36.** *The following implications hold.*

(1) *For every $c, d \in \mathbb{N}$ there exists $\ell \in \mathbb{N}$ such that if $\mathsf{SIZE}[n^c]$ can be learned in time $O(n^d)$ with accuracy $\varepsilon = 1/10$, then* Partial-MCSP *over dimension n can be solved on average in polynomial time with advantage $\gamma(n) \to_n 1$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples.*

(2) *If for every $c \in \mathbb{N}$ there exists $\ell \in \mathbb{N}$ such that* Partial-MCSP *over every dimension n can be solved on average in polynomial time with advantage $\gamma = 1/10$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples, then for every $a \in \mathbb{N}$ the class $\mathsf{SIZE}[n^a]$ can be learned in polynomial time with accuracy $\varepsilon(n) = 1/n$.*

In other words, Theorem 36 says that polynomial-size Boolean circuits over $\{0,1\}^n$ can be learned in polynomial time using random examples if and only if Partial-MCSP over $\{0,1\}^{\mathsf{poly}(n)}$ can be solved on average in polynomial time.

Proof Sketch. We start with the proof of (1). Let $\ell \stackrel{\text{def}}{=} n^{d+1} + n$. Assuming the existence of a learning algorithm A for $\mathsf{SIZE}[n^c]$ with accuracy $\varepsilon = 1/10$ and confidence $\delta = 1/n$, the algorithm B for average-case Partial-MCSP employs A as a sub-routine, and computes as follows. Given 1^n and a sequence $(z^1, a_1), \ldots, (z^t, a_t)$ in $\{0,1\}^{t\cdot(n+1)}$, where $t(n) \stackrel{\text{def}}{=} n^\ell$, B uses the first n^{d+1} pairs (z^i, a_i) to simulate the answers to the oracle calls made by A to its example oracle. Let C be the circuit output by $A^{\mathcal{EX}(\cdot)}(1^n)$ after its computation. B uses the last n input pairs (z^i, a_i) to compute the fraction $\alpha \in [0,1]$ of such pairs for which $C(x^i) \neq a_i$. Finally, B outputs 1 if and only if $\alpha \leq 1/3$. Note that B runs in polynomial time under the assumption that A runs in time $O(n^d)$.

Let $f \in \mathsf{SIZE}[n^c]$. In this case, $\Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] \geq 1 - 2/n$, since with probability at most $1/n$ algorithm A fails to output a circuit C that is $(1/10)$-close to f, and by a standard concentration bound with probability at most $1/n$ we have $\alpha > 1/3$. On the other hand, it is not hard to see by a standard concentration bound that $\Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \leq 1/n$, since in this case no matter the circuit C output by A, the last n input pairs of B contain random bits b_i that are uncorrelated with C. This shows that B has advantage $\gamma(n) \to_n 1$.

In order to prove (2), it is enough to conclude that for every $a \in \mathbb{N}$ there is $\ell \in \mathbb{N}$ such that the class $\mathsf{SIZE}[n^a]$ can be learned to accuracy $\varepsilon(n) = 1/2 - 1/n^{\ell+1}$ in polynomial time. This claim follows from a result of [15, Section 2] showing in particular that, when learning general polynomial-size Boolean circuits under the uniform distribution from random examples, one can boost the accuracy parameter from $\varepsilon(n) = 1/2 - 1/\mathsf{poly}(n)$ to $\varepsilon(n) = 1/\mathsf{poly}(n)$ with only a polynomial overhead in the running time. (From the discussion above, we also know that it is sufficient to achieve confidence $\delta(n) = 1 - 1/\mathsf{poly}(n)$.)

Proceeding with the proof of (2), we describe a learning algorithm A for $\mathsf{SIZE}[n^a]$ with accuracy $\varepsilon(n) = 1/2 - 1/100n^\ell$ using an algorithm B that solves Partial-MCSP over dimension n on average in polynomial time with advantage $\gamma = 1/10$ for $s(n) = n^c$ using $t(n) = n^\ell$ samples. The argument relies on Yao's connection between pseudorandomness and (un)predictability [81], which is established using a hybrid argument. We provide below a self-contained presentation of the argument.

By assumption, for every $f \in \mathsf{SIZE}[n^a]$,

$$\left| \Pr_{B,\,\{z^i\}_{i\in[t]}}[B(1^n, z^1, f(z^1), \ldots, z^t, f(z^t)) = 1\,] - \Pr_{B,\,\{z^i\}_{i\in[t]},\,b}[B(1^n, z^1, b_1, \ldots, z^t, b_t)) = 1\,] \right| \;\geq\; 1/10.$$

For $i \in \{0, 1, \ldots, t\}$, consider the distribution $\mathcal{D}_i$ supported over $\{0,1\}^{t\cdot(n+1)}$ obtained by sampling a string $x^1, b_1, \ldots, x^i, b_i, \ldots, x^t, b_t$, where each x^j is a random n-bit string, and each b_j is set to $f(x^j)$ if $j > i$ and to a random bit otherwise. For convenience, let $p_i \stackrel{\text{def}}{=} \Pr_{B,\,w\sim\mathcal{D}_i}[B(1^n, w) = 1]$. The inequality above implies that

$$\left| \sum_{i=0}^{t-1} (p_i - p_{i+1}) \right| \;\geq\; 1/10.$$

Consequently, for some index $j \in \{0, 1, \ldots, t-1\}$ that might depend on f,

$$\left| \Pr_{B,\,w\sim\mathcal{D}_j}[B(1^n, w) = 1] - \Pr_{B,\,w\sim\mathcal{D}_{j+1}}[B(1^n, w) = 1] \right| \;\geq\; 1/10t.$$

The only distinction between the distributions $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ is that, for a random $x^{j+1} \in \{0,1\}^n$, one generates the pair $(x^{j+1}, f(x^{j+1}))$, while the other generates the pair (x^{j+1}, b_{j+1}), where b_{j+1} is a random bit. Note that the other coordinates of these two distributions are identically distributed and can be generated using the randomness of the learner and its example oracle $\mathcal{EX}(f)$. For this reason, once one knows the index $j+1$, it is possible to use B (or its negation) to predict the value of f on a random input with advantage $\Omega(1/t)$ over a random guess.

It is not difficult to show that this can be used to design a randomized learning algorithm A that, with probability $\Omega(1/t)$ over its internal randomness and $\mathcal{EX}(f)$, outputs a circuit that is ε-close to f, where $\varepsilon(n) = 1/2 - 1/100t$. Finally, the running time of A is polynomial under the assumption that B runs in polynomial time. ◀

A consequence of this result is that we can base the hardness of learning on the worst-case assumption that $\mathsf{NP} \not\subseteq \mathsf{RP}$ if and only if the existence of an efficient algorithm for average-case Partial-MCSP implies the existence of an efficient (worst-case) algorithm for

Partial-MCSP. In order to see this, note that NP $\subseteq$ BPP if and only if NP $\subseteq$ RP (using a search-to-decision reduction). Now the inclusion NP $\subseteq$ BPP is equivalent to the easiness of (worst-case) Partial-MCSP by Theorem 4, while Theorem 36 establishes an equivalence between learnability and solving Partial-MCSP on average.

Finally, we remark that the connection between learning and Partial-MCSP described here generalizes to any circuit class $\mathcal{C}$ that can efficiently compute the parity function. (This is necessary in order to apply the uniform distribution boosting procedure from [15].) In other words, $\mathcal{C}$-Partial-MCSP is easy on average if and ony if $\mathcal{C}$ can be efficiently learned under the uniform distribution from random examples.

A Super-Quadratic Lower Bound for Depth Four Arithmetic Circuits

Nikhil Gupta
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
nikhilg@iisc.ac.in

Chandan Saha
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
chandan@iisc.ac.in

Bhargav Thankey
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India
thankeyd@iisc.ac.in

Abstract

We show an $\widetilde{\Omega}(n^{2.5})$ lower bound for general depth four arithmetic circuits computing an explicit n-variate degree-$\Theta(n)$ multilinear polynomial over any field of characteristic zero. To our knowledge, and as stated in the survey [88], no super-quadratic lower bound was known for depth four circuits over fields of characteristic $\neq 2$ before this work. The previous best lower bound is $\widetilde{\Omega}(n^{1.5})$ [85], which is a slight quantitative improvement over the roughly $\Omega(n^{1.33})$ bound obtained by invoking the super-linear lower bound for constant depth circuits in [73, 86].

Our lower bound proof follows the approach of the almost cubic lower bound for depth three circuits in [53] by replacing the shifted partials measure with a suitable variant of the projected shifted partials measure, but it differs from [53]'s proof at a crucial step – namely, the way "heavy" product gates are handled. Loosely speaking, a heavy product gate has a relatively high fan-in. Product gates of a depth three circuit compute products of affine forms, and so, it is easy to prune $\Theta(n)$ many heavy product gates by projecting the circuit to a low-dimensional affine subspace [53, 87]. However, in a depth four circuit, the second (from the top) layer of product gates compute products of polynomials having *arbitrary* degree, and hence it was not clear how to prune such heavy product gates from the circuit. We show that heavy product gates can also be eliminated from a depth four circuit by projecting the circuit to a low-dimensional affine subspace, unless the heavy gates together account for $\widetilde{\Omega}(n^{2.5})$ size. This part of our argument is inspired by a well-known greedy approximation algorithm for the weighted set-cover problem.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases depth four arithmetic circuits, Projected Shifted Partials, super-quadratic lower bound

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.23

Acknowledgements We would like to thank Neeraj Kayal and Ankit Garg for sitting through a presentation of this work and giving us useful feedback. Thanks specially to Ankit for bringing the work of Chen and Tell [20] to our notice. A part of this work is done at Microsoft Research India (MSRI), where CS is spending a sabbatical year. CS would like to thank MSRI for providing an excellent research environment and for the hospitality.

1 Introduction

The arithmetic circuit model is naturally well-suited for the study of optimality of algorithms for algebraic and linear algebraic problems. An arithmetic circuit consists of addition ($+$) and multiplication ($\times$) gates, it takes input $\{x_1, x_2, \ldots, x_n\}$ and field scalars, and computes a polynomial in $\{x_1, x_2, \ldots, x_n\}$. *Size* of a circuit is the number of wires in it, and *depth*

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 23; pp. 23:1–23:31


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

is the longest path from an input to an output gate. The two complexity measures – size and depth – of a circuit essentially capture the sequential and parallel complexity of the computation happening inside the circuit.

Arithmetic circuits are *weaker*[1] than Boolean circuits: An explicit lower bound on the size of Boolean circuits implies an explicit lower bound on the size of arithmetic circuits over finite fields[2] and also over fields of characteristic zero[3], but the converse is not true[4]. Still, the best known lower bound for arithmetic circuits is $\Omega(n \log n)$ [13, 90], which is barely super-linear. For arithmetic formulas, an $\Omega(n^2)$ lower bound is known [44]. Several other better lower bounds have been shown in the past few decades for various restricted models of arithmetic circuits (see Section A for a brief history of known lower bounds). But, very few lower bounds are known for circuit models that do not impose restrictions like homogeneity, multilinearity, monotonicity, bounded coefficients, bounded reads etc. One such result is the lower bound for constant depth circuits.

Shoup and Smolensky [86], and Raz [73], showed an $\Omega(\Delta n^{1+\frac{1}{\Delta}})$ lower bound for depth-Δ circuits, where $\Delta = O(\log n)$. In the special case of depth three circuits, an $\Omega(n^2)$ lower bound was shown in [87], which was improved to an $\widetilde{\Omega}(n^3)$ lower bound in [53]. However, for depth four circuits[5], the best lower bound was $\widetilde{\Omega}(n^{1.5})$ [85], which is a slight quantitative improvement over the roughly $\Omega(n^{1.33})$ lower bound obtained by specializing the constant depth lower bound in [73, 86] to depth four circuits.

In this work, we show an $\widetilde{\Omega}(n^{2.5})$ lower bound for depth four circuits. To the best of our knowledge, and as stated in the survey [88] (Section 1.4.2, page-13), this is the first super-quadratic lower bound for this model over fields of characteristic $\neq 2$.

1.1 Our Result

We state our result formally now. Without loss of generality, we will assume that a depth four circuit is a $\Sigma\Pi\Sigma\Pi$ circuit, i.e., the circuit has a $+$-gate on top followed by second layer of $\times$-gates, then a third layer of $+$-gates and finally a bottom layer of $\times$-gates.

▶ **Theorem 1** (Lower bound for depth four circuits). *Over any field of characteristic zero*[6]*, there exists a family of mulitilinear polynomials $\{f_n\}_{n\geq 1}$ in VNP, where f_n is a polynomial in $\Theta(n)$ variables and of degree $\Theta(n)$ such that any depth four circuit computing f_n has $\Omega\left(\frac{n^{2.5}}{(\log n)^6}\right)$ many wires/edges and $\Omega\left(\frac{n^{1.5}}{(\log n)^4}\right)$ many gates.*

---

[1] This is not to mean that arithmetic circuits cannot simulate Boolean circuits. It is easy to see that a Boolean circuit can be efficiently simulated by an arithmetic circuit over any field that contains the additive and the multiplicative identities 0 and 1 respectively.

[2] This is because an arithmetic circuit over a finite field can be simulated by a Boolean circuit with only a slight blow-up in size and depth (see [100]).

[3] It was shown in [16] (Corollary 4.6 in Chapter 4) that $\mathsf{NC}^3/\text{poly} \neq \mathsf{NP}/\text{poly}$ implies $\mathsf{VP} \neq \mathsf{VNP}$ over fields of characteristic zero, assuming the Generalized Riemann Hypothesis. The circuit classes VP and VNP are arithmetic analogues of non-uniform P and NP respectively [96].

[4] as computing a Boolean function is a weaker requirement than computing a specific polynomial representation of the function.

[5] Depth four circuits form a natural circuit class as the "optimal" circuit for an arbitrary polynomial turns out to be a depth four circuit: The *multiplicative complexity* of a polynomial f, denoted $M(f)$, is the minimum number of multiplication gates required to compute f. It is known that there exists an n-variate degree-d polynomial f for which $M(f) = \Omega(\sqrt{\binom{n+d}{d}})$ [21, 39]. On the other hand, every n-variate degree-d polynomial can be computed by a depth four circuit having $\sqrt{\binom{n+d}{d}} \cdot \text{poly}(n, d)$ many multiplication gates [61].

[6] The lower bound holds even if the characteristic is sufficiently large (see Section 4).

A word about the polynomial family. The polynomial family $\{f_n\}_{n\geq 1}$ is a variant of the Nisan-Wigderson design polynomial family, which has been used in proving several other previous lower bounds [49, 51–53, 57, 58, 73, 85]. The n-th member of the family, i.e., f_n is a polynomial in $\mathbf{x} = \{x_1, ..., x_{3m}\}$ and $\mathbf{y} = \{y_1, ..., y_{3m}\}$ variables, where m is an integer in $\left[\frac{n}{2}, 2n\right]$. The degree of the polynomial in $\mathbf{y}$ variables is $\deg_{\mathbf{y}}(f_n) = m$, while its degree in $\mathbf{x}$ variables is $\deg_{\mathbf{x}}(f_n) = d_{\mathbf{x}} = \Theta(\frac{\sqrt{m}}{\ln m})$. Informally, f_n contains multiple 'copies' of the design polynomial in different subsets of the $\mathbf{x}$ variables, while the $\mathbf{y}$ variables are used as 'prefixes' to uniquely identify each such copy. While the reason for having multiple copies is similar to [53], as we shall see in the next section, handling them is a little trickier in our case. Note that because of the way we have defined m and $d_{\mathbf{x}}$, proving that any depth four circuit computing f_n has $\Omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$ many edges and $\Omega\left(\frac{m d_{\mathbf{x}}}{(\ln m)^3}\right)$ many gates would establish the theorem.[7] The exact description of f_n is given in Section 4.

1.2 Proof Idea

Let $\mathtt{C}$ be a depth four circuit over a field $\mathbb{F}$. Like many other works on arithmetic circuit lower bounds, we use a rank based complexity measure to obtain our result. The measure we apply is a variant of the *projected shifted partials* measure, which has been used before in [49, 51, 58, 85] and other works. Our proof can be divided into four steps; the first three show a "small" upper bound on the measure of $\mathtt{C}$ while the last step shows a "large" lower bound on the measure of the hard polynomial f_n described above. We now briefly outline each of these steps.

Step 1: Restricting the bottom support of $\mathtt{C}$. We begin by removing all monomials computed by the bottom layer of $\mathtt{C}$ that have a "large" support. Such restrictions have been used in previous works [49, 51, 56, 58, 85] to control the degree of the (sparse) polynomials computed at the third layer of a depth four circuit so that a "small" upper bound on the measure of the circuit can be obtained. However, in our work, this step plays an even more significant role by enabling us to remove "heavy" gates (see below). While previous works use *random* restrictions, we use a *deterministic* procedure for restricting the bottom support – we briefly explain our reasons for doing so towards the end of this section.

Heavy gates. We call a product gate in the second layer of $\mathtt{C}$ *heavy* if the number of distinct third layer gates (computing sparse polynomials) feeding into it is $\widetilde{\Omega}(n^{1.5})$. The presence of heavy gates makes the task of obtaining a "small" upper bound on the measure of $\mathtt{C}$ difficult. The problem of dealing with heavy gates was also faced by previous works on depth three circuits [53, 87], and was dealt with by removing all heavy gates from the circuit before applying the measure to it. We too remove all heavy gates from $\mathtt{C}$, but our way of doing so differs from [53, 87]. Since a depth three circuit computes a sum of product of affine forms, [53, 87] were able to remove all heavy gates by going modulo affine factors of these gates thereby restricting the circuit to an affine subspace. While going modulo the sparse factors of heavy gates is a natural generalization of this technique for depth four circuits, we do not know how to adopt this method as the quotient ring so obtained might not be a polynomial ring. In the next step, we outline a technique of removing heavy gates from $\mathtt{C}$ which, in spirit, also restricts $\mathtt{C}$ to an affine subspace.

[7] We use log base e in the proof rather than log base 2 as it simplifies the analysis.

Step 2: Removing heavy gates from $\mathtt{C}$. We remove heavy gates from $\mathtt{C}$ by sequentially evaluating *exactly one* sparse factor of each heavy gate to zero. This can be done if $\mathbb{F}$ is algebraically closed, which one can assume without loss of generality (as argued in Section 5). In particular, we use the following greedy procedure: While there exists a heavy gate in $\mathtt{C}$, pick a sparse polynomial for which the ratio of the number of heavy gates connected to it to its fan-in is maximum, and evaluate it to zero. Intuitively, this allows us to remove a large number of heavy gates at the cost of evaluating a few monomials (computed by the bottom layer) to field constants. Then, as we have restricted the bottom support of $\mathtt{C}$ in Step 1, we are able to show that we can remove $\Theta(n)$ many heavy gates at a cost of setting only a few variables to constants (unless the already removed heavy gates account for $\widetilde{\Omega}(n^{2.5})$ size). Note that $\Theta(n)$ many heavy gates would immediately imply the desired lower bound. This greedy procedure is inspired by an approximation algorithm for the weighted set cover problem [99] (Section 2.1, page-16), however its analysis here is tailored to our needs. This step, which is the main contribution of this work, plays a crucial role in enabling us to prove a super-quadratic lower bound and we provide more details about it in Section 3.2.2.

Step 3: Analyzing the measure of $\mathtt{C}$. After all the heavy gates have been removed, a "small" upper bound on the measure of $\mathtt{C}$ is derived by closely following the "grouping" argument made in [53]. However, we replace the shifted partials measure by the projected shifted partials measure as the latter is suitable for controlling the degree of the sparse polynomials computed at the third layer of $\mathtt{C}$. In this step, we divide the factors of a polynomial T computed by a $\times$-gate in the second layer of $\mathtt{C}$ into "groups" of suitable sizes, and multiply out factors in the same group to reduce the effective number of factors of T. This then helps obtain a "small" upper bound on the projected shifted partials measure of T which, by sub-additivity, implies a "small" upper bound on the projected shifted partials measure of $\mathtt{C}$.

Step 4: Lower bound on the measure of f_n. The choice of the hard polynomial f_n is dictated by the above technique of removing heavy gates. As mentioned before, f_n has multiple copies of the Nisan-Wigderson design polynomial, denoted NW. The reason for having multiple copies is that if we work with only one copy, we might end up irreparably damaging it while removing heavy gates from $\mathtt{C}$. On the other hand, starting with multiple copies of NW, much like in [53], we are able to show that the procedure for removing heavy gates leaves an intact copy along with some 'damaged' parts from the other copies. Our use of a deterministic restriction in Step 1 makes it easier to show this. A "large" lower bound on the measure of NW was obtained in [49, 51, 58]. However, it is not clear how to obtain such a lower bound on NW in the presence of other "damaged" parts, and so we remove these parts. Although in [53], such parts were removed by simply setting a subset of variables to 0 and 1, here we need to augment the projected shifted measure appropriately to get rid of them.

2 Preliminaries

Notations. For $r \in \mathbb{N}$, $[r] := \{1, \ldots, r\}$. We use lowercase Greek alphabets like α, β for field constants, bold-face lowercase letters like $\mathbf{x}$ and $\mathbf{y}$ to denote sets of variables, f, g for polynomials in $\mathbb{F}[\mathbf{x}, \mathbf{y}]$, uppercase typewriter alphabets $\mathtt{C}, \mathtt{D}$ for arithmetic circuits over $\mathbb{F}$, uppercase Roman alphabets T, Q for the polynomials computed by the gates of a depth four circuit and M, M_1, M_2 for subsets of natural numbers. For $M \subseteq [3m]$, $\mathbf{x}_M := \{x_i : i \in M\}, \mathbf{y}_M := \{y_i : i \in M\}$. For $\mathbf{z} \subseteq \mathbf{x}_M \cup \mathbf{y}_M$ and $r \in \mathbb{N}$, $\mathbf{z}^{\leq \infty}$ and $\mathbf{z}^{\leq r}$ denote the set of all monomials in $\mathbf{z}$ variables and the set of all monomials in $\mathbf{z}$ variables with degree at most r respectively. For $S \subseteq \mathbb{F}[\mathbf{x}, \mathbf{y}]$, $\dim\langle S \rangle$ denotes the dimension of the $\mathbb{F}$-linear span of S.

Support and degree of a monomial. The support of a monomial η, denoted $\mathrm{Supp}(\eta)$, is the set of variables appearing in it. Also, for any $\mathbf{z} \subseteq \mathbf{x} \cup \mathbf{y}$ we will use $\deg_{\mathbf{z}}(\eta)$ to denote its degree in $\mathbf{z}$ variables. We will say that η is $\mathbf{z}$-multilinear if the degree of every $\mathbf{z}$ variable in η is at most one.

2.1 The complexity measure

Throughout this section, we will assume that $m \in \mathbb{N}$ is as stated in the paragraph following Theorem 1, $M \subseteq [3m]$, $|M| = m$, $f \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$ and $S \subseteq \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$. Note that the set M is not fixed and will depend on the circuit under analysis. Before defining the measure, let us define the operations that make up the measure.

1. **Partial derivatives.** Let $\eta = x_1 \cdots x_k$ be a monomial in $\mathbf{x}$ variables. Then, we define the partial derivative of f with respect to η as

$$\frac{\partial f}{\partial \eta} := \frac{\partial}{\partial x_1}\left(\frac{\partial}{\partial x_2}\left(\cdots\left(\frac{\partial f}{\partial x_k}\right)\right)\right).$$

   If the degree of η is k, then $\frac{\partial f}{\partial \eta}$ is said to be a k-th order partial derivative of f. We denote by $\partial^k_{\mathbf{x}} f$ the set of all k-th order partial derivatives of f taken with respect to multilinear monomials in $\mathbf{x}$ variables.
2. **The shift operation.** Let η be a degree ℓ multilinear monomial in $\mathbf{x}_M$ variables. We say that the polynomial $\eta \cdot f$ is obtained by *shifting* f by η. We denote by $\mathbf{x}^\ell_M f$ the set of polynomials obtained by shifting f by all degree ℓ multilinear monomials in $\mathbf{x}_M$ variables and $\mathbf{x}^\ell_M S := \left\{\mathbf{x}^\ell_M f : f \in S\right\}$.
3. **Multilinear projection.** We define a map $\pi_{\mathbf{x}} : \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M] \to \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$ with $\pi_{\mathbf{x}}(f)$ being the polynomial made up of exactly the $\mathbf{x}$-multilinear monomials of f. Formally, for a monomial η, $\pi_{\mathbf{x}}(\eta) = \eta$ if η is $\mathbf{x}$-multilinear and 0 otherwise. The map is then linearly extended for arbitrary polynomials and $\pi_{\mathbf{x}}(S) := \{\pi_{\mathbf{x}}(f) : f \in S\}$.
4. **A degree based projection.** For $i \in \mathbb{N}$ and $f \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$, we define $[f]_i$ to be the polynomial made up of only those monomials of f whose $\mathbf{y}$-*degree* is exactly i. Formally, for a monomial η, $[\eta]_i = \eta$ if $\deg_{\mathbf{y}}(\eta) = i$ and 0 otherwise. It is then linearly extended for arbitrary polynomials and $[S]_m := \{[f]_m : f \in S\}$.
5. **An evaluation map.** For $\alpha \in \mathbb{F}$ and $\mathbf{z} \subseteq \mathbf{x}_M \cup \mathbf{y}_M$, we define a map $\sigma_{\mathbf{z}=\alpha} : \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M] \to \mathbb{F}[\mathbf{x}_M \setminus \mathbf{z},\ \mathbf{y}_M \setminus \mathbf{z}]$ with $\sigma_{\mathbf{z}=\alpha}(f)$ being obtained from f by setting every variable in $\mathbf{z}$ to α and $\sigma_{\mathbf{z}=\alpha}(S) := \{\sigma_{\mathbf{z}=\alpha}(f) : f \in S\}$.

The operations given in 1, 2 and 3 constitute the projected shifted partials measure [49]. In this work, we define and use the measure $\mathsf{PSP}_{M,k,\ell}$, which is obtained by augmenting the projected shifted partials measure with the operations in 4 and 5 as follows.

▶ **Definition 2** (The measure). *For $m, k, \ell \in \mathbb{N}$, $M \subseteq [3m], |M| = m$ and $f \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$,*

$$\mathsf{PSP}_{M,k,\ell}(f) := \dim\left\langle \sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\mathbf{x}^\ell_M\ \partial^k_{\mathbf{x}} f\right)\right]_m\right)\right\rangle.$$

▶ **Observation 3** (Sub-additivity of the measure). *For any two polynomials $f, g \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$,*

$$\mathsf{PSP}_{M,k,\ell}(f+g) \leq \mathsf{PSP}_{M,k,\ell}(f) + \mathsf{PSP}_{M,k,\ell}(g).$$

The above observation is easy to prove and we omit its proof here.

2.2 Some numerical estimates

▶ **Proposition 4** (Estimating Binomial Coefficients). *For any $n, k \in \mathbb{N}$, $k \leq n$, $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} < \left(\frac{en}{k}\right)^k$.*

▶ **Proposition 5** ([33, 49]). *Let $a(n), f(n), g(n) : \mathbb{Z}_{>0} \to \mathbb{Z}$ be integer values functions such that $(|f| + |g|) = o(a)$. Then, $\ln \frac{(a+f)!}{(a-g)!} = (f+g)\ln(a) \pm O\left(\frac{f^2+g^2}{a}\right)$.*

3 Upper bounding the measure for a depth four circuit

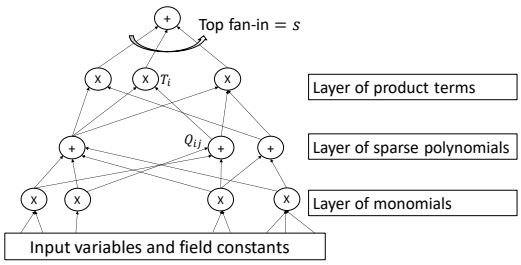
Let $\mathtt{C} = \sum_{i=1}^{s} T_i$ be a depth four circuit computing the polynomial $f = f_n$ (recall $\deg_{\mathbf{x}}(f_n) = d_{\mathbf{x}} = \Theta\left(\frac{\sqrt{m}}{\ln m}\right)$ from Section 1.1); $T_i = \prod_{j=1}^{a_i} Q_{ij}^{e_j}$ and Q_{ij}'s are distinct sparse polynomials computed by the $+$-gates in the third layer of $\mathtt{C}$, and $e_j \geq 1$. We assume that $\mathbb{F}$ is algebraically closed and argue in Section 5 why this holds without loss of generality. For brevity, we would use the terminologies 'product terms', 'sparse polynomials' and 'monomials' for the $\times$-gates, $+$-gates and $\times$-gates in the second, third and fourth layers of $\mathtt{C}$ respectively as shown in Figure 1a. The proof of the upper bound is divided into three steps:

Step 1: Restricting the bottom support. In this step, we show that if $\mathtt{C}$ has fewer than $\Omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$ distinct monomials then we can remove all monomials with support more than $\tau = \lfloor 20 \ln m \rfloor$ at a cost of setting m many $\mathbf{x}$ variables and m many $\mathbf{y}$ variables to zero. (Notice that if there are more than $\Omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$ many monomials then there is nothing to prove.) This step is required not only to remove heavy gates in Step 2 but also in Step 3 where using the fact that all monomials have support at most τ and multilinear projection, we will argue that the degree of all monomials is not too large. More details about this restriction are given in Section 3.2.1.

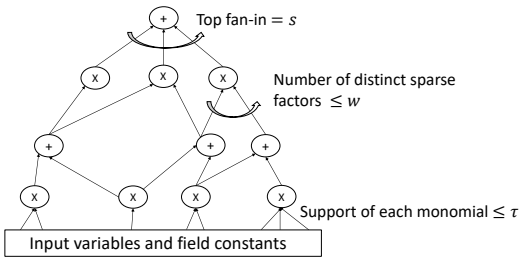
Step 2: Removing heavy gates. The transformed circuit $\mathtt{C}_1$, obtained after Step 1, computes a polynomial in the remaining $2m$ many $\mathbf{x}$ variables and $2m$ many $\mathbf{y}$ variables. Moreover, the number of gates and the fan-in of all gates in $\mathtt{C}_1$ is upper bounded by the number of gates and the fan-in of the corresponding gates in $\mathtt{C}$. A product term in $\mathtt{C}_1$ is called a *heavy gate* if at least $w = \left\lfloor \frac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3} \right\rfloor$ (λ_0 is a large enough constant fixed in Appendix C) many distinct sparse polynomials are connected to it. If there are more than m heavy gates, we are done. Otherwise, we remove all heavy gates using the following greedy procedure: While there is a heavy gate, evaluate a sparse polynomial that would kill the most number of heavy gates at the cost of evaluating as few monomials as possible. As we have already restricted the support of all monomials to τ, we are able to argue in Section 3.2.2 that this can be done at a cost of setting m many $\mathbf{x}$ and m many $\mathbf{y}$ variables to field constants.

These steps transform $\mathtt{C}$ to a 'pruned circuit', defined as follows and depicted in Figure 1b.

▶ **Definition 6.** *We say that a depth four circuit $\mathtt{D}$ is a* pruned circuit *if the support of all monomials in $\mathtt{D}$ is at most $\tau = \lfloor 20 \ln m \rfloor$, and it does not contain any heavy gate; i.e. the number of distinct sparse polynomials feeding into any product term in $\mathtt{D}$ is less than $w = \left\lfloor \frac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3} \right\rfloor$.*



(a) A depth four circuit C.



(b) The pruned depth four circuit D.

Figure 1 A depth four circuit and its pruned version.

Step 3: Analysing the measure. In this step, we analyse the measure $\mathsf{PSP}_{M,k,\ell}$ of the pruned circuit D, obtained after Steps 1 and 2, computing a polynomial in the remaining m many $\mathbf{x}$ and m many $\mathbf{y}$ variables. More details on this analysis are provided in the following section.

3.1 Upper bound on the measure of a pruned depth four circuit

Recall the definition of the measure $\mathsf{PSP}_{M,k,\ell}$ from Section 2. In Steps 1 and 2, we will ensure that if a variable x_i is set to a field constant then y_i is also set to a field constant and vice versa. The set M is then the set of indices of the remaining $\mathbf{x}$ (or $\mathbf{y}$) variables and $|M| = m$.

▶ **Lemma 7.** *Let D be a pruned depth four circuit with top fan-in s computing a polynomial in $\mathbf{x}_M$ and $\mathbf{y}_M$ variables, where $M \subseteq [3m], |M| = m$. Also, let $d_{\mathbf{x}}, \tau, w$ be as defined earlier, $t = \left\lfloor \frac{d_{\mathbf{x}}}{(\ln m)^3} \right\rfloor$, $\delta = \frac{1}{(\ln m)^2}$, $k = \left\lfloor \frac{\delta d_{\mathbf{x}}}{t} \right\rfloor$ and $\ell = \left\lfloor \frac{m}{m^{\delta/t}+1} \right\rfloor$. Then, for sufficiently large m,*

$$\mathsf{PSP}_{M,k,\ell}(\mathsf{D}) \leq s \cdot m^{O(1)} \binom{m}{\ell + 2kt\tau} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}.$$

We prove the lemma at the end of this section. As $\mathsf{D} = T_1 + \cdots + T_s$, where T_i is a product term and as $\mathsf{PSP}_{M,k,\ell}$ is sub-additive, to prove the lemma it suffices to show that for all $i \in [s]$,

$$\mathsf{PSP}_{M,k,\ell}(T_i) \leq m^{O(1)} \binom{m}{\ell + 2kt\tau} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}.$$

Consider any such product term $T = \prod_{i \in [a]} Q_i^{e_i}$, where $Q_i \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$, and since D is a pruned depth four circuit, $a \leq w$. Write $Q_i = Q_i' + Q_i''$, where Q_i' is the sum of all monomials of Q_i wherein the individual degree of every $\mathbf{x}$ variable is at most two and $Q_i'' = Q_i - Q_i'$.

Then,

$$T = \prod_{i\in[a]} (Q'_i + Q''_i)^{e_i} = \prod_{i\in[a]} {Q'_i}^{e_i} + Q'',$$

where Q'' is a polynomial whose every monomial has a $\mathbf{x}$ variable with degree at least three. Thus, $\mathsf{PSP}_{M,k,\ell}(Q'') = 0$ and hence from the sub-additivity of $\mathsf{PSP}_{M,k,\ell}$ we have that

$$\mathsf{PSP}_{M,k,\ell}(T) \leq \mathsf{PSP}_{M,k,\ell}\Big(\prod_{i\in[a]} {Q'_i}^{e_i}\Big).$$

Let $T' = \prod_{i\in[a]} {Q'_i}^{e_i}$. We will now upper bound $\mathsf{PSP}_{M,k,\ell}(T')$. First, we assume without loss of generality that $a = w$ since if $a < w$ then we can multiply with additional sparse polynomials all of which are 1. Next we divide the sparse polynomials into disjoint sets such that each set (except perhaps the last) has size exactly t. Then, we have that

$$T' = P_1 \cdots P_{\lceil \frac{w}{t} \rceil}, \text{ where } P_i = \prod_{j=(i-1)t+1}^{\min(it,w)} {Q'_j}^{e_j}.$$

▷ Claim 8. Let $P = {Q'_1}^{e_1} \cdots {Q'_t}^{e_t}$ be one of the polynomials P_i. For $k \geq 0$, let $P^{(k)} := \prod_{i\in[t]} {Q'_i}^{\max(e_i-k,0)}$. Then, $\partial^k_{\mathbf{x}} P \subseteq \mathbb{F}\text{-span}\{\mathbf{y}_M^{\leq\infty}\mathbf{x}_M^{\leq k(2t\tau-1)}P^{(k)}\}$.

The proof of the claim is straighforward and is given in Appendix B.1.

Proof of Lemma 7. Recall that it is enough to show the following

$$\mathsf{PSP}_{M,k,\ell}(T') \leq m^{O(1)}\binom{m}{\ell+2kt\tau}\binom{\lceil \frac{w}{t} \rceil + k - 1}{k},$$

where $T' = P_1 \cdots P_{\lceil \frac{w}{t} \rceil}$. Let $v = \lceil \frac{w}{t} \rceil$. Now,

$$\begin{aligned}
\partial^k_{\mathbf{x}} T' &\subseteq \mathbb{F}\text{-span}\left\{\partial^{k_1}_{\mathbf{x}} P_1 \cdots \partial^{k_v}_{\mathbf{x}} P_v \;\;:\; k_1 + \cdots + k_v = k\right\} \\
&\subseteq \mathbb{F}\text{-span}\left\{\mathbf{y}_M^{\leq\infty}\mathbf{x}_M^{\leq k_1(2t\tau-1)}P_1^{(k_1)} \cdots \mathbf{y}_M^{\leq\infty}\mathbf{x}_M^{\leq k_v(2t\tau-1)}P_v^{(k_v)} \;\;:\; k_1 + \cdots + k_v = k\right\} \\
&\subseteq \mathbb{F}\text{-span}\left\{\mathbf{y}_M^{\leq\infty}\mathbf{x}_M^{\leq k(2t\tau-1)}P_1^{(k_1)} \cdots P_v^{(k_v)} \;\;:\; k_1 + \cdots + k_v = k\right\},
\end{aligned}$$

where the second to last inclusion follows from Claim 8. Hence,

$$\mathbf{x}_M^{\ell}\partial^k_{\mathbf{x}} T' \subseteq \mathbb{F}\text{-span}\left\{\mathbf{y}_M^{\leq\infty}\mathbf{x}_M^{\leq \ell+k(2t\tau-1)}P_1^{(k_1)} \cdots P_v^{(k_v)} \;\;:\; k_1 + \cdots + k_v = k\right\}.$$

In other words, the space of shifted partials of T' is contained in the $\mathbb{F}$-span of polynomials of the form $Y \cdot X \cdot P_1^{(k_1)} \cdots P_v^{(k_v)}$ where Y is a monomial in $\mathbf{y}_M$ variables and X is a monomial in $\mathbf{x}_M$ variables of degree at most $\ell + k(2t\tau - 1)$. Let us analyse the effect of the operations $\sigma_{\mathbf{y}_M=1}$, $[\cdot]_m$ and $\pi_{\mathbf{x}}$ on one such polynomial. We will assume that $\deg_{\mathbf{y}}(Y) \leq m$ and X is multilinear for otherwise the polynomial will vanish after the operations are applied. Then, we have that,

$$\begin{aligned}
&\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(Y \cdot X \cdot P_1^{(k_1)} \cdots P_v^{(k_v)}\right)\right]_m\right) \\
&= X \cdot \sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\sigma_{\mathrm{Supp}(X)=0}\left(P_1^{(k_1)} \cdots P_v^{(k_v)}\right)\right)\right]_{m-\deg(Y)}\right).
\end{aligned}$$

Thus,

$$\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}T'\right)\right]_m\right) \subseteq \mathbb{F}\text{-span}\Big\{X\cdot\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\sigma_{\mathrm{Supp}(X)=0}\left(P_1^{(k_1)}\cdots P_v^{(k_v)}\right)\right)\right]_i\right): X \text{ is a multilinear monomial in } \mathbf{x}_M \text{ variables}, \deg(X) \text{ is at most } \ell+k(2t\tau-1), 0\le i\le m \text{ and } k_1+\cdots+k_v=k\Big\}.$$

Once we fix i, X, and $k_1,...,k_v$, $X\cdot\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\sigma_{\mathrm{Supp}(X)=0}\left(P_1^{(k_1)}\cdots P_v^{(k_v)}\right)\right)\right]_i\right)$ is fixed. So,

$$\begin{aligned}\mathsf{PSP}_{M,k,\ell}(T') &= \dim\left\langle\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}T'\right)\right]_m\right)\right\rangle\\ &\le (m+1)\cdot\sum_{j=0}^{\ell+k(2t\tau-1)}\binom{m}{j}\binom{v+k-1}{k}\\ &\le (m+1)\cdot(\ell+2kt\tau)\cdot\binom{m}{\ell+2kt\tau}\binom{v+k-1}{k}\\ &= m^{O(1)}\cdot\binom{m}{\ell+2kt\tau}\binom{\lceil\frac{w}{t}\rceil+k-1}{k},\end{aligned}$$

where the second last inequality follows from Claim 9 (proved in Appendix B.1). ◀

▷ Claim 9. Let ℓ, k, t and τ be as defined earlier. Then, $\ell+2kt\tau<\frac{m}{2}$.

3.2 Pruning a depth four circuit

As mentioned before, we will prune the circuit $\mathtt{C}$ computing f_n in two steps - first we will restrict the bottom support of $\mathtt{C}$ and then we will get rid of all heavy gates in it.

3.2.1 Step 1 - Restricting the bottom support of $\mathtt{C}$

If the number of monomials in $\mathtt{C}$ is more than $\left\lfloor\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right\rfloor$, there is nothing to prove. Otherwise, we show that we can get rid of all monomials with support more than $\tau=\lfloor 20\ln m\rfloor$.

▶ **Lemma 10.** *Let the number of monomials in $\mathtt{C}$ be at most $\left\lfloor\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right\rfloor$. Then, for sufficiently large m, there exists $M_1\subseteq[3m], |M_1|=m$ such that all monomials in $\mathtt{C}_1$ obtained from $\mathtt{C}$ by setting variables $\mathbf{x}_{M_1}$ and $\mathbf{y}_{M_1}$ to 0 have support at most τ.*

Proof. We first present a greedy procedure to remove all monomials with support more than τ and then argue that it sets m variables each from $\mathbf{x}$ and $\mathbf{y}$ to 0. In each iteration the procedure picks a pair of variables that appears in a large number of monomials with support more than τ and set them to 0.

Procedure 1 Restriction procedure.

1. $M_1\leftarrow\emptyset, \mathtt{C}_1\leftarrow\mathtt{C}, H :=$ set of all monomials of $\mathtt{C}_1$ with support more than τ.
2. For $j\in[3m]$, $e(j) :=$ number of monomials in H containing x_j or y_j.
3. **while** $H\neq\emptyset$ **do**
4. Pick $j'\in[3m]\backslash M_1$ such that $e(j')\ge e(j)$ for all $j\in[3m]$. Set $x_{j'}=0$ and $y_{j'}=0$. Update $M_1\leftarrow M_1\cup\{j'\}$, $\mathtt{C}_1\leftarrow$ circuit obtained from $\mathtt{C}_1$ by setting $x_{j'}$ and $y_{j'}$ to 0, $H\leftarrow$ set of all monomials of $\mathtt{C}_1$ with support more than τ, and $e(j)\leftarrow$ number of monomials in H containing x_j or y_j.
5. **end while**

It is clear that the bottom support of $\mathtt{C}_1$ obtained after the termination of the procedure is at most τ. Also, since we are only setting variables to 0, it trivially follows that the procedure does not increase the number of gates nor does it increase the fan-in of any gate in the circuit. Claim 11 (proved in Appendix B.2) implies that the procedure terminates in at most m iterations. If it terminates before m iterations, we arbitrarily add an appropriate number of $j \in [3m]$ to M_1 so that $|M_1| = m$ and set x_j and y_j to 0 for all such j. ◀

▷ Claim 11. Procedure 1 terminates in at most m iterations.

▶ Remark. Procedure 1 looks similar to an approximation algorithm for the Set Cover problem [102] (Section 1.2, page-6). This is because the problem of removing monomials with support more than τ can be formulated as an instance of Set Cover with the universe being all such monomials and with a set corresponding to each $j \in [3m]$. For a $j \in [3m]$, the corresponding set will contain all monomials with support more than τ in which at least one of x_j and y_j appears.

3.2.2 Step 2 - Pruning the heavy gates from $\mathtt{C}_1$

A sparse polynomial Q in $\mathtt{C}_1$ is said to be *light* if its fan-in is at most $\frac{m}{(\ln m)^2}$, i.e., at most $\frac{m}{(\ln m)^2}$ many non-zero monomials are present in Q. If the sum of fan-ins of all the light sparse polynomials is $\Omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$ then there is nothing to prove. So assume that the sum of fan-ins of all the light sparse polynomials is $O\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$. Recall that a product term is called heavy if it has at least $w = \left\lfloor \frac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3} \right\rfloor$ many distinct sparse polynomials connected to it (where λ_0 is a large enough constant fixed in Appendix C). Observe that one of the following cases is true:

1. There is a heavy gate in $\mathtt{C}_1$, that is connected to at most $\frac{m \cdot d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3}$ light sparse polynomials.
2. Every heavy gate is connected to at least $\frac{m \cdot d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3}$ light sparse polynomials.

Case 1 clearly implies a lower bound of $\Omega\big(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\big)$. Else, we prove the following lemma.

▶ **Lemma 12.** *Let $\mathtt{C}_1$ be the circuit (obtained from Lemma 10) having at most m heavy gates such that every heavy gate is connected to at least $\frac{m d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3}$ light sparse polynomials, and the sum of fan-ins of all the light sparse polynomials is at most $\frac{m^2 \cdot d_{\mathbf{x}}}{160 \cdot \lambda_0 \cdot (\ln m)^5}$. Then, there exist $M_2 \subseteq [3m] \setminus M_1$, $|M_2| = m$ and $\alpha_l, \beta_l \in \mathbb{F}$ for $l \in M_2$, such that setting $x_l = \alpha_l, y_l = \beta_l$ for all $l \in M_2$ removes all heavy gates from $\mathtt{C}_1$.*

Proof. We first present the pruning procedure and then argue its correctness. For any light sparse polynomial Q_j in $\mathtt{C}_1$, let b_j and c_j be equal to the fan-in of Q_j and the number of distinct heavy gates connected to Q_j in $\mathtt{C}_1$ respectively. As Q_j is a light sparse polynomial, $b_j \leq \frac{m}{(\ln m)^2}$. In this procedure, while all the heavy gates do not disappear from $\mathtt{C}_1$, we pick a sparse polynomial whose ratio of the number of distinct heavy gates connected to it and fan-in is maximum and set that to zero by evaluating it to one of its roots in $\mathbb{F}$. This can be done as $\mathbb{F}$ is algebraically closed. The restricted support of the monomials in $\mathtt{C}_1$ ensure that at the end of this procedure, only a small fraction of the variables are set.

Procedure 2 Pruning heavy gates from $\mathtt{C}_1$.

1. Set $i = 1$ and $M_2 = \emptyset$. Let $s_1 \leq m$ be the number of heavy gates in $\mathtt{C}_1$. Choose a non-constant light sparse polynomial Q_1 from $\mathtt{C}_1$ such that the ratio $\frac{c_1}{b_1}$ is maximum and add the indices of the variables appearing in Q_1 to M_2. As $b_1 \leq \frac{m}{(\ln m)^2}$, we have $\tau \cdot b_1 \leq m$.
2. Make Q_1 equal to zero by setting at most $\tau \cdot b_1$ many variables appearing in Q_1 to field constants. By doing so, at least c_1 many heavy gates vanish from $\mathtt{C}_1$.
3. **while** $(\tau(b_1 + \cdots + b_i) \leq m)$ **do**
4. Increment i by 1.
5. Let s_i be the number of heavy gates in the current circuit $\mathtt{C}_1$ obtained after the $(i-1)$-th iteration. Clearly, $s_i \leq s_{i-1} - c_{i-1}$. If $s_i = 0$ then exit the loop.
6. Otherwise, choose a non-constant light sparse polynomial Q_i from $\mathtt{C}_1$ having the maximum value of $\frac{c_i}{b_i}$ in $\mathtt{C}_1$ and add the indices of the variables appearing in Q_i to M_2.
7. Make Q_i equal to zero by setting at most $\tau \cdot b_i$ many variables appearing in Q_i to field constants. By doing so, at least c_i many heavy gates vanish from $\mathtt{C}_1$.
8. **end while**

▷ Claim 13. Let $\overline{M}_1 = [3m] \backslash M_1$. Procedure 2 sets at most m many variables in $\mathbf{x}_{\overline{M}_1} \cup \mathbf{y}_{\overline{M}_1}$ to field constants and removes all the heavy gates from $\mathtt{C}_1$.

The above claim is proved in Appendix B.3. The claim implies that $|M_2| \leq m$. If $|M_2| < m$, add appropriate number of elements from $[3m] \setminus M_1 \cup M_2$ to M_2 arbitrarily so that $|M_2| = m$. For every $l \in M_2$, if x_l or y_l is not set to a field constant then set $x_l = 0$ or $y_l = 0$ respectively. Clearly, we end up setting exactly m variables each from $\mathbf{x}_{\overline{M}_1}$ and $\mathbf{y}_{\overline{M}_1}$. ◀

▶ Remark. Procedure 2 resembles an approximation algorithm for the Weighted Set Cover problem [99] (Section 2.1, page-16). This is no coincidence as the problem of removing heavy gates can be formulated as an instance of Weighted Set Cover with the universe being all heavy gates and with a set corresponding to every sparse polynomial Q. The set corresponding to Q contains all heavy gates connected to Q and has a cost equal to the number of monomials feeding into Q.

4 An explicit polynomial family with high measure

We now describe the family $\{f_n\}_{n \geq 1}$, whose n-th member f_n is a polynomial in variables $\mathbf{x} = \{x_1, ..., x_{3m}\}$ and $\mathbf{y} = \{y_1, ..., y_{3m}\}$, where $m \in \left[\frac{n}{2}, 2n\right]$ will be fixed later.

$$f_n := \sum_{S \subseteq [3m], |S| = m} \left(\prod_{i \in S} y_i \right) \cdot \mathsf{NW}_r(\mathbf{x}_S),$$

where NW_r is a variant of the Nisan-Wigderson design polynomial (introduced in [52]), the construction of which is described later and r is a parameter fixed in this construction. Note that $\{f_n\}_{n \geq 1}$ is in VNP. Given a monomial, in order to find its coefficient in f_n, we first check if the monomial is multilinear and of degree m in $\mathbf{y}$ variables. If it is so and S is the set of the indices of the m many $\mathbf{y}$ variables in the monomial then simply return the coefficient of the part of the monomial in $\mathbf{x}$ variables in $\mathsf{NW}_r(\mathbf{x}_S)$ – this can be done as the Nisan-Wigderson polynomial family is in VNP.

Let M_1 and M_2 be as in Section 3 and $M = [3m] \setminus (M_1 \cup M_2)$. Let f_1 be the polynomial computed by the pruned circuit $\mathtt{D}$, which is obtained from $f = f_n$ by setting the variables $\mathbf{x}_{\overline{M}}$ and $\mathbf{y}_{\overline{M}}$ to field constants as in Section 3.2. Let us now see how $\mathsf{PSP}_{M,k,\ell}(f_1)$ is related to $\dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\,\mathsf{NW}_r\right)\right\rangle$.

▶ **Lemma 14.** *Let f_1 be as defined above. Then,* $\mathsf{PSP}_{M,k,\ell}(f_1) = \dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\,\mathsf{NW}_r(\mathbf{x}_M)\right)\right\rangle$.

Proof. The proof follows easily from the following two observations:

1. The two operations in $\mathbf{y}$ variables and the three operations in $\mathbf{x}$ variables (in the definition of $\mathsf{PSP}_{M,k,\ell}$) commute. That is, we have $\sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k} f_1)\right]_m\right) = \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\left(\sigma_{\mathbf{y}_M=1}\left([f_1]_m\right)\right)\right)$.
2. $f_1 = \left(\prod_{i\in M} y_i\right)\cdot \mathsf{NW}_r(\mathbf{x}_M) + f'$, where $f' \in \mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$ and $\deg_{\mathbf{y}}(f') < m$.

From these observations we have that

$$\begin{aligned}
\mathsf{PSP}_{M,k,\ell}(f_1) &= \dim\left\langle \sigma_{\mathbf{y}_M=1}\left(\left[\pi_{\mathbf{x}}(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k} f_1)\right]_m\right)\right\rangle \\
&= \dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\left(\sigma_{\mathbf{y}_M=1}\left(\left[\left(\prod_{i\in M} y_i\right)\cdot \mathsf{NW}_r(\mathbf{x}_M) + f'\right]_m\right)\right)\right)\right\rangle \\
&= \dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\,\mathsf{NW}_r(\mathbf{x}_M)\right)\right\rangle.
\end{aligned}$$

The last equality follows from the fact that $\left(\sigma_{\mathbf{y}_M=1}\left(\left[\left(\prod_{i\in M} y_i\right)\cdot \mathsf{NW}_r(\mathbf{x}_M) + f'\right]_m\right)\right) = \mathsf{NW}_r(\mathbf{x}_M)$. ◀

Construction of NW_r. Let $d_{\mathbf{x}} = \left\lfloor \frac{\sqrt{n}}{\ln n}\right\rfloor$. Pick an α such that $d_{\mathbf{x}}\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil \le n \le 2d_{\mathbf{x}}\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil$; this forces α to be $\Theta(\frac{\ln\ln n}{\ln n})$. Let q be a prime number between $\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil$ and $2\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil$ – such a prime exists [26] – and let $m = d_{\mathbf{x}}q$. Thus, $d_{\mathbf{x}}\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil \le m \le 2d_{\mathbf{x}}\left\lceil d_{\mathbf{x}}^{1+\alpha}\right\rceil$ and hence $\frac{n}{2} \le m \le 2n$; moreover, it can be easily verified that $d_{\mathbf{x}} \in \left[\frac{\sqrt{m}}{2\sqrt{2}\cdot\ln m}, \frac{2\sqrt{2}\cdot\sqrt{m}}{\ln m}\right]$; both being as required in Section 3. Also notice that this means $q = \Theta(\sqrt{n}\ln n)$. Let $\beta = \frac{1}{\ln m}$ and $r = \left\lfloor \frac{\alpha+\beta}{2(1+\alpha)} d_{\mathbf{x}}\right\rfloor - 1$, $\mathbf{u} = (u_{1,1}, ..., u_{1,q}, ..., u_{d_{\mathbf{x}},1}, ..., u_{d_{\mathbf{x}},q})$ and define

$$\mathsf{NW}_r(\mathbf{u}) := \sum_{h(z)\in\mathbb{F}_q[z],\ \deg(h)\le r} u_{1,h(1)}\cdots u_{d_{\mathbf{x}},h(d_{\mathbf{x}})}.$$

A lower bound on $\dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\,\mathsf{NW}_r\right)\right\rangle$ was proved in [51, 85]. Since their analysis continues to hold for our choice of parameters – which only slightly differ from the parameters in [85] – we omit the proof of the following theorem. Moreover, while they prove this lower bound over fields of characteristic zero, the same proof also works if the characteristic is greater than $q^{(r+1)\cdot\min\left\{\binom{m}{k}\binom{m}{\ell},\binom{m}{\ell-d_{\mathbf{x}}-k}\right\}}$.

▶ **Theorem 15** (Lemma 5.2 of [51], Lemma 4.1 of [85])**.**

$$\dim\left\langle \pi_{\mathbf{x}}\left(\mathbf{x}_M^{\ell}\partial_{\mathbf{x}}^{k}\,\mathsf{NW}_r(\mathbf{x}_M)\right)\right\rangle \ge \frac{1}{m^{O(1)}}\min\left\{\frac{1}{4^k}\cdot\binom{m}{\ell}\binom{m}{k}, \binom{m}{\ell+d_{\mathbf{x}}-k}\right\}.$$

Hence, from Lemma 14 and Theorem 15 we get

▶ **Lemma 16.** $\mathsf{PSP}_{M,k,\ell}(f_1) \ge \frac{1}{m^{O(1)}}\min\left\{\frac{1}{4^k}\cdot\binom{m}{\ell}\binom{m}{k}, \binom{m}{\ell+d_{\mathbf{x}}-k}\right\}$.

5 Proof of Theorem 1

As before, let $\mathtt{C}$ be a depth four circuit computing the polynomial f_n. Before proving the theorem, let us first justify the assumption that $\mathbb{F}$ is an algebraically closed field that we made in Section 3. Suppose not. Then, let $\overline{\mathbb{F}}$ be its algebraic closure. Since $\mathtt{C}$ is also a circuit over $\overline{\mathbb{F}}$ and f_n a polynomial over $\overline{\mathbb{F}}$, we can make all arguments assuming the underlying field to be $\overline{\mathbb{F}}$. Since the size of a circuit does not depend on the underlying field, the lower bound so obtained will continue to hold when we treat $\mathtt{C}$ as a circuit over $\mathbb{F}$.

First we will prove a lower bound on the number of wires of $\mathtt{C}$. If the number of monomials in $\mathtt{C}$ is $\left\lfloor \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \right\rfloor$ then there is nothing to prove. Otherwise from Lemma 10, we can obtain a circuit $\mathtt{C}_1$ such that the support of all the monomials of $\mathtt{C}_1$ is at most $\tau = \lfloor 20 \ln m \rfloor$, the number of gates in $\mathtt{C}_1$ is at most the number of gates in $\mathtt{C}$ and the fan-in of each gate in $\mathtt{C}_1$ is upper bounded by the fan-in of the corresponding gate in $\mathtt{C}$. Then, if $\mathtt{C}_1$ does not satisfy the hypothesis of Lemma 12, the size of $\mathtt{C}_1$ and hence the size of $\mathtt{C}$ is at least $\Omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$. Otherwise, we can obtain a pruned circuit $\mathtt{D}$ such that the top fan-in and the bottom support of $\mathtt{D}$ are upper bounded by the top fan-in and bottom support of $\mathtt{C}_1$ and so proving a lower bound on the top fan-in of $\mathtt{D}$ would suffice.

As $\mathtt{D}$ computes f_1 (defined in Section 4), $\mathsf{PSP}_{M,k,\ell}(\mathtt{D}) = \mathsf{PSP}_{M,k,\ell}(f_1)$. Lemma 7 and 16 imply

$$s \geq \frac{\frac{1}{m^{O(1)}} \min\left\{ \frac{1}{4^k} \cdot \binom{m}{\ell}\binom{m}{k}, \binom{m}{\ell + d_{\mathbf{x}} - k} \right\}}{m^{O(1)} \cdot \binom{m}{\ell + 2kt\tau} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}}, \tag{1}$$

and the required lower bound follows from the next claim, which is proved in Appendix C.

▷ Claim 17. The top fan-in s of $\mathtt{D}$ is $\omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$.

Now let us prove the lower bound on the number of gates. Notice that if the circuit $\mathtt{C}$ computing f has a heavy gate as defined in Section 3 then we are done. So assume that it does not have any heavy gates. Now, if the number of monomials in $\mathtt{C}$ is $\left\lfloor \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \right\rfloor$ then there is nothing to prove. Otherwise from Lemma 10, we can obtain a circuit $\mathtt{C}_1$ such that the support of all the monomials of $\mathtt{C}_1$ is at most $\tau = \lfloor 20 \ln m \rfloor$, the number of gates in $\mathtt{C}_1$ is at most the number of gates in $\mathtt{C}$ and the fan-in of each gate in $\mathtt{C}_1$ is upper bounded by the fan-in of the corresponding gate in $\mathtt{C}$. Obtain a circuit $\mathtt{D}$ from $\mathtt{C}_1$ by picking a set $M_2 \subseteq [3m] \setminus M_1$ (where M_1 is as in Lemma 10), $|M_2| = m$ arbitrarily and setting variables in $\mathbf{x}_{M_2}$ and $\mathbf{y}_{M_2}$ to 0 (notice that the top fan-in and bottom support of $\mathtt{D}$ are upper bounded by the top fan-in and bottom support of $\mathtt{C}_1$). Now $\mathtt{D}$ computes f_1 (where f_1 is as defined in Section 4) and just as it was done in the preceding paragraph, we can show that the top fan-in of $\mathtt{D}$ is $\omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$. However, we only get an $\Omega\left(\frac{m d_{\mathbf{x}}}{(\ln m)^3}\right)$ lower bound on the number of gates since the definition of a heavy gate is the bottleneck.

6 Conclusion

We conclude by stating a few questions/problems, some of which may not be very hard to answer/solve.

1. **Improving the depth four lower bound.** An affirmative answer to any of the following questions will strengthen or improve the lower bound shown in this work.
   - Can we prove an $\widetilde{\Omega}(n^{2.5})$ lower bound on the number of gates of depth four circuits? The almost cubic lower bound in [53] is on the number of gates of depth three circuits.

- Can we prove an $\widetilde{\Omega}(n^{2.5})$ lower bound for depth four circuits computing $\mathsf{IMM}_{2,n}$[8]? The same question may also be asked with regard to the $\widetilde{\Omega}(n^3)$ lower bound for depth three circuits.
- Can we prove an $\widetilde{\Omega}(n^3)$ lower bound for depth four circuits? The loss of a $\sqrt{n}$ factor (in Theorem 1) in comparison to the almost cubic lower bound for depth three circuits [53] is due to the use of the projected shifted partials measure in place of the shifted partials measure.

2. **A lower bound for depth five circuits.** As mentioned in Appendix A, an $\Omega(n^{1.8+\epsilon})$ lower bound on the number of gates of a depth five circuit computing $\mathsf{IMM}_{2,n}$ implies a super-cubic lower bound for depth three circuits. It is also interesting to note that the hard polynomial used in [12, 103] to prove an $\widetilde{\Omega}(n^3)$ lower bound for depth three circuits is computable by a poly(n)-size depth five circuit. As a natural next step, we pose the following problem:
   - Prove a super-quadratic lower bound for depth five circuits.
3. **Super-linear lower bound for constant depth circuits computing $\mathsf{IMM}_{2,n}$.** Recall that lower bounds of $\Omega(\Delta n^{1+\frac{1}{\Delta}})$ and roughly $\Omega(n^{1+\frac{1}{\Delta}})$ are shown for depth-Δ circuits in [86] and [73] respectively. We have argued in Appendix A that the same lower bound for depth-Δ circuits computing $\mathsf{IMM}_{2,n}$ would give a super-polynomial lower bound for constant depth circuits. It is thus natural to ask:
   - Can we prove a super-linear lower bound for constant depth circuits computing $\mathsf{IMM}_{2,n}$?
4. **Hardness magnification for commutative circuits.** It was shown in [18] that a sufficiently large super-linear lower bound for non-commutative circuits implies an arbitrarily large polynomial lower bound for general non-commutative circuits. It would be highly interesting to show a similar hardness magnification result for commutative circuits.

References

1 Manindra Agrawal, Eric Allender, and Samir Datta. On TC^0, AC^0, and Arithmetic Circuits. *J. Comput. Syst. Sci.*, 60(2):395–421, 2000. Conference version appeared in the proceedings of CCC 1997.

2 Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian Hits Circuits: Hitting Sets, Lower Bounds for Depth-D Occur-k Formulas and Depth-3 Transcendence Degree-k Circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016. Conference version appeared in the proceedings of STOC 2012.

3 Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75. IEEE Computer Society, 2008.

4 Miklós Ajtai. Σ^1_1-Formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

5 Boris Alexeev, Michael A. Forbes, and Jacob Tsimerman. Tensor rank: Some lower and upper bounds. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 283–291. IEEE Computer Society, 2011.

6 Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010. Conference version appeared in the proceedings of CCC 2008.

7 Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[8] The reason why lower bounds for $\mathsf{IMM}_{2,n}$ are interesting is mentioned in the discussion on hardness magnification in Appendix A.

8 Noga Alon, Mrinal Kumar, and Ben Lee Volk. Unbalancing Sets and an Almost Quadratic Lower Bound for Syntactically Multilinear Arithmetic Circuits. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 11:1–11:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

9 Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity Testing and Lower Bounds for Read-k Oblivious Algebraic Branching Programs. *TOCT*, 10(1):3:1–3:30, 2018. Conference version appeared in the proceedings of CCC 2016.

10 A. E. Andreev. On a method for obtaining more than quadratic effective lower bounds for thecomplexity of π-schemes. *Moscow Univ. Math. Bull.*, 42:63–66, 1987.

11 Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. *Computational Complexity*, 27(1):1–29, 2018. Conference version appeared in the proceedings of STOC 2010.

12 Nikhil Balaji, Nutan Limaye, and Srikanth Srinivasan. An almost cubic lower bound for $\Sigma\Pi\Sigma$ circuits computing a polynomial in VP. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:143, 2016. URL: `http://eccc.hpi-web.de/report/2016/143`.

13 Walter Baur and Volker Strassen. The Complexity of Partial Derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983.

14 Norbert Blum. A Boolean Function Requiring 3n Network Size. *Theor. Comput. Sci.*, 28:337–345, 1984. `doi:10.1016/0304-3975(83)90029-4`.

15 Allan Borodin, Alexander A. Razborov, and Roman Smolensky. On Lower Bounds for Read-K-Times Branching Programs. *Computational Complexity*, 3:1–18, 1993.

16 Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and computation in mathematics*. Springer, 2000.

17 Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.

18 Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness Amplification for Non-Commutative Arithmetic Circuits. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

19 Prerona Chatterjee, Mrinal Kumar, Adrian She, and Ben Lee Volk. A Quadratic Lower Bound for Algebraic Branching Programs. *Electronic Colloquium on Computational Complexity (ECCC)*, page 170, 2019. URL: `https://eccc.weizmann.ac.il/report/2019/170`.

20 Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 34–41. ACM, 2019.

21 Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial Derivatives in Arithmetic Complexity and Beyond. *Foundations and Trends in Theoretical Computer Science*, 6(1-2):1–138, 2011.

22 Suryajith Chillara, Christian Engels, Nutan Limaye, and Srikanth Srinivasan. A Near-Optimal Depth-Hierarchy Theorem for Small-Depth Multilinear Circuits. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 934–945. IEEE Computer Society, 2018.

23 Suryajith Chillara, Nutan Limaye, and Srikanth Srinivasan. Small-Depth Multilinear Formula Lower Bounds for Iterated Matrix Multiplication with Applications. *SIAM J. Comput.*, 48(1):70–92, 2019. Conference version appeared in the proceedings of STOC 2001.

24 Danny Dolev, Cynthia Dwork, Nicholas Pippenger, and Avi Wigderson. Superconcentrators, Generalizers and Generalized Connectors with Limited Depth (Preliminary Version). In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 42–51. ACM, 1983.

25 Zeev Dvir, Guillaume Malod, Sylvain Perifel, and Amir Yehudayoff. Separating multilinear branching programs and formulas. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 615–624. ACM, 2012.

26 Paul Erdős. Beweis eines Satzes von Tschebyschef. *Acta Litt. Sci. Szeged*, 5:194–198, January 1932.

27 Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A Better-Than-3n Lower Bound for the Circuit Complexity of an Explicit Function. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 89–98. IEEE Computer Society, 2016.

28 Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower Bounds for Depth-4 Formulas Computing Iterated Matrix Multiplication. *SIAM J. Comput.*, 44(5):1173–1201, 2015. Conference version appeared in the proceedings of STOC 2014.

29 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. Conference version appeared in the proceedings of FOCS 1981.

30 Michelangelo Grigni and Michael Sipser. Monotone Separation of Logarithmic Space from Logarithmic Depth. *J. Comput. Syst. Sci.*, 50(3):433–437, 1995. Conference version appeared in the proceedings of CCC 1991.

31 Dima Grigoriev and Marek Karpinski. An Exponential Lower Bound for Depth 3 Arithmetic Circuits. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 577–582, 1998.

32 Dima Grigoriev and Alexander A. Razborov. Exponential Lower Bounds for Depth 3 Arithmetic Circuits in Algebras of Functions over Finite Fields. *Appl. Algebra Eng. Commun. Comput.*, 10(6):465–487, 2000. Conference version appeared in the proceedings of FOCS 1998.

33 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *J. ACM*, 61(6):33:1–33:16, 2014. Conference version appeared in the proceedings of CCC 2013.

34 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. Conference version appeared in the proceedings of FOCS 2013.

35 Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986.

36 Johan Håstad. The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. Conference version appeared in the proceedings of FOCS 1993.

37 William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002. Conference version appeared in the proceedings of CCC 2001.

38 Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Non-commutative circuits and the sum-of-squares problem. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 667–676. ACM, 2010.

39 Pavel Hrubes and Amir Yehudayoff. Arithmetic complexity in ring extensions. *Theory of Computing*, 7(1):119–129, 2011.

40 Pavel Hrubes and Amir Yehudayoff. On Isoperimetric Profiles and Computational Complexity. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 89:1–89:12, 2016.

41 Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-Depth Tradeoffs for Threshold Circuits. *SIAM J. Comput.*, 26(3):693–707, 1997. Conference version appeared in the proceedings of STOC 1993.

42 Kazuo Iwama and Hiroki Morizumi. An Explicit Lower Bound of 5n - o(n) for Boolean Circuits. In Krzysztof Diks and Wojciech Rytter, editors, *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, volume 2420 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2002.

43 Mark Jerrum and Marc Snir. Some Exact Complexity Results for Straight-Line Computations over Semirings. *J. ACM*, 29(3):874–897, 1982.

44 K. Kalorkoti. A Lower Bound for the Formula Size of Rational Functions. *SIAM J. Comput.*, 14(3):678–687, 1985.

45 Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 633–643, 2016.

46 Mauricio Karchmer and Avi Wigderson. Monotone Circuits for Connectivity Require Super-Logarithmic Depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990. Conference version appeared in the proceedings of STOC 1988.

47 Mauricio Karchmer and Avi Wigderson. On Span Programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111. IEEE Computer Society, 1993.

48 Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012. URL: `http://eccc.hpi-web.de/report/2012/081`.

49 Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Formulas. *SIAM J. Comput.*, 46(1):307–335, 2017. Conference version appeared in the proceedings of FOCS 2014.

50 Neeraj Kayal and Chandan Saha. Lower Bounds for Sums of Products of Low arity Polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:73, 2015. URL: `http://eccc.hpi-web.de/report/2015/073`.

51 Neeraj Kayal and Chandan Saha. Lower Bounds for Depth-Three Arithmetic Circuits with small bottom fanin. *Computational Complexity*, 25(2):419–454, 2016. Conference version appeared in the proceedings of CCC 2015.

52 Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 146–153, 2014.

53 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. An Almost Cubic Lower Bound for Depth Three Arithmetic Circuits. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 33:1–33:15, 2016.

54 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.

55 Alexander S. Kulikov, Olga Melanich, and Ivan Mihajlin. A 5n - o(n) Lower Bound on the Circuit Size over U 2 of a Linear Boolean Function. In S. Barry Cooper, Anuj Dawar, and Benedikt Löwe, editors, *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, volume 7318 of *Lecture Notes in Computer Science*, pages 432–439. Springer, 2012.

56 Mrinal Kumar and Shubhangi Saraf. Superpolynomial lower bounds for general homogeneous depth 4 arithmetic circuits. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 751–762, 2014.

57 Mrinal Kumar and Shubhangi Saraf. Sums of Products of Polynomials in Few Variables: Lower Bounds and Polynomial Identity Testing. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 35:1–35:29, 2016.

58 Mrinal Kumar and Shubhangi Saraf. On the Power of Homogeneous Depth 4 Arithmetic Circuits. *SIAM J. Comput.*, 46(1):336–387, 2017. Conference version appeared in the proceedings of FOCS 2014.

59 Mrinal Kumar and Ben Lee Volk. Lower Bounds for Matrix Factorization. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:47, 2019.

60 Oded Lachish and Ran Raz. Explicit lower bound of *4.5n - o(n)* for boolean circuits. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 399–408. ACM, 2001.

61 Shachar Lovett. Computing polynomials with few multiplications. *Theory of Computing*, 7(1):185–188, 2011.

62 Jacques Morgenstern. Note on a Lower Bound on the Linear Complexity of the Fast Fourier Transform. *J. ACM*, 20(2):305–306, 1973.

63 Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901, 2018.

64 E. I. Nechiporuk. On a Boolean function. *Doklady of the Academy of Sciences of the USSR*, 164(4):765–766, 1966.

65 Noam Nisan. Lower Bounds for Non-Commutative Computation (Extended Abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418. ACM, 1991.

66 Noam Nisan and Avi Wigderson. Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity*, 6(3):217–234, 1997. Conference version appeared in the proceedings of FOCS 1995.

67 Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 65–76. IEEE Computer Society, 2018.

68 Pavel Pudlák. Communication in Bounded Depth Circuits. *Combinatorica*, 14(2):203–216, 1994.

69 Pavel Pudlák. A note on the use of determinant for proving lower bounds on the size of linear circuits. *Inf. Process. Lett.*, 74(5-6):197–201, 2000.

70 Ran Raz. On the Complexity of Matrix Product. *SIAM J. Comput.*, 32(5):1356–1369, 2003. Conference version appeared in the proceedings of STOC 2002.

71 Ran Raz. Separation of Multilinear Circuit and Formula Size. *Theory of Computing*, 2(6):121–135, 2006. Conference version appeared in the proceedings of FOCS 2004.

72 Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2):8:1–8:17, 2009. Conference version appeared in the proceedings of STOC 2004.

73 Ran Raz. Elusive Functions and Lower Bounds for Arithmetic Circuits. *Theory of Computing*, 6(1):135–177, 2010. Conference version appeared in the proceedings of STOC 2008.

74 Ran Raz. Tensor-Rank and Lower Bounds for Arithmetic Formulas. *J. ACM*, 60(6):40:1–40:15, 2013. Conference version appeared in the proceedings of STOC 2010.

75 Ran Raz and Pierre McKenzie. Separation of the Monotone NC Hierarchy. *Combinatorica*, 19(3):403–435, 1999. Conference version appeared in the proceedings of FOCS 1997.

76 Ran Raz and Amir Shpilka. Lower Bounds for Matrix Product in Bounded Depth Circuits with Arbitrary Gates. *SIAM J. Comput.*, 32(2):488–513, 2003. Conference version appeared in the proceedings of STOC 2001.

77 Ran Raz, Amir Shpilka, and Amir Yehudayoff. A Lower Bound for the Size of Syntactically Multilinear Arithmetic Circuits. *SIAM J. Comput.*, 38(4):1624–1647, 2008. Conference version appeared in the proceedings of FOCS 2007.

78 Ran Raz and Amir Yehudayoff. Balancing Syntactically Multilinear Arithmetic Circuits. *Computational Complexity*, 17(4):515–535, 2008.

79 Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009. Conference version appeared in the proceedings of CCC 2008.

80 A. A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical notes of the Academy of Sciences of the USSR*, 37(6):485–493, June 1985.

81 Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady*, 31:354–357, 1985.

82 John H. Reif and Stephen R. Tate. On Threshold Circuits and Polynomial Computation. *SIAM J. Comput.*, 21(5):896–908, 1992.

83 Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential Lower Bounds for Monotone Span Programs. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415. IEEE Computer Society, 2016.

84 Eli Shamir and Marc Snir. Lower bounds on the number of multiplications and the number of additions in monotone computations. Technical report, IBM RC 6757, 1977.

85 Abhijat Sharma. An Improved Lower Bound for Depth Four Arithmetic Circuits. Master's thesis, Indian Institute of Science, Bangalore, India, 2017.

86 Victor Shoup and Roman Smolensky. Lower Bounds for Polynomial Evaluation and Interpolation Problems. *Computational Complexity*, 6(4):301–311, 1997. Conference version appeared in the proceedings of FOCS 1991.

87 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. Conference version appeared in the proceedings of CCC 1999.

88 Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

89 Srikanth Srinivasan. Strongly Exponential Separation Between Monotone VP and Monotone VNP. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:32, 2019.

90 Volker Strassen. Die berechnungskomplexiät von elementarysymmetrischen funktionen und von iterpolationskoeffizienten. *Numerische Mathematik*, 20:238–251, 1973.

91 Volker Strassen. Vermeidung von divisionen. *The Journal für die Reine und Angewandte Mathematik*, 264:182–202, 1973.

92 Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

93 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Conference version appeared in the proceedings of MFCS 2013.

94 Leslie G. Valiant. On Non-linear Lower Bounds in Computational Complexity. In William C. Rounds, Nancy Martin, Jack W. Carlyle, and Michael A. Harrison, editors, *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, May 5-7, 1975, Albuquerque, New Mexico, USA*, pages 45–53. ACM, 1975.

95 Leslie G. Valiant. Graph-Theoretic Arguments in Low-Level Complexity. In *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*, pages 162–176, 1977.

96 Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979.

97 Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980. Conference version appeared in the proceedings of STOC 1979.

98 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

99 Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001. URL: `http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-65367-7`.

100 Joachim von zur Gathen and Gadiel Seroussi. Boolean Circuits Versus Arithmetic Circuits. *Inf. Comput.*, 91(1):142–154, 1991.

101 Ryan Williams. Nonuniform ACC Circuit Lower Bounds. *J. ACM*, 61(1):2:1–2:32, 2014. Conference version appeared in the proceedings of CCC 2011.

102 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: `http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE`.

103 Morris Yau. Almost cubic bound for depth three circuits in VP. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:187, 2016. URL: `http://eccc.hpi-web.de/report/2016/187`.

104 Amir Yehudayoff. Separating monotone VP and VNP. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 425–429. ACM, 2019.

A Known lower bounds

We give a brief account of known lower bounds for some of the important classes of arithmetic circuits by drawing parallels with similar results from the Boolean circuit literature. The reader may refer to the surveys [21, 88] or the book [17] for more details on arithmetic circuit lower bounds. We also state a few hardness magnification[9] or amplification results to show that proving seemingly modest lower bounds can be quite interesting and challenging even for constant depth circuits, provided the polynomials being computed have "low" complexity.

General circuits. The best known lower bound for general arithmetic circuits is $\Omega(n \log d)$, which was obtained nearly four decades ago for circuits computing the power symmetric polynomial $x_1^d + x_2^d + \ldots + x_n^d$ [13, 90][10]. Recently, [19] has shown an $\Omega(n^2)$ lower bound for "layered" algebraic branching programs (ABPs) computing the same polynomial. An $\Omega(n^2)$ lower bound for formulas computing the polynomial $\sum_{i,j\in[n]} x_i^j y_j$ was shown in [44]. The situation is similar for Boolean circuits. For circuits over the DeMorgan basis and over the full binary basis, the lower bounds $5n - o(n)$ [42, 55, 60] and $(3 + \frac{1}{86})n - o(n)$ [14, 27] respectively, are the best till date. For Boolean formulas, an $\widetilde{\Omega}(n^2)$ lower bound is known over the full binary basis [64], and an $\widetilde{\Omega}(n^3)$ lower bound is known over the DeMorgan basis [10, 36].

Monotone circuits. These are arithmetic circuits over $\mathbb{Q}$ or $\mathbb{R}$ that disallow negation. A lot more is known about this class of circuits. A near optimal $2^{\Omega(n)}$ lower bound on the monotone circuit complexity of the $n \times n$ permanent was shown in [43]. In fact, [97] showed an exponential separation between monotone and general circuits computing the perfect matching polynomial of a certain planar graph. Optimal separations are also known between monotone ABPs and monotone circuits [40] and between monotone formulas and monotone ABPs [84]. Recently, [104] gave an exponential separation between monotone VP and monotone VNP which was made stronger in [89]. One of the success stories on Boolean

[9] Borrowing terminology from [67].

[10] The bound also holds for the d-th elementary symmetric polynomial in n variables.

circuit complexity in the 80s is the exponential lower bound for monotone circuits computing the clique function [7, 81]. Building on these results, [92] showed an exponential separation between monotone and general Boolean circuits[11]. An exponential separation between monotone switching networks[12] and monotone circuits was given in [83][13]. A separation between monotone formulas and monotone switching networks follows from the work of [30]. Yet another interesting result is the separation of monotone-NC^i from monotone-NC^{i+1} for every $i \leq 1$ [46, 75].

Non-commutative and multilinear circuits. A non-commutative circuit computes a polynomial in non-commuting variables. This model has more structure than circuits over commuting variables and so one may hope that it is "easier" to prove lower bounds for non-commutative circuits[14]. The seminal work of [65] showed an exponential separation between non-commutative ABPs and non-commutative circuits. But, proving a super-polynomial lower bound for general non-commutative circuits[15] and showing a separation between non-commutative formulas and non-commutative ABPs continue to remain two important open problems. The techniques used to prove lower bounds for non-commutative circuits is closely related to that used to prove lower bounds for multilinear circuits. In a (syntactically) multilinear circuit, the sets of variables occurring in the subcircuits rooted at the children of a product gate are pairwise disjoint. It is an interesting model of computation as most natural families of polynomials, like the permanent, determinant, iterated matrix multiplication, elementary symmetric polynomials, design polynomials etc., are multilinear. Building on [77], an $\Omega(\frac{n^2}{(\log n)^2})$ lower bound for multilinear circuits has been recently shown in [8]. Prior to this, the breakthrough work of [72] culminated in an optimal separation between multilinear formulas and multilinear ABPs [25, 71, 78]. We do not know of any super-polynomial lower bound for multilinear ABPs.

Bounded coefficient circuits. In a bounded coefficient circuit over $\mathbb{C}$, we forbid any multiplication by a field element having absolute value larger than 1. An $\Omega(n \log n)$ lower bound for bounded coefficient circuits computing the Discrete Fourier Transform of a vector $(x_1, \ldots, x_n)$ was shown in [62]. Later, [70] gave an $\Omega(n \log n)$ lower bound for the same class of circuits computing the product of two $\sqrt{n} \times \sqrt{n}$ matrices.

Read-k circuits. A read-once oblivious algebraic branching program (ROABP) is a layered ABP in which every layer is indexed by a variable, and every variable indexes exactly one layer. The edges of a layer are labeled by univariate polynomials in the variable indexing the layer. In a certain sense, the ROABP model generalizes quite a few other interesting arithmetic circuit models, especially tensors. An exponential lower bound for ROABPs follows from the technique introduced in the work of [65]. A read-k oblivious ABP is defined similarly, with every variable indexing at most k layers. In [9], an $\exp(\frac{n}{k^{O(k)}})$ lower bound for read-k oblivious ABP was shown. Another related result is the $\exp(\frac{n}{k^2})$ lower bound for

[11] Prior to this, [80] showed a quasi-polynomial lower bound for monotone circuits computing the perfect matching function.
[12] We may think of monotone switching networks as the Boolean analogue of monotone ABPs.
[13] In fact, [83] gave an exponential lower bound for the more powerful model of monotone span programs that was introduced in [47].
[14] Besides, there is an interesting connection between the non-commutative determinant and the commutative permanent [11].
[15] In fact, nothing better than the $\Omega(n \log d)$ lower bound (which also holds for commutative circuits) is known. The hardness of proving a sufficiently strong super-linear lower bound for non-commutative circuits is explained in a recent work [18] (see the discussion below on hardness magnification).

depth four read-k formulas [2]. An exponential lower bound is also known for read-k Boolean branching programs [15], which is a stronger model than read-k oblivious Boolean branching programs.

Constant depth circuits

The best lower bound for constant depth circuits is a little better than that for general circuits. Shoup and Smolensky [86] showed an $\Omega(\Delta n^{1+\frac{1}{\Delta}})$ lower bound for depth-Δ circuits computing the polynomials $\{\sum_{j\in[n]} x_1^j y_j, \ldots, \sum_{j\in[n]} x_n^j y_j\}$. Using an argument similar to [86], Raz [73] showed a roughly $\Omega(n^{1+\frac{1}{\Delta}})$ lower bound for depth-Δ circuits computing polynomials of degree $\Theta(\Delta)$, i.e., the polynomials have constant degree for constant depth circuits. These bounds were essentially achieved by analyzing linear circuits[16]. Prior to these works, a barely super-linear lower bound of $n \cdot \lambda_\Delta(n)$ was known for depth-Δ linear circuits using *super-concentrators* [24,68,76,94], where $\lambda_\Delta(n)$ is a very slowly growing function[17]. Recently, [59] gave a $n^{1+\frac{1}{2\Delta}}$ lower bound for depth-Δ linear circuits computing a linear transformation that can be computed in $\exp(n^{1-\Omega(\frac{1}{d})})$ time. A similar lower bound was known before for bounded coefficient circuits – Pudlák [69] proved an $\Omega(\Delta n^{1+\frac{1}{\Delta}})$ lower bound for depth-Δ bounded coefficient circuits computing the DFT matrix. A lower bound of $\Omega(n^{1+\frac{1}{O(\Delta)}})$ was also shown in [70] for depth-Δ bounded coefficient circuits computing the product of two $\sqrt{n} \times \sqrt{n}$ matrices.

A lot better lower bounds are known for constant depth multilinear circuits. Raz and Yehudayoff [79] showed an $\exp(n^{\Omega(\frac{1}{\Delta})})$ lower bound for depth-Δ multilinear circuits computing the $n \times n$ determinant polynomial. More recently, [23] showed an $\exp(n^{\frac{1}{\Delta}})$ lower bound for depth-Δ multilinear circuits computing the product of n many 2×2 matrices. In fact, an exponential separation is known between depth-Δ and depth-$(\Delta+1)$ multilinear circuits [22], which improved upon a previous quasi-polynomial separation [79].

A circuit computing an n-variate homogeneous polynomial of $\text{poly}(n)$ degree can be homogenized with only a polynomial blow-up in size [91], but this process is not depth preserving. It is plausible that homogeneous depth-Δ circuits are weaker than general depth-Δ circuits for constant Δ. Indeed, such a statement is known to be true for $\Delta = 3$ and $\Delta = 4$. It was shown in the classical work [66] that any homogeneous depth three circuit computing the n-variate degree-d elementary symmetric polynomial $\text{ESym}_{n,d}$ has size $n^{\Omega(d)}$, although $\text{ESym}_{n,d}$ has a non-homogeneous (multilinear) depth three circuit[18] of size $O(n^2)$. A sequence of work [28,33,48,49,52,58] culminated in a $n^{\Omega(\sqrt{d})}$ lower bound for homogeneous depth four circuits computing the width-n, degree-d iterated matrix multiplication polynomial $\mathsf{IMM}_{n,d}$. On the other hand, the depth reduction result in [34,93], which built on [3,54,98], yields a non-homogeneous depth four circuit of size $n^{O(d^{\frac{1}{3}})}$ for $\mathsf{IMM}_{n,d}$.

An almost cubic lower bound for general depth three circuits was shown in [53], which improved upon the previous quadratic bound [87]. The bound in [87] is for the elementary symmetric polynomial, whereas the bound in [53] is for a variant of the Nisan-Wigderson design polynomial (which is in VNP). Subsequently, [12,103] showed near cubic lower bounds for depth three circuits computing polynomials that have $\text{poly}(n)$-size depth five circuits. Over

[16] A linear circuit has only addition gates and so it computes a linear transformation (i.e., a set of linear forms) in the input variables. If a set of linear forms is computable by a circuit of size s and depth-Δ then they are computable by a linear circuit of size $O(s)$ and depth Δ. Thus, a super-linear lower bound for linear circuits implies a super-linear lower bound for general circuits.

[17] For instance, $\lambda_4(n) = \log^* n$.

[18] Construction of this circuit is attributed to Michael Ben-Or.

fixed finite fields, an exponential lower bound is known for depth three circuits computing the determinant [31,32]. For depth three circuits with low bottom fan-in[19] (but without any homogeneity restriction), [51] proved an exponential lower bound[20]. As mentioned before, the previous best lower bound for general depth four circuits is $\widetilde{\Omega}(n^{1.5})$ [85], which is a slight improvement over the roughly $\Omega(n^{1.33})$ bound obtained by specializing the lower bound for constant depth circuits in [73,86] to depth four circuits[21]. Our work here improves these super-linear bounds to a super-quadratic lower bound for depth four circuits.

Now, coming to constant depth Boolean circuits, an exponential lower bound is known for constant depth Boolean circuits over the DeMorgan basis (i.e., AC^0 circuits). However, it appears to us that the "right" Boolean analogue of constant depth arithmetic circuits is TC^0 circuits (see [1,37,82])[22]. The exponential lower bounds for AC^0 circuits [4,29,35] and the quasi-polynomial lower bounds for ACC^0 circuits[23] [63,101] are two of the great achievements in Boolean circuit complexity, but we are yet to see these kind of bounds for TC^0 circuits. The best known lower bound for threshold circuits is the slightly super-linear $n^{1+\frac{1}{c^\Delta}}$ bound for depth-Δ TC^0 circuits computing the parity function, where $c > 1$ is a fixed constant [41]. For depth two TC^0 circuits, [45] showed an $\widetilde{\Omega}(n^{2.5})$ lower bound on the number of wires and an $\widetilde{\Omega}(n^{1.5})$ lower bound on the number of gates.

Hardness magnification

There are results in the arithmetic and Boolean circuit literature that show how to obtain strong lower bounds from seemingly weak ones. We state a few of these results below with the intent of demonstrating that sufficiently strong super-linear or super-quadratic lower bounds can be quite interesting and challenging to prove even for constant depth circuits.

It follows from [91] that a cubic form[24] has a depth three powering circuit[25] with $\Theta(s)$ gates if it has a circuit of size s. Thus, a super-linear lower bound on the number of gates of a depth three powering circuit computing an explicit cubic form implies a super-linear circuit lower bound. Stated differently, a super-linear lower bound on the (symmetric) tensor rank of an explicit (symmetric) tensor of order 3 implies a super-linear circuit lower bound[26]. In fact, Raz [74] showed that an $n^{r\cdot(1-o(1))}$ lower bound on the tensor rank of an explicit order-r tensor $T : [n]^r \to \mathbb{F}$ implies a super-polynomial formula lower bound, assuming r is a super-constant and $r \leq \frac{\log n}{\log\log n}$.

---

[19] The depth reduction in [34] yields a depth three circuit with low bottom fan-in. This is reminiscent of a result in [95], which showed that a strong exponential lower bound for depth three Boolean circuits with low bottom fan-in implies a super-linear lower bound for Boolean circuits having logarithmic depth and bounded fan-in.

[20] This result was extended to depth four circuits with low bottom fan-in in the works [50,57].

[21] For the reader's convenience, we show how the $\Omega(n^{1.33})$ bound can be derived from [73,86] in Appendix D.

[22] This is because iterated addition and multiplication of integers are in TC^0, and in the converse direction, it is known that TC^0 circuits can be simulated by constant depth arithmetic circuits using a single threshold gate [1]. A related fact (attributed to Michael Ben-Or) is that there is an $O(n^2)$ size depth three arithmetic circuit computing the n-variate degree-$\frac{n}{2}$ elementary symmetric polynomial, which is the arithmetic analogue of the majority function.

[23] also for ACC^0 circuits composed with a bottom layer of threshold gates

[24] i.e., a homogeneous degree-3 polynomial

[25] A depth three powering circuit has a top +-gate, a middle layer of powering gates, and a bottom layer of +-gates.

[26] The best known lower bound for an explicit order-3 tensor $T : [n]^3 \to \mathbb{F}$ is roughly $3n$ and for an explicit order-r tensor is roughly $2n^{\lfloor \frac{r}{2} \rfloor}$ [5].

Recently, Chen and Tell [20] showed that a super-linear lower bound for TC^0 circuits (computing certain NC^1-complete functions) which is slightly better than the lower bound in [41] would imply $\mathsf{TC}^0 \neq \mathsf{NC}^1$. Their result builds on the work of [6]. By mimicking the argument in [20] for arithmetic circuits one gets the following statement: If $\mathsf{IMM}_{2,n}$ is computable[27] by a depth-Δ_0 circuit of size n^k then it is computable by a depth-Δ circuit of size $O(\frac{\Delta}{\Delta_0} \cdot n^{1+\exp(-\frac{\Delta}{\Delta_0 k})})$. Recall that an $\Omega(\Delta n^{1+\frac{1}{\Delta}})$ lower bound is already known for depth-Δ arithmetic circuits [73, 86]. If the same lower bound is shown for depth-Δ circuits *computing* $\mathsf{IMM}_{2,n}$ then that would imply a super-polynomial lower bound for constant depth arithmetic circuits! Compare this with the best known upper bound for depth-Δ circuits computing $\mathsf{IMM}_{2,n}$ which is roughly $\exp(O(\Delta n^{\frac{1}{\Delta}}))$. Even for depth three circuits, we get the following interesting observation: An $\Omega(n^{1.8+\epsilon})$ lower bound on the number of gates of a depth five circuit computing $\mathsf{IMM}_{2,n}$, for any constant $\epsilon > 0$, implies a super-cubic lower bound for depth three circuits[28].

Hardness amplification results are also known for non-commutative circuits [18, 38]. A biquadratic polynomial f in the variables $\mathbf{x} = \{x_1, \ldots, x_n\}$ and $\mathbf{y} = \{y_1, \ldots, y_n\}$ is a homogeneous degree-4 polynomial in which every monomial is of the form $x_i x_j y_k y_l$. The bilinear complexity of a biquadratic polynomial f is the minimum r such that $f = g_1 h_1 + \ldots + g_r h_r$, where g_i, h_i are bilinear forms in $\mathbf{x}$ and $\mathbf{y}$. It was shown in [38] that Permanent requires non-commutative circuits of exponential size if there is an explicit biquadratic polynomial having bilinear complexity $\Omega(n^{1+\epsilon})$, for some constant $\epsilon > 0$. In other words, a super-cubic lower bound on the size of a homogeneous depth four (commutative) circuit computing an explicit biquadratic form implies an exponential lower bound for non-commutative circuits. In another appealing instance of hardness amplification, [18] showed that an $\Omega(n^{\frac{\omega}{2}+\epsilon})$ lower bound, where ω is the matrix multiplication exponent, for non-commutative circuits computing an explicit constant degree polynomial implies an exponential lower bound for non-commutative circuits; if the explicit polynomial has $\mathrm{poly}(n)$ degree then the lower bound is an arbitrarily large polynomial function.

B Missing proofs from Section 3

B.1 Proofs from Section 3.1

▷ Claim 8. Let $P = {Q'_1}^{e_1} \cdots {Q'_t}^{e_t}$ be one of the polynomials P_i. For $k \geq 0$, let $P^{(k)} := \prod_{i \in [t]} {Q'_i}^{\max(e_i - k, 0)}$. Then, $\partial^k_{\mathbf{x}} P \subseteq \mathbb{F}\text{-span}\{\mathbf{y}_M^{\leq \infty} \mathbf{x}_M^{\leq k(2t\tau - 1)} P^{(k)}\}$.

Proof. We prove the claim by induction on k. If $k = 0$, then $\partial^0_{\mathbf{x}_M} P = \{P\} = \{P^{(0)}\}$ and hence the claim is true. Assume that the claim is true for k. Let X be a multilinear monomial of degree $k+1$ in $\mathbf{x}$ variables. Then $X = xX'$ where X' is a multilinear monomial of degree k in $\mathbf{x}$ variables and x one of the $\mathbf{x}$ variables. From the induction hypothesis we have that,

$$\frac{\partial P}{\partial X'} = g \cdot P^{(k)}$$

where g is a polynomial in $\mathbb{F}[\mathbf{x}_M, \mathbf{y}_M]$ with $\mathbf{x}_M$ degree of g being at most $k(2t\tau - 1)$ while its $\mathbf{y}_M$ degree can be arbitrarily large.

[27] Here $\mathsf{IMM}_{2,n}$ is a collection of four polynomials corresponding to the entries of a product of n many 2×2 matrices whose entries are distinct formal variables.
[28] We attribute this observation to Ankit Garg.

Let $J := \{j \in [t] : e_j > k\}$. We have that,

$$\begin{aligned}
\frac{\partial P}{\partial X} &= \frac{\partial}{\partial x}\left(g \cdot P^{(k)}\right) \\
&= \frac{\partial}{\partial x}\left(g \cdot \prod_{j\in J} {Q'_j}^{e_j-k}\right) \\
&= \frac{\partial g}{\partial x} \cdot \prod_{j\in J} {Q'_j}^{e_j-k} + g \cdot \sum_{j\in J}(e_j-k) \cdot {Q'_j}^{e_j-k-1} \cdot \frac{\partial Q'_j}{\partial x} \cdot \prod_{i\in J\setminus\{j\}} {Q'_i}^{e_i-k} \\
&= \left(\frac{\partial g}{\partial x} \cdot \prod_{j\in J} Q'_j + g \cdot \sum_{j\in J}(e_j-k) \cdot \frac{\partial Q'_j}{\partial x} \cdot \prod_{i\in J\setminus\{j\}} Q'_i\right) \cdot \prod_{j\in J} {Q'_j}^{e_j-k-1}
\end{aligned}$$

Observe that as $\mathtt{D}$ is a pruned depth four circuit, the support of all monomials of Q'_j is upper bounded by τ and as in any monomial the individual degree of any $\mathbf{x}$ variable is at most two, $\deg_{\mathbf{x}}(Q'_j) \leq 2\tau$. Also, $|J| \leq t$ and hence

$$\deg_{\mathbf{x}}\left(\frac{\partial g}{\partial x} \cdot \prod_{j\in J} Q'_j + g \cdot \sum_{j\in J}(e_j-k) \cdot \frac{\partial Q'_j}{\partial x} \cdot \prod_{i\in J\setminus\{j\}} Q'_i\right) \leq (k+1)(2t\tau-1).$$

As $\prod_{j\in J} {Q'_j}^{e_j-k-1} = P^{(k+1)}$, the claim is true for $k+1$. ◁

▷ Claim 9. Let ℓ, k, t and τ be as defined earlier. Then, $\ell + 2kt\tau < \frac{m}{2}$.

Proof. We will show that the ratio $\frac{\frac{m}{2}-2kt\tau}{\ell} > 1$. Putting the values of k and ℓ,

$$\begin{aligned}
\frac{\frac{m}{2}-2kt\tau}{\ell} &= \frac{\frac{m}{2} - 2\left\lfloor\frac{\delta d_{\mathbf{x}}}{t}\right\rfloor t\tau}{\left\lfloor\frac{m}{m^{\delta/t}+1}\right\rfloor} \\
&\geq \left(\frac{1}{2} - \frac{2\delta d_{\mathbf{x}}\tau}{m}\right)(m^{\delta/t}+1).
\end{aligned}$$

So, we need to show that

$$\begin{aligned}
\frac{1}{\frac{1}{2}-\frac{2\delta d_{\mathbf{x}}\tau}{m}} < m^{\delta/t}+1 &\iff \frac{1}{\frac{1}{2}-\frac{2\delta d_{\mathbf{x}}\tau}{m}} - 1 < m^{\delta/t} \\
&\iff \frac{1+\frac{4\delta d_{\mathbf{x}}\tau}{m}}{1-\frac{4\delta d_{\mathbf{x}}\tau}{m}} < m^{\delta/t}.
\end{aligned}$$

For large enough m, $\frac{4\delta d_{\mathbf{x}}\tau}{m} \leq \frac{1}{2}$. Using $1+x \leq e^x$, which holds for all $x \in \mathbb{R}$, and $\frac{1}{1-x} \leq e^{2x}$, which holds for $0 \leq x \leq \frac{1}{2}$ we get:

$$\frac{1+\frac{4\delta d_{\mathbf{x}}\tau}{m}}{1-\frac{4\delta d_{\mathbf{x}}\tau}{m}} \leq e^{\frac{12\delta d_{\mathbf{x}}\tau}{m}}.$$

So showing that $e^{\frac{12\delta d_{\mathbf{x}}\tau}{m}} < m^{\delta/t}$ would suffice. Now,

$$e^{\frac{12\delta d_{\mathbf{x}}\tau}{m}} < m^{\delta/t} \iff e^{\frac{12 d_{\mathbf{x}} t\tau}{m}} < m.$$

Putting the values of $d_{\mathbf{x}}, t$ an τ, we get that $\frac{12 d_{\mathbf{x}} t\tau}{m} = \frac{12 d_{\mathbf{x}}\left\lfloor\frac{d_{\mathbf{x}}}{(\ln m)^3}\right\rfloor\lfloor 20\ln m\rfloor}{m} \leq \frac{12 d_{\mathbf{x}}^2 \cdot 20\ln m}{m(\ln m)^3} = \Theta\left(\frac{m}{m(\ln m)^2(\ln m)^2}\right) = \Theta\left(\frac{1}{(\ln m)^4}\right) = o(1)$ as $d_{\mathbf{x}} = \Theta\left(\frac{\sqrt{m}}{\ln m}\right)$. Thus $e^{\frac{12 d_{\mathbf{x}} t\tau}{m}} < m$. ◁

B.2 Proof from Section 3.2.1

▷ Claim 11. Procedure 1 terminates in at most m iterations.

Proof. Let H_i be the set H after the i-th iteration of the procedure. Since each monomial in H_i has support more than τ, for any such monomial there are at least $\frac{\tau}{2}$ distinct $j \in [3m]\backslash M_1$ such that at least one of x_j and y_j appears in it. Counting the number of times at least one of x_j and y_j appears in a monomial in H_i and summing up these counts for all $j \in [3m]\backslash M_1$, we get that

$$\sum_{j\in[3m]\setminus M_1} e(j) \geq \frac{\tau \cdot |H_i|}{2};$$

so from an averaging argument there exists a j such that

$$e(j) \geq \frac{\tau \cdot |H_i|}{6m}.$$

Hence, the size of H_{i+1} is upper bounded as

$$|H_{i+1}| \leq |H_i| \cdot \left(1 - \frac{\tau}{6m}\right).$$

So after i iterations of the procedure we get,

$$\begin{aligned}
|H_i| &\leq |H_0| \cdot \left(1 - \frac{\tau}{6m}\right)^i \\
&\leq \left\lfloor \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \right\rfloor \cdot \left(1 - \frac{\lfloor 20 \ln m \rfloor}{6m}\right)^i \\
&\leq \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \cdot \left(1 - \frac{(20 \ln m - 1)}{6m}\right)^i \\
&\leq \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \cdot e^{-\frac{3i \cdot \ln m}{m}} && \text{(for sufficiently large } m\text{)} \\
&= \frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \cdot m^{-\frac{3i}{m}}.
\end{aligned}$$

For $i = m, |H_i| < 1$ (for sufficiently large m), i.e., the procedure terminates in at most m iterations. ◁

B.3 Proof from Section 3.2.2

▷ Claim 13. Let $\overline{M}_1 = [3m]\backslash M_1$. Procedure 2 sets at most m many variables in $\mathbf{x}_{\overline{M}_1} \cup \mathbf{y}_{\overline{M}_1}$ to field constants and removes all the heavy gates from $\mathtt{C}_1$.

Proof. In each iteration, we evaluate a light sparse polynomial in $\mathtt{C}_1$ to zero. This can be done as $\mathbb{F}$ is an algebraically closed field. Since the support of every monomial in $\mathtt{C}_1$ is at most τ, we end up setting at most $\frac{\tau \cdot m}{(\ln m)^2} \leq \frac{20 \cdot m}{\ln m}$ many variables to field constants in each iteration. As we can afford to set m variables, Step 2 of the procedure executes successfully. For some $i \in \mathbb{N}$, the while loop terminates in the i-th iteration in either of the following two cases:

1. All the heavy gates get eliminated after the $(i-1)$-th iteration, i.e., $s_i = 0$.
2. $\tau(b_1 + \cdots + b_i) > m$. (We show in the following subclaim that all the heavy gates are eliminated before this happens. Hence, the procedure stops only in the above case.)

▷ Subclaim 18. Let $i \in \mathbb{N}$ be such that $\tau(b_1 + \cdots + b_{i-1}) \leq m$ but $\tau(b_1 + \cdots + b_i) > m$. Then, all the heavy gates in $\mathtt{C}_1$ get eliminated in the first $(i-1)$ iterations of Procedure 2. If we assume Subclaim 18 then Claim 13 is proved. ◁

Proof of Subclaim 18: For $1 \leq j < i$, let $(Q_{j,1}, \ldots, Q_{j,r_j})$ be the available light sparse polynomials in $\mathtt{C}_1$ after the $(j-1)$-th iteration. Recall that s_j is the number of heavy gates in $\mathtt{C}_1$ after the $(j-1)$-th iteration. Suppose, $s_j \geq 1$ (otherwise, we have nothing to prove). For every $l \in [r_j]$, $b_{j,l}$ and $c_{j,l}$ refer to the fan-in of $Q_{j,l}$ and the number of distinct heavy gates connected to $Q_{j,l}$ in $\mathtt{C}_1$ respectively. We may assume that $b_{j,l} \neq 0$ for every $l \in [r_j]$. It is given that

$$b_{j,1} + \ldots + b_{j,r_j} \leq \frac{m^2 d_{\mathbf{x}}}{160 \cdot \lambda_0 \cdot (\ln m)^5}. \tag{2}$$

Since every heavy gate is connected to at least $\frac{m \cdot d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3}$ many light sparse polynomials in $\mathtt{C}_1$,

$$s_j \cdot \frac{m d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3} \leq c_{j,1} + \cdots + c_{j,r_j}.$$

As $b_{j,1}, \ldots, b_{j,r_j}$ are all non-zero, we get

$$s_j \cdot \frac{m d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3} \leq \frac{c_{j,1}}{b_{j,1}} \cdot b_{j,1} + \cdots + \frac{c_{j,r_j}}{b_{j,r_j}} \cdot b_{j,r_j}.$$

Let $u \in [r_j]$ be such that $\frac{c_{j,u}}{b_{j,u}} = \max\left\{\frac{c_{j,1}}{b_{j,1}}, \ldots, \frac{c_{j,r_j}}{b_{j,r_j}}\right\}$. Let $c_j := c_{j,u}$ and $b_j := b_{j,u}$. Then, the above equation implies

$$s_j \cdot \frac{m d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3} \leq \frac{c_j}{b_j} \cdot (b_{j,1} + \cdots + b_{j,r_j}).$$

From Equation (2), we get that for every $1 \leq j < i$,

$$s_j \cdot \frac{m d_{\mathbf{x}}}{2 \cdot \lambda_0 \cdot (\ln m)^3} \leq \frac{c_j}{b_j} \cdot \frac{m^2 d_{\mathbf{x}}}{160 \cdot \lambda_0 \cdot (\ln m)^5},$$

which implies

$$\frac{80 \cdot s_j \cdot b_j \cdot (\ln m)^2}{m} \leq c_j. \tag{3}$$

Thus, by setting the light sparse polynomial $Q_{j,u}$ to zero in the j-th iteration, we get rid of at least $\frac{80 \cdot s_j \cdot b_j \cdot (\ln m)^2}{m}$ many heavy gates from $\mathtt{C}_1$. Recall that s_{j+1} is the number of available heavy gates after the j-th iteration. Then, for every $1 \leq j < i$,

$$s_{j+1} \leq s_j - c_j \leq s_j \cdot \left(1 - \frac{80 \cdot b_j \cdot (\ln m)^2}{m}\right). \tag{4}$$

Hence, for every $1 \leq j < i$,

$$s_{j+1} \leq s_1 \cdot \prod_{l=1}^{j} \left(1 - \frac{80 \cdot b_l \cdot (\ln m)^2}{m}\right). \tag{5}$$

Also, for every $l \leq j$, we have $c_l \leq s_l$. Thus, Equation (3) implies

$$\frac{80 \cdot b_l \cdot (\ln m)^2}{m} \leq 1. \tag{6}$$

As $1-a \leq e^{-\frac{a}{2}}$ for $0 \leq a \leq 1$, Equations (5) and (6) imply

$$s_{j+1} \leq s_1 \cdot \prod_{l=1}^{j} \left(e^{-\frac{40 \cdot b_l \cdot (\ln m)^2}{m}} \right) = s_1 \cdot e^{-\frac{40 \cdot (b_1 + \cdots + b_j) \cdot (\ln m)^2}{m}}. \tag{7}$$

It is given that $\tau(b_1 + \cdots + b_i) > m$, which implies $\tau(b_1 + \cdots + b_{i-1}) > m - \tau \cdot b_i$. As b_i is the fan-in of a light sparse polynomial, $b_i \leq \frac{m}{(\ln m)^2}$ and so

$$\tau(b_1 + \cdots + b_{i-1}) > m - \frac{m\tau}{(\ln m)^2}. \tag{8}$$

On substituting $j = i-1$, $\tau = \lfloor 20 \ln m \rfloor$ and the value of $\tau(b_1 + \cdots + b_{i-1})$ from Equation (8) in Equation (7), we get

$$s_i \leq s_1 \cdot e^{-\frac{40 \cdot (\ln m)^2}{m} \cdot \left(\frac{m}{20 \ln m} - \frac{m}{(\ln m)^2} \right)} = s_1 \cdot e^{-(2 \ln m - 40)}.$$

For large enough m, $s_i \leq \frac{s_1}{m^{1.9}}$. Since $s_1 \leq m$, we get $s_i = 0$ (as it is a natural number). In other words, we get rid of all the heavy gates within $(i-1)$ iterations. ◁

C Missing proofs from Section 5

▷ Claim 17. The top fan-in s of $\mathtt{D}$ is $\omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5}\right)$.

Proof. From Equation (1), we have

$$\begin{aligned}
s &\geq \frac{\frac{1}{m^{O(1)}} \min\left\{ \frac{1}{4^k} \cdot \binom{m}{\ell} \binom{m}{k}, \binom{m}{\ell + d_{\mathbf{x}} - k} \right\}}{m^{O(1)} \cdot \binom{m}{\ell + 2kt\tau} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \\
&\geq \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot \frac{\binom{m}{\ell+1}}{\binom{m}{\ell + 2kt\tau + 1}}, \frac{\binom{m}{\ell + d_{\mathbf{x}} - k}}{\binom{m}{\ell + 2kt\tau}} \right\} \\
&= \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot \frac{(m - \ell - 2kt\tau - 1)!}{(m - \ell - 1)!} \cdot \frac{(\ell + 2kt\tau + 1)!}{(\ell + 1)!}, \right. \\
&\qquad \left. \frac{(m - \ell - 2kt\tau)!}{(m - \ell - d_{\mathbf{x}} + k)!} \cdot \frac{(\ell + 2kt\tau)!}{(\ell + d_{\mathbf{x}} - k)!} \right\} \\
&= \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot e^{(-2kt\tau) \ln \frac{m-\ell-1}{\ell+1} \pm o(1)}, e^{(d_{\mathbf{x}} - 2kt\tau - k) \ln \frac{m-\ell}{\ell} \pm o(1)} \right\} \\
&\qquad \text{(Using Proposition 5.)} \\
&\geq \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot \left(\frac{m}{\ell + 1} - 1 \right)^{-2kt\tau}, \left(\frac{m}{\ell} - 1 \right)^{(d_{\mathbf{x}} - 2kt\tau - k)} \right\} \\
&= \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot \left(\frac{m}{\left\lfloor \frac{m}{m^{\delta/t} + 1} \right\rfloor + 1} - 1 \right)^{-2kt\tau}, \left(\frac{m}{\left\lfloor \frac{m}{m^{\delta/t} + 1} \right\rfloor} - 1 \right)^{(d_{\mathbf{x}} - 2kt\tau - k)} \right\} \\
&\geq \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot \left(\frac{m}{\frac{m}{m^{\delta/t} + 1}} - 1 \right)^{-2kt\tau}, \left(\frac{m}{\frac{m}{m^{\delta/t} + 1}} - 1 \right)^{(d_{\mathbf{x}} - 2kt\tau - k)} \right\} \\
&\geq \frac{1}{m^{O(1)} \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \min\left\{ \frac{\binom{m}{k}}{4^k} \cdot m^{-2k\delta\tau}, m^{(1 - 2\delta\tau - \frac{\delta}{t})k} \right\}
\end{aligned}$$

Since $\frac{\binom{m}{k}}{4^k} \cdot m^{-2k\delta\tau} = \frac{\binom{m}{k}}{4^k \cdot m^{(1-\frac{\delta}{t})k}} \cdot m^{(1-2\delta\tau-\frac{\delta}{t})k} \leq (\frac{em}{k})^k \cdot \frac{m^{\frac{\delta k}{t}}}{4^k m^k} \cdot m^{(1-2\delta\tau-\frac{\delta}{t})k}$. For our choice of parameters δ, k and t, $m^{\frac{\delta k}{t}} = O(1)$. Hence, $\frac{\binom{m}{k}}{4^k} \cdot m^{-2k\delta\tau} \leq m^{(1-2\delta\tau-\frac{\delta}{t})k}$ and thus,

$$
\begin{aligned}
s &\geq \frac{1}{m^{O(1)}} \cdot \frac{\binom{m}{k} \cdot m^{-2k\delta\tau}}{4^k \cdot \binom{\lceil \frac{w}{t} \rceil + k - 1}{k}} \\
&\geq \frac{1}{m^{O(1)}} \cdot \left(\frac{m \cdot k}{4e \cdot k \cdot m^{2\delta\tau} \cdot (\frac{w}{t} + k)} \right)^k && \text{(Using Proposition 4.)} \\
&\geq \frac{1}{m^{O(1)}} \cdot \left(\frac{m \cdot t}{8e \cdot m^{2\delta\tau} \cdot w} \right)^k && \text{(Since } kt \leq w = \left\lfloor \tfrac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3} \right\rfloor .) \\
&= \frac{1}{m^{O(1)}} \cdot \left(\frac{m \cdot \left\lfloor \frac{d_{\mathbf{x}}}{(\ln m)^3} \right\rfloor}{8e \cdot m^{2\delta\tau} \cdot \left\lfloor \frac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3} \right\rfloor} \right)^k \\
&\geq \frac{1}{m^{O(1)}} \cdot \left(\frac{m \cdot \frac{d_{\mathbf{x}}}{(\ln m)^3}}{16e \cdot m^{2\delta\tau} \cdot \frac{m d_{\mathbf{x}}}{\lambda_0 \cdot (\ln m)^3}} \right)^k \\
&\geq \frac{1}{m^{O(1)}} \cdot \left(\frac{\lambda_0}{16e \cdot m^{2 \cdot \frac{1}{(\ln m)^2} \cdot \lfloor 20 \ln m \rfloor}} \right)^{\ln m} && \text{(Since } k \geq \lfloor \ln m \rfloor .) \\
&= \frac{1}{m^{O(1)}} \cdot \left(\frac{\lambda_0}{16e \cdot e^{O(1)}} \right)^{\ln m} \\
&= \omega\left(\frac{m^2 d_{\mathbf{x}}}{(\ln m)^5} \right),
\end{aligned}
$$

if we choose λ_0 to be a large enough constant. ◁

D A brief review of the lower bounds from [86] and [73]

In this section, we present a short overview of the lower bounds for restricted depth arithmetic circuits with multiple output gates from [86] and [73] and focus mainly on depth four circuits. We would use (s, Δ)-arithmetic circuit to denote an arithmetic circuit of size-s and depth-Δ and $\mathbf{y}$ for the set of variables $\{y_1, \ldots, y_n\}$.

Lower bound from [86]. Let $n \in \mathbb{N}$ and $\Delta = O(\log n)$. Shoup and Smolensky showed that there exist n linear forms $g_1, \ldots, g_n \in \mathbb{C}[\mathbf{y}]$, such that the size of any depth-Δ normal-linear circuit[29] that computes $g_1, \ldots, g_n$ is $\Omega(\Delta n^{1+\frac{1}{\Delta}})$. The following proposition implies that the same lower bound holds for a depth-Δ arithmetic circuit, that also computes $g_1, \ldots, g_n$.

▶ **Proposition 19.** *Let $n \in \mathbb{N}$, $\mathbb{F}$ be an arbitrary field and $h_1, \ldots, h_n \in \mathbb{F}[\mathbf{y}]$ be linear forms computed by an (s, Δ)-arithmetic circuit. Then, there exists an (s, Δ)-normal-linear circuit that computes $h_1, \ldots, h_n$.*

This proposition is easy to prove; a proof of the same in given in Section 2 of [73]. We refer the reader to Section 3 of [86] for more details. In case of depth four arithmetic circuits over $\mathbb{C}$, if we substitute $\Delta = 4$ in the above mentioned result then we get a lower bound of

[29] An arithmetic circuit D over $\mathbb{F}$ is called a normal-linear circuit if all the gates in D are labelled by either variables or by $+$. Every gate in D computes a linear form in the underlying set of variables over $\mathbb{F}$.

$\Omega(n^{1.25})$, but we observe that this lower bound can be optimised roughly to $\Omega(n^{1.33})$ using the following claim. Claim 20 implies that the size of any depth four arithmetic circuit and any depth three normal-linear circuit computing the linear forms $g_1, \ldots, g_n$ given in [86] are same, which is roughly $\Omega(n^{1.33})$.

▷ Claim 20. Let $n \in \mathbb{N}$ and $h_1, \ldots, h_n \in \mathbb{F}[\mathbf{y}]$ be linear forms, computed by an $(s, 4)$-arithmetic circuit $\mathtt{C}$ over $\mathbb{F}$. Then, there exists an $(s, 3)$-normal-linear circuit over $\mathbb{F}$ that computes $h_1, \ldots, h_n$.

Proof. From Proposition 19, we obtain an $(s, 4)$-normal-linear circuit $\mathtt{D}$ over $\mathbb{F}$ that computes $h_1, \ldots, h_n$. We now argue that linearisation ensures that the fan-in of every gate in the bottom layer of $\mathtt{D}$ is exactly 1. It turns out that only those product gates survive the linearisation in the bottom layer of $\mathtt{C}$ which are connected to exactly one variable. Let v be a gate in the bottom layer of $\mathtt{C}$ with children $u_1, \ldots, u_r$. Then, there exists some $i \in [r]$ such that u_i is a variable and all other gates are labelled by field constants, in which case, we remove $u_j, j \in [r] \setminus \{i\}$ and multiply the label of the edge (v, u_i) with $\prod_{j \in [r]\setminus\{i\}} u_j$. As every gate in the bottom layer of $\mathtt{D}$ has fan-in 1, we can directly connect the input of every gate in this layer to its outputs, thereby yielding an $(s, 3)$-normal-linear circuit that computes $h_1, \ldots, h_n$. ◁

Lower bound from [73]. Let n be a prime number and $\Delta = O(\log n)$. Raz showed that there exist n explicit homogeneous polynomials of degree $\Theta(\Delta)$ in $\Theta(n)$ variables over $\mathbb{F}$, such that any depth-Δ arithmetic circuit that computes these polynomials has size $\Omega(n^{1+\frac{1}{2\cdot\Delta}})$. While the lower bound for depth-Δ arithmetic circuit given in [86] holds for n non-explicit linear forms over $\mathbb{C}$, the same lower bound also holds for n explicit homogeneous polynomials of $\Theta(n)$ degree in $\Theta(n)$ variables over $\mathbb{C}$. We first recall some definitions from [73] and then show that the lower bound for depth-Δ arithmetic circuits in [73] can be optimized slightly.

Let $n', m, t, s \in \mathbb{N}$. A polynomial mapping $f : \mathbb{F}^{n'} \to \mathbb{F}^m$ of degree t is an m tuple $(f_1, \ldots, f_m)$ of n' variate degree t polynomials over $\mathbb{F}$. The polynomial mapping f *eludes* a polynomial mapping $\Gamma : \mathbb{F}^s \to \mathbb{F}^m$ if $Image(f) \not\subset Image(\Gamma)$. Moreover, f is said to be (s, t)-*elusive* over $\mathbb{F}$ if it eludes every polynomial mapping $\Gamma : \mathbb{F}^s \to \mathbb{F}^m$ of degree at most t.

Let n be a prime, $\Delta = O(\log n)$, $m := n^2, \Delta' := a \cdot \Delta$, where $a \in \mathbb{N}$ is a constant, $n' := \Delta' \cdot n$ and $\mathbf{x} := \{x_{k,l} : k \in [\Delta'], l \in [n]\}$. Let $f : \mathbb{F}^{n'} \to \mathbb{F}^m$ be defined as follows:

For every $(i, j) \in [n] \times [n]$,

$$f_{(i,j)}(\mathbf{x}) := \prod_{k=1}^{\Delta'} x_{k,(i+j\cdot k) \bmod n}. \tag{9}$$

Further, for every $i \in [n]$,

$$\tilde{f}_i(\mathbf{x}, \mathbf{y}) := \sum_{j \in [n]} y_j \cdot f_{(i,j)}(\mathbf{x}). \tag{10}$$

[73] showed that for $a = 5$, any depth-Δ arithmetic circuit computing $\tilde{f}_1, \ldots, \tilde{f}_n$ requires size $\Omega(n^{1+\frac{1}{2\cdot\Delta}})$. The detailed proof is given in Section 4 of [73]. Here, we show how this lower bound can be optimized to $\Omega(n^{1+\frac{1}{\Delta}-\epsilon_{a,\Delta}})$, where $\epsilon_{a,\Delta} := \frac{2\cdot\Delta-1}{a\cdot\Delta^2}$. Note that as we increase the value of a, this lower bound gets closer to the one for depth-Δ arithmetic circuits given in [86]. The main ingredient of this improvement is the following optimization of Lemma 4.1 in [73].

▶ **Lemma 21.** *Let n be a prime, $m = n^2$, $\Delta = O(\log n)$, $\Delta' = a \cdot \Delta$, where $a \in \mathbb{N}$ is a constant and $n' = \Delta' \cdot n$. Let $\mathbb{G}$ be a field extension of $\mathbb{F}$ of size more than m and $f : \mathbb{G}^{n'} \to \mathbb{G}^m$ be the polynomial mapping defined in Equation (9)*[30]*. Then, f is (s, Δ)-elusive over $\mathbb{G}$, where $s = \lfloor n^{1+\frac{1}{\Delta}-\epsilon_{a,\Delta}} \rfloor$ and $\epsilon_{a,\Delta} = \frac{2\cdot\Delta-1}{a\cdot\Delta^2}$.*

Proof. Let $U := [n] \times [n]$, $r := \frac{1}{2}\lfloor n^{1-\frac{2}{\Delta'}} \rfloor$. For $A \subseteq U$, $f_A(\mathbf{x}) := \prod_{(i,j)\in A} f_{i,j}(\mathbf{x})$. A is said to be *retrievable* if for any $A' \subseteq U$, $f_A \neq f_{A'}$ implies $A \neq A'$. It is shown in Claim 4.2 of [73] that

$$\Pr_{A \in_R \binom{U}{r}}[A \text{ is not retrievable}] \leq \left(\frac{|A|}{n+1}\right)^{\Delta'} \cdot n^2,$$

where $A \in_R \binom{U}{r}$ means that A is a subset of U of size r chosen uniformly at random. On plugging the value of r in the above equation, we get

$$\Pr_{A \in_R \binom{U}{r}}[A \text{ is retrievable}] > \frac{1}{2}. \tag{11}$$

Let $\mathcal{L}$ be the set of degree r multilinear homogeneous polynomials of the type $g : \mathbb{G}^m \to \mathbb{G}$, such that every monomial of g corresponds to a retrievable set. Clearly, $\mathcal{L}$ is a $\mathbb{G}$-vector space. From Equation (11), we get $\dim_{\mathbb{G}}(\mathcal{L}) > \frac{1}{2}\binom{m}{r} \geq \frac{1}{2}\left(\frac{m}{r}\right)^r = \frac{1}{2}\left(2n^{1+\frac{2}{\Delta'}}\right)^r$. Fix a polynomial map $\Gamma : \mathbb{G}^s \to \mathbb{G}^m$ of degree Δ. Then, for every $g \in \mathcal{L}$, $g \circ \Gamma : \mathbb{G}^s \to \mathbb{G}$ is a polynomial of degree $r \cdot \Delta$. Let $\mathcal{K}$ be the set of all polynomials from $\mathbb{G}^s$ to $\mathbb{G}$ of degree at most $r \cdot \Delta$. Then, $\mathcal{K}$ is a $\mathbb{G}$-vector space and $\dim_{\mathbb{G}} \mathcal{K} \leq \binom{s+r\cdot\Delta}{r\cdot\Delta} \leq \left(\frac{e(s+r\cdot\Delta)}{r\cdot\Delta}\right)^{r\cdot\Delta} < \left(\frac{2es}{r}\right)^{r\cdot\Delta} = \left(12n^{\frac{1}{\Delta}+\frac{2}{\Delta'}-\epsilon_{a,\Delta}}\right)^{r\cdot\Delta}$.

On substituting the values of Δ' and $\epsilon_{a,\Delta}$ in $\dim_{\mathbb{G}} \mathcal{L}$ and $\dim_{\mathbb{G}} \mathcal{K}$, we get $\dim_{\mathbb{G}} \mathcal{K} < \dim_{\mathbb{G}} \mathcal{L}$. Now, for a fixed polynomial map $\Gamma : \mathbb{G}^s \to \mathbb{G}^m$ of degree Δ, define $\varphi_\Gamma : \mathcal{L} \to \mathcal{K}\,; g \mapsto g \circ \Gamma$. Clearly, φ_Γ is a $\mathbb{G}$-linear map and as $\dim_{\mathbb{G}} \mathcal{K} < \dim_{\mathbb{G}} \mathcal{L}$, φ_Γ is not an injective map. This means that there exists a non-zero $g_\Gamma \in \mathcal{L}$, such that $\varphi_\Gamma(g_\Gamma) = g_\Gamma \circ \Gamma = 0$. As $|\mathbb{G}| > m$, Claim 4.4 in [73] implies that $g_\Gamma \circ f : \mathbb{G}^{n'} \to \mathbb{G}$ is not the zero polynomial. Thus, for every polynomial mapping $\Gamma : \mathbb{G}^s \to \mathbb{G}^m$ of degree Δ, $Image(f) \not\subset Image(\Gamma)$. Hence, f is an (s, Δ)-elusive polynomial map over $\mathbb{G}$. ◀

The following is a corollary of Lemma 21 and Proposition 3.11 in [73].

▶ **Corollary 22.** *Let n be a prime, $\Delta = O(\log n)$ and $\Delta' = a \cdot \Delta$ for some constant $a \in \mathbb{N}$. Let $\tilde{f}_1, \ldots, \tilde{f}_n$ be $n(\Delta'+1)$ variate degree $\Delta'+1$ polynomials as defined in Equation (10). Then, any depth-Δ arithmetic circuit $\mathtt{C}$ over $\mathbb{F}$ computing $\tilde{f}_1, \ldots, \tilde{f}_n$ requires size $\Omega\left(n^{1+\frac{1}{\Delta}-\epsilon_{a,\Delta}}\right)$, where $\epsilon_{a,\Delta} = \frac{2\cdot\Delta-1}{a\cdot\Delta^2}$.*

Proof idea. In Proposition 3.11 in [73], $\tilde{f}_1, \ldots, \tilde{f}_n$ and $\mathtt{C}$ are viewed as linear polynomials in $\mathbf{y}$ variables over the function field $\mathbb{F}(\mathbf{x})$ and an arithmetic circuit over $\mathbb{F}(\mathbf{x})$ respectively. Then, using Proposition 19, $\mathtt{C}$ is converted to an (s, Δ)-normal-linear circuit over $\mathbb{F}(\mathbf{x})$, that also computes $\tilde{f}_1, \ldots, \tilde{f}_n$. After that, on invoking Lemma 21, we get the lower bound of $\Omega\left(n^{1+\frac{1}{\Delta}-\epsilon_{a,\Delta}}\right)$ on a depth-Δ arithmetic circuit computing $\tilde{f}_1, \ldots, \tilde{f}_n$. ◀

We now focus on depth four circuits. Let $s := n^{\frac{4}{3}-\epsilon_{a,3}}$, where $\epsilon_{a,3} := \frac{5}{9a}$. In the proof of Corollary 22, we use Claim 20 to obtain an $(s, 3)$-normal-linear circuit over $\mathbb{F}(\mathbf{x})$ from an $(s, 4)$-arithmetic circuit over $\mathbb{F}(\mathbf{x})$, such that both circuits compute $\tilde{f}_1, \ldots, \tilde{f}_n$. As Lemma 21 implies that the polynomial mapping f is $(s, 3)$-elusive, we get a lower bound of $\Omega(n^{\frac{4}{3}-\epsilon_{a,3}})$ for depth four circuits.

[30] f is naturally a polynomial mapping over $\mathbb{G}$ because from every $i, j \in [n]$, $f_{i,j}(\mathbf{x}) \in \mathbb{F}[\mathbf{x}] \subseteq \mathbb{G}[\mathbf{x}]$.

Multiparty Karchmer – Wigderson Games and Threshold Circuits

Alexander Kozachinskiy
Department of Computer Science, University of Warwick, Coventry, UK
Alexander.Kozachinskiy@warwick.ac.uk

Vladimir Podolskii
Steklov Mathematical Institute, Russian Academy of Sciences, Moscow, Russia
podolskii@mi-ras.ru

Abstract

We suggest a generalization of Karchmer – Wigderson communication games to the multiparty setting. Our generalization turns out to be tightly connected to circuits consisting of threshold gates. This allows us to obtain new explicit constructions of such circuits for several functions. In particular, we provide an explicit (polynomial-time computable) log-depth monotone formula for Majority function, consisting only of 3-bit majority gates and variables. This resolves a conjecture of Cohen et al. (CRYPTO 2013).

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases Karchmer-Wigderson Games, Threshold Circuits, threshold gates, majority function

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.24

Funding *Alexander Kozachinskiy*: Supported by the EPSRC grant EP/P020992/1 (Solving Parity Games in Theory and Practice).
Vladimir Podolskii: The work of V.V. Podolskii was performed at the Steklov International Mathematical Center and supported by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075-15-2019-1614).

Acknowledgements The authors are grateful to Alexander Shen for suggesting to generalize our initial results.

1 Introduction

Karchmer and Wigderson established tight connection between circuit depth and communication complexity [11] (see also [12, Chapter 9]). Namely, they showed that for each Boolean function f one can define a communication game which communication complexity *exactly* equals the depth of f in the standard De Morgan basis. This discovery turned out to be very influential in Complexity Theory. A lot of circuit depth lower bounds as well as formula size lower bounds rely on this discovery [10, 13, 5, 7, 4]. Karchmer – Wigderson games have been used also in adjacent areas like Proof Complexity (see, e.g., [14]).

Karchmer – Wigderson games represent a deep connection of *two-party* communication protocols with De Morgan circuits. Loosely speaking, in this connection one party is responsible for $\wedge$ gates and the other party is responsible for $\vee$ gates. In this paper we address the question of what would be a natural generalization of Karchmer – Wigderson games to the multiparty setting. Is it possible to obtain in this way a connection with other types of circuits?

We answer positively to this question: we suggest such a generalization and show its connection to circuits consisting of *threshold gates*. To motivate our results we first present applications we get from this new connection.

1.1 Applications to circuits

There are two classical constructions of $O(\log n)$-depth monotone formulas for the Majority function, MAJ_{2n+1}. The one was given by Valiant [15]. Valiant used probabilistic method which does not give an explicit construction. The other construction is the AKS sorting network [1]. This construction actually gives polynomial-time computable $O(\log n)$-depth $O(n \log n)$-size monotone circuit for MAJ_n.

Several authors (see, e.g., [6, 3]) noticed that the Valiant's probabilistic argument actually gives a $O(\log n)$-depth formula for MAJ_n, consisting only of MAJ_3 gates and variables. Is it possible to construct a $O(\log n)$-depth circuit for MAJ_{2n+1}, consisting only of MAJ_3 gates and variables, *deterministically in polynomial time*?[1]

This question was stated as a conjecture by Cohen et al. in [3]. First, they showed that the answer is positive under some cryptographic assumptions. Secondly, they constructed (unconditionally) a polynomial-time computable $O(\log n)$-depth circuit, consisting only of MAJ_3 gates and variables, which coincides with MAJ_n for all inputs in which the fraction of ones is bounded away from $1/2$ by $2^{-\Theta(\sqrt{\log n})}$.

We show that the conjecture of Cohen et al. is true (unconditionally).

▶ **Theorem 1.** *There exists polynomial-time computable $O(\log n)$-depth formula for MAJ_{2n+1}, consisting only of MAJ_3 gates and variables.*

In the proof we use the AKS sorting network. In fact, one can use any polynomial-time computable construction of $O(\log n)$-depth monotone circuit for MAJ_{2n+1}. We also obtain the following general result:

▶ **Theorem 2.** *If there is a monotone formula (i.e., formula, consisting of $\wedge, \vee$ gates and variables) for MAJ_{2n+1} of size s, then there is a formula for MAJ_{2n+1} of size $O(s \cdot n^{\log_2(3)}) = O(s \cdot n^{1.58\ldots})$, consisting only of MAJ_3 gates and variables.*

Transformation from the last theorem, however, is not efficient. We can make this transformation polynomial-time computable, provided $\log_2(3)$ is replaced by $1/(1-\log_3(2)) \approx 2.71$. In turn, we view Theorem 2 as a potential approach to obtain super-quadratic lower bounds on monotone formula size for MAJ_{2n+1}. However, this approach requires better than $n^{2+\log_2(3)}$ lower bound on formula size of MAJ_{2n+1} in the $\{\mathrm{MAJ}_3\}$ basis. Arguably, this basis may be easier to analyze than the standard monotone basis. The best known size upper bounds in the $\{\wedge, \vee\}$ basis and the $\{\mathrm{MAJ}_3\}$ basis are, respectively, $O(n^{5.3})$ and $O(n^{4.29})$ [8]. Both bounds are due to Valiant's method (see [8] also for the limitations of Valiant's method).

We also study a generalization of the conjecture of Cohen et al. to threshold functions. By THR_a^b we denote the following Boolean function:

$$\mathrm{THR}_a^b \colon \{0,1\}^b \to \{0,1\}, \qquad \mathrm{THR}_a^b(x) = \begin{cases} 1 & x \text{ contains at least } a \text{ ones,} \\ 0 & \text{otherwise.} \end{cases}$$

For some reasons (to be discussed below) a natural generalization would be a question of whether $\mathrm{THR}_{n+1}^{kn+1}$ can by computed by a $O(\log n)$-depth circuit, consisting only of THR_2^{k+1} gates and variables (initial conjecture can be obtained by setting $k = 2$). This question was also addressed by Cohen et al. in [3]. First, they observed that there is a construction of depth $O(n)$ (and exponential size). Secondly, they gave an explicit construction of depth $O(\log n)$, which coincides with $\mathrm{THR}_{n+1}^{kn+1}$ for all inputs in which the fraction of ones is bounded away from $1/k$ by $\Theta(1/\sqrt{\log n})$.

[1] Note that AKS sorting network does not provide a solution because it consists of $\wedge$ and $\vee$ gates.

However, no exact (even non-explicit) construction with sub-linear depth or sub-exponential size was known. In particular, Valiant's probabilistic construction does not work for $k \geq 3$. Nevertheless, in this paper we improve depth $O(n)$ to $O(\log^2 n)$ and size from $\exp\{O(n)\}$ to $n^{O(1)}$ for this problem:

▶ **Theorem 3.** *For any constant $k \geq 3$ there exists polynomial-time computable $O(\log^2 n)$-depth polynomial-size circuit for $\mathrm{THR}^{kn+1}_{n+1}$, consisting only of THR^{k+1}_2 gates and variables.*

1.2 Applications to Multiparty Secure Computations

The conjecture stated in [3] was motivated by applications to Secure Multiparty Computations. The paper [3] establishes an approach to construct efficient multiparty protocols based on protocols for a small number of players. More specifically, in their framework one starts with a protocol for a small number of players and a formula F computing a certain boolean function. Then one combines a protocol for a small number of players with itself recursively, where the recursion mimics the formula F.

It is shown in [3] that from our result it follows that for any n there is an explicit polynomial size protocol for n players secure against a passive adversary that controls any $t < \frac{n}{2}$ players. It is also implicit in [3] that from Theorem 3 for $k = 3$ it follows that for any n there is a protocol of size $2^{O(\log^2 n)}$ for n players secure against an active adversary that controls any $t < \frac{n}{3}$ players. An improvement of the depth of the formula in Theorem 3 to $O(\log n)$ would result in a polynomial size protocol. We refer to [3] for more details on the secure multiparty computations.

1.3 Multiparty Karchmer – Wigderson games

We now reveal a bigger picture to which the above results belong to. Namely, they can be put into framework of multiparty Karchmer – Wigderson games.

Before specifying how we define these games let us give an instructive example. Consider ordinary monotone Karchmer – Wigderson game for MAJ_{2n+1}. In this game Alice receives a string $x \in \mathrm{MAJ}^{-1}_{2n+1}(0)$ and Bob receives a string $y \in \mathrm{MAJ}^{-1}_{2n+1}(1)$. In other words, the number of ones in x is at most n and the number of ones in y is at least $n+1$. The goal of Alice and Bob is to find some coordinate i such that $x_i = 0$ and $y_i = 1$. Next, imagine that Bob flips each of his input bits. After that parties have two vectors in both of which the number of ones is at most n. Now Alice and Bob have to find any coordinate in which both vectors are 0.

In this form this problem can be naturally generalized to the multiparty setting. Namely, assume that there are k parties, and each receives a Boolean vector of length $kn+1$ with at most n ones. Let the task of parties be to find a coordinate in which all k input vectors are 0. How many bits of communication are needed for that?

For $k = 2$ the answer is $O(\log n)$, because there exists a $O(\log n)$-depth monotone circuit for MAJ_{2n+1} and hence the monotone Karchmer – Wigderson game for MAJ_{2n+1} can be solved in $O(\log n)$ bits of communication. For $k \geq 3$ we are only aware of a simple $O(\log^2 n)$-bit solution based on the binary search.

Now, let us look at the case $k \geq 3$ from another perspective and introduce multiparty Karchmer – Wigderson games. Note that each party receives a vector on which $\mathrm{THR}^{kn+1}_{n+1}$ equals 0. The goal is to find a common zero. Note that we can consider a similar problem for any function f satisfying so-called Q_k*-property*: any k vectors from $f^{-1}(0)$ have a common zero. In the next definition we define Q_k-property formally and also introduce related R_k-property.

▶ **Definition 4.** *Let Q_k be the set of all Boolean functions f satisfying the following property: for all $x^1, x^2, \ldots, x^k \in f^{-1}(0)$ there is a coordinate i such that $x^1_i = x^2_i = \ldots = x^k_i = 0$.*

Further, let R_k be the set of all Boolean functions f satisfying the following property: for all $x^1, x^2, \ldots, x^k \in f^{-1}(0)$ there is a coordinate i such that $x^1_i = x^2_i = \ldots = x^k_i$.

For $f \in Q_k$ let Q_k-*communication game* for f be the following communication problem. In this problem there are k parties. The jth party receives a Boolean vector $x^j \in f^{-1}(0)$. The goal of players is to find any coordinate i such that $x^1_i = x^2_i = \ldots = x^k_i = 0$.

Similarly we can define R_k-*communication games* for functions from R_k. In the R_k-communication games the objective of parties is slightly different: their goal is to find any coordinate i and a bit b such that $x^1_i = x^2_i = \ldots = x^k_i = b$.

Self-dual functions belong to R_2 and monotone self-dual functions belong to Q_2. It is easy to see that R_2-communication games are equivalent to Karchmer – Wigderson games for self-dual functions (one party should flip all the input bits). Moreover, Q_2-communication games are equivalent to monotone Karchmer – Widgerson games for monotone self-dual functions.

In this paper we consider R_k-communication games as a multiparty generalization of Karchmer – Wigderson games. In turn, Q_k-communication games are considered as a generalization of *monotone* Karchmer – Wigderson games. To justify this choice one should relate them to some type of circuit complexity.

1.4 Connection to threshold gates and the main result

Every function from Q_k can be *lower bounded* by a circuit, consisting only of THR_2^{k+1} gates and variables. More precisely, let us write $C \leq f$ for a Boolean circuit C and a Boolean function f if for all $x \in f^{-1}(0)$ we have $C(x) = 0$. Then the following proposition holds:

▶ **Proposition 5** ([3])**.** *The set Q_k is equal to the set of all Boolean functions f for which there exists a circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables.*

There is a similar characterization of the set R_k.

▶ **Proposition 6.** *The set R_k is equal to the set of all Boolean functions f for which there exists a circuit $C \leq f$, consisting only of THR_2^{k+1} gates and literals*[2]*.*

The proof from [3] of Proposition 5 with obvious modifications also works for Proposition 6.

Given $f \in Q_k$, what is the minimal depth of a circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables? We show that this quantity is equal (up to constant factors) the communication complexity of Q_k-communication game for f.

▶ **Theorem 7.** *Let $k \geq 2$ be any constant. Then for any $f \in Q_k$ the following two quantities are equal up to constant factors:*

- *the communication complexity of Q_k-communication game for f;*
- *minimal d for which there exists a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables.*

Similar result can be obtained for R_k-communication games.

[2] We stress that negations can only be applied to variables but not to THR_2^{k+1} gates.

▶ **Theorem 8.** *Let $k \geq 2$ be any constant. Then for any $f \in R_k$ the following two quantities are equal up to constant factors:*

- *the communication complexity of R_k-communication game for f;*
- *minimal d for which there exists a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and literals.*

Proofs of both theorems are divided into two parts:

(a) transformation of a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables (literals), into a $O(d)$-bit protocol computing $Q_k(R_k)$-communication game for f;

(b) transformation of a d-bit protocol computing $Q_k(R_k)$-communication game for f into a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables (literals).

The first part is simple and the main challenge is the second part. Later in this paper (Section 6) we also formulate refined versions of Theorems 7 and 8. Namely, we refine these theorems in the following two directions. Firstly, we take into account circuit size and for this we consider dag-like communication protocols. Secondly, we show that transformations (a-b) can be done in polynomial time (under some mild assumptions).

We derive our upper bounds on the depth of MAJ_{2n+1} and $\mathrm{THR}_{n+1}^{kn+1}$ (Theorems 1 and 3) from Theorem 7. We first solve the corresponding Q_k-communication games with small number of bits of communication. Namely, for the case of MAJ_{2n+1} we use AKS sorting network to solve the corresponding Q_2-communication game with $O(\log n)$ bits of communication. For the case of $\mathrm{THR}_{n+1}^{kn+1}$ with $k \geq 3$ we solve the corresponding Q_k-communication game by a simple binary search protocol with $O(\log^2 n)$ bits of communication. This is where we get depth $O(\log n)$ for Theorem 1 and depth $O(\log^2 n)$ for Theorem 3. Again, some special measures should be taken to make the resulting circuits polynomial-time computable and to control their size[3].

1.5 Our techniques: $Q_k(R_k)$-hypotheses games

As we already mentioned, the hard part of our main result is to transform a protocol into a circuit.

For this we develop a new language to describe circuits, consisting of threshold gates. Namely, for every f in Q_k (R_k) we introduce the corresponding *$Q_k(R_k)$-hypotheses game* for f. We show that strategies in these games exactly capture depth and size of circuits, consisting only of THR_2^{k+1} gates and variables (literals). It turns out that strategies are more convenient than circuits to simulate protocols, since they operate in the same top-bottom manner.

Once we establish the equivalence of circuits and hypotheses games, it remains for us to transform a communication protocol into a strategy in a hypotheses games. This is an elaborate construction that is presented in Propositions 20 and 24. Below in this section we introduce hypotheses games and as an illustration sketch the construction of a strategy in a hypothesis game that is used in the proof of Theorem 1.

Here is how we define these games. Fix $f\colon \{0,1\}^n \to \{0,1\}$. There are two players, Nature and Learner. Before the game starts, Nature privately chooses $z \in f^{-1}(0)$, which then can not be changed. The goal of Learner is to find some $i \in [n]$ such that $z_i = 0$. The game proceeds in rounds. At each round Learner specifies $k+1$ families $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_k \subset f^{-1}(0)$ to Nature. We understand this as if Learner makes the following $k+1$ hypotheses about z:

[3] We should only care about the size in case of Theorem 3, because depth $O(\log n)$ immediately gives polynomial size.

"$z \in \mathcal{H}_0$",
"$z \in \mathcal{H}_1$",
$\vdots$
"$z \in \mathcal{H}_k$".

Learner loses immediately if less than k hypotheses are true, i.e., if the number of $j \in \{0, 1, \ldots, k\}$ satisfying $z \in \mathcal{H}_j$ is less than k. Otherwise Nature points out to some hypothesis which is true. In other words, Nature specifies to Learner some $j \in \{0, 1, \ldots, k\}$ such that $z \in \mathcal{H}_j$. The game then proceeds in the same manner for some finite number of rounds. At the end Learner outputs an integer $i \in [n]$. We say that Learner wins if $z_i = 0$.

It is not hard to show that Learner has a winning strategy in Q_k-hypotheses game for f if and only if $f \in Q_k$. Since we will use similar arguments in the paper, let us go through the "if" part: if $f \in Q_k$, then Learner has a winning strategy. Denote by $\mathcal{Z}$ be the set of all z's which are compatible with Nature's answers so far. At the beginning $\mathcal{Z} = f^{-1}(0)$. If $|\mathcal{Z}| \geq k + 1$, Learner takes any distinct $z^1, z^2, \ldots, z^{k+1} \in \mathcal{Z}$ and makes the following hypotheses:

"$z \neq z^1$",
"$z \neq z^2$",
$\vdots$
"$z \neq z^{k+1}$".

At least k hypotheses are true, and the Nature's response strictly reduces the size of $\mathcal{Z}$. When the size of $\mathcal{Z}$ becomes k, Learner is ready to give an answer due to Q_k-property of f.

This strategy requires exponential in n number of rounds. This can be easily improved to $O(n)$ rounds. Indeed, instead of choosing $k + 1$ distinct elements of $\mathcal{Z}$ split $\mathcal{Z}$ into $k + 1$ disjoint almost equal parts. Then let the ith hypotheses be "z is not in the ith part". Nature's response to this reduces the size of $\mathcal{Z}$ by a constant factor, until the size of $\mathcal{Z}$ is k.

For $f \in Q_k$ we can now ask what is the minimal number of rounds on in a Learner's winning strategy. The following proposition gives an exact answer:

▶ **Proposition 9.** *For any $f \in Q_k$ the following holds. Learner has a d-round winning strategy in Q_k-hypotheses game for f if and only if there exists a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables.*

Proposition 9 is the core result for our applications. For instance, we prove Theorem 1 by giving an explicit $O(\log n)$-round winning strategy of Learner in Q_2-hypotheses game for MAJ_{2n+1}. Let us now sketch our argument (the complete proof can be found in Section 4).

Assume that Nature's input vector is z. We notice that in $O(\log n)$ rounds one can easily find *two* integers $i, j \in [2n + 1]$ such that either $z_i = 0$ or $z_j = 0$. However, we need to know for sure. For that we take any polynomial time computable $O(\log n)$-depth monotone formula F for MAJ_{2n+1} (for instance one that can be obtained from the AKS sorting network). We start to descend from the output gate of F to one of F's inputs. Throughout this descending we maintain the following invariant. If g is the current gate, then either $g(z) = 0 \wedge z_i = 0$ or $g(\neg z) = 1 \wedge z_j = 0$ (here $\neg$ denotes bit-wise negation). It can be shown that in one round one can either exclude i or j (which will already give us an answer) or replace g by some gate which is fed to g. If we reach an input to F, we output the index of the corresponding variable.

Similarly one can define R_k-hypotheses game for any $f\colon \{0,1\}^n \to \{0,1\}$. In R_k-hypotheses game Nature and Learner play in the same way except that now Learner's objective is to find some pair $(i,b) \in [n] \times \{0,1\}$ such that $z_i = b$. The following analog of Proposition 9 holds:

▶ **Proposition 10.** *For any $f \in R_k$ the following holds. Learner has a d-round winning strategy in R_k-hypotheses game for f if and only if there exists a d-depth circuit $C \leq f$, consisting only of THR_2^{k+1} gates and literals.*

1.6 Organization of the paper

In Section 2 we give Preliminaries. In Section 3 we define $Q_k(R_k)$-hypotheses games formally and derive Proposition 9 and 10. In Section 4 we obtain our results for Majority function (Theorems 1 and 2) using simpler arguments than in our general results. Then in Section 5 we prove these general results (Theorems 7 and 8). In Section 6 we refine Theorems 7 and 8 in order to take into account the circuit size and computational aspects (Theorems 27 and 30 below). In Section 7 we derive Theorem 3 and provide another proof for Theorem 1. Finally, in Section 8 we formulate some open problems.

2 Preliminaries

Let $[n]$ denote the set $\{1,2,\ldots,n\}$ for $n \in \mathbb{N}$. For a set W we denote the set of all subsets of W by 2^W. For two sets A and B by A^B we mean the set of all functions of the form $f\colon B \to A$.

We usually use subscripts to denote coordinates of vectors. In turn, we usually use superscripts to numerate vectors.

We use standard terminology for Boolean formulas and circuits [9]. We denote the size of a circuit C by $\mathrm{size}(C)$ and the depth by $\mathrm{depth}(C)$. By De Morgan formulas/circuits we mean formulas/circuits consisting of $\wedge, \vee$ gates of fan-in 2 and literals (i.e., we assume that negations are applied only to variables). By monotone formulas/circuits we mean formulas/circuits consisting of $\wedge, \vee$ gates of fan-in 2 and variables. We also consider formulas/circuits consisting only of THR_2^{k+1} gates and variables (literals). We stress that in such circuits we do not use constants. Allowing literals as inputs we allow to apply negations only to variables. We also assume that negations in literals do not contribute to the depth of a circuit.

We use the notion of deterministic communication protocols in the multiparty *number-in-hand* model. However, to capture the circuit size in our results we consider not only standard *tree-like* protocols, but also *dag-like* protocols. This notion was considered by Sokolov in [14]. We use slightly different variant of this notion, arguably more intuitive one. In the next subsection we provide all necessary definitions. To obtain a definition of a standard protocol one should replace dags by binary trees.

2.1 Dags and dag-like communication protocols

We use the following terminology for directed acyclic graphs (dags). Firstly, we allow more than one directed edge from one node to another. A terminal node of a dag G is a node with no out-going edges. Given a dag G, let

- $V(G)$ denote the set of nodes of G;
- $T(G)$ denote the set of terminal nodes of G.

For $v \in V(G)$ let $Out_G(v)$ be the set of all edges of G that start at v. A dag G is called t-ary if every non-terminal node v of G we have $|Out_G(v)| = t$. An ordered t-ary dag is a t-ary dag G equipped with a mapping from the set of edges of G to $\{0,1,\ldots,t-1\}$. This

mapping restricted to $Out_G(v)$ should be injective for every $v \in V(G) \setminus T(G)$. The value of this mapping on an edge e will be called the *label* of e. In terms of labels we require for ordered t-ary dags that any t edges, starting at the same node, have different labels.

By a path in G we mean a sequence of *edges* $\langle e_1, e_2, \ldots, e_m \rangle$ such that for every $j \in [m-1]$ edge e_i ends in the same node in which e_{j+1} starts. Note that there may be two distinct paths visiting same nodes (for instance, there may be two parallel edges from one node to another).

We say that a node w is a descendant of a node v if there is a path from v to w. We call w a successor of v if there is an edge from v to w. A node s is called *starting node* if any other node is a descendant of s. Note that any dag has at most one starting node.

If a dag G has the starting node s, then by depth of $v \in V(G)$ we mean the maximal length of a path from s to v. The depth of G then is the maximal depth of its nodes.

Assume that $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k, \mathcal{Y}$ are some finite sets.

▶ **Definition 11.** *A k-party dag-like communication protocol π with inputs from $\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \mathcal{X}_k$ and with outputs from $\mathcal{Y}$ is a tuple $\langle G, P_1, P_2, \ldots, P_k, \phi_1, \phi_2, \ldots, \phi_k, l \rangle$, where*

- *G is an ordered 2-ary dag with the starting node s;*
- *$P_1, P_2, \ldots, P_k$ is a partition of $V(G) \setminus T(G)$ into k disjoint subsets;*
- *ϕ_i is a function from $P_i \times \mathcal{X}_i$ to $\{0,1\}$;*
- *l is a function from $T(G)$ to $\mathcal{Y}$.*

The depth of π (denoted by $\mathrm{depth}(\pi)$) is the depth of G. The size of π (denoted by $\mathrm{size}(\pi)$) is $|V(G)|$.

The underlying mechanics of the protocol is as follows. Parties descend from s to one of the terminals of G. If the current node v is not a terminal and $v \in P_i$, then at v the ith party communicates a bit to all the other parties. Namely, the ith party communicates the bit $b = \phi_i(v, x)$, where $x \in \mathcal{X}_i$ is the input of the ith party. Among the two edges, starting at v, parties choose one labeled by b and descend to one of the successors of v along this edge. Finally, when parties reach a terminal t, they output $l(t)$.

We say that $x \in \mathcal{X}_i$ is i-compatible with an edge e from v to w if one of the following two condition holds:

- $v \notin P_i$;
- $v \in P_i$ and e is labeled by $\phi_i(v, x)$.

We say that $x \in \mathcal{X}_i$ is i-compatible with a path $p = (e_1, e_2, \ldots, e_m)$ of G if for every $j \in [m]$ it holds that x is i-compatible with e_j. Finally, we say that $x \in \mathcal{X}_i$ is i-compatible with a node $v \in V(G)$ if there is a path p from s to v such that x is i-compatible with v.

We say that an input $(x^1, x^2, \ldots, x^k) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \mathcal{X}_k$ visits a node $v \in V(G)$ if there is a path p from s to v such that for every $i \in [k]$ it holds that x^i is i-compatible with p. Note that there is unique $t \in T(G)$ such that $(x^1, x^2, \ldots, x^k)$ visits t.

To formulate an effective version of Theorems 7 and Theorem 8 we need the following definition.

▶ **Definition 12.** *The* light form *of a k-party dag-like communication protocol $\pi = \langle G, P_1, P_2, \ldots, P_k, \phi_1, \phi_2, \ldots, \phi_k, l \rangle$ is a tuple $\langle G, P_1, P_2, \ldots, P_k, l \rangle$.*

I.e., to obtain the light form of π we just forget about $\phi_1, \phi_2, \ldots, \phi_k$. In other words, the light form only contains the underlying graph of π, the partition of non-terminal nodes between parties and the labels of terminals. On the other hand, in the light form there is no information at all how parties communicate at the non-terminal nodes.

Protocol π *computes* a relation $S \subset \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_k \times \mathcal{Y}$ if the following holds. For every $(x^1, x^2, \ldots, x^k) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_k$ there exist $y \in \mathcal{Y}$ and $t \in T(G)$ such that $(x^1, \ldots, x^k)$ visits t, $l(t) = y$ and $(x^1, x^2, \ldots, x^k, y) \in S$.

Using language of relations, we can formally define Q_k- and R_k-communication games. Namely, given $f\colon \{0,1\}^n \to \{0,1\}, f \in Q_k$, we define Q_k-communication game for f as the following relation:

$$S \subset \underbrace{f^{-1}(0) \times \ldots \times f^{-1}(0)}_{k} \times [n],$$

$$S = \left\{(x^1, \ldots, x^k, j) \mid x^1_j = \ldots = x^k_j = 0\right\}.$$

Similarly, given $f\colon \{0,1\}^n \to \{0,1\}, f \in R_k$, we define R_k-communication game for f as the following relation:

$$S \subset \underbrace{f^{-1}(0) \times \ldots \times f^{-1}(0)}_{k} \times ([n] \times \{0,1\}),$$

$$S = \left\{(x^1, \ldots, x^k, (j,b)) \mid x^1_j = \ldots = x^k_j = b\right\}.$$

It is easy to see that a dag-like protocol for S can be transformed into a tree-like protocol of the same depth, but this transformation can drastically increase the size.

3 Formal treatment of $Q_k(R_k)$-hypotheses games

Fix $f \in Q_k$, $f\colon \{0,1\}^n \to \{0,1\}$. Here we define Learner's strategies in Q_k-hypotheses game for f formally. We consider not only tree-like strategies but also dag-like. To specify a Learner's strategy S in Q_k-hypotheses game we have to specify:

- An ordered $(k+1)$-ary dag G with the starting node s;
- a subset $\mathcal{H}_j(p)$ for every $j \in \{0,1,\ldots,k\}$ and for every path p in G from s to some node in $V(G) \setminus T(G)$;
- a number $i_t \in [n]$ for every terminal t.

The underlying mechanics of the game is as follows. Let Nature's vector be $z \in f^{-1}(0)$. Learner and Nature descend from s to one of the terminals of G. More precisely, a position in the game is determined by a path p, starting at s. If the endpoint of p is not a terminal, then Learner specifies some sets $\mathcal{H}_0(p), \mathcal{H}_1(p), \ldots, \mathcal{H}_k(p)$ as his hypotheses. If less than k of these sets contain z, then Nature wins. Otherwise Nature specifies some $j \in \{0,1,\ldots,k\}$ such that $z \in \mathcal{H}_j(p)$. Among $k+1$ edges that start at the endpoint of p players choose one which is labeled by j. After that they extend p by this edge. At some point parties reach some terminal t (i.e., the endpoint of p becomes equal t). Then the game ends and Learner output i_t.

We stress that Learner's output depends only on t but not on a path to t (unlike Learner's hypotheses). This property will be crucial in establishing connection of Q_k-hypotheses games to circuits.

We now proceed to a formal definition of what does it mean that S is winning for Learner.

We say that $z \in f^{-1}(0)$ is *compatible* with a path $p = \langle e_1, \ldots, e_m\rangle$, starting in s, if the following holds. If p is of length 0, then every $z \in f^{-1}(0)$ is compatible with p. Otherwise for every $i \in \{1,\ldots,e_m\}$ it should hold that $z \in \mathcal{H}_j(\langle e_1, \ldots, e_{i-1}\rangle)$, where j is the label of edge e_i. Informally this means that Nature, having z on input, can reach a position in the game which corresponds to a path p.

We say that strategy S is winning for Learner in Q_k-hypotheses game for f if for every path p, starting at s, and for every $z \in f^{-1}(0)$, compatible with p, the following holds:

- if the endpoint of p is not a terminal, then the number of $j \in \{0,1,\ldots,k\}$ such that $z \in \mathcal{H}_j(p)$ is at least k;
- if the endpoint of p is $t \in T(G)$, then $z_{i_t} = 0$.

We will formulate a stronger version of Proposition 9. For that we need the notion of the *light form* of the strategy S. Namely, the light form of S is its underlying dag G equipped with a mapping which to every $t \in T(G)$ assigns i_t. In other words, the light form contains a "skeleton" of S and Learner's outputs in terminals (and no information about Learner's hypotheses).

We can identify the light form of any strategy S with a circuit, consisting only of THR_2^{k+1} gates and variables. Namely, place THR_2^{k+1} gate in every $v \in V(G) \setminus T(G)$ and for every $t \in T(G)$ place a variable x_{i_t} in t. Set s to be the output gate.

▶ **Proposition 13.** *For all $f \in Q_k, f\colon \{0,1\}^n \to \{0,1\}$ the following holds:*

(a) *if S is a Learner's winning strategy in Q_k-hypotheses game for f, then its light form, considered as a circuit C consisting only of THR_2^{k+1} gates and variables, satisfies $C \leq f$.*

(b) *Assume that $C \leq f$ is a circuit, consisting only of THR_2^{k+1} gates and variables. Then there exists a Learner's winning strategy S in Q_k-hypotheses game for f such that the light form of S coincides with C.*

We omit the proof of ***(b)*** as in the paper we only use ***(a)***.

Proof of (a) of Proposition 13. For a node $v \in V(G)$ let $f_v\colon \{0,1\}^n \to \{0,1\}$ be the function, computed by the circuit C at the gate, corresponding to v.

We shall prove the following statement. For any path p, starting in s, and for any z which is compatible with p it holds that $f_v(z) = 0$, where v is the endpoint of p. To see why this implies $C \leq f$ take any $z \in f^{-1}(0)$ and note that z is compatible with the path of length 0. The endpoint of such path is s and hence $0 = f_s(z) = C(z)$.

We will prove the above statement by the backward induction on the length of p. The longest path p ends in some $t \in T(G)$. By definition $f_t = x_{i_t}$. On the other hand, since S is winning, $z_{i_t} = 0$ for any z compatible with p. In other words, $f_t(z) = 0$ for any z compatible with p. The base is proved.

Induction step is the same if p ends in some other terminal. Now assume that p ends in $v \in V(G) \setminus T(G)$. Take any $z \in f^{-1}(0)$ compatible with p. Let p_j be the extension of p by the edge which starts at v and is labeled by $j \in \{0, 1, \ldots, k\}$. Next, let v_j be the endpoint of p_j (nodes $v_0, v_1, \ldots, v_k$ are successors of v). Since S is winning, the number of $j \in \{0, 1, \ldots, k\}$ such that $z \in \mathcal{H}_j(p)$ is at least k. Hence by definition the number of $j \in \{0, 1, \ldots, k\}$ such that z is compatible with p_j is at least k. Finally, by the induction hypothesis this means that the number of $j \in \{0, 1, \ldots, k\}$ such that $f_{v_j}(z) = 0$ is at least k. On the other hand:

$$f_v = \mathrm{THR}_2^{k+1}(f_{v_0}, f_{v_1}, \ldots, f_{v_k}).$$

Therefore $f_v(z) = 0$, as required. ◀

One can formally define analogues notions for R_k-hypotheses games. We skip this as modifications are straightforwards and only formulate an analog of Proposition 13.

▶ **Proposition 14.** *For all $f \in R_k, f\colon \{0,1\}^n \to \{0,1\}$ the following holds:*

(a) *if S is a Learner's winning strategy in R_k-hypotheses game for f, then its light form, considered as a circuit C consisting only of THR_2^{k+1} gates and literals, satisfies $C \leq f$.*

(b) *Assume that $C \leq f$ is a circuit, consisting only of THR_2^{k+1} gates and literals. Then there exists a Learner's winning strategy S in R_k-hypotheses game for f such that the light form of S coincides with C.*

▶ Remark 15. It might be unclear why we prefer to construct strategies instead of constructing circuits directly, because beside the circuit itself we should also specify Learner's hypotheses. The reason is that strategies can be seen as *proofs* that the circuit we construct is correct.

4 Results for Majority

Proof of Theorem 1. There exists an algorithm which in $n^{O(1)}$-time produces a monotone formula F of depth $d = O(\log n)$ computing MAJ_{2n+1}. Below we will define a strategy S_F in the Q_2-hypotheses game for MAJ_{2n+1}. Strategy S_F will be winning for Learner. Moreover, its depth will be $d + O(\log n)$. In the end of the proof we will refer to Proposition 13 to show that S_F yields a $O(\log n)$-depth polynomial-time computable formula for MAJ_{2n+1}, consisting only of MAJ_3 gates and variables.

Strategy S_F has two phases. The first phase does not uses F at all, only the second phase does. The objective of the first phase is to find some distinct $i, j \in [2n+1]$ such that either $z_i = 0 \wedge z_j = 1$ or $z_i = 1 \wedge z_j = 0$, where z is the Nature's vector. This can be done as follows.

▶ **Lemma 16.** *One can compute in polynomial time a 3-ary tree T of depth $O(\log n)$ with the set of nodes $v(T)$ and a mapping $w\colon v(T) \to 2^{[2n+1]}$ such that the following holds:*

- *if r is the root of T, then $w(r) = [2n+1]$;*
- *if v is not a leaf of T and v_1, v_2, v_3 are 3 children of v, then every element of $w(v)$ is covered at least twice by $w(v_1), w(v_2), w(v_3)$;*
- *if l is a leaf of T, then $w(r)$ is of size 2.*

Proof. We start with a trivial tree, consisting only of the root, to which we assign $[2n+1]$. Then at each iteration we do the following. We have a 3-ary tree in which nodes are assigned to some subsets of $[2n+1]$. If every leaf is assigned to a set of size 2, we terminate. Otherwise we pick any leaf l of the current tree which is assigned to a subset $A \subset [2n+1]$ of size at least 3. We split A into 3 disjoint subsets A_1, A_2, A_3 of sizes $\lfloor |A|/3 \rfloor$, $\lfloor |A|/3 \rfloor$ and $|A| - 2\lfloor |A|/3 \rfloor$. We add 3 children to l (which become new leafs) and assign $A_1 \cup A_2, A_1 \cup A_3, A_2 \cup A_3$ to them.

It is easy to verify that the sizes of $A_1 \cup A_2, A_1 \cup A_3, A_2 \cup A_3$ are at least 2 and at most $\frac{4}{5} \cdot |A|$. Hence the size of the set assigned to a node of depth h is at most $\left(\frac{4}{5}\right)^h \cdot (2n+1)$. This means that the depth of the tree is at any moment at most $\log_{5/4}(2n+1) = O(\log n)$. Therefore we terminate in $3^{O(\log n)} = n^{O(1)}$ iterations, as at each iteration we add 3 new nodes. Each iteration obviously takes polynomial time. ◀

We use T to find two $i, j \in [2n+1]$ such that either $z_i = 0$ or $z_j = 0$. Namely, we descend from the root of T to one of its leafs. Learner maintains an invariant that the leftmost 0-coordinate of z is in $w(v)$, where v is the current node of T. Let v_1, v_2, v_3 be 3 children of v. Learner for every $i \in [3]$ makes a hypothesis that the leftmost 0-coordinate of z is in $w(v_i)$. Due to the properties of w at least two hypotheses are true. Nature indicates some v_i for which this is true, and Learner descends to v_i. When Learner reaches a leaf, he knows a set of size two containing the leftmost 0-coordinate of z. Let this set be $\{i, j\}$.

We know that either z_i or z_j is 0. Thus $z_i z_j \in \{00, 01, 10\}$. At the cost of one round we can ask Nature to identify an element of $\{00, 01, 10\}$ which differs from $z_i z_j$. If 10 is identified, then $z_i z_j \in \{00, 01\}$, and hence $z_i = 0$, i.e., we can already output i. Similar thing happens when 01 is identified. Finally, if 00 is identified, then the objective of the first phase is fulfilled and we can proceed to the second phase.

The second phase takes at most d rounds. In this phase Learner produces a sequence $g_0, g_1, \ldots, g_{d'}$, $d' \leq d$ of gates of F, where the depth of g_i is i, the last gate $g_{d'}$ is an input variable (i.e., a leaf of F) and each $g \in \{g_0, g_1, \ldots, g_{d'}\}$ satisfies:

$$(g(z) = 0 \wedge z_i z_j = 01) \vee (g(\neg z) = 1 \wedge z_i z_j = 10)\,. \tag{1}$$

Here $\neg z$ denotes the bit-wise negation of z.

At the beginning Learner sets $g_0 = g_{\text{out}}$ to be the output gate of F. Let us explain why (1) holds for g_{out}. Nature's vector is an element of $\text{MAJ}^{-1}_{2n+1}(0)$. I.e., the number of ones in z is at most n. In turn, in $\neg z$ there are at least $n+1$ ones. Since g_{out} computes MAJ_{2n+1}, we have that $g_{\text{out}}(z) = 0$ and $g_{\text{out}}(\neg z) = 1$. In turn, by the first phase it is guarantied that $z_i z_j = 01 \vee z_i z_j = 10$.

Assume now that the second phase is finished, i.e., Learner has produced some $g_{d'} = x_k$ satisfying (1). Then by (1) either $g_{d'}(z) = z_k = 0$ or $g_{d'}(\neg z) = (\neg z)_k = 1$. In both cases $z_k = 0$, i.e., Learner can output k.

It remains to explain how to fulfill the second phase. It is enough to show the following. Assume that Learner knows a gate g_l of F of depth l satisfying (1) and that g_l is not an input variable. Then in one round he can either find a gate g_{l+1} of depth $l+1$ satisfying (1) or give a correct answer to the game.

The gate g_{l+1} will be one of the two gates which are fed to g_l. Assume first that g_l is an $\wedge$-gate and $g_l = u \wedge v$. From (1) we conclude that from the following 3 statements exactly 1 is true for z:

$$u(z) = 0 \text{ and } z_i z_j = 01, \tag{2}$$

$$u(z) = 1, v(z) = 0 \text{ and } z_i z_j = 01, \tag{3}$$

$$u(\neg z) = v(\neg z) = 1 \text{ and } z_i z_j = 10. \tag{4}$$

At the cost of one round Learner can ask Nature to indicate one statement which is false for x. If Nature says that (2) is false for z, then (1) holds for $g_{l+1} = v$. Next, if Nature says that (3) is false for z, then (1) holds for $g_{l+1} = u$. Finally, if Nature says that (4) is false for z, then we know that $z_i z_j = 01$, i.e., Learner can already output i.

In the same way we can deal with the case when g_l is an $\vee$-gate and $g_l = u \vee v$. By (1) exactly 1 of the following 3 statements is true for z:

$$u(z) = v(z) = 0 \text{ and } z_i z_j = 01, \tag{5}$$

$$u(\neg z) = 1 \text{ and } z_i z_j = 10, \tag{6}$$

$$u(\neg z) = 0, v(\neg z) = 1 \text{ and } z_i z_j = 10. \tag{7}$$

Similarly, Learner asks Nature to indicate one statement which is false for z. If Nature says that (5) is false for z, then $z_i z_j = 10$, i.e., Learner can output j. Next, if Nature says that (6) is false for z, then (1) holds for $g_{l+1} = v$. Finally, if Nature says that (7) is false for z, then (1) holds for $g_{l+1} = u$.

Thus S_F is a $O(\log n)$-depth winning strategy of Learner. Apply Proposition 13 to S_F. We get a $O(\log n)$-depth formula $F' \leq \text{MAJ}_{2n+1}$, consisting only of MAJ_3 gates and variables. In fact, F' computes MAJ_{2n+1}. Indeed, $F' \leq \text{MAJ}_{2n+1}$ means that F' outputs 0 on every input with at most n ones. On the other hand, F' consists of MAJ_3 gates and hence F' computes a self-dual function. Therefore F' outputs 1 on every input with at least $n+1$ ones.

It remains to explain how to compute F' in polynomial time. To do so we have to compute in polynomial time the light form of S_F, i.e., the underlying tree of S_F and the outputs of Learner in the leafs. It is easy to see that one can do this as follows.

First, compute F and compute T from Lemma 16. For each leaf l of T do the following. Let $w(l) = \{i, j\}$. Add 3 children to l. Two of them will be leafs of S_F, in one Learner outputs i and in the other Learner outputs j. Attach a tree of F to the third child. Then add to each non-leaf node of F one more child so that now the tree of F is 3-ary. Each added child is a leaf of S_F. If a child was added to an $\wedge$-gate, then Learner outputs i in this child.

In turn, if a child was added to an $\vee$ gate, then Learner outputs j in it. Finally, there are leafs that were in F initially, each labeled by some input variable. In these nodes Learner outputs the index of the corresponding input variable. ◀

Proof of Theorem 2. How many rounds takes the first phase of the strategy S_F from the previous proof? Initially the left-most 0-coordinate of z takes $O(n)$ values. At the cost of one round we can shrink the number of possible values almost by a factor of $3/2$. Thus the first phase corresponds to a ternary tree of depth $\log_{3/2}(n) + O(1)$. The size of that tree is hence $3^{\log_{3/2}(n)+O(1)} = O(n^{1/(1-\log_3(2))}) = O(n^{2.70951\ldots})$. To some of its leafs we attach a tree of the same size as the initial formula F. As a result we obtain a formula F' of size $O(n^{2.70951\ldots} \cdot s)$ for MAJ_{2n+1}, consisting of MAJ_3 gates and variables (here s is the size of the initial formula F).

Let us show that we can perform the first phase in $\log_2(n) + O(1)$ rounds. This will improve the size of the previous construction to $O(3^{\log_2(n)+O(1)} \cdot s) = O(n^{\log_2(3)} \cdot s)$. However, the construction with $\log_2(n) + O(1)$ rounds will not be explicit. We need the following Lemma:

▶ **Lemma 17.** *There exists a formula D with the following properties:*

- *formula D is a complete ternary tree of depth $\lceil \log_2(n) \rceil + 10$;*
- *every non-leaf node of D contains a MAJ_3 gate and every leaf of D contains a conjunction of 2 variables;*
- *$D(x) = 0$ for every $x \in \{0,1\}^{2n+1}$ with at most n ones.*

Let us at first explain how to use formula D from Lemma 17 to fulfill the first phase. Recall that our goal is to find two indices $i, j \in [2n+1]$ such that either $z_i = 0$ or $z_j = 0$. To do so Learner descends from the output gate of D to some of its leafs. He maintains an invariant that for his current gate g of D it holds that $g(z) = 0$. For the output gate the invariant is true because by Lemma 17 D is 0 on all Nature's possible vectors. If we reached a leaf so that g is a conjuction of two variables z_i and z_j, then the first phase is fulfilled (by the invariant $z_i \wedge z_j = 0$). Finally, if g is a non-leaf node of D, i.e., a MAJ_3 gate, then we can descend to one of the children of g at the cost of one round without violating the invariant. Indeed, as $g(z) = 0$, then the same is true for at least 2 children of g. For each child g_i of g Learner makes a hypotheses that $g_i(z) = 0$. Any Nature's response allows us to replace g by some g_i.

Proof of Lemma 17. We will show existence of such D via probabilistic method. Namely, independently for each leaf l of D choose $(i, j) \in [2n+1]^2$ uniformly at random and put the conjuction $z_i \wedge z_j$ into l. It is enough to demonstrate that for any $x \in \{0,1\}^{2n+1}$ with at most n ones it holds that $\Pr[D(x) = 1] < 2^{-2n-1}$.

To do so we use the modification of the standard Valiant's argument. For any fixed x let p be the probability that a leaf l of D equals 1 on x. This probability is the same for all the leafs and is at most $1/4$. Now, $\Pr[D(x) = 1]$ can be expressed exactly in terms of p as follows:

$$\Pr[D(x) = 1] = \underbrace{f(f(f(\ldots f}_{\lceil \log_2(n) \rceil + 10}(p))\ldots),$$

where $f(t) = t^3 + 3t^2(1-t) = 3t^2 - 2t^3$. Observe that $3f(t) \leq (3t)^2$. Hence

$$3\Pr[D(x) = 1] \leq (3p)^{2^{\lceil \log_2(n) \rceil + 10}} \leq (3/4)^{1000n} < (1/2)^{-2n-1}.$$ ◀

◀

5 Proof of the main theorem

Theorem 7 follows from Proposition 18 (Subsection 5.1) and Proposition 20 (Subsection 5.2). In turn, Theorem 8 follows from Proposition 19 (Subsection 5.1) and Proposition 24 (Subsection 5.2).

5.1 From circuits to protocols

▶ **Proposition 18.** *For any constant $k \geq 2$ the following holds. Assume that $f \in Q_k$ and $C \leq f$ is a circuit, consisting only of THR_2^{k+1} gates and variables. Then there is a protocol π, computing Q_k-communication game for f, such that $\mathrm{depth}(\pi) = O(\mathrm{depth}(C))$.*

Proof. Let the inputs to parties be $z^1, \ldots, z^k \in f^{-1}(0)$. Parties descend from the output gate of C to one of the inputs. They maintain the invariant that for the current gate g of C it holds that $g(z^1) = g(z^2) = \ldots = g(z^k) = 0$. If g is not yet an input, then g is a THR_2^{k+1} gate and $g = \mathrm{THR}_2^{k+1}(g_1, \ldots, g_{k+1})$ for some gates $g_1, \ldots, g_{k+1}$. For each z^i we have $g(z^i) = \mathrm{THR}_2^{k+1}(g_1(z^i), \ldots, g_{k+1}(z^i)) = 0$. Hence for each z^i there is at most one gate out of $g_1, \ldots, g_{k+1}$ satisfying $g_j(z^i) = 1$. This means that in $O(1)$ bits of communication parties can agree on the index $j \in [k+1]$ satisfying $g_j(z^1) = g_j(z^2) = \ldots g_j(z^k) = 0$.

Thus in $O(\mathrm{depth}(\pi))$ bits of communication they reach some input of C. If this input contains the variable x_l, then by the invariant $z^1_l = z^2_l = \ldots = z^k_l = 0$, as required. ◀

Exactly the same argument can be applied to the following proposition.

▶ **Proposition 19.** *For any constant $k \geq 2$ the following holds. Assume that $f \in R_k$ and $C \leq f$ is a circuit, consisting only of THR_2^{k+1} gates and literals. Then there is a protocol π, computing R_k-communication game for f, such that $\mathrm{depth}(\pi) = O(\mathrm{depth}(C))$.*

5.2 From protocols to circuits

▶ **Proposition 20.** *For every constant $k \geq 2$ the following holds. Let $f \in Q_k$. Assume that π is a communication protocol computing Q_k-communication game for f. Then there is a circuit $C \leq f$, consisting of THR_2^{k+1} gates and variables, such that $\mathrm{depth}(C) = O(\mathrm{depth}(\pi))$.*

Proof. In the proof we will use the following terminology for strategies in Q_k-hypotheses game. Fix some strategy S. *A current play* is a finite sequence $r_1, r_2, r_3, \ldots r_j$ of integers from 0 to k. By r_i we mean Nature's response in the ith round. Given a current play, let $\mathcal{H}^i_0, \ldots, \mathcal{H}^i_k \subset f^{-1}(0)$ be $k+1$ hypotheses Learner makes in the ith round according to S if Nature's responses in the first $i-1$ rounds were $r_1, \ldots, r_{i-1}$. If after that Nature's response is r_i, then Nature's input vector z satisfies $z \in H^i_{r_i}$. We say that $z \in f^{-1}(0)$ is *compatible* with the current play $r_1, \ldots, r_j$ if $z \in H^1_{r_1}, \ldots, z \in H^j_{r_j}$. Informally, this means that Nature, having z on input, can produce responses $r_1, \ldots, r_j$ by playing against strategy S.

Set $d = \mathrm{depth}(\pi)$. By Proposition 13 it is enough to give a $O(d)$-round winning strategy of Learner in the Q_k-hypotheses game for f. Strategy proceeds in d iterations, each iteration takes $O(1)$ rounds.

As the game goes on, a sequence of Nature's responses $r_1, r_2, r_3 \ldots$ is produced. Assume that $r_1, \ldots, r_{h'}$ are Nature's responses in the first h iteration (here h' is the number of rounds in the first h iterations). Given any $r_1, r_2, r_3 \ldots$, by $\mathcal{Z}_h$ we denote the set of all $z \in f^{-1}(0)$ which are compatible with $r_1, \ldots r_{h'}$, . We also say that elements of $\mathcal{Z}_h$ are *compatible with the current play after h iterations.*

Let V be the set of all nodes of the protocol π and let T be the set of all terminals of the protocol π.

Consider a set $\mathcal{Z} \subset f^{-1}(0)$, a set of nodes $U \subset V$ and a function $g\colon \mathcal{Z} \to C$, where $|C| = k$. *A g-profile* of a tuple $(z^1, \dots, z^k) \in \mathcal{Z}$ is a vector $(g(z^1), \dots, g(z^k)) \in C^k$.

We say that $g\colon \mathcal{Z} \to C$ is *complete* for $\mathcal{Z}$ with respect to the set of nodes U if the following holds. For every vector $\bar{c} \in C^k$ there exists a node $v \in U$ such that all tuples from $\mathcal{Z}^k$ with g-profile $\bar{c}$ visit v in the protocol π.

We say that a set of nodes $U \subset T$ is complete for $\mathcal{Z}$ if there exists $g\colon \mathcal{Z} \to C$, $|C| = k$ which is complete for $\mathcal{Z}$ with respect to U.

Note that we can consider only complete sets of size at most k^k. Formally, if U is complete for $\mathcal{Z}$, then there is a subset $U' \subset U$ of size at most k^k which is also complete for $\mathcal{Z}$. Indeed, there are k^k possible g-profiles and for each we need only one node in U.

▶ **Lemma 21.** *Assume that $U \subset T$ is complete for $\mathcal{Z} \subset f^{-1}(0)$. Then there exists $i \in [n]$ such that $z_i = 0$ for every $z \in \mathcal{Z}$.*

Proof. If $\mathcal{Z}$ is empty, then there is nothing to prove. Otherwise let $g\colon \mathcal{Z} \to C$, $|C| = k$ be complete for $\mathcal{Z}$ with respect to U. Take any vector $\bar{c} = (c_1, \dots, c_k) \in C^k$ such that $\{c_i \mid i \in [k]\} = g(\mathcal{Z})$. There exists a node $v \in U$ such that any tuple from $\mathcal{Z}^k$ with g-profile $\bar{c}$ visits v. Note that v is a terminal of π and let i be the output of π in v. Let us show that for any $z \in \mathcal{Z}$ it holds that $z_i = 0$. Indeed, note that there exists a tuple $\bar{z} \in \mathcal{Z}^k$ which includes z and which has g-profile $\bar{c}$. This tuple visits v. Since π computes Q_k-communication game for f, every element of the tuple $\bar{z}$ should have 0 at the ith coordinate. In particular, this holds for z. ◀

After d iterations Learner should be able to produce an output. For that there should exist $i \in [n]$ such that for any $z \in \mathcal{Z}_d$ it holds that $z_i = 0$. We will use Lemma 21 to ensure that. Namely, we will ensure that there exists $U \subset T$ which is complete for $\mathcal{Z}_d$. Learner achieves this by maintaining the following invariant.

Let us say that a set of nodes U is h-low if every element of U is either a terminal or a node of depth at least h.

▶ **Invariant 22.** *There is a h-low set U which is complete for $\mathcal{Z}_h$.*

This invariant implies that Learner wins in the end, as any d-low set consists only of terminals.

A 0-low set which is complete for $\mathcal{Z}_0 = f^{-1}(0)$ is a set consisting only of the starting node of π.

Assume that Invariant 22 holds after h iterations. Let us show how to perform the next iteration to maintain the invariant. For that we need a notion of *communication profile*.

A communication profile of $z \in f^{-1}(0)$ with respect to a set of nodes $U \subset V$ is a function $p_z\colon U \to \{0, 1\}$. For $v \in U$ the value of $p_z(v)$ is defined as follows. If v is a terminal, set $p_z(v) = 0$. Otherwise let $i \in [k]$ be the index of the party communicating at v. Set $p_z(v)$ to be the bit transmitted by the ith party at v on input z. I.e., p_z for every $v \in U$ contains information where the protocol goes from the node v if the party, communicating at v, has z on input.

We also define a communication profile of the tuple $(z^1, \dots, z^k) \in (f^{-1}(0))^k$ as $(p_{z^1}, \dots, p_{z^k})$.

▶ **Lemma 23.** *Let $(z^1, \dots, z^k), (y^1, \dots, y^k) \in (f^{-1}(0))^k$ be two inputs visiting the same node $v \in V \setminus T$. Assume that their communication profiles with respect to $\{v\}$ coincide. Then these two inputs visit the same successor of v.*

Proof. Let their common communication profile with respect to $\{v\}$ be $(p_1, \ldots, p_k)$. Next, assume that i is the index of the party communicating at v. Then the information where these inputs descend from v is contained in p_i. ◀

Here is what Learner does during the $(h+1)$st iteration. He takes any h-low U of size at most k^k which is complete for $\mathcal{Z}_h$. Then he takes any $g\colon \mathcal{Z}_h \to C$, $|C| = k$ which is complete for $\mathcal{Z}_h$ with respect to U. He now devises a new function g' taking elements of the set $\mathcal{Z}_h$ on input. The value of $g'(z)$ is a pair $(p_z, g(z))$, where p_z is a communication profile of z with respect to U. There are at most $2^{|U|} \leq 2^{k^k}$ different communication profiles with respect to U. Hence $g'(z)$ takes at most $2^{k^k} \cdot k = O(1)$ values.

At each round of the $(h+1)$st iteration Learner asks Nature to identify some pair (p, c), where $p\colon U \to \{0,1\}$ and $c \in C$, such that $g'(z) \neq (p, c)$ for the Nature's vector z. Namely, we take any $k+1$ values of g' which are not yet rejected by Nature and ask Nature to reject one of them. We do so until there are only k possible values $(p_1, c_1), \ldots (p_k, c_k)$ left. This takes $O(1)$ rounds and the $(h+1)$st iteration is finished. Any $z \in f^{-1}(0)$ which is compatible with the responses Nature' gave during the $(h+1)$st iteration in the current play satisfies $g'(z) \in C' = \{(p_1, c_1), \ldots (p_k, c_k)\}$. In particular, any $z \in \mathcal{Z}_{h+1}$ satisfies $g'(z) \in C'$. I.e., the restriction of g' to $\mathcal{Z}_{h+1}$ is a function of the form $g'\colon \mathcal{Z}_{h+1} \to C'$. Let us show that $g'\colon \mathcal{Z}_{h+1} \to C'$ is complete for $\mathcal{Z}_{h+1}$ with respect to some $(h+1)$-low set U'. This will ensure that Invariant 22 is maintained after $h+1$ iterations.

We define U' is follows. Take any vector $\bar{c} \in (C')^k$. It is enough to show that all the inputs from $(\mathcal{Z}_{h+1})^k$ with g'-profile $\bar{c}$ visit the same node v' which is either a terminal or of depth at least $h+1$. Then we just set U' to be the union of all such v' over all possible g'-profiles.

All the tuples from $(\mathcal{Z}_{h+1})^k$ with the same g'-profile visit the same node $v \in U$. This is because g'-profile of a tuple determines its g-profile (the value of g' determines the value of g) , and hence we can use Invariant 22 for $\mathcal{Z}_{h-1}$ here. If v is a terminal, there is nothing left to prove. Otherwise, note that g'-profile of a tuple also determines its communication profile with respect to U and hence with respect to $\{v\} \subset U$. Therefore all the tuples with the same g'-profile by Lemma 23 visit the same successor of v. ◀

With straightforward modifications one can obtain a proof of the following:

▶ **Proposition 24.** *For every constant $k \geq 2$ the following holds. Let $f \in R_k$. Assume that π is a dag-like protocol computing R_k-communication game for f. Then there is a circuit $C \leq f$, consisting of THR_2^{k+1} gates and literals, satisfying $\mathrm{depth}(C) = O(\mathrm{depth}(\pi))$.*

▶ **Corollary 25** (Weak version of Theorem 3)**.** *For any constant $k \geq 2$ there exists $O(\log^2 n)$-depth formula for $\mathrm{THR}_{n+1}^{kn+1}$, consisting only of THR_2^{k+1} gates and variables.*

Proof. We will show that there exists $O(\log^2 n)$-depth protocol π computing Q_k-communication game for $\mathrm{THR}_{n+1}^{kn+1}$. By Proposition 20 this means that there is a $O(\log^2 n)$-depth formula $F \leq \mathrm{THR}_{n+1}^{kn+1}$, consisting only of THR_2^{k+1} gates and variables. It is easy to see that F actually coincides with $\mathrm{THR}_{n+1}^{kn+1}$. Indeed, assume that $F(x) = 0$ for some x with at least $n+1$ ones. Then it is easy to construct $x^2, \ldots, x^k$, each with n ones, such that there is no common 0-coordinate for $x, x^2, \ldots, x^k$. On all of these vectors F takes value 0. However, the function computed by F should belong to Q_k (Proposition 5).

Let π be the following protocol. Assume that the inputs to parties are $x^1, x^2, \ldots, x^k \in \{0,1\}^{kn+1}$, without loss of generality we can assume that in each x^r there are exactly n ones. For $x \in \{0,1\}^{kn+1}$ define $\mathrm{supp}(x) = \{i \in [kn+1] \mid x_i = 1\}$. Let T be a binary rooted tree of

depth $d = \log_2(n) + O(1)$ with $kn+1$ leafs. Identify leafs of T with elements of $[kn+1]$. For a node v of T let T_v be the set of all leafs of T which are descendants of v. Once again, we view T_v as a subset of $[kn+1]$.

The protocol proceeds in at most d iterations. After i iterations, $i = 0, 1, 2, \ldots, d$, parties agree on a node v of T of depth i, satisfying the following invariant:

$$\sum_{r=1}^{k} |\mathrm{supp}(x^r) \cap T_v| < |T_v|. \tag{8}$$

At the beginning Invariant (8) holds just because v is the root, $T_v = [kn+1]$ and each $\mathrm{supp}(x^r)$ is of size n.

After d iterations $v = l$ is a leaf of T. Parties output l. This is correct because by (8) we have $|T_l| = 1 \implies |\mathrm{supp}(x^r) \cap T_l| = 0 \implies x_l = 0$ for every $r \in [k]$.

Let us now explain what parties do at each iteration. If the current v is not a leaf, let v_0, v_1 be two children of v. Each party sends $|\mathrm{supp}(x^r) \cap T_{v_0}|$ and $|\mathrm{supp}(x^r) \cap T_{v_1}|$, using $O(\log n)$ bits. Since T_{v_0} and T_{v_1} is a partition of T_v, we have:

$$\sum_{b=0}^{1} \sum_{r=1}^{k} |\mathrm{supp}(x^r) \cap T_{v_b}| = \sum_{r=1}^{k} |\mathrm{supp}(x^r) \cap T_v| < |T_v| = \sum_{b=0}^{1} |T_{v_b}|.$$

Thus the inequality:

$$\sum_{r=1}^{k} |\mathrm{supp}(x^r) \cap T_{v_b}| < |T_{v_b}| \tag{9}$$

is true either for $b = 0$ or for $b = 1$. Let b^* be the smallest $b \in \{0, 1\}$ for which (9) is true. Parties proceed to the next iteration with v being replaced by v_{b^*}.

There are $d = O(\log n)$ iterations, at each parties communicate $O(\log n)$ bits. Hence π is $O(\log^2 n)$-depth, as required. ◀

▶ Remark 26. Strategy from the proof of Proposition 20 is efficient only in terms of the number of rounds. In the next section we give another version of this strategy. This version will ensure that circuits we obtain from protocols for Q_k-communication games are not only low-depth, but also polynomial-size and explicit. For that, however, we require a bit more from the protocol π.

6 Effective version

Fix $f \in Q_k$. We say that a dag-like communication protocol π *strongly* computes Q_k-communication game for f if for every terminal t of π, for every $x \in f^{-1}(0)$ and for every $i \in [k]$ the following holds. If x is i-compatible with t, then $x_j = 0$, where $j = l(t)$ is the label of terminal t in the protocol π.

Similarly, fix $f \in R_k$. We say that a dag-like communication protocol π *strongly* computes R_k-communication game for f if for every terminal t of π, for every $x \in f^{-1}(0)$ and for every $i \in [k]$ the following holds. If x is i-compatible with t, then $x_j = b$, where $(j, b) = l(t)$ is the label of terminal t in the protocol π.

Strong computability is close to the notion of computability that Sokolov gave in [14] for general relations. Strong computability implies more intuitive notion of computability that we gave in the Preliminaries. The opposite direction is false in general.

Next we prove an effective version of Proposition 20.

▶ **Theorem 27.** *For every constant $k \geq 2$ there exists a polynomial-time algorithm A such that the following holds. Assume that $f \in Q_k$ and π is a dag-like protocol which strongly computes Q_k-communication game for f. Then, given the light form of π, the algorithm A outputs a circuit $C \leq f$, consisting only of THR_2^{k+1} gates and variables, such that* $\mathrm{depth}(C) = O(\mathrm{depth}(\pi))$, $\mathrm{size}(C) = O\left(\mathrm{size}(\pi)^{O(1)}\right)$.

Proof. We will again give a $O(d)$-round winning strategy of Learner in the Q_k-hypotheses game for f. Now, however, we should ensure that the light form of our strategy is of size $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ and can be computed in time $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ from the light form of π. Instead of specifying the light form of our strategy directly we will use the following trick. Assume that Learner has a *working tape* consisting of $O(\log \mathrm{size}(\pi))$ cells, where each cell can store one bit. Learner memorizes all the Nature's responses so that he knows the current position of the game. But he *does not* store the sequence of Nature's responses on the working tape (there is no space for it). Instead, he first makes his hypotheses which depend on the current position. Then he receives a Nature's response $r \in \{0, 1, \ldots, k\}$. And then he *modifies* the working tape, but the result should depend only on the current content of the working tape and on r (and not on the current position in a game). Moreover, we will ensure that modifying the working tape takes $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ time, given the light form of π.

The main purpose of the working tape manifests itself in the end. Namely, at some point Learner decides to stop making hypotheses. This should be indicated on the working tape. More importantly, Learner's output should depend only on the content of working tape in the end (and not on the whole sequence of Nature's responses). Moreover, this should take $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ time to compute that output, given the light form of π.

If a strategy satisfies these restrictions, then its light form is computable in $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ time given the light form of π. Indeed, the underlying dag will consist of all possible configurations of the working tape. There are $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ of them, as working tape uses $O(\log \mathrm{size}(\pi))$ bits. For all non-terminal configurations c we go through all $r \in \{0, 1, \ldots, k\}$. We compute what would be a configuration c_r of the working tape if the current configuration is c and Nature's response is r. After that we connect c to $c_0, c_1, \ldots, c_k$. Finally, in all terminal configurations we compute the outputs of Learner. This gives a light form of our strategy in $O\left(\mathrm{size}(\pi)^{O(1)}\right)$ time.

Let V be the set of nodes of π and T be the set of terminals of π. Strategy proceeds in d iterations, each taking $O(1)$ rounds. We define sets $\mathcal{Z}_h$ exactly as in the proof of Proposition 20. We also use the same notion of communication profile. However, we define completeness in a different way. First of all, instead of working with sets of nodes with no additional structure we will work with *multidimensional arrays* of nodes. Namely, we will consider k-dimensional arrays in which every dimension is indexed by integers from $[k]$. Formally, such arrays are functions of the form $M\colon [k]^k \to V$. We will use notation $M[c_1, \ldots, c_k]$ for the value of M on $(c_1, \ldots, c_k) \in [k]^k$.

Consider any $\mathcal{Z} \subset f^{-1}(0)$. We say that $g\colon \mathcal{Z} \to [k]$ is complete for $\mathcal{Z}$ with respect to a multidimensional array $M\colon [k]^k \to V$ if for every $(c_1, \ldots, c_k) \in [k]^k$, for every $i \in [k]$ and for every $z \in \mathcal{Z}$ the following holds. If $c_i = g(z)$, then z is i-compatible with $M[c_1, \ldots, c_k]$.

We say that a multidimensional array $M\colon [k]^k \to V$ is complete for $\mathcal{Z}$ if there exists $g\colon \mathcal{Z} \to [k]$ which is complete with respect to M.

To digest the notion of completeness it is instructive to consider the case $k = 2$. In this case M is a 2×2 table containing four nodes of π. The function $g\colon \mathcal{Z} \to [2]$ is complete for $\mathcal{Z}$ with respect to M if the following holds. First, for every $z \in \mathcal{Z}$ two nodes in the $g(z)$th *row* of M should be 1-*compatible* with z. Second, for every $z \in \mathcal{Z}$ two nodes in the $g(z)$th *column* of M should be 2-*compatible* with z.

Let us now establish an analog of Lemma 21.

▶ **Lemma 28.** *Assume that $M\colon [k]^k \to T$ is complete for $\mathcal{Z} \subset f^{-1}(0)$. Let l be the output of π in the terminal $M[1,2,\dots,k]$. Then $z_l = 0$ for every $\mathcal{Z}$.*

Proof. Since π strongly computes Q_k-communication game for f, it is enough to show that every $z \in \mathcal{Z}$ is i-compatible with $M[1,2,\dots,k]$ for some i. Take $g\colon \mathcal{Z} \to [k]$ which is complete for $\mathcal{Z}$ with respect to M. By definition z is $g(z)$-compatible with $M[1,2,\dots,k]$. ◀

We now proceed to the description of the Learner's strategy. The working tape of Learner consists of:

- an integer *iter*;
- a multidimensional array $M\colon [k]^k \to V$;
- $O(1)$ additional bits of memory.

Integer *iter* will be at most $d \leq \text{size}(\pi)$ so to store all this information we need $O(\log(\text{size}(\pi)))$ bits, as required. Integer *iter* always equals the number of iterations performed so far (at the beginning $iter = 0$). The array M changes only at the moments when *iter* is incremented by 1. So let M_h denote the content of the array M when $iter = h$.

We call an array of nodes h-low if every node in it is either terminal or of depth at least h. Learner maintains the following invariant.

▶ **Invariant 29.** *M_h is h-low and M_h is complete for $\mathcal{Z}_h$.*

At the beginning Learner sets every element of M_0 to be the starting node of π so that Invariant 29 trivially holds.

Note that every node in M_d is a terminal of π. After d iterations Learner outputs the label of terminal $M_d[1,2,\dots,k]$ in the protocol π. As M_d is complete for $\mathcal{Z}_d$ due to Invariant 29, this by Lemma 28 will be a correct output in the Q_k-hypotheses game for f. Obviously producing the output takes polynomial time given the light form of π and the content of Learner's working tape in the end.

Now we need to perform an iteration. Assume that h iterations passed and Invariant 29 still holds. Let U_h be the set of all nodes appearing in M_h. Take any function $g\colon \mathcal{Z}_h \to [k]$ which is complete for $\mathcal{Z}_h$ with respect to M_h.

For any $z \in f^{-1}(0)$ we denote by p_z a communication profile of z with respect to U_h. Recall that p_z is an element of $\{0,1\}^{U_h}$, i.e., a function from U_h to $\{0,1\}$. At each round of the $(h+1)$st iteration Learner asks Nature to specify some pair $(p,c) \in \{0,1\}^{U_h} \times [k]$ such that $(p_z, g(z)) \neq (p,c)$, where z is the Nature's vector. Learner stores each (p,c) using his $O(1)$ additional bits on the working tape. Learner can do this until there are only k pairs from $(p_1,c_1),\dots,(p_k,c_k) \in \{0,1\}^{U_h} \times [k]$ left which are not rejected by Nature. When this moment is reached, the $(h+1)$st iteration is finished. The iteration takes $2^{|U_h|} \cdot k - k = O(1)$ rounds, as required. For any z compatible with the current play after $h+1$ iterations we know that $(p_z, g(z))$ is among $(p_1,c_1),\dots,(p_k,c_k)$, i.e,

$$(p_z, g(z)) \in \{(p_1,c_1),\dots,(p_k,c_k)\} \text{ for all } z \in \mathcal{Z}_{h+1}. \tag{10}$$

Learner writes $(p_1,c_1),\dots,(p_k,c_k)$ on the working tape (all the pairs that were excluded are on the working tape and hence he can compute the remaining ones). Learner then computes a $(h+1)$-low array M_{h+1} which will be complete for $\mathcal{Z}_{h+1}$. To compute M_{h+1} he will only need to know M_h, $(p_1,c_1),\dots,(p_k,c_k)$ (this information is on the working tape) and the light form of π.

Namely, Learner determines $M_{h+1}[d_1,\dots,d_k]$ for $(d_1,\dots,d_k)\in[k]^k$ as follows. Consider the node $v=M_h[c_{d_1},\dots,c_{d_k}]$. If v is a terminal, then set $M_{h+1}[d_1,\dots,d_k]=v$. Otherwise let $i\in[k]$ be the index of the party communicating at v. Look at p_{d_i}, which can be considered as a function of the form $p_{d_i}\colon U_h\to\{0,1\}$. Define $r=p_{d_i}(v)$. Among two edges, starting at v, choose one which is labeled by r. Descend along this edge from v and let the resulting successor of v be $M_{h+1}[d_1,\dots,d_k]$.

Obviously, computing M_{h+1} takes $O\left(\mathrm{size}(\pi)^{O(1)}\right)$. To show that Invariant 29 is maintained we have to show that ***(a)*** M_{h+1} is $(h+1)$-low and ***(b)*** M_{h+1} is complete for $\mathcal{Z}_{h+1}$.

The first part, ***(a)***, holds because each $M_{h+1}[d_1,\dots,d_k]$ is either a terminal or a successor of a node of depth at least h. For ***(b)*** we define the following function:

$$g'\colon \mathcal{Z}_{h+1}\to[k],\qquad g'(z)=i,\ \text{where } i \text{ is such that } (p_z,g(z))=(p_i,c_i).$$

By (10) this definition is correct. We will show that g' is complete for $\mathcal{Z}_{h+1}$ with respect to M_{h+1}.

For that take any $(d_1,\dots,d_k)\in[k]^k, z\in\mathcal{Z}_{h+1}$ and $i\in[k]$ such that $d_i=g'(z)$. We shall show that z is i-compatible with a node $M_{h+1}[d_1,\dots,d_k]$. By definition of g' we have that $g(z)=c_{d_i}$. As by Invariant 29 function g is complete for $\mathcal{Z}_h$ with respect to M_h, this means that z is i-compatible with $v=M[c_{d_1},\dots,c_{d_k}]$. If v is a terminal, then $M_{h+1}[d_1,\dots,d_k]=v$ and there is nothing left to prove.

Otherwise $v\in V\setminus T$. Let j be the index of the party communicating at v. By definition $M_{h+1}[d_1,\dots,d_k]$ is a successor of v. If $j\neq i$, i.e., not the ith party communicates at v, then any successor of v is i-compatible with z. Finally, assume that $j=i$. Node $M_{h+1}[d_1,\dots,d_k]$ is obtained from v by descending along the edge which is labeled by $r=p_{d_i}(v)$. Hence to show that z is i-compatible with $M_{h+1}[d_1,\dots,d_k]$ we should verify that at v on input z the ith party transmits the bit r. For that again recall that $g'(z)=d_i$, which means by definition of g' that $p_z=p_{d_i}$. I.e., p_{d_i} is the communication profile of z with respect to U_h. In particular, the value $r=p_{d_i}(v)$ is the bit transmitted by the ith party on input z at v, as required. ◀

In the same way one can obtain an analog of the previous theorem for the R_k-case.

▶ **Theorem 30.** *For every constant $k\geq 2$ there exists a polynomial-time algorithm A such that the following holds. Assume that $f\in R_k$ and π is a dag-like protocol which strongly computes R_k-communication game for f. Then, given the light form of π, the algorithm A outputs a circuit $C\leq f$, consisting only of THR_2^{k+1} gates and literals, such that $\mathrm{depth}(C)=O(\mathrm{depth}(\pi))$, $\mathrm{size}(C)=O\left(\mathrm{size}(\pi)^{O(1)}\right)$.*

7 Derivation of Theorems 1 and 3

In this section we obtain Theorems 1 and 3 by devising protocols strongly computing the corresponding Q_k-communication games. Unfortunately, establishing strong computability requires diving into straightforward but tedious technical details, even for simple protocols.

Alternative proof of Theorem 1. We will show that there exists $O(\log n)$-depth protocol π with polynomial-time computable light form, strongly computing Q_2-communication game for MAJ_{2n+1}. By Theorem 27 this means that there is a polynomial-time computable $O(\log n)$-depth formula $F\leq\mathrm{MAJ}_{2n+1}$, consisting only of MAJ_3 gates and variables. From self-duality of MAJ_{2n+1} and MAJ_3 it follows that F computes MAJ_{2n+1}.

Take a polynomial-time computable $O(\log n)$-depth monotone formula F' for MAJ_{2n+1}. Consider the following communication protocol π. The tree of π coincides with the tree of F'. Inputs to F' will be leafs of π. In a leaf containing input variable x_i the output of the protocol π is i. Remaining nodes of π are $\wedge$ and $\vee$ gates. In the $\wedge$ gates communicates the first party, while in the $\vee$ gates communicates the second party.

Fix an $\wedge$ gate g (which belongs to the first party). Let g_0, g_1 be gates which are fed to g, i.e., $g = g_0 \wedge g_1$. There are two edges, starting at g, one leads to g_0 (and is labeled by 0) and the other leads to g_1 (and is labeled by 1). Take an input $a \in \mathrm{MAJ}^{-1}_{2n+1}(0)$ to the first party. On input a at the gate g the first party transmits the bit $r = \min\{c \in \{0,1\} \mid g_c(a) = 0\}$. If the minimum is over the empty set, then we set $r = 0$.

Take now an $\vee$ gate h belonging to the second party. Similarly, there are two edges, starting at h, one leads to h_0 (and is labeled by 0) and the other leads to h_1 (and is labeled by 1). Here h_0, h_1 are two gates which are fed to h, i.e., $h = h_0 \vee h_1$. Take an input $b \in \mathrm{MAJ}^{-1}_{2n+1}(0)$ to the second party. On input b at the gate h the second party transmits the bit $r = \min\{c \in \{0,1\} \mid h_c(\neg b) = 1\}$. If the minimum is over the empty set, then we set $r = 0$. Here $\neg$ denotes the bit-wise negation. Description of the protocol π is finished.

Clearly, the protocol π is of depth $O(\log n)$ and its light form is polynomial-time computable. It remains to argue that the protocol strongly computes Q_2-communication game for MAJ_{2n+1}. Nodes of the protocol may be identified with the gates of F'. Consider any path $p = \langle e_1, \ldots, e_m \rangle$ in the protocol π. Assume that e_j is an edge from g^{j-1} to g^j and g^0 is the output gate of F'. We shall show that the following: if $a \in \mathrm{MAJ}^{-1}_{2n+1}(0)$ is 1-compatible with p, then $g^0(a) = g^1(a) = \ldots = g^m(a) = 0$. Indeed, $g^0(a) = 0$ holds because F' computes MAJ_{2n+1}. Now, assume that $g^j(a) = 0$ is already proved. If g^j is an $\vee$ gate, then $g^{j+1}(a) = 0$ just because g^{j+1} feds to g^j. Otherwise g^j is an $\wedge$ gate which therefore belongs to the first party. Let $r \in \{0,1\}$ is the label of the edge e_{j+1}. Note that $g^{j+1} = g^j_r$, where g^j_0, g^j_1 are two gates which are fed to g^j. . Since a is 1-compatible with p, it holds that r coincides with the bit that the first party transmits at g^j on input a, i.e., with $\min\{c \in \{0,1\} \mid g^j_c(a) = 0\}$. The set over which the minimum is taken is non-empty because $g^j(a) = 0$. In particular r belongs to this set, which means that $g^{j+1}(a) = g^j_r(a) = 0$, as required.

Similarly one can verify that if $b \in \mathrm{MAJ}^{-1}_{2n+1}(0)$ is 2-compatible with p, then $g^0(\neg b) = g^1(\neg b) = \ldots = g^m(\neg b) = 0$. Hence we get that if a leaf l is 1-compatible (2-compatible) with a (b) and l contains a variable x_i, then $a_i = 0$ ($\neg b_i = 1$). Hence the protocol strongly computes the Q_2-communication game for MAJ_{2n+1}. ◀

Proof of Theorem 3. We will realize the protocol from the proof of Corollary 25 in such a way that it will give us $O(\log^2 n)$-depth polynomial-size dag-like protocol with polynomial-time computable light form, strongly computing Q_k-communication game for $\mathrm{THR}^{kn+1}_{n+1}$. By Theorem 27 this means that there is a polynomial-time computable $O(\log^2 n)$-depth polynomial-size circuit $C \leq \mathrm{THR}^{kn+1}_{n+1}$, consisting only of THR^{k+1}_2 gates and variables. With the same argument as in Corollary 25 one can show that C coincides with $\mathrm{THR}^{kn+1}_{n+1}$.

We will use the same tree T as in the proof of Corollary 25. Let us specify the underlying dag G of our protocol π. For a node v of T let $\mathcal{S}_v$ be the set of all tuples $(s_1, s_2, \ldots, s_k) \in \{0, 1, \ldots, kn+1\}^k$ such that $s_1 + s_2 + \ldots + s_k < |T_v|$. For every node v of T and for every $(s_1, s_2, \ldots, s_k) \in \mathcal{S}_v$ the dag G will contain a node identified with a tuple $(v, s_1, s_2, \ldots, s_k)$. These nodes of G will be called the *main nodes* (there will be some other nodes too). The starting node of G will be $(r, n, \ldots, n)$, where r is the root of T. Note that if l is a leaf of T, then $|T_l| = 1$. Hence the only main node having l as the first coordinate is $(l, 0, \ldots, 0)$. The set of terminals of π will coincide with the set of all main nodes of the form $(l, 0, \ldots, 0)$, where l is a leaf of T. The output of π in $(l, 0, \ldots, 0)$ is l.

For an integer $s \leq kn+1$ let $W(s)$ be a binary tree of depth $O(\log n)$ with $|\{(a,b) \mid a,b \in \{0,1,\ldots,s\}, a+b=s\}|$ leaves. We assume that leaves of $W(s)$ are identified with elements of $\{(a,b) \mid a,b \in \{0,1,\ldots,s\}, a+b=s\}$. We use $W(s)$ in the construction of G. Namely, take any main node $(v,s_1,s_2,\ldots,s_k)$ with a non-leaf v. Attach $W(s_1)$ to it. Then attach to every leaf of $W(s_1)$ a copy of $W(s_2)$. Next, to every leaf of the resulting tree attach a copy of $W(s_3)$ and so on. In this way we obtain a binary tree $W(v,s_1,\ldots,s_k)$ of depth $O(\log n)$ growing at $(v,s_1,\ldots,s_k)$. Its leaves can be identified with tuples of integers $(a_1,b_1,\ldots,a_k,b_k)$ satisfying $a_1,b_1,\ldots,a_k,b_k \geq 0, a_1+b_1=s_1,\ldots,a_k+b_k=s_k$. We will merge every leaf of $W(v,s_1,\ldots,s_k)$ with some main node. Namely, take a leaf $(a_1,b_1,\ldots,a_k,b_k)$. If $a_1+\ldots+a_k < |T_{v_0}|$, then we merge $(a_1,b_1,\ldots,a_k,b_k)$ with the main node $(v_0,a_1,\ldots,a_k)$. Otherwise it should hold that $b_1+\ldots+b_k < |T_{v_1}|$. In this case we merge $(a_1,b_1,\ldots,a_k,b_k)$ with the main node $(v_1,b_1,\ldots,b_k)$.

Description of the dag of π is finished. Since k is constant, there are $n^{O(1)}$ main nodes and to each we attach a tree of depth $O(\log n)$. Hence π is $O(\log^2 n)$-depth and $n^{O(1)}$-size. Let us define a partition of non-terminal nodes between parties. Take a main node $(v,s_1,\ldots,s_k)$, where v is not a leaf of T. The tree $W(v,s_1,\ldots,s_k)$, growing from $(v,s_1,\ldots,s_k)$ consists of copies of $W(s_1),\ldots,W(s_k)$. We simply say that the ith party communicates in copies of $W(s_i)$. After that we conclude that the light form of π is polynomial-time computable.

Now let us specify how the ith party communicates inside $W(s_i)$. Assume that $x \in \{0,1\}^{kn+1}$ is the input to the ith party. If $|T_v \cap \mathrm{supp}(x)| \neq s_i$, then the ith party communicates arbitrarily. Now, assume that $|T_v \cap \mathrm{supp}(x)| = s_i$. Then the ith party communicates in such a way that the resulting path descends from the root of $W(s_i)$ to the leaf identified with a pair of integers $(|T_{v_0} \cap \mathrm{supp}(x)|, |T_{v_1} \cap \mathrm{supp}(x)|)$.

From this we immediately get the following observation. Let p be a path from the root of $W(v,s_1,\ldots,s_k)$ to a leaf identified with a tuple $(a_1,b_1,\ldots,a_k,b_k)$. Further, assume that $x \in (\mathrm{THR}^{kn+1}_{n+1})^{-1}(0)$, satisfying $|T_v \cap \mathrm{supp}(x)| = s_i$, is i-compatible with p. Then $a_i = |T_{v_0} \cap \mathrm{supp}(x)|$ and $b_i = |T_{v_1} \cap \mathrm{supp}(x)|$. Indeed, any such p passes though a copy $W(s_i)$ and leaves $W(s_i)$ in a leaf identified with $(|T_{v_0} \cap \mathrm{supp}(x)|, |T_{v_1} \cap \mathrm{supp}(x)|)$.

From this observation one can easily deduce that if $x \in (\mathrm{THR}^{kn+1}_{n+1})^{-1}(0)$ is i-compatible with a main node $(v,s_1,\ldots,s_k)$, then $|T_v \cap \mathrm{supp}(x)| = s_i$. Indeed, we can obtain this by induction on the depth of v. Induction step easily follows from the previous paragraph. As for induction base we notice that $|T_r \cap \mathrm{supp}(x)| = n$ for the root r of T (as in the proof of Corollary 25 we assume that $|\mathrm{supp}(x)| = n$ as party can always add missing 1's).

In particular, this means that π strongly computes Q_k-communication game for $\mathrm{THR}^{kn+1}_{n+1}$. Indeed, any terminal of π is of the form $(l,0,\ldots,0)$, where l is a leaf of T. If $x \in (\mathrm{THR}^{kn+1}_{n+1})^{-1}(0)$ is i-compatible with $(l,0,\ldots,0)$, then, as shown in the previous paragraph, $|T_l \cap \mathrm{supp}(x)| = |\{l\} \cap \mathrm{supp}(x)| = 0$. This means that $x_l = 0$ and hence the output of the protocol is correct. ◀

8 Open problems

- Can Q_k-communication game for $\mathrm{THR}^{kn+1}_{n+1}$ be solved in $O(\log n)$ bits of communication for $k \geq 3$? Equivalently, can $\mathrm{THR}^{kn+1}_{n+1}$ be computed by $O(\log n)$-depth circuit, consisting only of THR^{k+1}_2 and variables? Can a deeper look into the construction of AKS sorting network help here (note that we only use this sorting network as a black-box)?
- Can at least R_k-communication game for $\mathrm{THR}^{kn+1}_{n+1}$ be solved in $O(\log n)$ bits of communication for $k \geq 3$? Again, this is equivalent to asking whether $\mathrm{THR}^{kn+1}_{n+1}$ can be computed by $O(\log n)$-depth circuit, consisting only of THR^{k+1}_2 and *literals*. Note that if we allow

literals (along with $\wedge$ and $\vee$ gates), then there are much simpler constructions of a $O(\log n)$-depth formula for MAJ_n and, in fact, for every symmetric Boolean function [16]. Moreover, this can be done in terms of communication complexity [2]. A natural approach would be to apply ideas of [2] to R_k-communication games.

- Are there any other interesting functions in Q_k and R_k which can be analyzed with our technique?

References

1. Miklós Ajtai, János Komlós, and Endre Szemerédi. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9, 1983. doi:10.1145/800061.808726.
2. Gerth Stølting Brodal and Thore Husfeldt. *A communication complexity proof that symmetric functions have logarithmic depth*. BRICS, Department of Computer Science, Univ., 1996.
3. Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D Rothblum. Efficient multiparty protocols via log-depth threshold formulae. In *Annual Cryptology Conference*, pages 185–202. Springer, 2013. doi:10.1007/978-3-642-40084-1_11.
4. Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *computational complexity*, 27(3):375–462, 2018. doi:10.1007/s00037-017-0159-x.
5. Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson. Toward better formula lower bounds: an information complexity approach to the KRW composition conjecture. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 213–222, 2014. doi:10.1145/2591796.2591856.
6. Oded Goldreich. Valiant's polynomial-size monotone formula for majority, 2011. URL: http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-maj.pdf.
7. Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 847–856, 2014. doi:10.1145/2591796.2591838.
8. Arvind Gupta and Sanjeev Mahajan. Using amplification to compute majority with small majority gates. *Computational Complexity*, 6(1):46–63, 1996. doi:10.1007/BF01202041.
9. Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012. doi:10.1007/978-3-642-24508-4.
10. Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3-4):191–204, 1995. doi:10.1007/BF01206317.
11. Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990. doi:10.1137/0403021.
12. Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020. doi:10.1017/9781108671644.
13. Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 234–243. IEEE, 1997. doi:10.1109/SFCS.1997.646112.
14. Dmitry Sokolov. Dag-like communication and its applications. In *International Computer Science Symposium in Russia*, pages 294–307. Springer, 2017. doi:10.1007/978-3-319-58747-9_26.
15. Leslie G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5(3):363–366, 1984. doi:10.1016/0196-6774(84)90016-6.
16. Ingo Wegener. *The complexity of Boolean functions*. BG Teubner, 1987.

Optimal Error Pseudodistributions for Read-Once Branching Programs

Eshan Chattopadhyay
Department of Computer Science, Cornell University, Ithaca, NY, USA
eshanc@cornell.edu

Jyun-Jie Liao
Department of Computer Science, Cornell University, Ithaca, NY, USA
jjliao@cs.cornell.edu

Abstract

In a seminal work, Nisan (Combinatorica'92) constructed a pseudorandom generator for length n and width w read-once branching programs with seed length $O(\log n \cdot \log(nw) + \log n \cdot \log(1/\varepsilon))$ and error ε. It remains a central question to reduce the seed length to $O(\log(nw/\varepsilon))$, which would prove that $\mathbf{BPL} = \mathbf{L}$. However, there has been no improvement on Nisan's construction for the case $n = w$, which is most relevant to space-bounded derandomization.

Recently, in a beautiful work, Braverman, Cohen and Garg (STOC'18) introduced the notion of a *pseudorandom pseudo-distribution* (PRPD) and gave an explicit construction of a PRPD with seed length $\tilde{O}(\log n \cdot \log(nw) + \log(1/\varepsilon))$. A PRPD is a relaxation of a pseudorandom generator, which suffices for derandomizing $\mathbf{BPL}$ and also implies a hitting set. Unfortunately, their construction is quite involved and complicated. Hoza and Zuckerman (FOCS'18) later constructed a much simpler hitting set generator with seed length $O(\log n \cdot \log(nw) + \log(1/\varepsilon))$, but their techniques are restricted to hitting sets.

In this work, we construct a PRPD with seed length

$$O(\log n \cdot \log(nw) \cdot \log\log(nw) + \log(1/\varepsilon)).$$

This improves upon the construction by Braverman, Cogen and Garg by a $O(\log\log(1/\varepsilon))$ factor, and is optimal in the small error regime. In addition, we believe our construction and analysis to be simpler than the work of Braverman, Cohen and Garg.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Derandomization, explicit constructions, space-bounded computation

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.25

Funding *Eshan Chattopadhyay*: Supported by NSF grant CCF-1849899.
Jyun-Jie Liao: Supported by NSF grant CCF-1849899.

Acknowledgements We thank anonymous reviewers for helpful comments.

1 Introduction

A major challenge in computational complexity is to understand to what extent randomness is useful for efficient computation. It is widely believed that randomness does not provide substantial savings in time and space for algorithms. Indeed, under plausible assumption, every randomized algorithm for decision problem can be made deterministic with only a polynomial factor slowdown in time ($\mathbf{BPP} = \mathbf{P}$) [16] or a constant factor blowup in space ($\mathbf{BPL} = \mathbf{L}$) [20].

However, it remains open for decades to prove these results unconditionally. For derandomization in the time-bounded setting, it is known that proving $\mathbf{BPP} = \mathbf{P}$ implies circuit lower bounds which seem much beyond reach with current proof techniques [18]. However no

such implications are known for the space-bounded setting, and there has been some progress. Savitch's theorem [30] implies that $\mathbf{RL} \subseteq \mathbf{L}^2$. Borodin, Cook, Pippenger [3] and Jung [17] proved that $\mathbf{PL} \subseteq \mathbf{L}^2$, which implies $\mathbf{BPL} \subseteq \mathbf{L}^2$. Nisan [23, 24] constructed a pseudorandom generator for log-space computation with seed length $O(\log^2 n)$, and used it to show that $\mathbf{BPL}$ can be simulated with $O(\log^2 n)$ space and polynomial time. Saks and Zhou [29] used Nisan's generator in a non-trivial way to show that $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$, which remains the best known result so far. We refer the interested reader to the beautiful survey by Saks [28] for more background and relevant prior work.

We introduce the notion of a read-once branching programs, which is a non-uniform model for capturing algorithms that use limited memory.

▶ **Definition 1** (Read-once branching program)**.** *A (n, w)-read-once branching program (ROBP) B is a directed graph on the vertex set $V = \bigcup_{i=0}^{n} V_i$, where each set V_i contains w nodes. Every edge in this directed graph is labeled either 0 or 1. For every $i < n$, and every node $v \in V_i$, there exists exactly two edges starting from v, one with label 0 and the other with label 1. Every edge starting from a node in V_i connects to a node in V_{i+1}. We say n is the* length *of B, w is the* width *of B and V_i is the i-th* layer *of B.*

Moreover, there exists exactly one starting state $s \in V_0$, and exactly one accepting state $t \in V_n$. For every $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, we define $B(x) = 1$ if starting from s we will reach t following the edges labeled by $x_1, \ldots, x_n$. Otherwise we define $B(x) = 0$.

It is well-known the computation of a probabilistic Turing machine that uses space S and tosses n coins, on a given input y, can be carried out by a $(n, 2^{O(S)})$-ROBP B_y. In particular, if the string $x \in \{0,1\}^n$ corresponds to the n coin tosses, then $B_y(x)$ is the output of the Turing machine.

A standard derandomization technique is via *pseudorandom generators*. We define this notion for the class of ROBPs.

▶ **Definition 2** (Pseudorandom generator)**.** *A function $G : \{0,1\}^s \to \{0,1\}^n$ is a (n, w, ε)-pseudorandom generator (PRG) if for every (n, w)-ROBP B,*

$$\left| \mathop{\mathbb{E}}_{x \in \{0,1\}^n} [B(x)] - \mathop{\mathbb{E}}_{r \in \{0,1\}^s} [B(G(r))] \right| \leq \varepsilon.$$

The seed length *of G is s. G is explicit if G is computable in $O(s)$ space.*

To derandomimze space-bounded computation given an explicit (n, w, ε)-PRG, one can enumerate $B(G(r))$ for every $r \in \{0,1\}^s$ with $O(s)$ additional space to compute an ε-approximation of the quantity $\mathbb{E}_x [B(x)]$.

Nisan [23] constructed a (n, w, ε)-PRG with seed length $O(\log n \cdot \log(nw/\varepsilon))$, which implies $\mathbf{BPL} \subseteq \mathbf{L}^2$. While there is a lot of progress in constructing PRG with better seed length for restricted family of ROBP (see, e.g., [25, 1, 6, 2, 5, 21, 9, 31, 22] and references therein), Nisan's generator and its variants [23, 15, 26] remain the best-known generators in the general case.

1.1 Pseudorandom pseudodistribution

Recently, a beautiful work of Braverman, Cohen and Garg [4] introduced the notion of a *pseudorandom pseudodistribution* (PRPD) that relaxes the definition of a PRG.

▶ **Definition 3** (Pseudorandom pseudodistribution). *A pair of functions* $(G, \rho) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ *generates a* (n, w, ε)*-pseudorandom pseudodistribution (PRPD) if for every* (n, w)*-ROBP* B*,*

$$\left| \mathop{\mathbb{E}}_{x \in \{0,1\}^n} [B(x)] - \mathop{\mathbb{E}}_{r \in \{0,1\}^s} [\rho(r) \cdot B(G(r))] \right| \leq \varepsilon.$$

We say s *is the* seed length *of* (G, ρ)*. We say* (G, ρ) *is* k*-bounded if* $|\rho(x)| \leq k$ *for every* $x \in \{0,1\}^s$*. We say* (G, ρ) *is explicit if they are computable in space* $O(s)$*.*

Note that a (n, w, ε)-PRG G of seed length s with a constant function $\rho(x) = 1$ generates a 1-bounded (n, w, ε)-PRPD. Similar to a PRG, it is possible to derandomize **BPL** by enumerating all seeds of a PRPD and computing an ε-approximation for $\mathbb{E}_x [B(x)]$. In [4] they observe that given (G, ρ) which generates an (n, w, ε)-PRPD, the function G itself is an ε-hitting set generator for (n, w)-ROBP.

The main result in [4] is an explicit construction of a (n, w, ε)-PRPD with seed length

$$O\left((\log n \cdot \log(nw) + \log(1/\varepsilon)) \cdot \log\log(nw/\varepsilon)\right),$$

which is $\mathrm{poly}(nw/\varepsilon)$-bounded.[1] This improves on the seed-length of Nisan's generator and provides near optimal dependence on error.

Unfortunately, the construction and analysis in [4] is highly complicated. Hoza and Zuckerman [13] provided a dramatically simpler hitting set generator with slightly improved seed length. However, it is not clear how to extend their techniques for constructing a PRPD (or PRG).

1.2 Main result

In this paper, we construct a PRPD with optimal dependence on error (up to constants).

▶ **Theorem 4.** *There exists an explicit* (n, w, ε)*-PRPD generator* (G, ρ) *with seed length*

$$O\left(\log n \cdot \log(nw) \cdot \log\log(nw) + \log(1/\varepsilon)\right),$$

which is $\mathrm{poly}(1/\varepsilon)$*-bounded.*

This improves upon the construction in [4] by a factor of $O(\log\log(1/\varepsilon))$, for any $\varepsilon < n^{-\Omega(\log(nw)\log\log(nw))}$.

As observed in [4], the small-error regime is well motivated for application to derandomizing space-bounded computation. In particular, Saks and Zhou [29] instantiated Nisan's PRG with error $n^{-\omega(1)}$ to obtain the result $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$. We note that one can replace the PRG in the Saks-Zhou scheme with a PRPD which is $\mathrm{poly}(w, 1/\epsilon)$-bounded, and hence improvements to our result will lead to improved derandomization of **BPL**. We sketch a proof in Appendix A.

Our construction uses a strategy similar to [4] with the following key differences.

[1] Note that in [4], they define $\sum_r \rho(r) B(G(r))$ to be the approximation of $\mathbb{E}_x [B(x)]$. Here we define $\mathbb{E}_r [\rho(r) B(G(r))]$ to be the approximation instead to emphasize the possible loss when plugged into the Saks-Zhou scheme. (See Appendix A for more details.) Therefore a k-bounded PRPD in their definition is actually $2^s k$-bounded in our definition. Nevertheless, it is possible to show that their construction is still $\mathrm{poly}(nw/\varepsilon)$-bounded with our definition.

- The construction in [4] has a more *bottom-up* nature: their construction follows the binary tree structure in Nisan's generator [23], but in each node they maintain a sophisticated "leveled matrix representation" (LMR) which consists of many pieces of small-norm matrices, and they show how to combine pieces in two LMRs one by one to form a LMR in the upper level. Our construction follows the binary tree structure in Nisan's generator, but has a more *top-down* spirit. We give a clean recursive formula which generates a "robust PRPD" for (n, w)-PRPD given robust PRPDs for $(n/2, w)$-ROBP, where a robust PRPD is a family of pseudodistributions such that the approximation error of pseudodistribution drawn from this family is small on average. (A formal definition can be found in Definition 32.) The top-down nature of our construction significantly simplifies the construction and analysis.
- Following [4], we use an averaging sampler in our recursive construction, but we further observe that we can apply a simple "flattening" operation to limit the growth of seed length. With this observation, we not only improve the seed length but also simplify the construction and analysis by avoiding some special case treatments that are necessary in [4]. (Specifically, we do not need the special multiplication rule "outer product" in [4].)

Independent work

Independent work of Cheng and Hoza [7] remarkably prove that a hitting set generator (HSG) for ROBPs can be used for derandomizing **BPL**. Their first result shows that every (n, w)-ROBP f can be deterministically approximated within error ε with an explicit HSG for $(\mathrm{poly}(\frac{nw}{\varepsilon}), \mathrm{poly}(\frac{nw}{\varepsilon}))$-ROBP with seed length s. The space complexity of their first derandomization is $O(s + \log(nw/\varepsilon))$. Their second result shows that every (n, w)-ROBP f can be deterministically approximated within error ε with an explicit HSG for $(n, \mathrm{poly}(w))$-ROBP with seed length s. Their second derandomization has space complexity $O(s + w \log(n/\varepsilon))$, and only requires black-box access to f.

Their first result does not imply better derandomization algorithms with the state-of-art HSGs so far. Plugging in the HSG from [13], their second result gives a black-box derandomization algorithm for (n, w)-ROBP in space $O(\log(n) \log(nw) + w \log(n/\varepsilon))$. This is better than the black-box derandomization with our PRPD for the restricted case of $w = O(1)$. We note that an advantage of PRPDs (over hitting sets) is that they are applicable in the Saks and Zhou's scheme [29] (as mentioned in Appendix A, when applied with Armoni's sampler trick [1]).

Organization

In Section 2, we present the matrix representation of ROBPs, see how a pseudodistribution can be interpreted as matrices, and introduce some basic rules for translating between matrix operations and operations on pseudodistribution. We use Section 3 to present an outline of our main construction and proof. Section 4 contains necessary preliminaries. In Section 5, we formally prove several lemmas about using samplers on approximate matrix multiplication. In Section 6, we present and prove correctness of our main construction. We conclude with possible future directions in Section 7.

2 ROBPs and Matrices

We introduce the matrix representation of ROBPs and some related definitions that are useful in the rest of the paper. First, we setup some notation.

Notation: Given two strings x, y, we use $x\|y$ to denote the concatenation of x and y. For every $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We denote a collection of objects A_i^j with subscript $i \in S$ and superscript $j \in T$ by $[A]_S^T$ for short.

Given a (n, w)-ROBP B with layers $V_0, \ldots, V_n$, we can represent the transition from layer V_{t-1} to V_t by two stochastic matrices M_t^0 and M_t^1 as follows: suppose layer V_j consists of the nodes $\{v_{j,1}, \ldots, v_{j,w}\}$. The entry $(M_t^0)_{i,j} = 1$ if and only if there exist a 0-labeled edge from $v_{t-1,i}$ to $v_{t,j}$ (else $(M_t^0)_{i,j} = 0$). The matrix M_t^1 is defined similarly according to the edges that labeled 1 between layers V_{t-1} and V_t. More generally, we can also represents multi-step transition by a stochastic matrix. That is, for every $0 \le a \le b \le n$, and every $r = (r_{a+1}, \ldots, r_b) \in \{0,1\}^{b-a}$, we can define

$$M_{a..b}^r = \prod_{t=a+1}^{b} M_t^{r_t}$$

which corresponds to the transition matrix from layer a to layer b following the path labeled by r. Note that every row of $M_{a,b}^r$ contains exactly one 1, and the other entries are 0.

An n-step random walk starting from the first layer can be represented with the following matrix:

$$M_{0..n} = \frac{1}{2^n} \sum_{r \in \{0,1\}^n} M_{0..n}^r = \prod_{t=1}^{n} \frac{1}{2} \left(M_t^0 + M_t^1\right).$$

By definition of M_t^0, M_t^1 one can observe that the (i, j) entry of $M_{0..n}$ is the probability that a random walk from $v_{0,i} \in V_0$ reaches $v_{n,j} \in V_n$. Therefore, suppose $v_{0,i} \in V_0$ is the starting state of B, $v_{n,j} \in V_n$ is the accepting state of B, then $\mathbb{E}_x [B(x)]$ equals the (i, j) entry of $M_{0..n}$.

Recall that a generator of a (n, w, ε)-PRPD is a pair of function (G, ρ) such that for every (n, w)-ROBP B,

$$\left| \mathop{\mathbb{E}}_{r} [\rho(r) \cdot B(G(r))] - \mathop{\mathbb{E}}_{x \in \{0,1\}^n} [B(x)] \right| \le \varepsilon.$$

Equivalently, for every transition matrices $M_1^0, M_1^1, \ldots, M_n^0, M_n^1$, we have

$$\left\| \mathop{\mathbb{E}}_{r} \left[\rho(r) \cdot M_{0..n}^{G(r)} \right] - M_{0..n} \right\|_{\max} \le \varepsilon,$$

where $\|A\|_{\max}$ denotes $\max_{i,j} |A(i, j)|$.

Therefore it is natural to represents a PRPD (G, ρ) with a mapping $\mathcal{G} : \{0,1\}^s \to \mathbb{R}^{w \times w}$ where $\mathcal{G}(r) = \rho(r) \cdot M_{0..n}^{G(r)}$. More generally, we will use a notation similar to the "matrix bundle sequence" (MBS) introduced in [4] to represent a PRPD.

▶ **Definition 5.** *Consider a (n, w)-ROBP $[M]_{[n]}^{\{0,1\}}$ and a pair of functions $(G, \rho) : \{0,1\}^{s_{\text{out}}} \times [S_{\text{in}}] \to \{0,1\}^n \times \mathbb{R}$. The* matrix form *of (G, ρ) on $[M]_{[n]}^{\{0,1\}}$ is a mapping $\mathcal{A} : \{0,1\}^{s_{\text{out}}} \times [S_{\text{in}}] \to \mathbb{R}^{w \times w}$ such that for every $x \in \{0,1\}^{s_{\text{out}}}$ and $y \in [S_{\text{in}}]$,*

$$\mathcal{A}(x, y) = \rho(x, y) \cdot M_{0..n}^{G(x,y)}.$$

For every $x \in \{0,1\}^{s_{\text{out}}}$ we abuse the notation and define

$$\mathcal{A}(x) = \mathop{\mathbb{E}}_{y} [\mathcal{A}(x, y)].$$

Besides, we define $\langle \mathcal{A} \rangle = \mathbb{E}_{x,y} [\mathcal{A}(x, y)]$. We say s_{out} is the outer seed length *of $\mathcal{A}$, denoted by $s_{\text{out}}(\mathcal{A})$, and S_{in} is the* inner size *of $\mathcal{A}$, denoted by $S_{\text{in}}(\mathcal{A})$. We also define $s_{\text{in}}(\mathcal{A}) = \lceil \log S_{\text{in}} \rceil$ to be the* inner seed length *of $\mathcal{A}$, and $s(\mathcal{A}) = s_{\text{out}}(\mathcal{A}) + s_{\text{in}}(\mathcal{A})$ to be the* seed length *of $\mathcal{A}$.*

▶ Remark 6. For every fixed x, the collection $\{\mathcal{A}(x,y) : y \in [S_{\text{in}}]\}$ corresponds to the "matrix bundle" in [4]. This should be treated as a collection of matrices which "realizes" the matrix $\mathcal{A}(x)$. The whole structure $\mathcal{A}$ corresponds to the "matrix bundle sequence" in [4], and should be treated as a uniform distribution over the set $\{\mathcal{A}(x) : x \in \{0,1\}^{s_{\text{out}}}\}$.

When the ROBP $[M]_{[n]}^{\{0,1\}}$ is clear in the context, we will use the matrix form $\mathcal{A}$ to represent the pseudodistribution (G,ρ) directly. We will apply arithmetic operations on matrices $\mathcal{A}(x)$, and these operations can be easily translated back to operations on pseudodistributions as follows.

▶ **Definition 7.** *Consider a (n,w)-ROBP $[M]_{[n]}^{\{0,1\}}$, and a pair of function $(F,\sigma) : [S] \to \{0,1\}^n \times \mathbb{R}$. The matrix that is* realized *by (F,σ) on $M_{0..n}$ is $\mathbb{E}_{i\in[S]}\left[\sigma(i)\cdot M_{0..n}^{F(i)}\right]$. We say S is the* size *of (F,σ).*

Scaling the matrix corresponds to scaling the coefficients in the pseudodistribution.

▷ Claim 8. Consider a (n,w)-ROBP $[M]_{[n]}^{\{0,1\}}$, let A be a matrix realized by matrix bundle (F_A,σ_A) on $M_{0..n}$. Then cA is realized by a matrix bundle (F'_A,σ'_A) of size S_A s.t. $F'_A = F_A$ and $\sigma'_A(x) = c\sigma_A(x)$ for every $x \in [S]$.

The summation on matrices corresponds to re-weighting and union on pseudodistributions.

▷ Claim 9. Consider a (n,w)-ROBP $[M]_{[n]}^{\{0,1\}}$, let A be a matrix realized by matrix bundle (F_A,σ_A) of size S_A on $M_{0..n}$ and B be a matrix realized by matrix bundle (F_B,σ_B) of size S_B on $M_{0..n}$. Then $A+B$ is realized by a matrix bundle (F',σ') of size S_A+S_B on $M_{0..n}$ s.t.

$$F'(x) = \begin{cases} F_A(x) & \text{if } x \le S_A \\ F_B(x-S_A) & \text{if } x > S_A \end{cases} \text{ and } \sigma'(x) = \begin{cases} \frac{S_A+S_B}{S_B}\cdot\sigma_A(x) & \text{if } x \le S_A \\ \frac{S_A+S_B}{S_B}\cdot\sigma_B(x-S_A) & \text{if } x > S_A \end{cases}$$

The multiplication on matrices corresponds to concatenation of pseudodistributions.

▷ Claim 10. Consider a (n,w)-ROBP $[M]_{[n]}^{\{0,1\}}$, let A be a matrix realized by matrix bundle (F_A,σ_A) of size S_A on $M_{0..n/2}$ and B be a matrix realized by matrix bundle (F_B,σ_B) of size S_B on $M_{n/2..n}$. Fix a bijection $\pi : [S_A]\times[S_B] \to [S_A\cdot S_B]$. Then AB is realized by a matrix bundle (F',σ') of size $S_A\cdot S_B$ s.t. for every $a \in [S_A], b \in [S_B]$,

$$F'(\pi(a,b)) = F_A(a)\|F_B(b) \text{ and } \sigma'(\pi(a,b)) = \sigma(a)\cdot\sigma(b).$$

3 Proof Overview

In this section we give an outline of our construction and proof. In Section 3.1, we briefly recap how a sampler is used in [4] to achieve better seed length in the small-error regime. We discuss our construction ideas in Section 3.2.

3.1 The sampler argument

Nisan's generator and its variants recursively use a lemma of the following form.

▶ **Lemma 11.** *Consider a (n,w)-ROBP $[M]_{[n]}^{\{0,1\}}$. Let $\mathcal{A}$ be the matrix form of a distribution on $M_{0..n/2}$, and $\mathcal{B}$ be the matrix form of a distribution on $M_{n/2..n}$. Suppose $s(\mathcal{A}) = s(\mathcal{B}) = s$. Then there exists a distribution whose matrix form $\mathcal{C}$ on $M_{0..n}$ of seed length $s + O(\log(w/\delta))$ such that*

$$\|\langle\mathcal{C}\rangle - \langle\mathcal{A}\rangle\langle\mathcal{B}\rangle\|_{\max} \le \delta.$$

This lemma is usually achieved with a pseudorandom object. For example, the INW generator [15] uses a bipartite expander with degree $\mathrm{poly}(w/\delta)$ to construct the distribution $\mathcal{C}$ in the above lemma. That is, for every edge (x, y) in the expander G, they add $\mathcal{A}(x)\mathcal{B}(y)$ into $\mathcal{C}$. A similar lemma can also be obtained with universal hash functions [23] or seeded extractors [26]. By recursively constructing good approximations of $M_{0..n/2}$ and $M_{n/2..n}$ and applying Lemma 11, one can obtain a PRG which has seed length $O(\log n \cdot \log(nw/\varepsilon))$ (δ is taken to be ε/n because of a union bound). Observe that in such constructions, one needs to pay $O(\log(1/\varepsilon))$ (in seed length) per level of recursion.

The crucial idea in [4] is to amortize this cost over all $\log n$ levels. What makes this possible is the following argument, which we will refer to as the *sampler argument*. First we define the notion of an averaging sampler.

▶ **Definition 12.** *A function $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is an (ε, δ)-(averaging) sampler if for every function $f : \{0,1\}^m \to [0,1]$,*

$$\Pr_{x\in\{0,1\}^n}\left[\left|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}\left[f(g(x,s))\right] - \mathop{\mathbb{E}}_{y\in\{0,1\}^m}\left[f(y)\right]\right| \le \varepsilon\right] \ge 1-\delta.$$

The crucial observation in [4] is that if one uses a sampler to prove Lemma 11, the error actually scales with the norm of one of the matrix forms.

▶ **Lemma 13** ([4])**.** *Consider a (n, w)-ROBP with matrix representation $[M]_{[n]}^{\{0,1\}}$. Let $\mathcal{A}$ and $\mathcal{B}$ be (pseudo)distributions in matrix form on $M_{0..n/2}$ and $M_{n/2..n}$ respectively. Let $n = s_{\mathrm{out}}(\mathcal{A})$, $m = s_{\mathrm{out}}(\mathcal{B})$. Suppose $\forall x \in \{0,1\}^n, \|\mathcal{A}(x)\| \le 1$ and $\forall y \in \{0,1\}^m, \|\mathcal{B}(y)\| \le 1$. Let $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (ε, δ) sampler. Then there exists a (pseudo)distribution $\mathcal{C}$ such that*

$$\|\langle C\rangle - \langle A\rangle \langle B\rangle\| \le O\left(w^2\left(\delta + \varepsilon \mathop{\mathbb{E}}_x\left[\|\mathcal{A}(x)\|\right]\right)\right).$$

Besides, $\mathcal{C}$ has outer seed length $n = s_{\mathrm{out}}(\mathcal{A})$, and for every $x \in \{0,1\}^n$,

$$\mathcal{C}(x) = \mathop{\mathbb{E}}_s\left[\mathcal{A}(x)\mathcal{B}\left(g(x,s)\right)\right].$$

Note that $s_{\mathrm{in}}(\mathcal{C}) = s_{\mathrm{in}}(\mathcal{A}) + s_{\mathrm{in}}(\mathcal{B}) + d$.

The intuition behind this approximation is as follows. If we want to compute the matrix product precisely, we take every $\mathcal{A}(x)$ and multiply it with $\mathbb{E}_y[\mathcal{B}(y)]$. However, with the help of sampler, we can use x as our seed to select some samples from $\mathcal{B}$, and take their average as an estimate of $\mathbb{E}_y[\mathcal{B}(y)]$. The error of this approximation comes in two different way. For those x which are not good choices of a seed for the sampler, the samples chosen with such an x can deviate from the average arbitrarily. However, only δ fraction of x can be bad, so they incur at most δ error. The second kind of error is the estimation error between average of samples $\mathbb{E}_s[\mathcal{B}(g(x,s))]$ and the real average $\mathbb{E}_y[\mathcal{B}(y)]$, which can be at most ε. Since this gets multiplied with $\mathcal{A}(x)$, this kind of error actually scales with $\|\mathcal{A}(x)\|$. Although the first kind of error (which is δ) does not benefit from $\|\mathcal{A}\|$ being small, in [4] they observe that, the parameter δ has almost no influence on the seed length in some cases. To discuss this more precisely, we first recall explicit constructions of samplers.

▶ **Lemma 14** ([27, 10])**.** *For every $\delta, \varepsilon > 0$ and integer m, there exists a space efficient (ε, δ)-sampler $f : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ s.t. $d = O(\log\log(1/\delta) + \log(1/\varepsilon))$ and $n = m + O(\log(1/\delta)) + O(\log(1/\varepsilon))$.*

Note that in Lemma 13, $s(\mathcal{C}) = s(\mathcal{A}) + d + s_{\mathrm{in}}(\mathcal{B})$. Therefore if $n \ge m + O(\log(1/\delta)) + O(\log(1/\varepsilon))$, δ has almost no impact on the seed length.

To use the above ideas, it boils down to working with matrices with small norm, and making sure that every multiplication is "unbalanced" enough so that δ has no impact. [4] applies a delicate telescoping sum trick (which they called "delta sampler") to divide an ε-approximation into a base approximation with $1/\mathrm{poly}(n)$ error and several "correcting terms" which have small norm. By carefully choosing the samplers and discarding all the non-necessary terms, they roughly ensure the following properties: first, a matrix with large seed length must have small norm; second, every matrix multiplication is unbalanced enough so that δ has no impact on the seed length.

With these properties and the sampler argument, they show that the total seed length is bounded by $\tilde{O}(\log(1/\varepsilon) + \log n \log(nw))$.

3.2 Our construction

In executing the ideas sketched above, the construction and analysis in [4] turns out to be quite complicated and involved. One thing which complicates the construction and analysis is its *bottom-up* nature. That is, when multiplying two terms, they create more terms with the telescoping sum trick. Moreover, in the telescoping sum trick one needs to choose the parameters of each sampler very carefully to make sure the seed length of each term does not exceed its "smallness".

Our first step toward a simpler construction is the following *top-down* formula, which we will apply recursively to compute an approximation of $M_{0..n}$:

▶ **Lemma 15.** *Let $\|\cdot\|$ be a sub-multiplicative matrix norm, and A, B be two matrices s.t. $\|A\|, \|B\| \le 1$. Let $k \in \mathbb{N}$ and $\gamma < 1$. For every $0 \le i \le k$, let A_i be a γ^{i+1}-approximation of A, and let B_i be a γ^{i+1}-approximation of B. Then*

$$\sum_{i=0}^{k} A_i B_{k-i} - \sum_{i=0}^{k-1} A_i B_{k-1-i}$$

is a $((k+2)\gamma^{k+1} + (k+1)\gamma^{k+2})$-approximation of AB.

Proof. We have,

$$\begin{aligned}
&\left\| (\sum_{i=0}^{k} A_i B_{k-i} - \sum_{i=0}^{k-1} A_i B_{k-1-i}) - AB \right\| \\
&= \left\| \sum_{i=0}^{k} (A - A_i)(B - B_{k-i}) - \sum_{i=0}^{k-1} (A - A_i)(B - B_{k-1-i}) + (A_k - A)B + A(B_k - B) \right\| \\
&\le \sum_{i=0}^{k} \|A - A_i\| \cdot \|B - B_{k-i}\| + \sum_{i=0}^{k-1} \|A - A_i\| \cdot \|B - B_{k-1-i}\| \\
&+ \|A_k - A\| \cdot \|B\| + \|A\| \cdot \|B_k - B\| \\
&\le (k+2)\gamma^{k+1} + (k+1)\gamma^{k+2}
\end{aligned}$$

◀

This formula shares an important property with the BCG construction: we never need a γ^k-approximation (which implies large seed length) on both sides simultaneously. The benefit of our top-down formula is that we are treating the PRPD as one object instead of the sum of many different terms. One obvious effect of such treatment is we don't need to analyze the "smallness" of each term and the accuracy of the whole PRPD separately.

In this top-down formula, we do not explicitly maintain small-norm matrices as in [4]. However, observe that in the proof of Lemma 15, we are using the fact that $A_k - A$ is a small norm matrix. Our goal is to apply the sampler argument (Lemma 13) on these "implicit" small-norm matrices. The following is our main technical lemma.

▶ **Lemma 16** (main lemma, informal). *Let $A, B \in \mathbb{R}^{w\times w}$, $k \in \mathbb{N}$ and $\gamma < 1$. Suppose for every $i \le k$ there exists pseudodistribution $\mathcal{A}_i, \mathcal{B}_i$ such that $\mathbb{E}_x\left[\|\mathcal{A}_i(x) - A\|\right] \le \gamma^{i+1}$, $\mathbb{E}_x\left[\|\mathcal{B}_i(x) - B\|\right] \le \gamma^{i+1}$, and $\|\mathcal{A}_i(x)\|, \|\mathcal{B}_i(x)\| \le 1$ for every x. Then there exists a pseudodistribution $\mathcal{C}_k$ such that*

$$\mathop{\mathbb{E}}_{x}\left[\|\mathcal{C}_k(x) - AB\| \le O(\gamma)^{k+1}\right],$$

where $\mathcal{C}_k(x) = \sum_{i+j=k} A_{x,i}B_{x,j} - \sum_{i+j=k-1} A_{x,i}B_{x,j}$. $A_{x,i}$ and $B_{x,i}$ are defined as follows.

- *If $i > \lceil k/2\rceil$, $A_{x,i} = \mathcal{A}_i(x)$ and $B_{x,i} = \mathcal{B}_i(x)$.*
- *If $i \le \lceil k/2\rceil$, $A_{x,i} = \mathbb{E}_s\left[\overline{\mathcal{A}_i}(g_i(x,s))\right]$ and $B_{x,i} = \mathbb{E}_s\left[\overline{\mathcal{B}_i}(g_i(x,s))\right]$, where g_i is a $(\gamma^{i+1}, \gamma^{k+1})$-sampler, and $\overline{\mathcal{A}_i}, \overline{\mathcal{B}_i}$ denote the "flattened" form of $\mathcal{A}_i$ and $\mathcal{B}_i$.*

We leave the explanation of "flattened" for later and explain the intuition behind the lemma first. Our goal is to construct $\mathcal{C}_k$ such that $\mathcal{C}_k(x)$ is a good approximation of AB *on average* over x. We know that $\mathcal{A}_i$ and $\mathcal{B}_i$ are γ^{i+1}-approximation of A and B on average. Our hope is to use x to draw samples A_i and B_i from $\mathcal{A}_i$ and $\mathcal{B}_i$, and apply the formula in Lemma 15 to get a good approximation of AB. In particular, a natural choice would be setting $A_{x,i} = \mathcal{A}_i(x)$ and $B_{x,i} = \mathcal{B}_i(x)$ for every $i \le k$. However, if there exists a term $A_{x,i}B_{x,j}$ such that $A_{x,i}$ and $B_{x,j}$ are both bad approximation for a large enough fraction of x, we cannot guarantee to get a $O(\gamma^{k+1})$-approximation on average.

To avoid the above case, for every $i \le \lceil k/2\rceil$ we use a sampler to approximate $\langle\mathcal{A}_i\rangle$ and $\langle\mathcal{B}_i\rangle$. This ensure that the chosen samples $A_{x,i}$ and $B_{x,i}$ are good *with high probability*. This guarantees that in each term $A_{x,i}B_{x,j}$, at least one of $A_{x,i}$ or $B_{x,j}$ will be a good choice with high probability over x. If $A_{x,i}$ is a good choice with high probability, we can apply the average-case guarantee on $B_{x,i}$ to get an average-case guarantee for $\mathcal{C}_k$, and vice versa. (Indeed, this is the sampler argument.) Therefore we can ensure that $\mathcal{C}_k(x)$ is good on average. Note that we only apply a sampler on $\mathcal{A}_i$ (or $\mathcal{B}_i$) when $i \le \lceil k/2\rceil$, which means $\mathcal{A}_i$ (or $\mathcal{B}_i$) has small seed length. Therefore we don't need to add too much redundant seed to make the sampler argument work.

In executing the above sketched idea, we run into the following problem: in each multiplication, the inner seed on both sides aggregates to the upper level. If we start with pseudodistributions with non-zero inner seed in the bottom level, the inner seed would become $\Omega(n)$ in the topmost level. Therefore we need a way to limit the aggregation of inner seed.

In [4], they run into a similar problem. To deal with this, they apply a different multiplication rule, "outer product", in some special cases to deal with this. However, the outer product does not seem applicable in our construction. Nevertheless, we observe that whenever we use a sampler to select matrix $A_{x,i}$, we only care about whether $\langle\mathcal{A}_i\rangle$ is close to A, and we don't need most of $\mathcal{A}_i(x)$ to be close to A anymore. Therefore we will "flatten" $\mathcal{A}_i$ whenever we apply a sampler. That is, recall that each $\mathcal{A}_i(x)$ is realized by the average of some matrices, $\mathbb{E}_y\left[\mathcal{A}_i(x,y)\right]$. We define the flattened form of $\mathcal{A}_i$, denoted by $\overline{\mathcal{A}_i}$, such that $\overline{\mathcal{A}_i}(x\|y) = \mathcal{A}_i(x,y)$. Observe that $\langle\overline{\mathcal{A}_i}\rangle = \langle\mathcal{A}_i\rangle$ and $s_{\text{in}}(\overline{\mathcal{A}_i}) = 0$. This guarantees that the inner seed length of $\mathcal{A}_i$ will not aggregate in $\mathcal{C}_k$. Moreover, while the flattening will increase the outer seed length of $\overline{\mathcal{A}_i}$, this is almost for free since we only flatten $\mathcal{A}_i$ when $i \le \lceil k/2\rceil$, i.e. when $\mathcal{A}_i$ has relatively small seed length. As a result, this operation also helps us save a $O(\log\log(1/\varepsilon))$ factor in the seed length.

We conclude by briefly discussing the seed length analysis. First note that we set $\gamma = 1/\mathrm{poly}(n)$ to make sure that the error is affordable after a union bound. Now consider the inner seed length. Consider a term A_iB_j such that $i \geq j$. In this term, part of the inner seed of $\mathcal{C}$ is passed to $\mathcal{A}_i$, and the other is used for the sampler on B_j. Since the seed length of the sampler only needs to be as large as the "precision gap" between $\mathcal{A}_i$ and $\mathcal{C}_k$, the inner seed length of $\mathcal{C}_k$ can be maintained at roughly $O(k\log(1/\gamma)) = O(\log(1/\varepsilon))$. However, after each multiplication, there's actually a $O(\log(nw/\gamma)) = O(\log(nw))$ additive overhead. Note that this is necessary since the $k = 0$ case degenerates to the INW generator. Therefore after $\log n$ levels of recursion, the inner seed length will be $O(\log(1/\varepsilon) + \log n \cdot \log(nw))$.

Besides, we also need the outer seed length of $\mathcal{C}_k$ to be long enough so that we can apply a sampler on $\mathcal{A}_{\lceil k/2\rceil}$ and $\mathcal{B}_{\lceil k/2\rceil}$. The seed length caused by approximation accuracy ε can be bounded similarly as the inner seed length. However, the $O(\log n \cdot \log(nw))$ inner seed length will be added to the outer seed length several times, because of the flattening operation. Nevertheless, since we only do flattening for $\mathcal{A}_i$ and $\mathcal{B}_i$ where $i \leq \lceil k/2\rceil$, this ensures that the flattening operation happens at most $\log k$ times. So the total outer seed length will be bounded by $O(\log(1/\varepsilon) + \log k \cdot \log n \cdot \log(nw)) = O(\log(1/\varepsilon) + \log\log(1/\varepsilon) \cdot \log n \cdot \log(nw))$, which is bounded by $O(\log(1/\varepsilon) + \log\log(nw) \cdot \log n \cdot \log(nw))$ since $O(\log(1/\varepsilon))$ is the dominating term when $\log(1/\varepsilon) \geq \log^3(nw)$.

4 Preliminaries

4.1 Averaging samplers

▶ **Definition 17.** *A function $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (ε, δ)* (averaging) sampler *if for every function $f : \{0,1\}^m \to [0,1]$,*

$$\Pr_{x\in\{0,1\}^n}\left[\left|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}[f(g(x,s))] - \mathop{\mathbb{E}}_{y\in\{0,1\}^m}[f(y)]\right| \leq \varepsilon\right] \geq 1-\delta.$$

It's easy to show that samplers also work for f with general range by scaling and shifting.

▷ Claim 18. Let $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (ε,δ)-sampler, and let $\ell < r \in \mathbb{R}$. Then for every $f : \{0,1\}^m \to [\ell, r]$,

$$\Pr_{x\in\{0,1\}^n}\left[\left|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}[f(g(x,s))] - \mathop{\mathbb{E}}_{y\in\{0,1\}^m}[f(y)]\right| \leq \varepsilon(r-\ell)\right] \geq 1-\delta.$$

Proof. Let f' be the function such that $f'(y) = (f(y)-\ell)/(r-\ell)$. Observe that the range of f' is in $[0,1]$. By definition of sampler,

$$\Pr_{x\in\{0,1\}^n}\left[\left|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}[f'(g(x,s))] - \mathop{\mathbb{E}}_{y\in\{0,1\}^m}[f'(y)]\right| \leq \varepsilon\right] \geq 1-\delta.$$

By multiplying $(r-\ell)$ on both sides of the inequality inside the probability above we prove the claim. ◁

In our construction, we will use the following sampler which is explicitly computable with small space.

▶ **Lemma 19** ([27, 10])**.** *For every $\delta, \varepsilon > 0$ and integer m, there exists a (ε,δ)-sampler $f : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ s.t. $d = O(\log\log(1/\delta) + \log(1/\varepsilon))$ and $n = m + O(\log(1/\delta)) + O(\log(1/\varepsilon))$. Moreover, for every x, y, $f(x,y)$ can be computed in space $O(m + \log(1/\delta) + \log(1/\varepsilon))$.*

▶ Remark 20. The original sampler in [27] has a restriction on ε. Such a restriction will cause a $2^{O(\log^*(nw/\varepsilon))}$ factor in our construction, as in [4]. However, [27] pointed out that the restriction is inherited from the extractor in [33], which breaks down when the error is extremely small. As observed in [10], this restriction can be removed by plugging in a more recent extractor construction in [12]. Note that there exists a space-efficient implementation of [12] in [19], so the resulting sampler is also space-efficient. For completeness we include a proof in Appendix B.

4.2 Matrix norms

As in [4], we will use the infinity norm in this paper.

▶ **Definition 21.** *For every matrix* $A \in \mathbb{R}^{w\times w}$, $\|A\| = \max_i \sum_j |A_{i,j}|$.

We record some well known properties of the infinity norm.

▷ Claim 22. Let $A, B \in \mathbb{R}^{w\times w}$, $c \in \mathbb{R}$. Then

- $\|cA\| = |c|\,\|A\|$
- $\|A\| + \|B\| \le \|A + B\|$
- $\|AB\| \le \|A\|\,\|B\|$
- $\max_{i,j} |A_{i,j}| \le \|A\|$
- If A is stochastic, then $\|A\| = 1$

Note that for any (n, w)-ROBP represented by $w \times w$ matrices $M_{[n]}^{\{0,1\}}$, $\|M_{i..j}\| = 1$ for every $0 \le i \le j \le n$.

5 Approximate Matrix Multiplication via Samplers

In this section we formally prove the sampler arguments which will be used in our construction. Our proof strategy resembles that of [4], with the following two crucial differences. First, we will define two different notions of "smallness" for our flattening idea. Second, in our construction we need the case where we use samplers to select matrices on both sides (Lemma 27).

We will consider mappings $\mathcal{A} : \{0,1\}^n \to \mathbb{R}^{w\times w}$ which correspond to the implicit small norm matrices we discussed in the previous section. Borrowing notation from Definition 5, we use $\langle\mathcal{A}\rangle$ to denote $\mathbb{E}_x[\mathcal{A}(x)]$. First we define two different norms for the mapping $\mathcal{A}$. The *robust norm* is similar to the notion of "smallness" in [4], i.e. the average of norm of $\mathcal{A}(x)$, while the *norm* of $\mathcal{A}$ is simply the norm of $\langle\mathcal{A}\rangle$, i.e. the norm of average of $\mathcal{A}(x)$.

▶ **Definition 23.** *For every function* $\mathcal{A} : \{0,1\}^n \to \mathbb{R}^{w\times w}$, *we define the* norm *of* $\mathcal{A}$ *to be* $\|\mathcal{A}\| = \left\|\mathbb{E}_{x\in\{0,1\}^n}[\mathcal{A}(x)]\right\|$, *and the* robust norm *of* $\mathcal{A}$ *to be* $\|\mathcal{A}\|_r = \mathbb{E}_{x\in\{0,1\}^n}[\|\mathcal{A}(x)\|]$. *Besides, we define the weight of* $\mathcal{A}$ *to be* $\mu(\mathcal{A}) = \max_x \|\mathcal{A}(x)\|$.

▷ Claim 24. $\|\mathcal{A}\| \le \|\mathcal{A}\|_r \le \mu(\mathcal{A})$.

Proof. $\|\mathcal{A}\| \le \|\mathcal{A}\|_r$ is by sub-additivity of $\|\cdot\|$, and $\|\mathcal{A}\|_r \le \mu(\mathcal{A})$ since $\|\mathcal{A}\|_r$ is the average of values no larger than $\mu(\mathcal{A})$. ◁

Next we show a simple lemma which will be used later. That is, a sampler for functions with range $[0, 1]$ is also a sampler for matrix-valued functions, where the error is measured with infinity norm.

▶ **Lemma 25.** *For every function $\mathcal{A} : \{0,1\}^m \to \mathbb{R}^{w\times w}$ and every (ε,δ)-sampler $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$,*

$$\Pr_{x\in\{0,1\}^n}\left[\left\|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}\left[\mathcal{A}(g(x,s))\right]-\langle\mathcal{A}\rangle\right\| \le 2w\mu(\mathcal{A})\varepsilon\right] \ge 1-w^2\delta.$$

Proof. Let $\mathcal{E}(y) = \mathcal{A}(y) - \langle\mathcal{A}\rangle$. For every $i,j\in[w]$, observe that

$$\max_y \mathcal{E}(y)_{i,j} - \min_y \mathcal{E}(y)_{i,j} = \max_y \mathcal{A}(y)_{i,j} - \min_y \mathcal{A}(y)_{i,j}$$

By the property of sampler it follows that

$$\Pr_{x\in\{0,1\}^n}\left[\left|\mathop{\mathbb{E}}_{s}\left[\mathcal{E}(g(x,s))_{i,j}\right]\right| \le 2\varepsilon\mu(\mathcal{A})\right] \ge 1-\delta.$$

Using a union bound,

$$\Pr_{x\in\{0,1\}^n}\left[\forall i,j\in[w], \left|\mathop{\mathbb{E}}_{s}\left[\mathcal{E}(g(x,s))_{i,j}\right]\right| \le 2\varepsilon\mu(\mathcal{A})\right] \ge 1-w^2\delta.$$

Thus by definition of the infinity norm, we can conclude that

$$\Pr_{x\in\{0,1\}^n}\left[\left\|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}\left[\mathcal{E}(g(x,s))\right]\right\| \le 2w\mu(\mathcal{A})\varepsilon\right] \ge 1-w^2\delta.$$

which by sub-additivity of $\|\cdot\|$ implies

$$\Pr_{x\in\{0,1\}^n}\left[\left\|\mathop{\mathbb{E}}_{s}\left[\mathcal{A}(g(x,s))\right]\right\| \le \|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon\right] \ge 1-w^2\delta. \quad ◀$$

▶ **Corollary 26.** *For every function $\mathcal{A} : \{0,1\}^m \to \mathbb{R}^{w\times w}$ and every (ε,δ)-sampler $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$,*

$$\Pr_{x\in\{0,1\}^n}\left[\left\|\mathop{\mathbb{E}}_{s\in\{0,1\}^d}\left[\mathcal{A}(g(x,s))\right]\right\| \le \|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon\right] \ge 1-w^2\delta.$$

Proof. By sub-additivity of $\|\cdot\|$, $\left\|\mathbb{E}_{s\in\{0,1\}^d}\left[\mathcal{A}(g(x,s))\right]-\langle A\rangle\right\| \le 2w\mu(\mathcal{A})\varepsilon$ implies $\left\|\mathbb{E}_{s\in\{0,1\}^d}\left[\mathcal{A}(g(x,s))\right]\right\| \le \|\langle\mathcal{A}\rangle\| + 2w\mu(\mathcal{A})\varepsilon$. The claim now directly follows from Lemma 25. ◀

Now we introduce three different matrix multiplication rules. The first one is applying a sampler on both sides, and the second and third are applying sampler on only one side.

▶ **Lemma 27** (symmetric product)**.** *Consider $\mathcal{A} : \{0,1\}^n \to \mathbb{R}^{w\times w}$ and $\mathcal{B} : \{0,1\}^m \to \mathbb{R}^{w\times w}$. Let $f : \{0,1\}^k \times \{0,1\}^{d_A} \to \{0,1\}^n$ be a (δ,ε_A) sampler, and $g : \{0,1\}^k \times \{0,1\}^{d_B} \to \{0,1\}^m$ be a (δ,ε_B) sampler. Then*

$$\mathop{\mathbb{E}}_{z}\left[\left\|\mathop{\mathbb{E}}_{x,y}\left[\mathcal{A}(f(z,x))\mathcal{B}(g(z,y))\right]\right\|\right] \le 2w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \left(\|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon_A\right)\left(\|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right).$$

Proof. Let

$$E_A = \left\{z : \left\|\mathop{\mathbb{E}}_{x}\left[\mathcal{A}(f(z,x))\right]\right\| > \|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon_A\right\},$$

and

$$E_B = \left\{z : \left\|\mathop{\mathbb{E}}_{y}\left[\mathcal{B}(g(z,y)\right]\right\| > \|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right\}.$$

Define $E = E_A \cup E_B$. By Lemma 26 and union bound, $\Pr_z[z \in E] < 2w^2\delta$. Therefore

$$\begin{aligned}
\mathop{\mathbb{E}}_z\left[\left\|\mathop{\mathbb{E}}_{x,y}[\mathcal{A}(f(z,x))\mathcal{B}(g(z,y))]\right\|\right] &= \Pr[z \in E]\mathop{\mathbb{E}}_{z\in E}\left[\left\|\mathop{\mathbb{E}}_{x,y}[\mathcal{A}(f(z,x))\mathcal{B}(g(z,y))]\right\|\right] \\
&\quad + \Pr[z \notin E]\mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathop{\mathbb{E}}_{x}[\mathcal{A}(f(z,x))]\mathop{\mathbb{E}}_{y}[\mathcal{B}(g(z,y))]\right\|\right] \\
&\le 2w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathop{\mathbb{E}}_{x}[\mathcal{A}(f(z,x))]\right\|\left\|\mathop{\mathbb{E}}_{y}[\mathcal{B}(g(z,y))]\right\|\right] \\
&\le 2w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \left(\|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon_A\right)\left(\|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right).
\end{aligned}$$

The second last inequality is by the fact that $\|\cdot\|$ is non-negative and sub-multiplicative. ◀

▶ **Lemma 28** (left product). *Consider $\mathcal{A} : \{0,1\}^k \to \mathbb{R}^{w\times w}$ and $\mathcal{B} : \{0,1\}^m \to \mathbb{R}^{w\times w}$. Let $g : \{0,1\}^k \times \{0,1\}^{d_B} \to \{0,1\}^m$ be a (δ, ε_B) sampler. Then*

$$\mathop{\mathbb{E}}_z\left[\left\|\mathop{\mathbb{E}}_{y}[\mathcal{A}(z)\mathcal{B}(g(z,y))]\right\|\right] \le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \|\mathcal{A}\|_r\left(\|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right).$$

Proof. Let

$$E = \left\{z : \left\|\mathop{\mathbb{E}}_{y}[\mathcal{B}(g(z,y)]\right\| > \|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right\}.$$

By Lemma 26, $\Pr_z[z \in E] < w^2\delta$. Therefore

$$\begin{aligned}
\mathop{\mathbb{E}}_z\left[\left\|\mathop{\mathbb{E}}_{y}[\mathcal{A}(z)\mathcal{B}(g(z,y))]\right\|\right] &= \Pr[z \in E]\mathop{\mathbb{E}}_{z\in E}\left[\left\|\mathop{\mathbb{E}}_{y}[\mathcal{A}(z)\mathcal{B}(g(z,y))]\right\|\right] \\
&\quad + \Pr[z \notin E]\mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathop{\mathbb{E}}_{y}[\mathcal{A}(z)\mathcal{B}(g(z,y))]\right\|\right] \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \Pr[z \notin E]\cdot\mathop{\mathbb{E}}_{z\notin E}\left[\|\mathcal{A}(z)\|\left\|\mathop{\mathbb{E}}_{y}[\mathcal{B}(g(z,y))]\right\|\right] \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \Pr[z \notin E]\mathop{\mathbb{E}}_{z\notin E}[\|\mathcal{A}(z)\|]\cdot(\|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B) \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \|\mathcal{A}\|_r\left(\|\mathcal{B}\| + 2w\mu(\mathcal{B})\varepsilon_B\right).
\end{aligned}$$

The third last inequality is by sub-multiplicativity of $\|\cdot\|$, the second last inequality is by non-negativity of $\|\cdot\|$, and the last inequality is by the fact that

$$\Pr[z \notin E]\cdot\mathop{\mathbb{E}}_{z\notin E}[\|\mathcal{A}(z)\|] = \mathop{\mathbb{E}}_z[\|\mathcal{A}(z)\|\cdot\mathbb{1}(z \notin E)] \le \|\mathcal{A}\|_r.$$ ◀

▶ **Lemma 29** (right product). *Consider $\mathcal{A} : \{0,1\}^k \to \mathbb{R}^{w\times w}$ and $\mathcal{B} : \{0,1\}^m \to \mathbb{R}^{w\times w}$. Let $f : \{0,1\}^k \times \{0,1\}^{d_A} \to \{0,1\}^n$ be a (δ, ε_A) sampler. Then*

$$\mathop{\mathbb{E}}_z\left[\left\|\mathop{\mathbb{E}}_{x}[\mathcal{A}(f(z,x))\mathcal{B}(z)]\right\|\right] \le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B}) + \left(\|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon_A\right)\|\mathcal{B}\|_r.$$

Proof. Let

$$E = \left\{z : \left\|\mathop{\mathbb{E}}_{x}[\mathcal{A}(f(z,x)]\right\| > \|\mathcal{A}\| + 2w\mu(\mathcal{A})\varepsilon_A\right\}.$$

By Lemma 26, $\Pr_z[z \in E] < w^2\delta$. Therefore

$$\begin{aligned}
\mathop{\mathbb{E}}_{z}\left[\left\|\mathop{\mathbb{E}}_{x}\left[\mathcal{A}(f(z,x))\mathcal{B}(z)\right]\right\|\right] &= \Pr\left[z\in E\right]\mathop{\mathbb{E}}_{z\in E}\left[\left\|\mathop{\mathbb{E}}_{x}\left[\mathcal{A}(f(z,x))\mathcal{B}(z)\right]\right\|\right] \\
&\quad+\Pr\left[z\notin E\right]\mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathop{\mathbb{E}}_{x}\left[\mathcal{A}(f(z,x))\mathcal{B}(z)\right]\right\|\right] \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B})+\Pr\left[z\notin E\right]\cdot\mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathop{\mathbb{E}}_{x}\left[\mathcal{A}(f(z,x))\right]\right\|\left\|\mathcal{B}(z)\right\|\right] \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B})+\left(\left\|\mathcal{A}\right\|+2w\mu(\mathcal{A})\varepsilon_A\right)\cdot\Pr\left[z\notin E\right]\mathop{\mathbb{E}}_{z\notin E}\left[\left\|\mathcal{B}(z)\right\|\right] \\
&\le w^2\delta\mu(\mathcal{A})\mu(\mathcal{B})+\left(\left\|\mathcal{A}\right\|+2w\mu(\mathcal{A})\varepsilon_A\right)\left\|\mathcal{B}\right\|_r. \qquad \blacktriangleleft
\end{aligned}$$

6 Main Construction

In this section we show our main construction and prove its correctness. We first introduce several definitions.

▶ **Definition 30.** *For every mapping $\mathcal{A}:\{0,1\}^n\to\mathbb{R}^{w\times w}$ and every matrix $A\in\mathbb{R}^{w\times w}$, we define $\mathcal{A}-A$ to be the mapping s.t. $(\mathcal{A}-A)(x)=\mathcal{A}(x)-A$.*

▶ **Definition 31.** *Consider $A\in\mathbb{R}^{w\times w}$ and $\mathcal{A}:\{0,1\}^n\to\mathbb{R}^{w\times w}$. $\mathcal{A}$ is a* ε-approximator *of A if $\left\|\mathbb{E}_x\left[\mathcal{A}(x)\right]-A\right\|\le\varepsilon$, i.e. $\left\|\mathcal{A}-A\right\|\le\varepsilon$. $\mathcal{A}$ is a* ε-robust approximator *of A if $\mathbb{E}_x\left[\left\|\mathcal{A}(x)-A\right\|\right]\le\varepsilon$, i.e. $\left\|\mathcal{A}-A\right\|_r\le\varepsilon$.*

Now we define a robust PRPD. Note that a (n,w,ε)-robust PRPD (G,ρ) is also a $\mu(G,\rho)$-bounded (n,w,ε)-PRPD.

▶ **Definition 32.** *$(G,\rho):\{0,1\}^{s_{\text{out}}}\times\{0,1\}^{s_{\text{in}}}\times[\mu]\to\{0,1\}^n\times\mathbb{R}$ is a (n,w,ε)-*robust PRPD *if for every (n,w)-ROBP and its matrix representation $[M]_{[n]}^{\{0,1\}}$ the following holds. Let $\mathcal{A}:\{0,1\}^{s_{\text{out}}}\times\{0,1\}^{s_{\text{in}}}\to\mathbb{R}^{w\times w}$ denote the mapping*

$$\mathcal{A}(x,y)=\mathop{\mathbb{E}}_{i\in[\mu]}\left[\rho(x,y,i)\cdot M_{0..n}^{G(x,y,i)}\right].$$

- *Every $\rho(x,y,i)$ is either μ or $-\mu$. In other word, $\mathcal{A}(x,y)$ is the summation of transition matrices with coefficient ±1.*
- *Let $\widehat{\mathcal{A}}$ denote the mapping $\widehat{\mathcal{A}}(x)=\mathbb{E}_y\left[\mathcal{A}(x,y)\right]$. Then $\widehat{\mathcal{A}}$ is a ε-robust approximator for $M_{0..n}$.*

We say μ is the weight of (G,ρ), denoted by $\mu(G,\rho)$. s_{out} is the outer seed length *of (G,ρ), denoted by $s_{\text{out}}(G,\rho)$. s_{in} is the* inner seed length *of (G,ρ), denoted by $s_{\text{in}}(G,\rho)$. We write $s(G,\rho)=s_{\text{out}}(G,\rho)+s_{\text{in}}(G,\rho)$ for short. We say (G,ρ) is explicit if it can be computed in $O(s(G,\rho))$ space.*

We say $\mathcal{A}$ is the matrix form *of (G,ρ) on $M_{0..n}$, and the definition of $s_{\text{out}},s_{\text{in}},\mu$ on (G,ρ) also apply to $\mathcal{A}$. We say $\widehat{\mathcal{A}}$ is the* robust matrix form *of (G,ρ) on $M_{0..n}$.*

▶ Remark 33. The above definition is similar to Definition 5, but each matrix $\mathcal{A}(x,y)$ is realized with μ matrices instead of one matrix. These μ matrices will never be separated even after flattening. We do this in order to ensure that the matrix form always take bit-strings as input. This ensures that we can increase the outer and inner seed length of $\mathcal{A}$ arbitrarily: we can construct the new mapping $\mathcal{A}':\{0,1\}^{s'_{\text{out}}}\times\{0,1\}^{s'_{\text{in}}}$ such that $\mathcal{A}'(x,y)=\mathcal{A}(x_p,y_p)$ where x_p is the length-$s_{\text{out}}(\mathcal{A})$ prefix of x and y_p is the length-$s_{\text{in}}(\mathcal{A})$ prefix of y. In other word, $\mathcal{A}'$ computes the output only with prefix of necessary length of the input, and ignore the remaining bits. It is easy to verify that $\mathcal{A}'$ is also the matrix form of a (n,w,ε)-robust PRPD.

The following is some additional basic properties about robust PRPD and its flattened form.

▷ Claim 34. Let $(G,\rho):\{0,1\}^{s_{\text{out}}}\times\{0,1\}^{s_{\text{in}}}\times[\mu]\to\{0,1\}^n\times\mathbb{R}$ be a (n,w,ε)-*robust PRPD*. For every (n,w)-ROBP $M_1^0,M_1^1,\dots,M_n^0,M_n^1$ the following holds.

- Let $\widehat{\mathcal{A}}$ be the robust matrix form of (G,ρ) on $M_{0..n}$. Then $\mu(\widehat{\mathcal{A}})\le\mu(G,\rho)$.
- Let $\mathcal{A}$ denote the matrix form of (G,ρ) on $M_{0..n}$. Let $\overline{\mathcal{A}}:\{0,1\}^{s_{\text{out}}+s_{\text{in}}}\to\mathbb{R}^{w\times w}$ denote the mapping $\overline{\mathcal{A}}(x\|y)=\mathcal{A}(x,y)$. We say $\overline{\mathcal{A}}$ is the *flattened matrix form* of (G,ρ) on $M_{0..n}$. Then $\overline{\mathcal{A}}$ is an ε-approximator for $M_{0..n}$, and $\mu(\overline{\mathcal{A}})\le\mu(G,\rho)$.

Proof. Recall that for every string $r\in\{0,1\}^n$, $\|M^r_{0..n}\|=1$. By sub-additivity of $\|\cdot\|$ we have $\|\mathcal{A}(x,y)\|\le\mu(G,\rho)$ for every x,y, which implies $\mu(\overline{\mathcal{A}})\le\mu(G,\rho)$. By sub-additivity and scalability of $\|\cdot\|$, we have $\mu(\mathcal{A}')\le\mu(\mathcal{A})$. To show that $\overline{\mathcal{A}}$ is a ε-approimxator of $M_{0..n}$, observe that $\mathcal{A}'$ is also an ε-approximator of $M_{0..n}$ by Claim 24, and note that $\langle\mathcal{A}\rangle=\langle\mathcal{A}'\rangle$. ◁

Now we prove our main lemma. The following lemma allows us to construct robust PRPDs for $(2m,w)$ ROBPs from robust PRPDs for (m,w) ROBPs, without increasing the seed length too much. We will recursively apply this lemma for $\log n$ levels to get a (n,w,ε)-robust PRPD. The basic idea is as described in Lemma 16.

▶ **Lemma 35.** *Suppose there exists $s_{\text{out}},s_{\text{in}}$ such that the following conditions hold.*

- *For every $0\le i\le k$, there exists a (m,w,γ^{i+1})-robust PRPD (G_i,ρ_i) s.t. $\mu(G_i,\rho_i)\le\binom{m-1}{i}$ and $s_{\text{out}}(G,\rho)\le s_{\text{out}}$. Moreover, for every $0\le i\le\lceil k/2\rceil$, $s(G_i,\rho_i)\le s_{\text{out}}$.*
- *For every $i\le\lceil k/2\rceil$, there exists a (ε_i,δ)-sampler $g_i:\{0,1\}^{s_{\text{out}}}\times\{0,1\}^{d_i}\to\{0,1\}^{s(G_i,\rho_i)}$, where $\varepsilon_i\le\gamma^{i+1}/(w\cdot\binom{m-1}{i})$ and $\delta\le\gamma^{k+1}/(w^2\cdot\binom{2m-1}{i})$.*
- *For every $i\ge j\ge0$ s.t. $i+j\le k$, if $j\le i\le\lceil k/2\rceil$, then $d_i+d_j\le s_{\text{in}}$. If $i>\lceil k/2\rceil$, then $s_{\text{in}}(G_i,\rho_i)+d_j\le s_{\text{in}}$.*

Then there exists a $(2m,w,(11\gamma)^{k+1})$-robust PRPD (G,ρ) s.t. $s_{\text{out}}(G,\rho)=s_{\text{out}}$, $s_{\text{in}}(G,\rho)=s_{\text{in}}$ and $\mu(G,\rho)\le\binom{2m-1}{k}$.

Proof. Fix any $(2m,w)$-ROBP with matrix representation $M^{\{0,1\}}_{[2m]}$. Let $A=M_{0..m}$ and $B=M_{m..2m}$. For every $0\le i\le k$, let $\mathcal{A}_i,\widehat{\mathcal{A}}_i,\overline{\mathcal{A}}_i$ denote the matrix form, robust matrix form and flattened matrix form of (G,ρ) on $M_{0..m}$ respectively. Let $\mathcal{B}_i,\widehat{\mathcal{B}}_i,\overline{\mathcal{B}}_i$ denote the matrix form, robust matrix form and flattened matrix form of (G,ρ) on $M_{m..2m}$ respectively. By definition, $\widehat{\mathcal{A}}_i$ and $\widehat{\mathcal{B}}_i$ are γ^{i+1}-robust approximator for A and B respectively. By Claim 34, $\overline{\mathcal{A}}_i$ and $\overline{\mathcal{B}}_i$ are γ^{i+1}-approximator for A and B respectively. Moreover, we will increase the outer seed length of $\mathcal{A}_i$ and $\mathcal{B}_i$ to match the length of the given input when necessary. (See Remark 33)

Now for every x,y we define a mapping $\mathcal{C}_k:\{0,1\}^{s_{\text{out}}}\times\{0,1\}^{s_{\text{in}}}\to\mathbb{R}^{w\times w}$ as follows. Note that $\mathcal{C}_k$ corresponds to the matrix form of (G,ρ) on $M_{0..2m}$.

(1) For every $0\le i\le\lceil k/2\rceil$, let a_i be the prefix of y of length d_i and b_i be the suffix of y of length d_i. Define $A_{x,y,i}=\overline{\mathcal{A}_i}(g_i(x,a_i))$ and $B_{x,y,i}=\overline{\mathcal{B}_i}(g_i(x,b_i))$.

(2) For every $\lceil k/2\rceil<i\le k$, let a_i be the prefix of y of length $s_{\text{in}}(\mathcal{A}_i)$ and b_i be the suffix of y of length $s_{\text{in}}(\mathcal{B}_i)$. Define $A_{x,y,i}=\mathcal{A}_i(x,a_i)$ and $B_{x,y,i}=\mathcal{B}_i(x,b_i)$.

(3) Define $\mathcal{C}_k(x,y)=\sum_{i+j=k}A_{x,y,i}B_{x,y,j}-\sum_{i+j=k-1}A_{x,y,i}B_{x,y,j}$.

Note that for every $i+j\le k$, prefix a_i and suffix b_j of y never overlap.

By expanding every $A_{x,y,i}B_{x,y,j}$ term with distributive law, we can see that each small term in $A_{x,y,i}B_{x,y,j}$ has coefficient ±1, which satisfies the first condition of robust PRPD. Moreover, the total number of terms after expanding is

$$\mu(\mathcal{C}_k)\le\sum_{i+j=k}\binom{m-1}{i}\cdot\binom{m-1}{j}+\sum_{i+j=k-1}\binom{m-1}{i}\cdot\binom{m-1}{j}=\binom{2m-1}{k}.$$

It remains to show that $\mathcal{C}_k$ satisfies the second condition of robust PRPD, i.e. $\mathbb{E}_y[\mathcal{C}_k(x,y)]$ is a good approximation of $M_{0..2m} = AB$ on average over x. Observe that

$$\begin{aligned}
\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[\mathcal{C}_k(x,y)] - AB\right\|\right] &= \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[\mathcal{C}_k(x,y) - AB]\right\|\right] \\
&\le \sum_{i+j=k} \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,i} - A)(B_{x,y,j} - B)]\right\|\right] \\
&+ \sum_{i+j=k-1} \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,i} - A)(B_{x,y,j} - B)]\right\|\right] \\
&+ \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,k} - A)B]\right\|\right] + \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[A(B_{x,y,k} - B)]\right\|\right],
\end{aligned}$$

by decomposing $\mathcal{C}_k(x,y) - AB$ with the equation in the proof of Lemma 15 and applying sub-additivity of $\|\cdot\|$.

First we consider the last two terms. Since $\|B\| = 1$, by sub-multiplicativity we have

$$\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,k} - A)B]\right\|\right] \le \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[A_{x,y,k} - A]\right\|\right].$$

Now consider two cases. If $k \ge 2$, then

$$\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[A_{x,y,k} - A]\right\|\right] = \mathop{\mathbb{E}}_x\left[\left\|\widehat{\mathcal{A}}_k(x) - A\right\|\right] \le \gamma^{k+1}$$

by definition. If $k < 2$, then

$$\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[A_{x,y,k} - A]\cdot B\right\|\right] = \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_{a_k}[\overline{\mathcal{A}_k}(g_k(x,a_k)) - A]\right\|\cdot B\right].$$

Apply Lemma 29 on $\overline{\mathcal{A}_k} - A$ and the dummy mapping $\mathcal{B}$ s.t. $\mathcal{B}(x) = B$ for every x, we can derive that the above formula is bounded by $w^2\delta\binom{m-1}{k} + 3\gamma^{i+1}$. For the term $\mathbb{E}_x\left[\|\mathbb{E}_y[A(B_{x,y,k} - B)]\|\right]$ we can get the same bound with a similar proof.

Now consider the terms in the form $\mathbb{E}_x\left[\|\mathbb{E}_y[(A_{x,y,i} - A)(B_{x,y,j} - B)]\|\right]$. First consider the case $i, j \le \lceil k/2\rceil$. Then

$$\begin{aligned}
&\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,i} - A)(B_{x,y,j} - B)]\right\|\right] \\
&= \mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_{a_i}[\overline{\mathcal{A}_i}(g_i(x,a_i)) - A]\mathop{\mathbb{E}}_{b_j}[\overline{\mathcal{B}_k}(g_j(x,b_j)) - B]\right\|\right] \text{ (since } a_i, b_j \text{ don't overlap)} \\
&\le 2w^2\delta\cdot\binom{m-1}{i}\cdot\binom{m-1}{j} + 9\gamma^{i+j+2}. \text{ (by Lemma 27)}
\end{aligned}$$

Next consider the case $i > \lceil k/2\rceil, j \le \lceil k/2\rceil$. Then

$$\begin{aligned}
&\mathop{\mathbb{E}}_x\left[\left\|\mathop{\mathbb{E}}_y[(A_{x,y,i} - A)(B_{x,y,j} - B)]\right\|\right] \\
&= \mathop{\mathbb{E}}_x\left[\left\|\widehat{\mathcal{A}}_i(x)\cdot\mathop{\mathbb{E}}_{b_j}[\overline{\mathcal{B}_k}(g_j(x,b_j)) - B]\right\|\right] \text{ (since } a_i, b_j \text{ don't overlap)} \\
&\le w^2\delta\cdot\binom{m-1}{i}\cdot\binom{m-1}{j} + 3\gamma^{i+j+2}. \text{ (by Lemma 29)}
\end{aligned}$$

Similarly for the case that $i \le \lceil k/2 \rceil, j > \lceil k/2 \rceil$ we can show that

$$\mathop{\mathbb{E}}_{x}\left[\left\|\mathop{\mathbb{E}}_{y}\left[(A_{x,y,i}-A)(B_{x,y,j}-B)\right]\right\|\right] \le w^2\delta \cdot \binom{m-1}{i}\cdot\binom{m-1}{j} + 3\gamma^{i+j+2}$$

by Lemma 28. Finally, note that the case $i, j > \lceil k/2 \rceil$ does not exist because $i + j \le k$.

Taking the summation of all the cases, we get

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x}\left[\left\|\mathop{\mathbb{E}}_{y}\left[\mathcal{C}_k(x,y)\right]-AB\right\|\right] \\
&\le 2w^2\delta\cdot\left(\sum_{i+j=k}\binom{m-1}{i}\binom{m-1}{j}+\sum_{i+j=k-1}\binom{m-1}{i}\binom{m-1}{j}+\binom{m-1}{k}\right) \\
&+(k+1)\cdot 9\gamma^{k+2}+k\cdot 9\gamma^{k+1}+2\cdot 3\gamma^{k+1} \\
&\le 4w^2\delta\cdot\binom{2m-1}{k}+(9k+9)\gamma^{k+2}+(9k+6)\gamma^{k+1} \\
&\le (10k+11)\gamma^{k+1} \\
&\le (11\gamma)^{k+1}.
\end{aligned}$$

Moreover, note that $AB = M_{0..2m}$, and the construction of $\mathcal{C}_k$ does not depend on the matrices $M_{[2m]}^{\{0,1\}}$. (See Section 2 for how the arithmetic operations in $\mathcal{C}_k(x,y)$ are translated back to operations on pseudo-distributions.) Therefore there exists a $(2m, w, (11\gamma)^{k+1})$-robust PRPD (G,ρ). ◀

Finally we analyze the seed length of the recursive construction, and present the main theorem.

▶ **Theorem 36.** *There exists an explicit (n, w, ε)-robust PRPD (G,ρ) such that*

- $s_{\text{out}}(G,\rho) = O\left(\log(1/\varepsilon)+\log n\log(nw)\log\left(\frac{\log(1/\varepsilon)}{\log n}\right)\right)$
- $s_{\text{in}}(G,\rho) = O\left(\log(1/\varepsilon)+\log n\log(nw)\log\left(\frac{\log(1/\varepsilon)}{\log n}\right)\right)$
- $\mu(G,\rho) = \text{poly}(1/\varepsilon)$

Moreover, for every B the approximator $\mathcal{G}$ has the same corresponding pseudodistribution.

Proof. Let c be the constant such that for every $\varepsilon, \delta > 0$ there exists a (ε,δ)-sampler $g : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that $n = m + c\log(1/\varepsilon) + c\log(1/\delta)$ and $d = c\log(1/\varepsilon) + c\log\log(1/\delta)$, as guaranteed in Lemma 19. WLOG assume that n is a power of 2. Define $\gamma = 1/n^4$. For every $0 \le h \le \log n$, every $k \ge 0$, we will inductively prove that there exists a $(2^h, w, (11^h\gamma)^{k+1})$-robust PRPD $(G_{h,k}, \rho_{h,k})$ with the following parameters.

- If $k \le 1$, $s_{\text{out}}(G_{h,k},\rho_{h,k}) \le h\cdot(3ck\log(n/\gamma)+7c\log(w/\gamma))$
- If $k > 1$, $s_{\text{out}}(G_{h,k},\rho_{h,k}) \le 4ck\log(n/\gamma)+(\lceil\log k\rceil+1)\cdot h\cdot(10c\log(w/\gamma))$
- If $k \le 1$, $s_{\text{in}}(G_{h,k},\rho_{h,k}) \le ck\log(n/\gamma)+4c\log(w/\gamma)$
- If $k > 1$, $s_{\text{in}}(G_{h,k},\rho_{h,k}) \le ck\log(n/\gamma)+h\cdot(4c\log(kw/\gamma))$
- $\mu(G_{h,k},\rho_{h,k}) \le \max(1,\binom{2^h-1}{k})$

We will write $s_{\text{out},h,k} = s_{\text{out}}(G_{h,k},\rho_{h,k})$ and $s_{\text{in},h,k} = s_{\text{out}}(G_{h,k},\rho_{h,k})$ for short. First consider the terminal case $2k \ge 2^h$ or $h = 0$. In this case we simply take $s_{\text{out},h,k} = 0$, $s_{\text{in},h,k} = 2^h \le 2k$ and $\mu(G_{h,k},\rho_{h,k}) = 1$ s.t. $G_{h,k}(x,y,i) = y$ and $\rho_{h,k}(x,y,i) = 1$. For the other cases, we show that we can get the intended parameters by constructing $(G_{h,k},\rho_{h,k})$ with the recursion in Lemma 35. Note that based on the induction hypothesis we can assume $\mathcal{G}_{a,h-1,k}$ and $\mathcal{G}_{a+2^{h-1},h-1,k}$ have exactly the same parameters, so we consider the

parameter of $\mathcal{G}_{a,h-1,k}$ only. We have seen that the bound for $\mu(G_{h,k}, \rho_{h,k})$ is correct. First we show that the bound for $s_{\text{in},h,k}$ is correct. Recall that in the recursion we take parameters $d_i = c\log(1/\varepsilon_i) + c\log\log(1/\delta) \leq ci\log(n/\gamma) + 2c\log(knw/\gamma)$, based on the fact that $\binom{2^h-1}{i} \leq n^i$. Now consider the restriction on $s_{\text{in}}(\mathcal{G}_k)$ in our recursion. For $i + j \leq k$ and $j \leq i \leq \lceil k/2 \rceil$, we need

$$d_i + d_j \leq ck\log(n/\gamma) + 4c\log(knw/\gamma) \leq s_{\text{in},h,k}$$

which is true. For $i + j \leq k$ and $i > \lceil k/2 \rceil$, we need

$$\begin{aligned} s_{\text{in},h-1,i} + d_j &\leq ci\log(1/\gamma) + (h-1)\cdot 4c\log(inw/\gamma) + (cj\log(1/\gamma) + 2c\log(knw/\gamma)) \\ &\leq ck\log(1/\gamma) + h\cdot 4c\log(knw/\gamma) \\ &\leq s_{\text{in},h,k} \end{aligned}$$

which is also true. Moreover, observe that when $k \leq 1$ it is always the case that $i, j \leq \lceil k/2 \rceil$. Therefore the third condition is also true. Finally we show that the bound for $s_{\text{out},h,k}$ is also correct. First observe that the restriction $s_{\text{out},h-1,i} \leq s_{\text{out},h,k}$ is trivially true. Then the only condition left is that for every $i \leq \lceil k/2 \rceil$,

$$s_{\text{out},h-1,i} + s_{\text{in},h-1,i} + c\log(1/\delta) + c\log(1/\varepsilon_i) \leq s_{\text{out},h,k}.$$

Since $s_{\text{out},h-1,i} \leq s_{\text{out},h-1,\lceil k/2 \rceil}$ and $s_{\text{in},h-1,i} \leq s_{\text{in},h-1,\lceil k/2 \rceil}$ for every i, it suffices to show that

$$s_{\text{out},h-1,\lceil k/2 \rceil} + s_{\text{in},h-1,\lceil k/2 \rceil} + c\log(1/\delta) + c\log(1/\varepsilon_{\lceil k/2 \rceil}) \leq s_{\text{out},h,k}.$$

First we consider $k \leq 1$, which is the case that $\lceil k/2 \rceil = k$. Then

$$\begin{aligned} &s_{\text{out},h-1,\lceil k/2 \rceil} + s_{\text{in},h-1,\lceil k/2 \rceil} + c\log(1/\delta) + c\log(1/\varepsilon_{\lceil k/2 \rceil}) \\ &\leq s_{\text{out}}(\mathcal{G}_{a,h-1,k}) + 3ck\log(n/\gamma) + 7c\log(n/\gamma) \\ &\leq h\cdot(3ck\log(n/\gamma) + 7c\log(n/\gamma)) \\ &\leq s_{\text{out},h,k}. \end{aligned}$$

Finally we consider the case $k > 1$. Observe that

$$\begin{aligned} &s_{\text{out},h-1,\lceil k/2 \rceil} + s_{\text{in},h-1,\lceil k/2 \rceil} + c\log(1/\delta) + c\log(1/\varepsilon_{\lceil k/2 \rceil}) \\ &\leq s_{\text{out},h-1,\lceil k/2 \rceil} + s_{\text{in},h-1,\lceil k/2 \rceil} + \frac{3k+1}{2}\cdot c\log(n/\gamma) + 3c\log(w/\gamma) \\ &\leq s_{\text{out},h-1,\lceil k/2 \rceil} + (2k+1)\cdot c\log(n/\gamma) + (h-1)\cdot 4c\log(w/\gamma) + 7c\log(w/\gamma) \\ &\leq 4c\cdot\frac{k+1}{2}\cdot\log(n/\gamma) + \left(\lceil\log\lceil\frac{k}{2}\rceil\rceil + 1\right)\cdot(h-1)\cdot(10c\log(nw/\gamma)) \\ &+ (2k+1)\cdot c\log(n/\gamma) + (h-1)\cdot 4c\log(w/\gamma) + 7c\log(w/\gamma) \\ &\leq 4ck\log(n/\gamma) + \left(\lceil\log\lceil\frac{k}{2}\rceil\rceil + 1\right)\cdot(h-1)\cdot(10c\log(nw/\gamma)) + h\cdot 10c\log(nw/\gamma) \\ &\leq s_{\text{out},h,k}. \end{aligned}$$

In the last inequality we use the fact that $\lceil\log k\rceil = \lceil\log(\lceil k/2\rceil)\rceil + 1$ for every $k > 1$.

Finally, note that $(11^{\log n}\gamma) = n^{\log_2 11}\cdot n^{-4} \leq n^{-0.5}$. By taking $h = \log n$ and $k = \frac{\log(1/\varepsilon)}{\log(1/n^{0.5})}$, we get a (n, w, ε)-robust PRPD. ◀

▶ Remark 37. To get the seed length we claimed in Theorem 4, observe that the $\log(1/\varepsilon)$ term is dominating when $\log(1/\varepsilon) \geq \log^3(nw)$. Therefore we can simply replace the $\log\log(1/\varepsilon)$ factor on the $O(\log n\log(nw))$ term with $\log\log(nw)$.

7 Discussion and Open Questions

We discuss some natural questions that arise from our work.

- In our construction, we applied the sampler argument in [4] without constructing small-norm matrices explicitly. This is probably hinting that negative weight is not essentially required for the sampler argument. Is it possible to apply the sampler argument to construct a PRG (instead of PRPD) with improved dependency on error?
- Is there an explicit PRPD which matches the seed length of the hitting set generator in [13], i.e. $O(\log(w/\varepsilon))$ when $n = \operatorname{poly}\log(w)$? A possible direction is to adapt our construction to a t-ary recursion tree where $t = \log^{1-\Omega(1)}(n)$ instead of a binary tree, as in [25, 1]. However, a direct adaption requires us to apply samplers on $(t-1)$-children in each recursion, and for every sampler we need to pay some randomness for "inner seed" which cannot be recycled. In our construction we see that the inner seed of a sampler contains a $\log w$ term. Therefore in each recursion we need to pay at least $(t-1)\log w$ which is too expensive. Is it possible to make the sampler argument work with a shorter inner seed?
- Is it possible to improve the seed length to $\tilde{O}(\log^2 n + \log(w/\varepsilon))$, even in some restricted settings? We note that there are two things which cause the $\Omega(\log n \cdot \log w)$ term in our construction. The first one is the inner seed of sampler, which is related to the question above. The second one is the restriction on the outer seed length, which is analogous to "entropy loss" if we view the samplers as extractors. Note that [26] shows how to "recycle entropy" in the INW generator in some restricted settings, but it is not clear how to apply the extractor-type analysis of INW generator in our construction.

References

1. Roy Armoni. On the derandomization of space-bounded computations. In Michael Luby, José D. P. Rolim, and Maria J. Serna, editors, *Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings*, volume 1518 of *Lecture Notes in Computer Science*, pages 47–59. Springer, 1998. doi:10.1007/3-540-49543-6_5.
2. Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9:283–293, 2013. doi:10.4086/toc.2013.v009a007.
3. Allan Borodin, Stephen A. Cook, and Nicholas Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control*, 58(1-3):113–136, 1983. doi:10.1016/S0019-9958(83)80060-6.
4. Mark Braverman, Gil Cohen, and Sumegha Garg. Hitting sets with near-optimal error for read-once branching programs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 353–362. ACM, 2018. doi:10.1145/3188745.3188780.
5. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. doi:10.1137/120875673.
6. Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 30–39. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.10.
7. Kuan Cheng and William Hoza. Hitting sets give two-sided derandomization of small space. *Electronic Colloquium on Computational Complexity (ECCC)*, 2020. URL: https://eccc.weizmann.ac.il/report/2020/016/.

8 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. doi:10.1137/0217015.

9 Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 221–231. IEEE Computer Society, 2011. doi:10.1109/CCC.2011.23.

10 Oded Goldreich. A sample of samplers: A computational perspective on sampling. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011. doi:10.1007/978-3-642-22670-0_24.

11 Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Struct. Algorithms*, 11(4):315–343, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4<315::AID-RSA3>3.0.CO;2-1.

12 Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. doi:10.1145/1538902.1538904.

13 William Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 59–64. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00015.

14 William M. Hoza and Chris Umans. Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 629–640. ACM, 2017. doi:10.1145/3055399.3055414.

15 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 356–364. ACM, 1994. doi:10.1145/195058.195190.

16 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997. doi:10.1145/258533.258590.

17 H. Jung. Relationships between probabilistic and deterministic tape complexity. In Jozef Gruska and Michal Chytil, editors, *Mathematical Foundations of Computer Science 1981, Strbske Pleso, Czechoslovakia, August 31 - September 4, 1981, Proceedings*, volume 118 of *Lecture Notes in Computer Science*, pages 339–346. Springer, 1981. doi:10.1007/3-540-10856-4_101.

18 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

19 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. Revisiting norm estimation in data streams. *CoRR*, abs/0811.3648, 2008. arXiv:0811.3648.

20 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

21 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 263–272. ACM, 2011. doi:10.1145/1993636.1993672.

22 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 626–637. ACM, 2019. doi:10.1145/3313276.3316319.

23 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

24 Noam Nisan. RL <= SC. *Computational Complexity*, 4:1–11, 1994. doi:10.1007/BF01205052.

25 Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.

26 Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 159–168. ACM, 1999. doi:10.1145/301250.301294.

27 Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(18), 2001. URL: http://eccc.hpi-web.de/eccc-reports/2001/TR01-018/index.html.

28 Michael Saks. Randomization and derandomization in space-bounded computation. In *Proceedings of Computational Complexity (Formerly Structure in Complexity Theory)*, pages 128–149. IEEE, 1996.

29 Michael E. Saks and Shiyu Zhou. BP $_{\mathrm{h}}$space(s) subseteq dspace(s$^{3/2}$). *J. Comput. Syst. Sci.*, 58(2):376–403, 1999. doi:10.1006/jcss.1998.1616.

30 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970. doi:10.1016/S0022-0000(70)80006-X.

31 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:83, 2012. URL: http://eccc.hpi-web.de/report/2012/083.

32 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

33 David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4<345::AID-RSA4>3.0.CO;2-Z.

A Using PRPDs in the Saks-Zhou Scheme

In this section, we will briefly introduce Saks and Zhou's proof for $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$ [29] and Armoni's trick for replacing Nisan's PRG with any PRG in this proof [1]. Then we will see why a poly(nw/ε)-bounded PRPD suffices for this scheme. Since our purpose here is to go over the possible difference between using PRGs and PRPDs in this scheme, we will only include a sketch of Saks and Zhou's proof. We recommend interested readers to check [29, 1] for formal proofs and also [14] for a beautiful summary.

A.1 Saks and Zhou's Scheme

It is well-known that derandomizing **BPL** can be reduced to approximating M^n where M is any $n \times n$ stochastic matrix. The first step of Saks and Zhou is to turn M^n into the following recursive computation:

▶ **Fact 1.** *Let n_1, n_2 be integers such that $n_1^{n_2} = n$. Define $M_0 = M$, and $M_i = M_{i-1}^{n_1}$ for every positive integer i. Then $M_{n_2} = M^n$.*

To approximate M_{n_2}, it suffices to compute $M_i = M_{i-1}^{n_1}$ with small enough error in each step. However, if we need s bits of space to approximate the n_1-th power of a stochastic matrix, we will need $O(sn_2)$ bits of space in total. This doesn't really save any space (over approximating M^n directly) if we approximate $M_{i-1}^{n_1}$ with PRGs such as Nisan's generators. The first idea of Saks and Zhou is to utilize the "high probability" property of Nisan's generator:

▶ **Lemma 38** ([23]). *For every n, w, ε there exists an algorithm $\widehat{\mathrm{Pow}}_n$ which takes a $w \times w$ (sub)stochastic matrix M and a string $y \in \{0,1\}^{O(\log n \log(nw/\varepsilon))}$ as input, and outputs a $w \times w$ matrix such that*

$$\Pr_y\left[\left\|\widehat{\mathrm{Pow}}_n(M,y) - M^n\right\|_{\max} < \varepsilon\right] \geq 1-\varepsilon$$

in space $O(log(nw/\varepsilon))$.

In other word, derandomization with Nisan's generator has the following structure. First it fixes an "offline randomness" $y \in \{0,1\}^r$ and considers it as a part of input. Then it takes s bits of additional "processing space" to compute an approximation of M^n. Then the output will be a good approximation with high probability over y. (This is called an "offline randomized algorithm" in [29].) Furthermore $s \ll r$. With these properties, the main idea of Saks and Zhou is to reuse the same offline randomness for each level of recursion. If computing M^{n_1} takes r bits of offline randomness and s bits of processing space, then computing M^n will take r bits of offline randomness and $O(sn_2)$ bits of processing space. The space complexity would be $O(r + sn_2)$ which is better than approximating M^n directly since the offline randomness part was the original bottleneck.

However, there's a problem in this construction: if we compute $\widehat{M_1} = \widehat{\mathrm{Pow}}_{n_1}(M,y)$, and try to use $\widehat{\mathrm{Pow}}_{n_1}(\widehat{M_1},y)$ to approximate $M_2 = M_1^{n_1}$, it might be possible that $\widehat{\mathrm{Pow}}_{n_1}(\widehat{M_1},y)$ is always a bad approximation because $\widehat{M_1}$ depends on y. To resolve this issue, the second idea of Saks and Zhou is to break the dependency with a randomized rounding operation. We will borrow the name "snap" from [14] for this operation.

▶ **Definition 39.** *Given value $x \in \mathbb{R}$, string $y \in \{0,1\}^d$, define*

$$\mathrm{Snap}_d(x,y) = \max(\lfloor x \cdot 2^d - 2^{-d}y \rfloor \cdot 2^{-d}, 0).$$

For a $w \times w$ matrix M, define $\mathrm{Snap}_d(M,y)$ to be the matrix M' such that $M'_{i,j} = \mathrm{Snap}_d(M_{i,j},y)$ for every $i,j \in [w]$.

In other word, in a snap operation, we randomly perturb the matrix with a offset in $[0, 2^{-2d}]$, then round the entries down to d bits of precision. It's not hard to prove the following lemma:

▶ **Lemma 40** ([29]). *For any matrix M, M' such that $\|M - M'\|_{\max} \leq \varepsilon$,*

$$\Pr_y[\mathrm{Snap}_d(M,y) \neq \mathrm{Snap}_d(M',y)] \leq w^2(2^d\varepsilon + 2^{-d}).$$

Proof. The snap operation is equivalent to randomly choose a grid of length 2^{-d} and round each value to the closest grid point the left. Therefore two values a, b are rounded to different points only if there is a grid point between them, which happens with probability at most $2^d|a-b| + 2^{-d}$. By union bound and the fact that $\|M - M'\|_{\max} \leq \varepsilon$ the lemma follows. ◀

With the lemma above, we can see that by taking $\widehat{M_1} = \mathrm{Snap}_d(\widehat{\mathrm{Pow}}_{n_1}(M, y), z)$ instead, $\widehat{M_1}$ will be equivalent to $\mathrm{Snap}_d(M^{n_1}, z)$ with high probability, which is independent of y. Therefore we can use y as the offline randomness to compute the n_1-th power of $\widehat{M_1}$. Moreover, if the rounding precision is high enough, the snapped matrix is still a good approximation. Finally we get Saks-Zhou theorem:

▶ **Lemma 41** ([29]). *Let n_1, n_2 be integers such that $n_1^{n_2} = n$. Suppose there exists an offline randomized algorithm $\widehat{\mathrm{Pow}}_{n_1}$ which takes r bits of randomness and s bits of processing space such that for every substochastic matrix M,*

$$\Pr_x \left[\left\| \widehat{\mathrm{Pow}}_{n_1}(M, y) - M^{n_1} \right\|_{\max} \leq \varepsilon \right] \geq 1 - \varepsilon.$$

Now consider uniform random bits $y \in \{0,1\}^r$ and $z_1, z_2, \ldots, z_{n_2} \in \{0,1\}^d$. Let $\widehat{M_0} = M$, and $\widehat{M_i} = \mathrm{Snap}_d(\widehat{\mathrm{Pow}}_{n_1}(\widehat{M_{i-1}}, y), z_i)$ for every $i \in [n_2]$. Then with probability at least $1 - O(w^2 n_2(2^d\varepsilon + 2^{-d}))$ over $y, z_1, \ldots, z_{n_2}$,

$$\left\| \widehat{M_{n_2}} - M^n \right\| \leq nw2^{-d+1}.$$

Moreover, the space complexity of computing $\widehat{M_{n_2}}$ is $O(r + n_2(s + d))$.

Proof sketch. Define $\overline{M_0} = M$, $\overline{M_i} = \mathrm{Snap}((\overline{M_{i-1}})^{n_1}, z_i)$. By union bound, the following events happen simultaneously with probability $1 - O(w^2 n_2(2^d\varepsilon + 2^{-d}))$:

1. For every $i \in [n_2]$, $\left\| \widehat{\mathrm{Pow}}_{n_1}(\overline{M_{i-1}}, y) - (\overline{M_{i-1}})^{n_1} \right\|_{\max} \leq \varepsilon$.
2. For every $i \in [n_2]$, conditioned on $\widehat{M_{i-1}} = \overline{M_{i-1}}$ and $\left\| \widehat{\mathrm{Pow}}_{n_1}(\overline{M_{i-1}}, y) - \overline{M_{i-1}}^{n_1} \right\|_{\max} \leq \varepsilon$, $\widehat{M_i} = \overline{M_i}$.

When the above events occur, we have $\widehat{M_{n_2}} = \overline{M_{n_2}}$. Moreover, note that for every $i \in [n_2]$

$$\left\| \overline{M_i} - (\overline{M_{i-1}})^{n_1} \right\|_{\max} \leq 2^{-d+1}.$$

To see why this is true, observe that in a snap operation we change the given value by at most 2^{-2d} from perturbation and 2^{-d} from rounding. [2] This implies

$$\left\| \overline{M_i} - (\overline{M_{i-1}})^{n_1} \right\| \leq 2^{-d+1},$$

where $\|\cdot\|$ denotes the matrix infinity norm. By Lemma 5.4 in [29],

$$\left\| \overline{M_{n_2}} - M^n \right\| \leq nw2^{-d+1}.$$

For the space complexity, observe that we can compute $\widehat{M_{n_2}}$ with n_2 levels of recursive calls, and each recursive call takes $O(s + d)$ bits. Moreover, we need r bits to store the offline randomness. Therefore the space complexity is $O(n_2(s + d) + r)$ ◀

If we take $n_2 = \sqrt{\log n}$, $n_1 = 2^{\sqrt{\log n}}$, $d = O(\log(n))$ and $\varepsilon = 2^{-2d}$ and plugging in Nisan's generator, the above lemma shows that $\mathbf{BPL} \subseteq \mathbf{L}^{3/2}$.

[2] Note that capping the lowest possible value to be 0 can only reduce the error, because the snapped value was non-negative.

A.2 Armoni's Trick

We saw that in Saks and Zhou's proof, we need a "offline randomized algorithm" for substochastic matrix exponentiation such that when given r bits of randomness as additional input, the algorithm only requires additional $s \ll r$ bits of space to compute a good approximation with high probability. This is in fact the only place where we need PRGs in Saks and Zhou's proof. However, not every PRG has such property, so it might be hard to tell whether an improvement over Nisan's PRG will actually give a better derandomization for **BPL**. Fortunately, Armoni [1] observed that one can turn any PRG into a derandomization algorithm with the required property by simply composing the PRG with an averaging sampler.

Before we go through Armoni's claim, first we generalize Lemma 41 for a larger class of algorithms $\widehat{\mathrm{Pow}}$.

▶ **Definition 42.** *We say an offline randomized algorithm requires s bits of* sensitive processing space *and t bits of* reusable processing space *if*

- *During the execution of this algorithm, only t bits of processing space is required.*
- *Before each time a bit is read from the real input (not including the offline randomness), only s bits of processing space is being used at the time.*

In the above definition, think of each input bit as generated from a recursive call. Thus the "reusable processing space" can be interpreted as "recursion-friendly processing space", which can be erased before every recursive call. With this new definition we can generalize Lemma 41 as follows:

▶ **Lemma 43** ([29], generalized)**.** *Let n_1, n_2 be integers such that $n_1^{n_2} = n$. Suppose there exists an offline randomized algorithm $\widehat{\mathrm{Pow}}_{n_1}$ which takes r bits of randomness, s bits of sensitive processing space and t bits of reusable processing space, such that for every substochastic matrix M,*

$$\Pr_x\left[\left\|\widehat{\mathrm{Pow}}_{n_1}(M,y) - M^{n_1}\right\|_{\max} \le \varepsilon\right] \ge 1-\varepsilon.$$

Now consider uniform random bits $y \in \{0,1\}^r$ and $z_1, z_2, \ldots, z_{n_2} \in \{0,1\}^d$. Let $\widehat{M_0} = M$, and $\widehat{M_i} = \mathrm{Snap}_d(\widehat{\mathrm{Pow}}_{n_1}(\widehat{M_{i-1}}, y), z_i)$ for every $i \in [n_2]$. Then with probability at least $1 - O(w^2 n_2(2^d\varepsilon + 2^{-d}))$ over $y, z_1, \ldots, z_{n_2}$,

$$\left\|\widehat{M_{n_2}} - M^n\right\| \le nw2^{-d+1}.$$

Moreover, the space complexity of computing $\widehat{M_{n_2}}$ is $O(r + t + n_2(s+d))$.

We omit the proof because it's Exactly the same as Lemma 41.

For technicality, we also need to define a ROBP with larger "step size".

▶ **Definition 44.** *A (n,w,d)-ROBP is a ROBP of n layers, w nodes in each layer, and 2^d branches from each node.*

That is, a (n,w,d)-ROBP is a ROBP which can read d bits at once. Note that derandomizing (n,w,d)-ROBP corresponds to derandomizing the exponentiation of a stochastic matrix which has d bits of precision in each entry.

Now we are ready to introduce Armoni's Lemma.

▶ **Lemma 45** ([1])**.** *Suppose there exists an explicit PRG for $(n, w+1, \log(3nw/\varepsilon))$-ROBP with error $\varepsilon/3$ which has seed length s. Then there exists an offline randomized algorithm which approximates the n-th power of any substochastic matrix within error ε with probability*

at least $1-\varepsilon$*. Moreover, such algorithm requires* $s+O(\log(w/\varepsilon))$ *bits of randomness,* $O(s+O(\log(w/\varepsilon)))$ *bits of reusable processing space and* $O(\log(nw/\varepsilon))$ *bits of sensitive processing space.*

Proof. Given an input M, first we round each entry down to $d=\log(3nw/\varepsilon)$ bits of precision. Then we will get a substochastic matrix M' such that each entry of M' is a multiple of 2^{-d}, and $\|M-M'\|_{\max}\le\varepsilon/3nw$. Then we have

$$\|M^n-(M')^n\|_{\max}\le\|M^n-(M')^n\|\le n\,\|M-M'\|\le nw\,\|M-M'\|_{\max}\le\frac{\varepsilon}{3}.$$

Then we construct a $(n,w+1,d)$-ROBP B as follows. For each $t\in[n]$, we connect k edges from node $(t-1,i)$ to node (t,j) if $M'_{i,j}=k\cdot 2^{-d}$. Then for each node $(t-1,i)$ which doesn't have 2^d outgoing edges yet, we connect more edges from $(t-1,i)$ to a dummy node $(t,w+1)$. For each dummy node we connect 2^d edges to the dummy node in the next layers. It is easy to observe that $(M'^n)_{i,j}$ is exactly the probability that we start a random walk from $(0,i)$ and reach (n,j). Now for every $i,j\in[w]$, define $B_{i,j}(x)$ to be the indicator for whether we will reach (t,j) if we start from $(0,i)$ and follow $x\in(\{0,1\}^d)^n$. Then $\mathbb{E}_x\left[B_{i,j}(x)\right]=(M'^n)_{i.j}$. Take the given PRG G, we have

$$\left|\mathop{\mathbb{E}}_r\left[B_{i,j}(G(r))\right]-\mathop{\mathbb{E}}_x\left[B_{i,j}(x)\right]\right|\le\frac{\varepsilon}{3}.$$

Now define the offline randomized algorithm $\widehat{\mathrm{Pow}}$ to be

$$\widehat{\mathrm{Pow}}(M,y)_{i,j}=\mathop{\mathbb{E}}_z\left[B_{i,j}(G(\mathrm{Samp}(y,z)))\right],$$

where Samp is a $(\varepsilon/3,\varepsilon/w^2)$-sampler. By definition of sampler, with probability at least $(1-(\varepsilon/w^2))$ over the choice of y, we have

$$\left|\widehat{\mathrm{Pow}}(M,y)_{i,j}-\mathop{\mathbb{E}}_r\left[B_{i,j}(G(r))\right]\right|\le\frac{\varepsilon}{3}.$$

By union bound, with probability at least $(1-\varepsilon)$,

$$\left\|\widehat{\mathrm{Pow}}(M,y)_{i,j}-\mathop{\mathbb{E}}_r\left[B_{i,j}(G(r))\right]\right\|_{\max}\le\frac{\varepsilon}{3}$$

for every $i,j\in[w]$. Therefore by triangle inequality we have

$$\left\|\widehat{\mathrm{Pow}}(M,y)-M^n\right\|_{\max}\le\varepsilon$$

with probability at least $1-\varepsilon$.

Finally we compute the complexity of $\widehat{\mathrm{Pow}}$. By Lemma 19, the required randomness in this offline randomized algorithm is $s+O(\log(1/\varepsilon)+\log\log(w/\varepsilon))$. The required processing space is the processing space for samplers and PRGs. Observe that the only sensitive data is the second input for sampler (i.e. z); the current node in the ROBP, which takes $\log(nw)$ bits to store; and a d-bit block in $G(\mathrm{Samp}(y,z))$ indicating which entry of M we should check. Therefore the required sensitive processing space is only $O(\log(nw/\varepsilon))$ bits. ◀

With Armoni's sampler trick, if we have any PRG for $(n,w+1,\log(3nw/\varepsilon))$-ROBP, we can always plug it into the Saks-Zhou scheme regardless of whether it has the high-probability property. Specifically, as suggested in [4], if we have a PRG of seed length $O(\log^2(n)+\log^{4/3}(w/\varepsilon))$, we can even prove that $\mathbf{BPL}\subseteq\mathbf{L}^{4/3}$.

A.3 Saks-Zhou-Armoni Scheme with PRPDs

Finally we see how to apply a PRPD in the above scheme.

▶ **Lemma 46.** *Suppose there exists an explicit* $\mathrm{poly}(nw/\varepsilon)$*-bounded PRPD* (G, ρ) *for* $(n, w+1, \log(3nw/\varepsilon))$*-ROBP with error* $\varepsilon/3$ *which has seed length* s*. Then there exists an offline randomized algorithm which approximates the* n*-th power of any substochastic matrix within error* ε *with probability at least* $1-\varepsilon$*. Moreover, such algorithm requires* $s + O(\log(w/\varepsilon))$ *bits of randomness,* $O(s + O(\log(w/\varepsilon)))$ *bits of reusable processing space and* $O(\log(nw/\varepsilon))$ *bits of sensitive processing space.*

Proof. The proof is basically the same as Lemma 45, with the following two difference.

- $\widehat{\mathrm{Pow}}(M, y)_{i,j}$ is defined as $\mathbb{E}_z\left[\rho(\mathrm{Samp}(y, z)) \cdot B_{i,j}(G(\mathrm{Samp}(y, z)))\right]$ instead.
- If (G, ρ) is k-bounded, then we will choose Samp as a $(\varepsilon/6k, \varepsilon/w^2)$ sampler instead.

It's not hard to verify the correctness. (With Claim 18 which shows that samplers can also be used for functions with output range $[-k, k]$.) The required sensitive processing space is increased to $O(\log(nw/\varepsilon) + \log(k))$, which is still $O(\log(nw/\varepsilon))$ if $k = \mathrm{poly}(nw/\varepsilon)$. ◀

One may notice that there might have negative output in our new definition of $\widehat{\mathrm{Pow}}$. However, this is not a problem when applying Saks-Zhou argument because we only rely on the non-negativeness of matrices $\overline{M_i}$, which is independent of the approximation algorithm we use. With the above lemma we have the following corollary, which better motivates the problem of getting improved seed length for PRPDs:

▶ **Corollary 47.** *If there exists a* $\mathrm{poly}(nw/\varepsilon)$*-bounded explicit PRPD for* (n, w, d)*-ROBP with error* ε *which has seed length* $O(\log^2(n) + (\log(w/\varepsilon) + d)^{4/3})$*, then* $\mathbf{BPL} \subseteq \mathbf{L}^{4/3}$*.*

Proof. Apply the Saks-Zhou scheme (Lemma 43), and take $n_1 = 2^{\log^{2/3}(n)}$, $n_2 = \log^{1/3}(n)$, $d = 10\log(n)$ and $\varepsilon = 2^{-2d}$. The required subprocedure $\widehat{\mathrm{Pow}}$ would be approximating the n_1-th power of $n \times n$ substochastic matrices within error ε. By Lemma 46, there exists an offline randomized algorithm which approximates M^{n_1} within error $\varepsilon = 2^{-2d} = \mathrm{poly}(1/n)$, which requires sensitive processing space $O(\log(n))$ and offline randomness + reusable processing space $O(\log^2(n_1) + \log^{4/3} n) = O(\log^{4/3}(n))$. Therefore the total space complexity is $O(\log(n) \cdot n_2 + \log^{4/3}(n)) = O(\log^{4/3}(n))$. ◀

▶ Remark 48. Note that while we only construct PRPDs for (n, w)-ROBP in this paper, it is possible to adapt our construction to get PRPDs for (n, w, d)-ROBP with seed length $O(\log n \log(nw) \log\log(nw) + \log(1/\varepsilon) + d)$: simply replace the base case with a sampler with d-bit output. Since it doesn't imply better derandomization for **BPL** anyway, we keep $d = 1$ for simplicity.

B Proof of Lemma 19

▶ **Lemma 49** (Lemma 19, restated. [27, 10])**.** *For every* $\delta, \varepsilon > 0$ *and integer* m*, there exists a* (ε, δ)*-sampler* $f : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *s.t.* $d = O(\log\log(1/\delta) + \log(1/\varepsilon))$ *and* $n = m + O(\log(1/\delta)) + O(\log(1/\varepsilon))$*. Moreover, for every* x, y*,* $f(x, y)$ *can be computed in space* $O(m + \log(1/\delta) + \log(1/\varepsilon))$*.*

We will use the equivalence between seeded randomness extractor and oblivious sampler by Zuckerman [33]. To achieve the parameter we need, we need a "high-entropy seeded extractor" such that the seed length only depends on entropy loss but not the length of source. We will use the standard "block-source" construction for high-entropy extractor

which can be found in [11, 27]. For simplicity, we will use simple composition instead of zig-zag composition [27] because we are not aiming for optimal entropy loss. We will use the following standard lemmas for the extractor construction. Some of the following lemmas are implicit in their original source, and we recommend the readers to see [12, 32] for a proof.

▶ **Definition 50** ([8]). *(X_1, X_2) is a (k_1, k_2)-*block source *if X_1 is a k_1-source, and for every $x_1 \in \mathrm{Supp}(X)$, X_2 conditioned on $X_1 = x_1$ is a k_2-source.*

▶ **Lemma 51** ([11]). *Let $X \in \{0,1\}^n$ be a $(n - \Delta)$ source. Then for every integer $0 \le t \le n$, X is ε-close to a $(t - \Delta, n - t - \Delta - \log(1/\varepsilon))$-block source (X_1, X_2) where $X_1 \in \{0,1\}^t$ and $X_2 \in \{0,1\}^{n-t}$.*

▶ **Lemma 52** ([25]). *Let $E_1 : \{0,1\}^{n_1} \times \{0,1\}^d \to \{0,1\}^{d_2}$ be a (k_1, ε_1) extractor and $E_2 : \{0,1\}^{n_2} \times \{0,1\}^{d_2} \to \{0,1\}^m$ be a (k_2, ε_2) extractor. Define $E((x_1, x_2), s) = E_1(x_2, E_2(x_1, s))$. Then for every (k_1, k_2)-block source $(X_1, X_2) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$, $E((X_1, X_2), U_d)$ is $(\varepsilon_1 + \varepsilon_2)$-close to uniform.*

▶ **Lemma 53** ([11]). *For every $\varepsilon, \Delta > 0$ and integer n there exists a $(n - \Delta, \varepsilon)$ extractor $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^n$ with $d = O(\Delta + \log(1/\varepsilon))$, and for every x, y, $E(x, y)$ can be computed in space $O(n + \log(1/\varepsilon))$.*

▶ **Lemma 54** ([12, 19]). *For every $\varepsilon > 0$, integer $m > 0$ and $n \ge 2m$, there exists a $(2m, \varepsilon)$ extractor $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log m + \log(1/\varepsilon))$, and for every x, y, $E(x, y)$ can be computed in space $O(m + \log(1/\varepsilon))$.*

▶ **Lemma 55** ([33]). *Every $(n - \log(1/\delta) - 1, \varepsilon)$-extractor is a (ε, δ)-sampler.*

Now we show how to construct the sampler we need, and that it is indeed space efficient.

Proof. Let $\Delta = \log(1/\delta) + 1$. Let $E_1 : \{0,1\}^m \times \{0,1\}^{d_1} \to \{0,1\}^m$ be an $(m - \Delta, \varepsilon/3)$-extractor from Lemma 53, w.l.o.g. assume that $d_1 \ge \Delta + \log(3/\varepsilon)$. Then let $E_2 : \{0,1\}^{3d_1} \times \{0,1\}^d \to \{0,1\}^{d_1}$ be an $(2d_1, \varepsilon/3)$-extractor from Lemma 54. Then we claim that $E : \{0,1\}^{m+3d_1} \times \{0,1\}^d \to \{0,1\}^m$, defined as $E((x_1, x_2), s) = E_1(x_1, E_2(x_2, s))$, is a $(m + 3d_1 - \Delta, \varepsilon)$ extractor, and hence a (ε, δ) sampler by Lemma 55.

To prove the claim, consider any $(m + 3d_1 - \Delta)$-source X. By Lemma 51, X is $(\varepsilon/3)$-close to a $(m - \Delta, 3d_1 - \Delta - \log(3/\varepsilon))$-block source $(X_1, X_2) \in \{0,1\}^{3d_1} \times \{0,1\}^m$. By Lemma 52, $E_1(X_1, E_2(X_2, U_d))$ is $2\varepsilon/3$-close to uniform. Since $E(X, U_d)$ is $\varepsilon/3$-close to $E_1(X_1, E_2(X_2, U_d))$, by triangle inequality it is ε-close to uniform. Moreover, $d = O(\log(d_1/\varepsilon)) = O(\log\log(1/\delta) + \log(1/\varepsilon))$, $n = m + 3d_1 = m + O(\log(1/\delta) + \log(1/\varepsilon))$, and the required space to compute E is $O(m + d_1 + \log(1/\varepsilon)) = O(m + \log(1/\varepsilon) + \log(1/\delta))$. ◀

Circuit Lower Bounds from NP-Hardness of MCSP Under Turing Reductions

Michael Saks
Rutgers University, Piscataway, NJ, USA
saks@math.rutgers.edu

Rahul Santhanam
University of Oxford, UK
rahul.santhanam@cs.ox.ac.uk

Abstract

The fundamental Minimum Circuit Size Problem is a well-known example of a problem that is neither known to be in P nor known to be NP-hard. Kabanets and Cai [18] showed that if MCSP is NP-hard under "natural" m-reductions, superpolynomial circuit lower bounds for exponential time would follow. This has triggered a long line of work on understanding the power of reductions to MCSP.

Nothing was known so far about consequences of NP-hardness of MCSP under general Turing reductions. In this work, we consider two structured kinds of Turing reductions: *parametric honest* reductions and *natural* reductions. The latter generalize the natural reductions of Kabanets and Cai to the case of Turing-reductions. We show that NP-hardness of MCSP under these kinds of Turing-reductions imply superpolynomial circuit lower bounds for exponential time.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Circuit complexity; Theory of computation → Complexity classes

Keywords and phrases Minimum Circuit Size Problem, Turing reductions, circuit lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.26

Funding *Rahul Santhanam*: This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement No. 615075.

1 Introduction

The Minimum Circuit Size Problem (MCSP) is a fundamental problem in computational complexity that has been studied in various forms since the 1950s [24, 1]. Given the truth table of a Boolean function f and a parameter s, the question is whether f has Boolean circuits of size at most s. It is easy to see that MCSP is in NP: we can simply guess a circuit C of size at most s, and verify that for each input of f that C computes f correctly. Since we are given the truth table of f explicitly, this verification can be done in polynomial time.

MCSP is a rare example of a natural problem in NP for which we neither know that the problem is in P nor that it is NP-complete. There is some evidence that the problem is not in P. The celebrated results on "natural proofs" by Razborov and Rudich [23] can be interpreted as saying that MCSP is not in P if one-way functions exist [18, 2].

Regarding the question of NP-completeness, however, the evidence is more mixed. It is reported that Levin delayed publishing his seminal results on NP-completeness because he was hoping to show that MCSP is NP-complete [8]. More than 50 years later, we still have no idea.

It has been known for a while that the analogue of MCSP for DNFs, where we ask whether the function corresponding to a given truth table has small DNFs, is NP-complete [20, 10, 5]. Intuitively it would appear that checking if a given function has small circuits is at least as hard a problem as checking if a given function has small DNFs. But it does not seem easy to make this intuition precise in terms of giving a reduction from the latter to the former[1].

Looking at the proof that the analogue of MCSP for DNFs is NP-hard, one notices that an essential component of the reduction is an explicit function, namely Parity, that is hard for DNFs. This leads one to wonder whether the difficulty of coming up with an NP-hardness reduction for MCSP is related to the difficulty of proving Boolean circuit lower bounds for explicit functions.

Kabanets and Cai show that the answer is yes, in an influential paper [18] which led to a resurgence of interest in MCSP among complexity theorists. They define a notion of "natural reduction" to MCSP. A natural reduction is a many-one reduction in which the parameter as well as the size of the output depend only on the size of the input. Kabanets and Cai observe that standard reductions showing NP-hardness of various well-studied problems are natural in this sense. They then show that any natural reduction of SAT to MCSP implies that exponential time requires super-polynomial size Boolean circuits. Note that this does not give evidence *against* NP-hardness. However, given the well-known difficulty of proving circuit lower bounds for exponential time, it does indicate that *arguing* for NP-hardness of MCSP via natural reductions is likely to be difficult.

The work of Kabanets and Cai has led to a long line of results on reductions to MCSP and their implications [2, 3, 4, 7, 21, 17, 16, 6, 22, 15, 14]. Much of this work has focused on reductions implementable within low-depth complexity classes [7, 21, 6, 22] or on many-one and truth-table reductions [7, 21, 17, 16].

In this work, our focus is on polynomial-time Turing reductions to MCSP, for which very little is known. On the one hand, we are motivated by the question of NP-hardness of MCSP, and whether more powerful reductions help in this regard. Allender and Das [3] showed that every problem in Statistical Zero Knowledge reduces to MCSP via probabilistic Turing reductions, so at least for some problems that are believed to be hard, Turing reductions have been found to be useful in reducing to MCSP.

On the other hand, from the perspective of complexity theorists who care about circuit lower bounds, we find the connections between reductions to MCSP and circuit lower bounds intriguing, and would like to understand the extent to which these connections hold.

1.1 Our Results

We consider two kinds of structured Turing reductions - *parametric honest* reductions, which were defined in the context of many-one reductions by Hitchcock and Pavan [17], and *natural* reductions, which generalize the notion defined by Kabanets and Cai [18]. Parametric honest reductions are Turing reductions where there is a polynomial lower bound on the parameter of each query, as a function of input size. Natural reductions are Turing reductions where the parameter of any query made on an input only depends on the size of the input.

Our first main result shows that NP-completeness of MCSP under parametric honest Turing reductions implies super-polynomial circuit lower bounds for E.

▶ **Theorem 1.** *If* MCSP *is* NP*-hard under parametric honest Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.

[1] Similar issues come up when trying to prove hardness of properly learning Boolean circuits.

We remark that every language in P reduces to MCSP under parametric honest Turing reductions, so we crucially need to exploit the assumption that we reduce from hard languages in NP in order to derive the conclusion of Theorem 1.

Our second main result shows that NP-completeness of MCSP under natural Turing reductions implies super-polynomial circuit lower bounds for E.

▶ **Theorem 2.** *If* MCSP *is* NP*-hard under natural Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

The proof of Theorem 2 builds upon the ideas of Theorem 1, but is significantly more involved technically, and requires several new ideas.

We next describe the main ideas behind our proofs.

1.2 Main Ideas

Our starting point is the idea of Kabanets and Cai [18] for deriving circuit lower bounds from structured many-one reductions. Suppose we could efficiently generate NO instances of any length for some language L that reduces to MCSP via an m-reduction. We could hope to efficiently generate truth tables of hard functions as follows: we generate a NO instance x_n of L of length n, and then apply the m-reduction f from L to MCSP to generate a truth table y_n and a parameter s_n. Since x_n is a NO instance of L and f is an m-reduction, we are guaranteed that y does not have circuits of size s.

If s is large, say at least n^ϵ for some constant ϵ, then we are done, since we have efficiently generated a truth table y of a Boolean function that does not have sub-exponential size Boolean circuits as a function of its input length. But a priori we do not have control over the parameter s associated with y. Imagine for example a reduction that magically knows that each x_n is a NO instance and therefore produces s_n so small that it gives us no non-trivial information about circuit lower bounds.

Kabanets and Cai get around this problem by considering natural reductions, where the parameter s_n depends only on the input length n. Now there are two cases: either all but finitely many input lengths yield small parameter $s_n \leq \log(n)^{\log\log(n)}$, or infinitely many input lengths yield parameter $s_n > \log(n)^{\log\log(n)}$. In the first case, Kabanets and Cai use a standard indirect diagonalization argument to argue that circuit lower bounds follow. In the second case, we directly get hard truth tables for infinitely many n by composing the reduction with the efficient generation of NO instances. Thus, in either case, we get $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

Note that in the case where infinitely many input lengths yield hard parameters, the Kabanets-Cai argument does not actually exploit the hardness of the language L from which we are reducing. The argument works equally well starting from a language in polynomial time. When we deal with Turing reductions rather than m-reductions, this feature of their argument is an issue, since any language L in P can be decided trivially by Turing reductions that ignore the answers to any queries they ask and simulate the polynomial-time algorithm for L directly to arrive at an answer.

Let us consider first the case of parametric honest reductions. Parametric honesty means that we don't need to worry about the parameter being small. Suppose that SAT reduces to MCSP via a parametric honest Turing reduction implemented by an oracle machine M. Our first idea is simple: we define a simulation A of SAT which simply runs M and answers all queries of M by "yes". This algorithm A clearly runs in polynomial time. If it is correct on all but finitely many instances, we have that $\mathsf{NP} = \mathsf{P}$ and we can then derive circuit lower bounds using a standard indirect diagonalization technique.

If A is wrong on infinitely many instances, however, it is not obvious how to use this. Here we exploit the main idea of Gutfreund, Shaltiel and Ta-Shma [12], who show how any efficient algorithm A that fails to solve SAT can be used to efficiently produce, for infinitely many n, a small set of instances S of sizes polynomially related to n, such that A fails on some instance in S. This is very useful for us, as an instance on which A fails must ask a query to M that is answered incorrectly. But since we always answer queries by "yes" in A, a query is only answered incorrectly if the true answer is "no", i.e., if the string y that is queried is the truth table of a hard Boolean function. We don't know which of the queries this is, but we do know that there is a *first* wrongly answered query, and that by the parametric honesty condition, this query has large parameter. Hence, by simply concatenating the queries asked by A on the instances in S, we obtain hard truth tables of size $\text{poly}(n)$ for infinitely many n, and this enables us to conclude that E requires super-polynomial size circuits.

This is the argument for the proof of Theorem 1. The assumption of parametric honesty is an important part of this argument. For the proof of Theorem 2, where we consider natural reductions, we need to work much harder and use several new ideas.

We would like to adopt the following strategy. Suppose there is a natural Turing-reduction from SAT to MCSP. Choose a "threshold" parameter s and try to solve SAT by running the reduction and answering all queries with parameter less than the threshold correctly by doing brute force search, and all queries with parameter greater than the threshold by "yes".

Either this simulation succeeds, or it fails. If it succeeds, and the threshold parameter s is small, NP is easy, and using the standard indirect diagonalization argument, we get that E requires large circuits. If it fails, we would like to argue that we can efficiently find instances on which the simulation fails. Such an instance must ask queries with parameters larger than the threshold that are wrongly answered "yes", so the concatenation of such queries must be a hard truth table.

To efficiently find instances on which the simulation fails, we can again to try to use the strategy of [12]. Unfortunately, there is a catch. The time taken by the algorithm of [12] to find hard instances is at least the time taken for the simulation, which is exponential in s. It might be that the parameter for queries on these instances is just slightly larger than s. Thus we will take time exponential in s to find truth tables requiring circuits only slightly larger than s, which won't give us sufficiently strong circuit lower bounds for E.

To remedy this issue, we consider two different simulations, M and M'. M is guaranteed to be correct and proceeds by just evaluating queries using brute force search. If M is relatively efficient, we are in good shape, as we can use indirect diagonalization to get the consequence we want. If M is not always efficient, we get an infinite sequence of input lengths on which large parameter queries are made. Here "large" means at least s, where s is super-polylogarithmic in n, but still small enough so that the indirect diagonalization helps us get circuit lower bounds in case M succeeds.

In fact, by using the paddability of SAT in our argument, we get not just an infinite sequence of input lengths yielding large parameters, but a sequence of long intervals, within which all input lengths yield large parameters. This will be important later, as the [12] algorithm does not produce hard instances at a given input length, but rather at one of two input lengths which are polynomially related.

The simulation M' will use a smaller parameter threshold s', chosen so that s' is superpolylogarithmic in n, but also so that s is superpolynomial in s'. The simulation M' implements our initial idea: we answer queries with parameter at most s' by brute force search, and other queries by "yes".

Now either this simulation is correct on all but finitely many input lengths, or it fails on infinitely many. If it is correct on all but finitely many input lengths, we can use indirect diagonalization as before, but it is not clear how to use the failure on infinitely many, without more structural information on where the failures occur.

So we argue instead as follows, using a notion of "robust simulation" introduced in [11]. If the simulation is correct, robustly often, than we can carry the indirect diagonalization through. Otherwise, the simulation fails on at least one input length in every large enough sequence of input lengths. Using the large stretches of input lengths with large parameters from our first simulation, we get that there are infinitely many long sequences of input lengths where the simulation M' fails and the corresponding parameters of queries are large.

Now we use the strategy of [12]. We use simulation M' itself to find a small set of inputs on which M' fails, and on which the parameter of correctly produced queries is at least s. M' will take time only exponential in s', and since s is super-polynomial in s', the time taken to find this small set of inputs will be sub-exponential in s. By concatenating the queries made on this small set of inputs, we get a hard truth table, and hence can arrive at our desired conclusion.

There are numerous technical details which we have ignored in the above discussion, but hopefully this description helps in understanding the proof of Theorem 2.

1.3 Related Work

As mentioned earlier in the Introduction, there have been several works in recent years studying reductions to MCSP and their consequences. We briefly survey this work in the context of our results. We focus on work that either shows interesting consequences of hardness for MCSP under certain types of reductions, or unconditionally rules out hardness, as this is closest in spirit to our work here.

Kabanets and Cai [18] defined natural m-reductions, and showed that NP-hardness of MCSP under natural m-reductions implies super-polynomial circuit lower bounds for E. Murray and Williams [21] showed unconditionally that MCSP is unconditionally not NP-hard under very efficient local reductions where each output bit only depends on a limited number of input bits. In fact, their technique even rules out P-hardness of MCSP under local reductions. They also show that NP-hardness of MCSP under m-reductions (without the "natural" constraint of [18]) implies EXP $\neq$ ZPP. Hitchcock and Pavan showed that NP-hardness of MCSP under general truth-table reductions implies EXP $\neq$ ZPP. Allender, Holden and Kabanets [7] consider the question of hardness for various oracle versions of MCSP and show some unlikely consequences of hardness under efficient reductions when the oracle is powerful.

The work that is closest to ours is that of Hirahara and Watanabe [16], who also consider Turing-reductions to MCSP. They consider a kind of reduction they call "oracle-independent", and show that no language outside of P reduces to MCSP using an oracle-independent reduction. However, there are examples of useful reductions known that are not oracle-independent [6] [2]. It is also shown in [16] that Turing-hardness of approximating the minimum circuit size implies EXP $\neq$ ZPP. We note that the hardness of approximation factors required for this result are large - the assumption is that it is hard to approximate the logarithm of the circuit size to within any constant factor. Moreover, [16] do not derive a circuit lower bound from this assumption, but the weaker statement that EXP $\neq$ ZPP.

[2] On the other hand, we are not aware of any hardness results for MCSP using reductions that are not natural.

2 Preliminaries

2.1 Basic Notions

We refer to the book by Arora and Barak [9] for definitions of standard complexity classes.

Given a circuit class $\mathfrak{C}$, we use $\mathfrak{C}[s(n)]$ to refer to the class of functions computed by $\mathfrak{C}$-circuits of size $s(n)$.

Given a language $L \subseteq \{0,1\}^*$, co-L denotes the complement of L. Given language L and $n \in \mathbb{N}$, L_n is used to refer to $L \cap \{0,1\}^n$.

Given a set $S \subseteq \mathbb{N}$ and languages L and L', we say L agrees with L' on S if for all $n \in S$, $L_n = L'_n$.

A time-constructible function $T : \mathbb{N} \to \mathbb{N}$ is a function such that there is a Turing machine transducer M, which for each n, on input 1^n halts with output $T(n)$ within $O(T(n))$ steps.

We need a generalization of the notion of robustly-often simulation from [11]. Given $g : \mathbb{N} \to \mathbb{N}$, a set $S \subseteq \mathbb{N}$ is called *g-robust* if for each constant k, there is $m \in \mathbb{N}$ such that for all n such that $m \leq n \leq g(m)^k$, $n \in S$. Note that any g-robust set is also poly(g)-robust.

Given a function g, a language L and a complexity class $\mathcal{C}$, we say L is g-robustly often in $\mathcal{C}$ if there is $L' \in \mathcal{C}$ for which there is a g-robust set S such that L agrees with L' on S. Given function g and complexity classes $\mathcal{C}$ and $\mathcal{C}'$, we say that $\mathcal{C}$ is g-robustly often in $\mathcal{C}'$ if for all languages $L \in \mathcal{C}$, L is g-robustly often in $\mathcal{C}'$. We use the term "robustly often" as a synonym for g-robustly often with g the identity function.

The proposition below is a parameterized version of Theorem 12 in [11].

▶ **Proposition 3.** *Let T be a time-constructible function such that $T(n) \geq n$ for all n. If* SAT *is $T(\text{poly}(n))$-robustly often in* DTIME(T)*, then* Σ_2P *is robustly often in* DTIME$(T(\text{poly}(T(\text{poly}(n)))))$.

The following is a simple application of standard diagonalization [13].

▶ **Proposition 4.** *Let T be a time-construtible function such that $T = o(2^n)$. Then* E *is not robustly often in* DTIME(T).

2.2 Resource-Bounded Kolmogorov Complexity

We study various notions of resource-bounded Kolmogorov complexity, and associated decision problems.

Fix a universal Turing machine U. The Kt complexity of a string is defined as follows: $\mathsf{Kt}(x) = min\{|p| + \log(t) | U(p)$ halts and outputs x within t steps$\}$.

In the MKtP problem, the instance is a string x together with a parameter s, and the question is whether $\mathsf{Kt}(x) \leq s$.

It is known that MKtP is complete for E under polynomial-size truth-table reductions [2], but completeness under uniform polynomial-time reductions is unknown.

In the MCSP problem, the instance is a string x together with a parameter s, and the question is whether x is the truth table of a Boolean function with circuit complexity at most s.

MCSP is easily seen to be in NP, but it is unknown whether MCSP is NP-hard. A brute-force search strategy of trying all circuits C of size at most s and checking if any of them computes the function with truth table x can easily be implemented to run in time poly$(|x|)s^{O(s)}$.

2.3 Reductions

We will work with constrained notions of polynomial-time Turing reduction.

▶ **Definition 5.** *A polynomial-time Turing reduction from a language L to* MCSP *or* MKtP *is called parametric honest if there is a constant $\epsilon > 0$ such that for every $x \in \{0,1\}^*$, every query made by the reduction on x has parameter bounded below by $|x|^\epsilon$.*

Note that parametric honesty is not implied by the standard notion of honesty for polynomial-time reductions, which only requires the output length to be bounded below by some polynomial in the input length. Implicit in the definition is that the lower bound on parameters of queries is only required to hold when previous queries are answered correctly; when queries are answered wrongly, "all bets are off".

▶ **Definition 6.** *A polynomial-time Turing reduction from a language L to* MCSP *or* MKtP *is called natural if for any input $x \in \{0,1\}^*$, the parameter of any query made by the reduction on x depends only on $|x|$.*

This notion of naturalness generalizes the notion used by Kabanets and Cai for m-reductions to Turing reductions. In fact, it is slightly weaker than their notion when restricted to m-reductions, since it does not constrain the length of the output, merely the parameter. As with the definition of parametric honesty, the condition on parameters is only required to hold when previous queries are answered correctly.

Naturalness is incomparable to parametric honesty - parametric honesty allows the parameter to vary but restricts its range of variation, while naturalness allows the parameter to have a large range of variation but insists that it is the same for all queries made on any input of a given length.

3 Parametric Honest Reductions

Our first result unconditionally rules out hardness of MKtP for E under parametric honest Turing reductions. For many results on hardness of MCSP and consequences thereof, MKtP is a "test case" where analogous results can be shown unconditionally and where the ideas are often simpler [8, 16].

▶ **Theorem 7.** MKtP *is not hard for* E *under parametric honest Turing reductions.*

Proof. Let L be a tally set that is in E but not in P. The existence of such a tally set L follows by a standard diagonalization argument.

Now suppose, for the purpose of contradiction, that MKtP is hard for E under parametric honest Turing reductions. This implies that there is an oracle Turing machine M running in time at most cn^k, where c and $k \geq 1$ are constants, such that M decides L correctly with oracle access to MKtP, and moreover there is a constant $\epsilon > 0$ such that each query made by M on an input of length n has parameter at least n^ϵ. We show how to use M to decide $L \in$ P, contrary to the assumption on L.

Consider the following polynomial-time machine A that attempts to decide L. Given input 0^n, A runs the oracle machine M, answering each query made by M with "yes". When the simulation of M is complete, A accepts iff M accepts. Since M is polynomial-time, so is A. We show that A correctly decides L for large enough n.

Given input 0^n, M makes some sequence of oracle queries $q_1, q_2 \ldots q_t$ during the simulation by A, where $0 \leq t \leq cn^k$. We show that for each i, q_i has Kt complexity $O(\log(n))$. Observe that q_i can be generated by the universal machine U implicit in the definition of Kt complexity

in time poly(n) given descriptions of i, n and the oracle machine M: it simply needs to run M on 0^n answering the first $i-1$ queries made by M with "yes", and then output the i'th query. The description length required is $O(\log(n))$ since $i \leq cn^k$, and moreover the time taken by U is polynomial in n, hence using the definition of Kt complexity, we have an $O(\log(n))$ upper bound on the Kt complexity of q_i. Let C be a constant such that the Kt complexity of each q_i is bounded above by $C\log(n)$.

By the assumption that the reduction is parametric honest, the parameter of q_i is at least n^ϵ, which for large enough n is greater than $C\log(n)$. Hence for large enough n and for each i, it follows that A's assumed answer for the oracle query q_i is correct for each i, and hence that A outputs the same answer as M would given the true oracle MKtP. Since M is an oracle Turing machine correctly implementing a reduction from L to MKtP, A is a polynomial-time Turing machine correctly deciding L for large enough n, contradicting the assumption on L. ◀

We now re-state and prove Theorem 1.

▶ **Theorem 8.** *If* MCSP *is* NP*-hard under parametric honest Turing reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$.

Proof. Suppose that MCSP is NP-hard under parametric honest Turing reductions. Let M be a polynomial-time oracle Turing reduction implementing a parametric honest reduction from SAT to MCSP, and let ϵ be a constant such that the parameter of any query made by M on any SAT instance of length x is bounded below by $|x|^\epsilon$.

Consider the following polynomial-time machine A which attempts to decide SAT. On an input x of SAT, A runs the oracle Turing machine M. Assume wlog that all queries asked by M have length a power of two - any queries for which this is not the case may be eliminated by assuming the answer is 'no'. For every query asked by M with length a power of two, A assumes the answer is "yes", and continues the simulation of M. When the simulation concludes, A accepts iff M accepts.

Clearly A runs in polynomial time. If A decides SAT correctly, then we have that $\mathsf{NP} = \mathsf{P}$. Hence the Polynomial Hierarchy collapses to P. But this implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$; if the contrary were true, then E would collapse to the Polynomial Hierarchy using the Karp-Lipton theorem for E, and hence to P, contradicting the hierarchy theorem for deterministic time.

Suppose now that $\mathsf{NP} \neq \mathsf{P}$. In particular, this means that there are infinitely many instances on which A does not decide SAT correctly. If not, we could hardcode the finitely many instances on which A fails to compute SAT into a new polynomial-time machine A' which runs in polynomial time and solves SAT correctly on all instances, contradicting the assumption that $\mathsf{NP} \neq \mathsf{P}$.

Now, we can use the main result of Gutfreund, Shaltiel and Ta-Shma. They show that if $\mathsf{NP} \neq \mathsf{P}$, then for any polynomial-time algorithm A that fails to compute SAT, there is a polynomial-time Turing machine B and a constant d, such on input 1^n, B outputs three instances x_1, x_2, x_3 each of length n or length n^d such that for infinitely many n, A fails to decide SAT correctly on at least one of these three instances.

We show how to use B to argue again that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$. Consider the three instances x_1, x_2, x_3 output by B on input 1^n. For each $i, 1 \leq i \leq 3$, let Q_i be the concatenation of all queries made by M when M is simulated by A on input x_i. Let Y be the concatenation of Q_1, Q_2, Q_3. Note that $|Y| = \text{poly}(n)$, since M is a polynomial-time oracle Turing machine. Also note that each query made by M on an input of length n has parameter at least n^ϵ, by the parametric honesty of the reduction, as long as all previous queries were answered correctly. Each such query was assumed by A to have a "yes" answer. It could not be the

case that all such answers were correct for all queries made by M on inputs x_1, x_2, x_3 - if they were, then A would correctly solve SAT on x_1, x_2, x_3, using the fact that M implements a correct polynomial-time Turing reduction from SAT.

Thus we get that there is a first query made by M on one of x_1, x_2, x_3 that has parameter at least n^ϵ and is wrongly answered "yes". Fix such a query y. Since M implements a Turing reduction from SAT to MCSP, y is the truth table of a Boolean function on $O(\log(n))$ bits that requires circuits of size n^ϵ. We have that y is a substring of Y, hence Y must also require circuits of size $\Omega(n^\epsilon)$. We are using here the simple fact that if y is a substring (with length a power of two) of the truth table Y of a Boolean function with circuits of size s, then y is the truth table of a Boolean function with circuits of size $O(s)$.

Hence, for infinitely many n, on input 1^n, we can compute the truth table Y of a Boolean function on $O(\log(n))$ bits such that the function requires circuits of size n^ϵ, i.e., of exponental size in the length of the input to the Boolean function. This implies $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$. ◀

4 Natural Reductions

We require a technical lemma allowing us to use indirect diagonalization based on robustly-often simulations.

▶ **Lemma 9.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function such that $T(\mathrm{poly}(T(\mathrm{poly}(n)))) = o(2^n)$. If* SAT *is $T(\mathrm{poly}(n))$-robustly often in time $T(n)$, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$.

Proof. Suppose SAT is $T(\mathrm{poly}(n))$-robustly often in time $T(n)$. By Proposition 3, we have that $\Sigma_2\mathsf{P}$ is robustly often in time $T(\mathrm{poly}(T(\mathrm{poly}(n))))$. Assume for the sake of contradiction that $\mathsf{E} \subseteq \mathsf{SIZE}(\mathrm{poly})$. The Karp-Lipton theorem for E, credited to Meyer [19], implies that E collapses to $\Sigma_2\mathsf{P}$. Note that if T is time-constructible, $T' = T(\mathrm{poly}(T(\mathrm{poly}(n))))$ is also time-constructible. Thus we have that E is robustly often in time T' for a time-constructible $T' = o(2^n)$, contradicting Proposition 4. ◀

We now re-state and prove Theorem 2. In the proof below, we use the fact that SAT is paddable - there is a polynomial-time computable *padding* function f such that for each $x \in \{0,1\}^*$, and for each $m \geq |x|$, $|f(x, 1^m)| = m$ and $f(x, 1^m)$ is in SAT iff x is in SAT. Of course, this depends on the encoding of SAT, but it suffices for us that there is some encoding for which paddability is true in the form above, and all other standard properties of SAT continue to hold.

▶ **Theorem 10.** *If* MCSP *is* NP*-hard under natural Turing-reductions, then* $\mathsf{E} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$

Proof. Suppose that MCSP is NP-hard under natural Turing reductions. Let M be an oracle Turing machine implementing a natural reduction from SAT to MCSP. By the definition of natural reduction, this means that there is a function $s : \mathbb{N} \to \mathbb{N}$ such that all queries of M on input of size n for SAT have parameter $s(n)$. If there are no queries that M makes on an input of size n, we set $s(n) = 0$.

Let $s_1(n) = \log(n)^{\log\log(n)}$, and $s_2(n) = \log(n)^{\log\log\log(n)}$. Moreover let $r(n) = n^{\log(n)}$. We define two Turing machines M_1 and M_2 that attempt to decide SAT using the oracle machine M.

Machine M_1 operates as follows. On input x of size n, M_1 computes strings $x_1 \ldots x_{r(n)-n}$, where $x_i = f(x, 1^{n+i})$. Here f is the padding function for SAT. Let $x_0 = x$. M_1 iteratively does the following starting with $i = 0$. It simulates M on x_i. If an oracle query is made with parameter k, it checks if k is at most $s_1(n+i)$. Note that since M implements a natural reduction, $k = s(n+i)$. If k is at most $s_1(n+i)$, it solves the corresponding MCSP instance

by brute force search in time at most $\text{poly}(n+i)2^{s_1(n+i)^2}$ and continues the simulation. Note that any succeeding queries must also have parameter at most $s_1(n+i)$, therefore the simulation continues to completion. M_1 accepts iff M accepts. If no oracle queries are made, then too the simulation continues to completion, and M_1 returns the same answer as M.

In case the parameter k for a query is greater than $s_1(n+i)$, M_1 increments i, and repeats the process, as long as $i \leq r(n) - n$. If $i > r(n) - n$, M_1 simply runs the simulation of M on x to completion, now answering all queries by brute-force search regardless of the parameter size.

$\triangleright$ Claim. Either SAT is decidable in time $\text{poly}(r(n))2^{s_1(r(n))^2}$, or there is an infinite sequence of disjoint intervals $I_j = [n_j, n'_j)$ of input lengths such that $n'_j = r(n_j)$ for each j, and moreover for each j, if $n \in I_j$, then $s(n) > s_1(n)$.

We establish the Claim. Suppose that SAT is not decidable in time $\text{poly}(r(n))2^{s_1(r(n))^2}$. We argue that for any $n_0 \in \mathbb{N}$, there exists an input length $n > n_0$ such that $s(n) \geq s_1(n)$, and moreover $s(m) \geq s_1(m)$ for every $n < m < r(n)$. Indeed, if this were not the case, for every $n > n_0$ one of the iterations of M_1 would succeed for some $i < r(n) - n$, and this would mean that the entire simulation would complete in time at most $\text{poly}(r(n))2^{s_1(r(n))^2}$. Note that when the simulation completes, it does solve SAT correctly, as M is an oracle Turing reduction correctly reducing SAT to MCSP, and any completing simulation uses correct answers to oracle queries.

Now we pick the intervals I_j inductively as follows. Let n_1 be the least integer such that $s(m) > s_1(m)$ for every $n_1 \leq m \leq r(n_1)$, and let $n'_1 = r(n_1)$. Inductively, suppose we have picked $n_1 \ldots n_j$. We pick n_{j+1} to be the least integer such that $n_{j+1} > r(n_j)$, and moreover $s(m) > s_1(m)$ for each $n_{j+1} \leq m \leq r(n_{j+1})$. Such an integer exists by the argument of the previous paragraph. Let $n'_{j+1} = r(n_j)$, and set $I_j = [n_j, n'_j)$. This sequence of intervals is clearly disjoint, and satisfies the property in the Claim.

The first disjunct of the Claim implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$, using Lemma 9 and the fact that $T(n) = \text{poly}(r(n))n^{s_1(r(n))}$ is time-constructible and satisfies $T(\text{poly}(T(\text{poly}(n)))) = o(2^n)$, for the given choice of r and s_1. Hence, in the rest of our proof, we assume that the second disjunct holds, i.e., that there is an infinite sequence of long intervals within which each input length generates queries with a large parameter.

Now we define machine M_2. On input x of length n, M_2 uses M to try to solve the search version of SAT, i.e., to find a satisfying assignment for x if one exists. M_2 uses the standard search-to-decision reduction for SAT based on self-reducibility, simulating M to answer any decision questions as described below. In order to find a satisfying assignment for x, this search-to-decision procedure might call the decision procedure for SAT on inputs x' such that $|x'| < |x|$; in such cases, M_2 pads up x' to length $|x|$ by using the padding function $f(x', 1^{|x|})$ and runs the simulation of M on $f(x', 1^{|x|})$. This guarantees that all inputs on which M is run during the simulation have the same length. At the end of the search-to-decision procedure, if a satisfying assignment is successfully found, M_2 accepts, else it rejects.

Next we describe how M_2 uses M to decide if an input is in SAT. M_2 simulates the oracle machine M on the input. If M asks a query with parameter greater than $s_2(n)$, M_2 assumes the answer is "yes" and continues the simulation. If M asks a query with parameter at most $s_2(n)$, M_2 solves the query by brute force search and continues the simulation. At the end of the simulation, M_2 accepts iff M accepts.

M_2 always halts within time $\text{poly}(n)2^{s_2(n)^2}$. Unlike machine M_1, machine M_2 is not guaranteed to solve SAT correctly. However, since M_2 has been modified to solve the search version of SAT, the only way M_2 can make a mistake on x is to answer "no" when the true answer is "yes". We will argue that whether or not M_2 solves SAT correctly, circuit lower bounds follow.

We consider two cases. The first is that there is a $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly. In this case, we can apply Lemma 9 with $T(n) = \text{poly}(n)2^{s_2(n)^2}$, noting that T is time-constructible and that $T(\text{poly}(n)) < 2^{s_2(n)^3}$ for large enough n. Thus we get that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$ in this case.

The remaining case is that there is no $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly. In this case, we apply the main idea of [12], using in addition our assumption that there is an infinite sequence $\{I_j\}$ of disjoint intervals $[n_j, r(n_j))$ of input lengths such that for each j and $n \in [n_j, r(n_j))$, we have that $s(n) > s_1(n)$.

If there is no $2^{s_2(n)^3}$-robust set S on which M_2 solves SAT correctly, there must be a constant k such that for each n, there is $m < 2^{ks_2(n)^3}$ such that M_2 fails to solve SAT correctly on some input of length m. Now we use the existence of the sequence $\{I_j\}$ of intervals from the Claim. Since $r(n) = n^{\log(n)}$ is significantly larger than $2^{ks_2(n)^3}$ for large enough n, it follows that there is an infinite sequence of input lengths $\{m_i\}$ such that the following hold: (i) M_2 fails to solve SAT correctly on some input x of length m_i such that M on input x asks a query with parameter at least $s_1(m_i)$ (ii) For all input lengths m such that $m_i \leq m \leq 2^{s_2(m_i)^4}$, $s(m) \geq s_1(m)$.

Now we apply the main idea of [12], by attempting to use the simulation M_2 itself to find a small set of inputs for which M_2 fails to decide SAT correctly on at least one of these inputs. We define a Turing machine A as follows. On input 1^m, A formulates the question of whether there is an instance of size m on which M_2 fails to solve SAT correctly and asks a query with parameter at least $s_2(m)$ as an instance y of SAT. Since M_2 runs in time $\text{poly}(n)2^{s_2(n)^2}$, such an instance y of size at most $2^{s_2(m)^3}$ for large enough m can be computed in time at most $2^{s_2(m)^3}$. A then runs M_2 on y to find a satisfying assignment for y, as in the proof of the main result of [12]. For all $m = m_i$ for some i, either A succeeds, in which case it finds an instance x of length m on which M_2 fails, or it finds a set of at most three instances y_1, y_2, y_3 such that $|y_j| = |y|$ for each $1 \leq j \leq 3$, and M_2 fails on at least one of these three instances. Note that by item (ii) in the para above, in the first case x must ask some query with parameter at least $s_1(m)$ that is answered wrongly, and in the second case, one of y_1, y_2, y_3 must ask some query with parameter at least $s_1(|y|)$ that is answered wrongly. In either case, a wrongly answered query must be the truth table of a hard function. In the first case, we have that the concatenation Z_i of all queries asked by x is the truth table of a function on $O(\log(m))$ input bits that does not have circuits of size $s_1(m)$, and in the second case we have that the concatenation Z_i of all queries asked by y_1, y_2, y_3 must be the truth table of a function on $O(\log(|y|))$ input bits that does not have circuits of size $s_1(|y|)$. A outputs Z_i.

In either case, A runs in time $\text{poly}(|Z_i|)2^{s_2(|Z_i|)^3}$ and outputs a string Z_i that does not have circuits of size $s_1(|Z_i|)$. Since $s_2(|Z|) = s_1(|Z|)^{o(1)}$, this implies that $\mathsf{E} \not\subseteq \mathsf{SIZE}(\text{poly})$. ◀

5 Open Problems

The main open problem is to derive circuit lower bounds from unrestricted Turing reductions from SAT to MCSP. One obstacle here is that we don't we know that MKtP is not hard for E under Turing reductions - this is a potentially simpler "test case" that could be indicative. We note that [16] have shown that a gap version of MKtP is not hard for E, when the gap between the upper and lower thresholds for Kt-complexity is $\omega(\log(n))$.

An easier problem is to derive circuit lower bounds from unrestricted many-one reductions from SAT to MCSP. Even this is unknown, though we do know that such reductions imply $\mathsf{EXP} \neq \mathsf{ZPP}$ [21, 17].

References

1 Eric Allender. The complexity of complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.

2 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

3 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.

4 Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *CoRR*, abs/1511.08189, 2015. `arXiv:1511.08189`.

5 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing DNF formulas and AC0 circuits given a truth table. *Electronic Colloquium on Computational Complexity (ECCC)*, 126, 2005.

6 Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *International Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 54:1–54:14, 2017.

7 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In *International Symposium on Theoretical Aspects of Computer Science* (STACS), pages 21–33, 2015.

8 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.

9 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1st edition, 2009.

10 Sebastian Czort. The complexity of minimizing disjunctive normal form formulas. Master's Thesis, University of Aarhus, 1999.

11 Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Information and Computation*, 256:149–159, 2017.

12 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007.

13 Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, pages 285–306, 1965.

14 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference, CCC 2018*, pages 5:1–5:31, 2018.

15 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.

16 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In *Conference on Computational Complexity* (CCC), pages 18:1–18:20, 2016.

17 John M. Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *Conference on Foundation of Software Technology and Theoretical Computer Science* (FSTTCS), pages 236–245, 2015.

18 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

19 Richard Karp and Richard Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309, 1980.

20 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

21 Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

22 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

23 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

24 Boris A. Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

Finding Small Satisfying Assignments Faster Than Brute Force: A Fine-Grained Perspective into Boolean Constraint Satisfaction

Marvin Künnemann
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
marvin@mpi-inf.mpg.de

Dániel Marx
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
dmarx@mpi-inf.mpg.de

Abstract

To study the question under which circumstances small solutions can be found faster than by exhaustive search (and by how much), we study the fine-grained complexity of Boolean constraint satisfaction with size constraint exactly k. More precisely, we aim to determine, for any finite constraint family, the optimal running time $f(k)n^{g(k)}$ required to find satisfying assignments that set precisely k of the n variables to 1.

Under central hardness assumptions on detecting cliques in graphs and 3-uniform hypergraphs, we give an almost tight characterization of $g(k)$ into four regimes:

1. Brute force is essentially best-possible, i.e., $g(k) = (1 \pm o(1))k$,
2. the best algorithms are as fast as current k-clique algorithms, i.e., $g(k) = (\omega/3 \pm o(1))k$,
3. the exponent has sublinear dependence on k with $g(k) \in [\Omega(\sqrt[3]{k}), O(\sqrt{k})]$, or
4. the problem is fixed-parameter tractable, i.e., $g(k) = O(1)$.

This yields a more fine-grained perspective than a previous FPT/W[1]-hardness dichotomy (Marx, Computational Complexity 2005). Our most interesting technical contribution is a $f(k)n^{4\sqrt{k}}$-time algorithm for SubsetSum with precedence constraints parameterized by the target k – particularly the approach, based on generalizing a bound on the Frobenius coin problem to a setting with precedence constraints, might be of independent interest.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Fine-grained complexity theory, algorithmic classification theorem, multivariate algorithms and complexity, constraint satisfaction problems, satisfiability

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.27

Funding *Dániel Marx*: Research of the second author was supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement SYSTEMATICGRAPH (No. 725978).

1 Introduction

Extensive research in complexity theory has established methods to give precise qualitative results on the computational hardness of problems. In this context, a basic question that we would like to answer is: When are there algorithms better than a brute force search, and if there are, how much improvement is possible compared to brute force? In problem settings where the task is to find a solution of size k, typically it is easy to obtain algorithms with running time of the form $\mathcal{O}(n^{k+\mathcal{O}(1)})$ by a brute force search of every possible solution. In such cases, beating brute force could involve having an algorithm with a term $(1-\epsilon)k + O(1)$

in the exponent for some $\epsilon > 0$, or having sublinear (e.g, $O(k/\log k)$ or $O(\sqrt{k})$) dependence on k in the exponent, or we might be able to completely remove k from the exponent of n with an $f(k)n^{O(1)}$ time algorithm.

In this paper, we study the above question in the context of the class of Boolean Constraint Satisfaction problems. Fixing a *constraint family* $\mathcal{F}$ of Boolean functions, the task is to determine an assignment to Boolean variables $x_1, \ldots, x_n$ satisfying a given conjunction of constraints of the form $f(x_{i_1}, \ldots, x_{i_r})$ with $f \in \mathcal{F}$ and $i_1, \ldots, i_r \in [n]$. Here, the natural notion of the solution size k is the number of variables set to 1 and we consider the task of determining a satisfying assignment with precisely k ones. This class indeed contains a variety of problems: basic graph problems such as the vertex cover problem ($\mathcal{F}$ consists of the binary OR) and the independent set problem in graphs ($\mathcal{F}$ consists of the binary NAND) or d-uniform hypergraphs ($\mathcal{F}$ consists of the d-ary NAND), but also other natural problems such as a formulation of SubsetSum parameterized by the target k ($\mathcal{F}$ consists of binary equality)[1], finding a solution of a (sparse) linear system over GF(2) where each linear equality involves at most a constant number r of variables and the solution must have precisely k ones ($\mathcal{F}$ consists of all linear constraints of arity at most r), as well as finding a closed set of size k in a directed graph ($\mathcal{F}$ consists of the binary implication). Note that the last problem can be seen to be equivalent to a variant of SubsetSum that prescribes precedence constraints on the items and uses an unary encoding for all item sizes.

The time complexity inside this class varies widely: Vertex cover is famously fixed-parameter tractable when parameterized by k, with a best current running time bound of $\mathcal{O}(kn + 2^{O(k)})$ [16]. It is even simpler to solve the SubsetSum formulation in time $\mathcal{O}(m + k^2) = \mathcal{O}(n^2)$ (where m is the number of edges in the graph) by a straightforward algorithm[2]. The fastest known algorithm for independent set [29], however, relies on the sophisticated techniques for matrix multiplication, and achieves a running time of $\mathcal{O}(n^{(\omega/3)k})$ for k divisible by 3, where $\omega \leq 2.373$ is the matrix multiplication exponent. For finding closed sets of size k, a surprisingly simple $\mathcal{O}(n^{k/2})$-time algorithm[3] improves over brute force even without matrix multiplication, but a priori there is little indication for the optimality of this approach. Finally, for finding independent sets in 3-uniform hypergraphs, no substantially faster-than-brute-force algorithm is known.

The central purpose of this paper is to give a detailed understanding of the time complexity of Boolean constraint satisfaction parameterized by solution size k, particularly when k is considered a (large) constant: How precisely can we determine the running time $f(k)n^{g(k)}$, with $g(k)$ as small as possible? Note that for large constant k, we have $f(k)n^{g(k)} = \mathcal{O}(n^{g(k)})$ and aim to determine its optimal polynomial-time complexity.

A classification of the second author [28] resolves the qualitative question for which $\mathcal{F}$ the problem is solvable in FPT time (assuming $\mathsf{FPT} \neq \mathsf{W}[1]$), i.e., when $g(k)$ can be bounded by a constant independent of k. In particular, from this classification, we obtain that among

[1] To see the correspondence, note that if $\mathcal{F}$ consists of the binary equality, $\mathsf{SAT}(\mathcal{F})$ asks to find a union of connected components of total size k. By representing each connected component by its size (after linear-time preprocessing), this is precisely the SubsetSum problem with target k.

[2] Determine all connected components in time $\mathcal{O}(m)$ and solve a SubsetSum instance on the component sizes in time $\mathcal{O}(k^2)$ using Bellman's pesudopolynomial-time algorithm or recent improvements [23, 8].

[3] Without loss of generality, it suffices to solve the following problem: given a node-weighted DAG $G = (V, E)$ and $k \in \mathbb{N}$, find a weight-k subset $S \subseteq V$ such that $u \in S$ and $(u, v) \in E$ implies $v \in S$. If S contains a set S' of at most $k/2$ *sources* (i.e., vertices that have no incoming edges from other vertices in S), we can simply guess S' and check that S' and the set of all descendants of S' have total weight k. If S contains no such set S' of size at most $k/2$, we can guess all $\leq k/2$ non-sources S'', remove all incoming edges to S'' and find a weight-$(k - |S''|)$ set of vertices with out-degree 0.

the above examples, vertex cover, SubsetSum with target k, and the sparse linear systems over GF(2) can be solved in time $f(k)n^c$, while for independent set (in both graphs and hypergraphs) as well as SubsetSum with precedence constraints, the exponent of n must depend on k (unless $\mathsf{FPT} = \mathsf{W}[1]$). Can we obtain tight bounds on $g(k)$ when it must depend on k? In particular, can we determine for which $\mathcal{F}$ the brute-force $\mathcal{O}(n^{k+c})$-time solution is essentially optimal?

1.1 Our Results

Let us formally state our problems and results.

▶ **Problem 1.1.** *Let $\mathcal{F}$ be a finite* constraint family *of Boolean functions. The problem* $\mathsf{SAT}(\mathcal{F})$ *asks to determine whether a given formula ϕ on Boolean variables $x_1, \dots, x_n$ is satisfiable by an assignment with k ones, where ϕ is a conjunction of m constraints C of the form $f(\mathbf{x})$, where $f : \{0,1\}^r \to \{0,1\}$ is a constraint function in $\mathcal{F}$ and $\mathbf{x}$ is an r-tuple of variables among $x_1, \dots, x_n$.*

Note that if all $f \in \mathcal{F}$ have arity bounded by r, then there are at most $\mathcal{O}(n^r)$ possible constraints, and exhaustive search solves $\mathsf{SAT}(\mathcal{F})$ in time $\mathcal{O}(n^{k+r})$.

We will show that the complexity of $\mathsf{SAT}(\mathcal{F})$ is tightly characterized by the set of functions *expressible as restrictions of constraint functions $f \in \mathcal{F}$*. To formally introduce this concept, let $f : \{0,1\}^r \to \{0,1\}$ be an arbitrary Boolean function. We say that $g : \{0,1\}^s \to \{0,1\}$ is a *restriction of f* if it is obtained from g by replacing each argument of f by either the constant 0, the constant 1, or an argument of g, i.e., we can partition $[r]$ into $X_1, \dots, X_s, Z_0, Z_1$ such that

$$g(x_1, \dots, x_s) = f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_s \dots x_s}^{X_s}, \overbrace{0 \dots 0}^{Z_0}, \overbrace{1 \dots 1}^{Z_1}).$$

Here, $\overbrace{y \dots y}^{Y}$ denotes plugging in y for all (not necessarily contiguous) positions $Y \subseteq [r]$, see Section 2.

▶ **Definition 1.2.** *Let $g : \{0,1\}^d \to \{0,1\}$ be an arbitrary Boolean function. A constraint family $\mathcal{F}$* represents *g if there is some $f \in \mathcal{F}$ such that g is a restriction of f. If $\mathcal{F}$ does not represent g, we say that $\mathcal{F}$* avoids *g.*

Let $\mathrm{IMPL} : \{0,1\}^2 \to \{0,1\}$ and $\mathrm{NAND}_d : \{0,1\}^d \to \{0,1\}$ be the binary implication and d-ary NAND function, respectively, i.e.,

$$\mathrm{IMPL}(y_1, y_2) \coloneqq \overline{y_1} \vee y_2,$$
$$\mathrm{NAND}_d(y_1, \dots, y_d) \coloneqq \overline{\bigwedge\nolimits_{i=1}^{d} y_i}.$$

In [28], it is shown that assuming $\mathsf{FPT} \neq \mathsf{W}[1]$, $\mathsf{SAT}(\mathcal{F})$ is solvable in FPT time $f(k)n^c$ if and only if $\mathcal{F}$ is *weakly separable*, which is a condition equivalent to $\mathcal{F}$ avoiding NAND_2 and IMPL. We show an almost tight characterization of $g(k)$ (under plausible assumptions from fine-grained complexity theory) that depends only on whether or not $\mathcal{F}$ represents IMPL, NAND_2 or NAND_d for higher order $d \geq 3$. Specifically, we obtain the following main theorem, illustrated in Figure 1.

▶ **Theorem 1.3.** *Let $\mathcal{F}$ be a finite constraint family.*

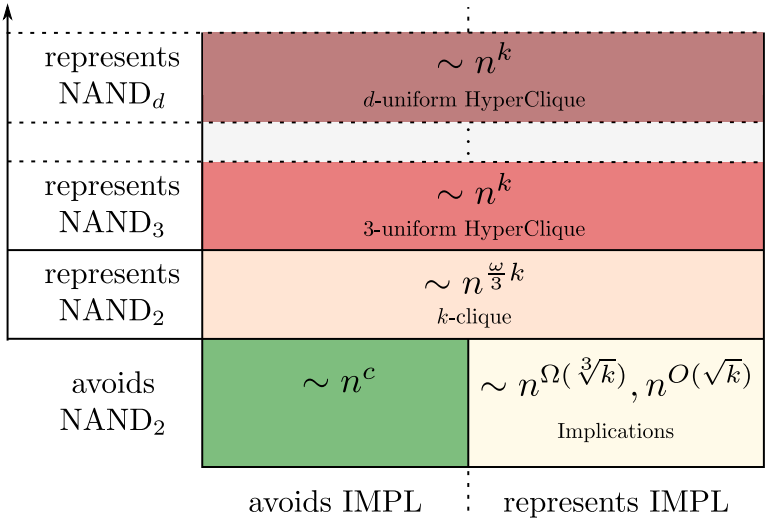


Figure 1 Overview over our main results. The parts of the diagram to the right of the vertical IMPL line depict $\mathcal{F}$ representing IMPL, while the parts to the left avoid IMPL. Analogously, the parts of the diagram above a NAND$_d$ line depict NAND$_d$-representing $\mathcal{F}$, while those below avoid NAND$_d$. For each cell, we illustrate our (typically matching) algorithmic and hardness results, together with a problem that is complete for this cell (in a certain sense). For clarity of presentation, we drop additional $f(k)n^c$-factors of stated running times.

1. **[FPT regime]**
   If $\mathcal{F}$ avoids both NAND$_2$ *and* IMPL*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{c_\mathcal{F}}$.*
2. **[Subexponential regime]**
   If $\mathcal{F}$ represents IMPL*, but avoids* NAND$_2$*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{4\sqrt{k}+c_\mathcal{F}}$;*
   furthermore, for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$, SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c_\mathcal{F}}$, unless the k-clique conjecture fails.*
3. **[Clique regime]**
   If $\mathcal{F}$ represents NAND$_2$*, but avoids* NAND$_3$*, then there is a computable $f(k)$ and constant $c_\mathcal{F}$ such that* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/3)k+c_\mathcal{F}}$;*
   furthermore for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$, SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(\omega/3-\epsilon)k+c_\mathcal{F}}$, unless the k-clique conjecture fails.*
4. **[Brute-force regime]**
   If $\mathcal{F}$ represents NAND$_3$*, then for no computable $f(k)$ and constants $c_\mathcal{F}, \epsilon > 0$,* SAT$(\mathcal{F})$ *can be solved in time $f(k)n^{(1-\epsilon)k+c_\mathcal{F}}$, unless the 3-uniform k-HyperClique conjecture fails.*

That is, we only have four regimes: $g(k)$ is either constant, sublinear in k with a value between essentially $(\omega/6)\sqrt[3]{k}$ and $4\sqrt{k}$, the clique detection bound of essentially $(\omega/3)k$, or the brute force bound of essentially k. Note that we do not try to optimize the bounds on $f(k)$, which generally are bounded by $r^{\mathcal{O}(k^3)}$, where r is the arity of $\mathcal{F}$.

Let us briefly discuss our hardness assumptions and their plausibility (for a detailed discussion, we refer to Section 2.1): The k-clique conjecture postulates that there is no $\mathcal{O}(n^{(\omega/3-\epsilon)k+c})$ time algorithm for detecting a k-clique in a given graph, with a matching upper bound of $\mathcal{O}(n^{(\omega/3)k+1})$ known since 1985 [29]. By now, it has been used, e.g., to justify (conditional) optimality of Valiant's parser for context free grammars [2] and to give

conditional lower bounds for string problems [10, 1], average-case settings [5], and more. Notably, the only k-clique algorithm known to break brute force by a polynomial factor makes crucial use of fast matrix multiplication techniques – unfortunately, these techniques do not extend to finding cliques in *hypergraphs*. This has led to the d-uniform HyperClique conjecture (for arbitrary $d \geq 3$): This conjecture states that there is no algorithm beating brute force, i.e., no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time algorithm, for detecting a k-clique in a given d-uniform hypergraph. It has been used to expose hardness of problems in sparse graphs [27], for first-order queries to relational databases (specifically, in model-checking [9] and enumeration contexts [13]), and for the orthogonal vectors problem [3]; furthermore, it is known that its refutation requires giving a $\mathcal{O}((2-\epsilon)^n)$-time algorithm for Max-3SAT – we refer to [2, 27] for more detailed discussions of the plausibility of the (d-uniform Hyper-)Clique conjecture.

Interestingly, our classification does not fundamentally rely on the validity of the d-uniform HyperClique conjecture: If, for some $d \geq 3$, the d-uniform HyperClique conjecture is eventually refuted, we obtain faster-than-brute-force algorithms for all NAND_{d+1}-avoiding families!

Coarser Classification. While we state our results under very fine-grained hardness assumptions on clique and hyperclique detection, we may also state a coarser classification assuming only the assumption that k-clique cannot be solved in time $f(k)n^{o(k)}$. Already under this assumption, which is implied by the Exponential Time Hypothesis (see [14, 15]), our reductions and algorithms show that there exists an FPT regime where $g(k)$ is a constant, a subexponential regime where $g(k)$ is between $\Omega(\sqrt[3]{k})$ and $\mathcal{O}(\sqrt{k})$, and a linear regime where $g(k) = \Theta(k)$. However, based on the Exponential Time Hypothesis only, we cannot distinguish problems solvable in time $f(k)n^{(1\pm o(1))k}$ and $f(k)n^{(\omega/3\pm o(1))k}$, and thus cannot differentiate in the linear regime.

Examples. From our general classification, we can draw some interesting specific corollaries (assume here that k is a large constant):

- **3-SAT:** Finding satisfying assignments with k ones for 3-CNF formulas ($\mathcal{F}$ consists of all ternary functions with a single falsifying assignment) requires brute force time $n^{(1-o(1))k}$ under the 3-uniform HyperClique conjecture. However, if we drop a single function from $\mathcal{F}$ (specifically NAND_3, i.e., each constraint must have at most two negative literals), the problem can be solved in time $\mathcal{O}(n^{(\omega/3)k+c})$, which is essentially optimal under the k-Clique conjecture.
- **Subexponential cases:** We obtain $n^{\mathcal{O}(\sqrt{k})}$-time algorithms for interesting special cases: Beyond precedence-constrained SubsetSum with target k (i.e, $\mathsf{SAT}(\{\mathrm{IMPL}\})$), this includes $\mathsf{SAT}(\{f\})$ with $f(y_1, y_2, y_3) \coloneqq y_1 \Rightarrow (y_2 \vee y_3)$, and, more generally, every finite set of *dual-Horn constraints* (i.e., constraints that can be represented by clauses with at most a single negative literal)[4]. This also includes examples beyond dual-Horn constraints such as $\mathsf{SAT}(\{\mathrm{IMPL}, f'\})$ with f' being defined by $f'(y_1, y_2, y_3) = 1$ iff $(y_1, y_2, y_3) \in \{(0,0,0), (1,0,1), (1,1,0)\}$. Interestingly, all of these problems have the same (conditionally optimal) time complexity of $f(k)n^{\Theta(k^\alpha)}$ with $1/3 \leq \alpha \leq 1/2$; determining the precise value of α remains a challenge for future work.

[4] It is known that a constraint is dual-Horn if and only if it its satisfying assignments are closed under union, which immediately implies that it cannot contain NAND_2 as a restriction.

1.2 Technical Overview

We give an overview of the technical challenges that are handled in our work, from the highest running time regime to the lowest running time regime:

Brute-force regime. It is straightforward to obtain hardness for NAND_3-representing families by the following intuitive approach: To reduce from k-clique in a 3-uniform hypergraph G, we let x_i denote whether we include vertex v_i in our k-clique. By the standard observation that a clique in a hypergraph G is an independent set of its complement graph $\overline{G}$, we only need to ensure that for each edge $e = (v_a, v_b, v_c)$ of $\overline{G}$, not all vertices are included in our clique, i.e., $\mathrm{NAND}_3(x_a, x_b, x_c)$ holds. Since $\mathcal{F}$ represents NAND_3, we can express this constraint using an appropriate restriction of some $f \in \mathcal{F}$. Here, there is a technical issue of how we can generate the constants 0 or 1 to obtain the desired restrictions – using not particularly difficult, but careful constructions, we show that we can always simulate these constants as needed (Section 6).

Moderately hard regime. While the hardness of NAND_3-representing families is straightforward, it is surprising that this condition is in fact necessary for the brute-force approach to be (conditionally) optimal: If NAND_3 is *not* representable, we give a $f(k)n^{(\omega/3)k+c_{\mathcal{F}}}$-time algorithm via reduction to k-Clique.

The essential idea for this reduction is the following win-win argument. Let us denote by $a_{x,y}$ the weight-2 assignment setting only x and y to 1. Fix any weight-k satisfying assignment a. If there are two variables $x_i = x_{i'} = 1$ in a such that $a_{x_i,x_{i'}}$ is not satisfying, then we can use this pair of variables to "guide" our search towards a. We guess $x_i, x_{i'}$, identify a falsified constraint (of arity r) and guess an additional third variable from the at most r unguessed variables in this constraint. This means that by guessing two variables (n^2 possibilities), we obtain an additional variable almost for free (guessing r possibilities). That is, in the considered case we can identify 3 variables of a with a guess of rn^2 possibilities, which is a significant gain compared to the n^3 possibilities of brute force. Otherwise, if a has no such pair of variables, we observe that a satisfies already a simpler formula that uses only NAND_2's: specifically, the conjunction of $\mathrm{NAND}(x_i, x_{i'})$ for all i, i' such that assignment $a_{x_i,x_{i'}}$ violates the original formula. Furthermore, we show that since NAND_3 is not representable, any solution of the simpler formula indeed remains a solution of the original formula.

Interestingly, this reduction generalizes also to hypergraphs so that a refutation of the d-uniform HyperClique conjecture would give a $f(k)n^{(1-\epsilon)k+c_{\mathcal{F}}}$-time algorithm for NAND_{d+1}-avoiding families.

On the hardness side, analogously to the brute-force regime, it is rather straightforward to show that k-clique running time is indeed necessary for NAND_2-representing constraint families (see Section 6), which thus concludes a tight bound on $g(k)$ of essentially $(\omega/3)k$ in this regime.

Mildly hard regime. This is the technically most interesting regime. If NAND_2 is not representable, then $\mathsf{SAT}(\mathcal{F})$ might still not have an FPT algorithm, specifically, if it represents IMPL. Implicit in the $\mathsf{W}[1]$-hardness proof in [28] is a fine-grained lower bound of $n^{\Omega(\log k)}$ under the k-clique conjecture. By giving a careful adaptation of the lower bound of [28], we can strengthen this lower bound to $n^{\Omega(\sqrt[3]{k})}$. While it is conceivable that this lower bound can be strengthened to $n^{\Omega(\sqrt{k})}$, the structure of the construction suffers from a fundamental obstacle that makes a lower bound beyond $n^{\Omega(\sqrt{k})}$ seem unlikely. This raises the suspicion that

a $n^{o(k)}$-time algorithm for NAND_2-avoiding families could exist – and indeed, we manage to develop a $n^{\mathcal{O}(\sqrt{k})}$-time algorithm, which is perhaps the most interesting technical contribution of our paper.

To illustrate our approach, consider the problem Weighted DAG Implications: Given a DAG $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and a parameter $k \in \mathbb{N}$, the task is to find a set $S \subseteq V$ such that (1) $u \in S$ and $(u, v) \in E$ implies $v \in S$ and (2) S has total weight $\sum_{s \in S} w(s) = k$. Without edges, this problem simplifies to SubsetSum which we could solve in $\mathsf{poly}(k)$ time [23, 8]. However, to enable a generalization to our precedence setting, we describe a different approach based on a combinatorial property inspired by the famous Frobenius coin problem: Given coins of denominations $2 \leq d_1 < d_2 < \cdots < d_\ell$ with $\gcd(d_1, \ldots, d_\ell) = 1$, what is the largest number x not representable as $x = \sum_{i=1}^{\ell} \alpha_i d_i$ for some non-negative values $\alpha_i \geq 0$? A proof attributed to Schur (see [7, 30, 21]) yields an upper bound of $x \leq (d_1 - 1)(d_\ell - 1)$. Consequently, if $w_1 \leq \cdots \leq w_\ell$ with $\gcd(w_1, \ldots, w_\ell) \mid k$ are the weights occurring in an edgeless G, and $w_\ell \leq \sqrt{k}$, then there always exists a set S of total weight k, provided each weight occurs sufficiently often (say, at least k times). Thus, if we can preprocess the instance such that each weight is bounded by $\sqrt{k}$ and occurs sufficiently often, we can determine the answer to the instance by simply computing the gcd of the weights. Intuitively, this is possible in time $n^{\mathcal{O}(\sqrt{k})}$ by guessing the $\mathcal{O}(\sqrt{k})$ vertices of weight larger than $\sqrt{k}$, as well as brute-forcing vertices of each weight class containing only few vertices.

Interestingly, this approach can be lifted to the setting with precedence constraints. To this end, assume that the graph consists of layers $V_1, \ldots, V_\ell$ such that each V_i consists of a sufficiently large number of vertices of weight w_i and that all edges respect the layering (i.e., an edge between a vertex in V_i and a vertex in V_j implies $i > j$). We show the following property, which gives a generalization of Schur's bound to the precedence setting:

If for each vertex v, the total weight of its descendants (including v itself) is at most $\sqrt{k/2}$, then there exists a solution of total size k *if and only if* $\gcd(w_1, \ldots, w_\ell) \mid k$.

By an $n^{\mathcal{O}(\sqrt{k})}$-time preprocessing analogous to the intuitive arguments for the edge-less case, we can ensure that the preconditions are satisfied. We give the details of this approach in Section 3.

The above algorithmic insight solves the Weighted DAG Implications problem in time $\mathcal{O}(n^{4\sqrt{k}})$. To obtain such a bound for all NAND_2-avoiding families, we use a randomized reduction to Weighted DAG Implications. On a very high level, the approach is to create a Weighted DAG Implications instance G that contains only solutions that satisfy the given formula ϕ by iteratively choosing random implications consistent with certain solutions of ϕ. Doing this in an appropriate manner, a fixed feasible solution survives this process with $1/f(k)$ probability, which gives an algorithm running in time essentially $\mathcal{O}(f(k)n^{4\sqrt{k}})$. We give the details in Section 4.

Fast regime. For the remaining regime of families avoiding both IMPL and NAND_2, an $f(k)n^c$-time algorithm follows from [28], concluding the characterization.

1.3 Related work

Dichotomy theorems for constraint satisfaction have a rich history, starting with Schaefer's Theorem classifying Boolean Constraint Satisfaction Problems (CSPs) into either polynomial-time solvable or NP-complete [31]. The subsequent *Dichotomy conjecture* [20], which postulated that Schaefer's Theorem can be extended to any constant domain size beyond

Boolean, was resolved positively only recently by Bulatov [11] and Zhuk [34]. Further classifications have been investigated in a number of related settings, including quantified CSP (see, e.g., [18, 35]) and optimization variants (see, e.g. [17, 22]). Parameterizing by the solution size (as we do here), corresponding dichotomies have been obtained for Boolean [28] and larger domain sizes [12, 26], with a characterization of kernelization for Boolean domain given in [24] and a study of parameterized approximability given in [6]. A parameterized dichotomy for related local search tasks has been given in [25].

On a conceptual level, our work is related to a fine-grained classification result for model-checking first-order properties with a bounded number of quantifiers [9], where a fine-grained dichotomy under the 3-uniform HyperClique conjecture is given. Note, however, that the hardness criterion and techniques developed there are substantially different due to the different nature of the problem settings.

1.4 Open Problems

The main open problem raised by our work is to close the gap in the subexponential regime: Can we solve Implications $=$ $\mathsf{SAT}(\{\text{IMPL}\})$ already in $f(k)n^{\mathcal{O}(\sqrt[3]{k})}$ or can we improve our lower bound to $f(k)n^{\Omega(\sqrt{k})}$? Note that by our reductions, improved bounds directly transfer to all NAND_2-avoiding families.

Second, a natural direction is to extend our classification beyond the Boolean domain, i.e., give a fine-grained perspective building on [12, 26].

Finally, interesting related settings include natural problem variants with different size restrictions (*at most* k or *at least* k), local search tasks as well as optimization settings with weights on the variables or on the constraints.

2 Preliminaries

We write $[n] \coloneqq \{1, \dots, n\}$ and for any set S and integer d, let $\binom{S}{d}$ denote the set of d-element subsets of S.

For a finite constraint family $\mathcal{F}$, we say its *arity* r is the maximum arity of a function $f \in \mathcal{F}$. Since in the constraints of $\mathsf{SAT}(\mathcal{F})$, we may use variables in arbitrary order, we use the following notation for convenience: For any $f : \{0,1\}^r \to \{0,1\}$ and partition $X_1, \dots, X_s$ of $[r]$, we write

$$f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_s \dots x_s}^{X_s})$$

to denote the value of $f(u_1, \dots, u_r)$ where we plug in x_j for each u_i with $i \in X_j$. Correspondingly $g : \{0,1\}^d \to \{0,1\}$ can be obtained as a restriction of f if and only if there is a partition $X_1, \dots, X_d, Z_0, Z_1$ of $[r]$ such that

$$g(x_1, \dots, x_d) = f(\overbrace{x_1 \dots x_1}^{X_1}, \dots, \overbrace{x_d \dots x_d}^{X_d}, \overbrace{0 \dots 0}^{Z_0}, \overbrace{1 \dots 1}^{Z_1}).$$

We say that an assignment $a : [n] \to \{0,1\}$ has *weight* k if $\sum_{i=1}^{n} a(i) = k$. Furthermore, we say that a is dominated by an assignment $a' : [n] \to \{0,1\}$, written $a \leq a'$, if for all $i \in [n]$, we have $a(i) \leq a'(i)$. For a subset $S \subseteq [n]$, we let a_S denote the assignment that sets $a(i) = 1$ if and only if $i \in S$. We let $\text{ones}(a) \coloneqq \{x_i \mid a(i) = 1\}$ denote the set of 1-variables of a. For any constraint $C = f(\mathbf{x})$ where $\mathbf{x} = (x_{i_1}, \dots, x_{i_r})$ with $i_1, \dots, i_r \in [n]$, we let $\text{vars}(C) = \{x_{i_1}, \dots, x_{i_r}\}$ denote the variable set involved in C.

All graphs considered in this paper are simple, i.e., we disallow multiple edges and self-loops. If $G = (V, E)$ is a directed graph, we call $S \subseteq V$ a *closed* set if for all $(u, v) \in E$, we have that $u \in S$ implies that $v \in S$. We say that v is a *descendant* of u if v is reachable by a path from u and let $D(u)$ denote the set of descendants of u (including u itself). Analogously, if v is a descendant of u, we call u an *ancestor* of v. We extend the notation naturally to sets $S \subseteq V$ by defining $D(S) \coloneqq \bigcup_{u \in S} D(u)$. For a graph $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and $S \subseteq V$, we write $w(S) \coloneqq \sum_{v \in S} w(v)$. For any $S \subseteq V$, we let $G[S]$ denote the subgraph of G induced by S, i.e., the subgraph obtained by deleting all vertices in $V \setminus S$ and adjacent edges.

2.1 Hardness Assumptions

Let k-clique denote the following problem: Given a (simple) undirected graph $G = (V, E)$, determine whether there is a *clique* of size k, i.e., $S \subseteq V, |S| = k$ such that for all $\{u, v\} \in \binom{S}{2}$ we have $\{u, v\} \in E$. A simple algorithm [29] solves k-clique in time $\mathcal{O}(n^{(\omega/3)k})$ when k is divisible by 3, which extends to time $\mathcal{O}(n^{\lfloor k/3 \rfloor \omega + (k \bmod 3)})$ for arbitrary k (for more precise bounds, see [19]). This running time is conjectured to be best possible, in the following sense.

▶ **Hypothesis 2.1** (k-Clique Conjecture). *For no $c, \epsilon > 0$ and $f(k)$, there is an $f(k)n^{(\omega/3-\epsilon)k+c}$-time algorithm for k-Clique.*[5]

As without the use of matrix multiplication, no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time algorithms are known, a variant of the conjecture postulates that there are even no $\mathcal{O}(n^{(1-\epsilon)k+c})$-time *combinatorial* algorithms, i.e., algorithms avoiding the sophisticated algebraic techniques underlying current matrix multiplication algorithms.

By now, the k-clique conjecture has been used to explain hardness barriers in various contexts, such as the optimality of Valiant's parser for context-free grammar recognition [2], pattern matching in uncompressed and compressed strings [10, 1], average-case hardness [5] and more. For a more detailed discussion of this hardness assumption, we refer to [2].

The k-clique problem naturally extends to hypergraphs: Given a d-uniform hypergraph $G = (V, E)$, the d-uniform k-HyperClique problem asks to determine whether there is a *(hyper-)clique* of size k, i.e., $S \subseteq V, |S| = k$ such that for all subsets $S' \in \binom{S}{d}$, we have $S' \in E$.

▶ **Hypothesis 2.2** (d-Uniform k-HyperClique Conjecture). *Let $d \geq 3$. For no $c, \epsilon > 0$ and $f(k)$, there is an $f(k)n^{(1-\epsilon)k+c}$-time algorithm for d-uniform k-HyperClique.*

Similarly to the k-Clique conjecture, this hardness conjecture reveals hardness barriers in a number of contexts, such as hardness for problems on sparse graphs [27], for deciding or enumerating answers to first-order queries [9, 13] and for the study of fine-grained average-case complexity [5]. It is known that it implies the Orthogonal Vectors conjecture [3], however, refuting this conjecture requires (at least) to give an $\mathcal{O}((2-\epsilon)^n)$-time exact algorithm for Max3SAT; for details and further discussion of the plausibility of this conjecture, see [27].

[5] Note: sometimes, the k-clique conjecture is stated as

$$\inf\{F \mid 3k\text{-clique can be solved in time } n^{Fk+o(1)} \text{ for all (sufficiently large) constant } k\} = \omega,$$

which can be seen to be equivalent to the above formulation via a standard self-reduction for k-clique.

3 Algorithm for Implications

In this section, we give an algorithm for the problem Implications $=$ SAT($\{$IMPL$\}$) that is much faster than brute force and achieves $O(\sqrt{k})$ dependence of k in the exponent n. For convenience, we reduce Implications to the following problem. (Recall that for any graph $G = (V, E)$, we say that $S \subseteq V$ is *closed*, if for all $(u, v) \in E$, we have $u \in S$ implies $v \in S$.)

▶ **Problem 3.1** (Weighted DAG Implications)**.** *Given a DAG $G = (V, E)$ with node weights $w : V \to \mathbb{N}$ and parameter $k \in \mathbb{N}$, determine whether there is a closed set $S \subseteq V$ of weight exactly k, i.e., $w(S) = k$.*

The easy reduction works as follows. For each variable x_i, we introduce a corresponding vertex x_i of weight 1 and introduce an edge (x_i, x_j) for every implication constraint $x_i \Rightarrow x_j$ of ϕ. We contract each strongly connected component $C = \{v_1, v_2, \ldots, v_\ell\}$ in G to a single vertex v_C of weight $\sum_{i=1}^{\ell} w(v_i)$ in time $\mathcal{O}(n + m) = \mathcal{O}(n^2)$ [33]. Observe that the resulting graph is a DAG which has a closed set of weight k if and only if ϕ has satisfying assignment of weight k.

Recall that for any $v \in V$, we let $D(v)$ denote the set of descendants of v, i.e., the set of nodes reachable from v (including v).

As we will formally argue later, by a $f(k)n^{\mathcal{O}(\sqrt{k})}$-time preprocessing it is not difficult to preprocess a Weighted DAG Implications instance into the following form, which we call *Frobenius instance*, as it admits a combinatorial characterization of solvability that is analogous to Schur's bound for the Frobenius coin problem.

▶ **Definition 3.2.** *A* Frobenius instance with parameter k *is a weighted directed graph $G = (V, E, w)$ with ℓ parts $V = V_1 \cup V_2 \cup \cdots \cup V_\ell$ and weight function $w : V \to \mathbb{N}$ such that the following properties hold:*

(P1) *there are weights $w_1, \ldots, w_\ell$ such that $w(v) = w_i$ for all $v \in V_i$ and $i \in [\ell]$.*
(P2) *for any edge $(u, v) \in E$, we have $u \in V_i$ and $v \in V_j$ for some $\ell \geq i > j \geq 1$,*
(P3) *for all $i \in [\ell]$, we have $|V_i| \geq k$,*
(P4) *for all $v \in V$, we have $w(D(v)) \leq \sqrt{k/2}$.*

Intuitively, the necessary preprocessing follows from the following arguments: To ensure (P4), note that any weight-k closed set S has at most $\sqrt{2k}$ many vertices $v \in S$ with $w(D(v)) > \sqrt{k/2}$, which we can exhaustively enumerate with $n^{\mathcal{O}(\sqrt{k})}$-time overhead. By suitably arranging remaining nodes among the layers, it is straightforward to ensure (P1), (P2) and additionally that $\ell \leq f(k)$, since by (P4), each node has at most $\mathcal{O}(\sqrt{k})$ descendants. Finally, to ensure (P3), if any part V_i is small (i.e., $|V_i| < k$), we can exhaustively try out including any subset of V_i, introducing an overhead of only $2^{\mathcal{O}(k)}$ per V_i; since $\ell \leq f(k)$, this additional overhead is bounded by $f(k)2^{\mathcal{O}(k)}$.

If a Frobenius instance had no edges, then Schur's bound on the Frobenius coin problem implies that it has a solution if and only if $\gcd(w_1, \ldots, w_\ell) \mid k$. We prove that this criterion holds even in the setting of precedence constraints.

▶ **Lemma 3.3.** *Let G be a Frobenius instance with parameter k. Then G has a closed set of weight k if and only if $\gcd(w_1, \ldots, w_\ell) \mid k$.*

Proof. Since $\gcd(w_1, \ldots, w_\ell) \mid w(S)$ for any $S \subseteq V$, the condition $\gcd(w_1, \ldots, w_\ell) \mid k$ is necessary for G to have a closed set of weight k.

We show that this condition is also sufficient via induction on ℓ. In the base case $\ell = 1$, let $S \subseteq V_1 = V$ be an arbitrary subset of k/w_1 vertices (note that by $k/w_1 \leq k \leq |V_1|$, such a set indeed exists). By construction, S has weight $|S|w_1 = k$ and is closed, as G cannot contain any edges.

Thus let us assume that the claim holds for all $\ell' \leq \ell - 1$ and consider a Frobenius instance with $d' := \gcd(w_1, \ldots, w_\ell) \mid k$. Let $d := \gcd(w_1, \ldots, w_{\ell-1})$. We may assume that $d \nmid k$; otherwise, already the Frobenius instance $G[V_1 \cup \cdots \cup V_{\ell-1}]$ satisfies the assumption $\gcd(w_1, \ldots, w_{\ell-1}) \mid k$ and we obtain a closed set by inductive hypothesis.

Intuitively, we want to use the variables in V_ℓ to reach the target weight k modulo d; then we can reduce to a simpler instance where every weight (including the target weight) is divided by d. Note that we may assume

$$2 \leq d \leq \sqrt{k/2}, \tag{1}$$

where the lower bound follows from $d \nmid k$ and the upper bound follows from $d \leq \min_{i \in [\ell-1]} w_i \leq \sqrt{k/2}$, as $w(v) \leq w(D(v)) \leq \sqrt{k/2}$ for any $v \in V$.

Let b be the smallest non-negative integer such that $b \cdot w_\ell \equiv k \pmod{d}$. Such an integer exists and satisfies $b < d$: By Bézout's identity, since $\gcd(w_\ell, d) = d' \mid k$, there are coefficients β, γ such that $\beta w_\ell + \gamma d = k$, and thus any b with $b \equiv \beta \pmod{d}$ achieves the desired congruence.

Let $S \subseteq V_\ell$ be an arbitrary subset of size $b < d$; such a set indeed exists as $d \leq \sqrt{k/2} \leq k \leq |V_\ell|$. We observe that S satisfies

$$w(D(S)) \leq \sum_{s \in S} w(D(s)) \leq |S|\sqrt{k/2} \leq d\sqrt{k/2} \leq \frac{k}{2}, \tag{2}$$

where we used (P4) for the second inequality, and (1) for the last inequality. Consider the graph $G' = (V', E')$ obtained as a copy from G from which we delete $V_\ell \cup D(S)$ and define the node weights $w'(v') = w(v)/d$ for any $v \in V \setminus (V_\ell \cup D(S))$. We claim that G' is a Frobenius instance with parameter $k' := (k - w(D(S)))/d$ (observe that k' is indeed integer, as $w(D(S)) \equiv bw_\ell \equiv k \pmod{d}$, and that $k' \geq 0$ by (2)). If this is indeed the case, then by inductive hypothesis G' has a closed set S' with $w'(S') = k'$, since the gcd of the weights w' is 1. Observe that by construction, $D(S) \cup S'$ is a closed set in G of weight $w(D(S)) + d \cdot w'(S') = w(D(S)) + (k - w(D(S))) = k$, as desired.

It remains to prove that G' is indeed a Frobenius instance with parameter k'. First, observe G' has $\ell - 1$ layers $V'_i := V_i \setminus D(S), i \in [\ell - 1]$ and that w' is well defined, as $d \mid w_i$ for all $i \in [\ell - 1]$. Conditions (P1) and (P2) of being Frobenius are fulfilled as G' is a subgraph of G. To see (P3), note that

$$|V'_i| \geq |V_i| - |D(S)| \geq |V_i| - w(D(S)) \geq k - w(D(S)) \geq k'.$$

To see (P4), we observe that by (2) and (1), we have

$$k' = \frac{k - w(D(S))}{d} \geq \frac{k - k/2}{d} = \frac{k}{2d} \geq \frac{k}{d^2}.$$

Thus, for any $v' \in V'$, we obtain

$$w'(D(v')) \leq \frac{w(D(v))}{d} \leq \frac{\sqrt{k/2}}{d} = \sqrt{\frac{k}{2d^2}} \leq \sqrt{k'/2},$$

where we used condition (P4) of G in the second inequality. Thus, G' is indeed a Frobenius instance with parameter k', concluding the claim and thus the proof of our lemma. ◀

The above criterion is the main technical tool in the algorithmic result of the session. What remains is to show that the instance can be preprocessed in a way that it becomes a Frobenius instance.

▶ **Theorem 3.4.** *We can solve* Weighted DAG Implications *in time* $f(k)n^{4\sqrt{k}}$.

Proof. Consider the following recursive algorithm, which proceeds in 4 steps:

Step 1: For every $v \in V$ with $w(D(v)) \geq \sqrt{k/2}$, we return YES if a recursive call determines that $G[V \setminus D(v)]$ has a closed set of weight $k - w(D(v))$; otherwise, we delete v and all its ancestors from G. From now on, G satisfies $w(D(v)) \leq \sqrt{k/2}$ for all $v \in V$.

Step 2: We construct layers $L_1, \ldots, L_{\sqrt{k/2}}$ by the following iterative process: for every $i = 1, \ldots, \sqrt{k/2}$, we let L_i consists of all vertices in $V \setminus (L_1 \cup \cdots \cup L_{i-1})$ whose outgoing edges end in $L_1 \cup \cdots \cup L_{i-1}$. Note that $L_1, \ldots, L_{\sqrt{k/2}}$ partitions V; in particular, every vertex is included in some L_i, since if there was a vertex $v \in V \setminus (L_1 \cup \cdots \cup L_{\sqrt{k/2}})$, then by construction there exists a path from v containing strictly more than $\sqrt{k/2}$ vertices, leading to the contradiction $w(D(v)) \geq |D(v)| > \sqrt{k/2}$.

We observe that each layer L_i can be partitioned into sublayers $L_{i,j}, j \in \{1, \ldots, \sqrt{k/2}\}$ such that each $v \in L_{i,j}$ has weight $w(v) = j$: there can be no vertex of larger weight, as otherwise $w(D(v)) \geq w(v) > \sqrt{k/2}$ yields a contradiction. We consider layers $L_{i,j}$ in increasing lexicographic order of (i, j): If $|L_{i,j}| < k$, then for every $v \in L_{i,j}$, we return YES if a recursive call determines that $G[V \setminus D(v)]$ contains a closed set of size $k - w(D(v))$, and otherwise we delete v and all its ancestors from G. Observe that by the lexicographic ordering, we never delete vertices from already processed layers, so that at the end of the process, each $L_{i,j}$ is either empty or contains at least k vertices.

Step 3: We let $V_1, \ldots, V_\ell$ be an enumeration of all non-empty sublayers $L_{i,j}$ by the lexicographic order on (i, j) so that any vertex $v \in V_i$ has only edges to vertices in $V_1 \cup \cdots \cup V_{i-1}$. Observe that by construction, this yields a Frobenius instance. Let $w_1, \ldots, w_\ell$ be the weights of the Frobenius instance. We return YES if $\gcd(w_1, \ldots, w_\ell) \mid k$ and NO otherwise.

Using Lemma 3.3, the correctness of the algorithm is easy to see.

▷ Claim 3.5. The above algorithm is correct.

Proof. If the algorithm returns YES, indeed there is a closed set of size k: If we return YES in Steps 1 or 2, we have found a vertex v and a closed set S' in $G[V \setminus D(v)]$ of size $k - w(D(v))$, which yields a closed set $S' \cup D(v)$ in G of size k, as desired. Otherwise, we have arrived at a Frobenius instance and returned YES since $\gcd(w_1, \ldots, w_\ell) \mid k$, which implies that G has a closed set of size k by Lemma 3.3.

Conversely, fix a closed set S of size k, and we show that the algorithm returns YES: If S contains a vertex v investigated in Steps 1 or 2, then the recursive call to $G[V \setminus D(v)]$ (for the first such vertex v) will find a solution of size $|S| - w(D(v))$ (note that $D(v) \subseteq S$ if $v \in S$). Otherwise, we have arrived at a Frobenius instance which must satisfy $\gcd(w_1, \ldots, w_\ell)$ by Lemma 3.3, and we return YES. ◁

Finally, we need to bound the running time of the recursive algorithm. The analysis relies on the observation that the algorithm makes at most n recursive calls with a parameter decrease of at least $\sqrt{k/2}$, and at most $O(k^2)$ recursive calls with a parameter decrease of one.

▷ Claim 3.6. The above algorithm can be implemented in time $f(k)n^{4\sqrt{k}}$.

Proof. Let U be the set of vertices of small layers ($|L_{i,j}| < k$) considered in Step 2. We observe that the above algorithm can be implemented recursively with the following recurrence on its running time $T(n, k)$ on instances with n vertices and parameter k.

$$T(n,k) \leq \sum_{v \in V, w(D(v)) \geq \sqrt{k/2}} T(n, k - w(D(v))) + \sum_{u \in U} T(n, k - w(D(u))) + \mathcal{O}(n^2)$$

We claim by induction on k that this yields a bound of $T(n,k) \leq f(k)n^{4\sqrt{k}}$ for some $f(k) = k^{\mathcal{O}(k)}$. It is not difficult to see that for $k \leq 2$, we can solve the problem in time $\mathcal{O}(n^2) = \mathcal{O}(n^{4\sqrt{k}})$, yielding the base case. For $k \geq 3$, we thus obtain the following bound, using that in Step 2, we process less than k vertices for each "small" sublayer $L_{i,j}, 1 \leq i,j \leq \sqrt{k/2}$, i.e., $|U| \leq k(k/2) = k^2/2$,

$$\begin{aligned} T(n,k) &\leq \mathcal{O}(n \cdot f(k-\sqrt{k/2})n^{4\sqrt{k-\sqrt{k/2}}} + k^2 f(k-1)n^{4\sqrt{k-1}} + n^2) \\ &\leq (f(k)/2)(n^{4\sqrt{k-\sqrt{k/2}}+1} + n^{4\sqrt{k}}) \leq f(k)n^{4\sqrt{k}}, \end{aligned}$$

where the second bound follows from choosing $f(k) = k^{\mathcal{O}(k)}$ large enough to ensure $k^2 f(k-1) \leq f(k)/2$ and the last bound follows from the observation that $4\sqrt{k-\sqrt{k/2}}+1 \leq 4\sqrt{k}$ if and only if

$$\begin{aligned} & \left(4\sqrt{k-\sqrt{k/2}}+1\right)^2 \leq 16k \\ \iff \quad & 16(k-\sqrt{k/2}) + 8\sqrt{k-\sqrt{k/2}} + 1 \leq 16k \\ \iff \quad & 8\sqrt{k-\sqrt{k/2}} + 1 \leq 16\sqrt{k/2}, \end{aligned}$$

where the last inequality holds since $8\sqrt{k}+1 \leq 16\sqrt{k/2}$ as $k \geq 3$. ◁

Claims 3.5 and 3.6 show the correctness of our algorithm for Weighted DAG Implications. By the reduction described at the beginning of the section, a similar algorithm follows for Implications. ◀

4 Algorithms for NAND_2-avoiding $\mathcal{F}$: Reduction to Implications

In this section, we show that for any NAND_2-avoiding constraint family $\mathcal{F}$, we can reduce $\mathsf{SAT}(\mathcal{F})$ to Implications. Specifically, we obtain the following theorem.

▶ **Theorem 4.1.** *Let $\mathcal{F}$ be a NAND_2-avoiding constraint family and let $T_{\mathrm{IMPL}}(n,k)$ denote the optimal running time to solve Implications. There is a constant $c_{\mathcal{F}}$ and computable $f(k)$ such that we can solve $\mathsf{SAT}(\mathcal{F})$ in time $f(k)(T_{\mathrm{IMPL}}(n,k) + n^{c_{\mathcal{F}}})\log n$.*

Together with Theorem 3.4, this gives an $f(k)n^{4\sqrt{k}+c_{\mathcal{F}}}$-time algorithm for any NAND_2-avoiding constraint family $\mathcal{F}$.

To prove the above theorem, we prepare some notation and helpful facts. Let ϕ be an arbitrary formula. For any assignment a, we call a' a *minimal satisfying extension* of a, if a' satisfies ϕ, $a \leq a'$, and no other satisfying assignment $a'' \notin \{a, a'\}$ fulfills $a \leq a'' \leq a'$. The following lemma shows that there are only $f(k)$ many minimal extensions of weight at most k, and these minimal extensions can be computed in time $f(k)n^c$ for some constant c independent of k. Intuitively, this follows by using the bounded search tree technique over violated constraints, where the depth of the search tree is bounded by k and each branching step has at most r possibilities.

▶ **Lemma 4.2** ([12, Lemma 2.3])**.** *Let $\mathcal{F}$ be a finite constraint family of bounded arity r. There is a constant $c'_{\mathcal{F}}$ such that given any instance ϕ of $\mathsf{SAT}(\mathcal{F})$ and assignment a, there are at most $\mathcal{O}(r^k)$ minimal extensions of a of weight k, and we can compute these extensions in time $\mathcal{O}(r^k n^{c'_{\mathcal{F}}})$.*

As an immediate useful consequence, we obtain that for our algorithmic results, we may assume without loss of generality that $\mathcal{F}$ is 0-valid, i.e., each $f \in \mathcal{F}$ is satisfied by the all-zeroes assignment.

▶ **Corollary 4.3** (see also [28, Lemma 4.1])**.** *We can reduce any instance of* $\mathsf{SAT}(\mathcal{F})$ *with parameter* k *to* $\mathcal{O}(r^k)$ *many instances of* $\mathsf{SAT}(\mathcal{F}')$ *with a parameter bounded by* k*, where* $\mathcal{F}'$ *is the set of all* 0*-valid* f' *that are represented by* $\mathcal{F}$*.*

By definition, if $\mathcal{F}$ does not represent NAND_2, then also $\mathcal{F}'$ does not represent NAND_2, and it remains to give an $f(k)(T_{\mathrm{IMPL}}(n,k) + n^{c_{\mathcal{F}}}) \log n$-time algorithm for 0-*valid* NAND_2-avoiding $\mathcal{F}$.

In the remainder of this section, we will use the graph formulation of the Implications problem: We are given a directed graph $G = (V, E)$ and the task is to find a closed set S (recall that S is closed, if for all $(u, v) \in E$ we have that $u \in S$ implies $v \in S$) of size k. Recall that for any vertex set $S \subseteq V$, $D(S)$ denotes the set of descendants of any vertex $s \in S$ (including the vertices in S).

Our aim is the following: Given a formula ϕ of $\mathsf{SAT}(\mathcal{F})$, we give a randomized construction of an Implications instance G such that

- **(i)** any closed set S in G corresponds to a satisfying assignment of ϕ, and
- **(ii)** with large enough probability, G contains a closed set of size k if ϕ has a weight-k solution.

To this end, we let $V = \{x_1, \ldots, x_n\}$ and recall that, for any set $S \subseteq V$, we let $a_S : [n] \to \{0, 1\}$ denote a corresponding assignment with $a_S(i) = 1$ iff $x_i \in S$. From now on, we often synonymously speak of closed sets $S \subseteq V$ in G and the corresponding assignment a_S for ϕ.

The rough outline is as follows: we start with the graph $G = (V, \emptyset)$, and try to repeatedly "fix" some closed set S that violates ϕ, by determining a (random) implication consistent with a minimal satisfying extension of S. The main insight is that if $\mathcal{F}$ avoids NAND_2, then it suffices to make sure that all sets $D(v)$ for $v \in V$ are satisfying and this will automatically ensure that every closed set is satisfying.

Let us formally describe the algorithm:

1. Given ϕ, initialize $G = (V, E)$ with $V = \{x_1, \ldots, x_n\}$ and $E = \emptyset$.
2. While there exists some $v \in V$ such that $a_{D(v)}$ violates ϕ, do the following:
   - **a.** Compute the set A_v of minimal satisfying extensions of $a_{D(v)}$ of weight at most k.
   - **b.** Let X consist of all $x_i \in V \setminus D(v)$ such that there is some $a \in A_v$ with $a(i) = 1$.
   - **c.** If $X = \emptyset$, delete all ancestors of v (including v) from G. Otherwise, pick x uniformly at random from X and add the edge (v, x) to E.

The important properties of the algorithm are captured in the following lemma.

▶ **Lemma 4.4.** *Let* $\mathcal{F}$ *be a finite* 0*-valid constraint family. There is a constant* $c_{\mathcal{F}}$ *and a function* $g(k)$ *such that the following properties hold.*

- **(P1)** *During the process, each vertex* v *is considered at most* k *times in the while loop. Thus, the algorithm can be implemented to run in time* $\mathcal{O}(g(k)n^{c_{\mathcal{F}}})$*.*
- **(P2)** *If* ϕ *has a satisfying assignment of weight* k*, then with probability at least* $g(k)^{-1}$*, there is a closed set* S *in* G *of size* k*.*
- **(P3)** *If* $\mathcal{F}$ *avoids* NAND_2*, any closed set* $S \subseteq V$ *in the constructed graph yields a satisfying assignment* a_S *for* ϕ*.*

Proof. For (P1), note that whenever $v \in V$ is considered in the while loop, it is either deleted, or an edge (v, x) with $x \notin D(v)$ is added to the graph. Thus, when v is considered for the k-th time, we have $|D(v)| \geq k$, and thus there can be no satisfying extension of $a_{D(v)}$

of weight at most k. Consequently, we must have $A_v = \emptyset$, and thus $X = \emptyset$, which forces v to be deleted. Thus, we have at most kn iterations of the while loop, where each iteration can be implemented in time $\mathcal{O}(r^k n^{c'_{\mathcal{F}}})$ by Lemma 4.2.

For (P2), assume that there is a set S of size k such that a_S satisfies ϕ. We show that with large enough probability, we will maintain as invariant that $D(v) \subseteq S$ for every $v \in S$, and thus S will be a closed set in G. To this end, we first observe that for $D(v) \subseteq S$ to hold for all $v \in S$, it suffices that the following property holds:

In each iteration that considers a vertex $v \in S$, the selected vertex x is in S. (3)

Indeed, if this is the case, then no $v \in S$ is ever deleted. Furthermore, we have that $D(v) \subseteq S$ for all $v \in S$, and thus S is a closed set in G. It remains to give a lower bound on the probability that (3) holds throughout the process.

To this end, consider the event that some $v \in V$ is considered in the while loop, conditioned that (3) has not been violated in a previous iteration. Under this event, $D(v) \subseteq S$, and thus there is a minimal satisfying extension $D(v) \subsetneq S' \subseteq S$ such that $a_{S'}$ satisfies ϕ and thus $a_{S'} \in A_v$. Let $s \in S' \setminus D(v)$ be arbitrary, then $s \in X$ by construction (note that s has not been deleted). By Lemma 4.2, we have that $|A_v| \leq \mathcal{O}(r^k)$. Since each $a \in A_v$ has weight at most k, this yields $|X| \leq k|A_v| \leq \mathcal{O}(kr^k)$. Thus, the probability that the random choice is $x = s$ is at least $1/|X| \geq \Omega(1/(kr^k))$. Finally, we observe that by (P1), for each $v \in S$, there are at most k iterations considering v, where each iteration has a probability of at least $\Omega(1/(kr^k))$ of not violating (3). Thus, we obtain that (3) holds with probability at least $\Omega(1/(kr^k)^{k|S|}) = \Omega(1/(kr^k)^{k^2})$, and the claim follows by setting $g(k) \coloneqq (kr^k)^{-k^2}$.

Finally, for (P3), note that at the end of the process, the property holds that

For all (remaining) $v \in V, a_{D(v)}$ satisfies ϕ. (4)

We will leverage this fact to show that a_S satisfies ϕ for *all* closed sets $S = D(v_1) \cup ... \cup D(v_\ell)$ for $v_1, \ldots, v_\ell \in V$. We first transform the graph G to a DAG by contracting all strongly connected components $C = \{v_1, \ldots, v_c\}$ to a single vertex v_C representing the set C. Note that the closed sets in the DAG remain in a one-to-one correspondence to the closed sets of the original graph (and the corresponding assignments to ϕ), thus this transformation is without loss of generality. Thus, we may assume that G has a topological ordering $v_1, \ldots, v_{n'}$ of its vertices ($n' \leq n$). We will prove by induction on $i = n', ..., 1$ that for all closed sets $S \subseteq \{v_i, ..., v_{n'}\}$, a_S satisfies ϕ.

For the base case $i = n'$, we only need to verify that (i) the all-0 assignment satisfies ϕ, which holds by 0-validity of $\mathcal{F}$, and (ii) that $a_{v_{n'}}$ satisfies ϕ, which holds by (4) (as $D(v_{n'}) = \{v_{n'}\}$). Thus, for $i < n'$, let us assume that the claim holds for $i+1$. Consider any closed set $U \subseteq \{v_i, \ldots, v_{n'}\}$. If U does not contain v_i, the claim follows by inductive assumption, thus let us assume that $v_i \in U$ and thus $U \supseteq D(v_i)$, as U is closed. If $U = D(v_i)$, a_U satisfies ϕ by (4). Thus, it remains to consider $U \supsetneq D(v_i)$, for which we assume for contradiction that a_U violates ϕ. Let $W \coloneqq U \setminus D(v_i)$, and note that $D(W) \subseteq U$ is a closed set in $\{v_{i+1}, \ldots, v_{n'}\}$. Thus, by inductive assumption, $a_{D(W)}$ satisfies ϕ. Furthermore, observe that $Z \coloneqq D(v_i) \cap D(W)$ is a closed set in $\{v_{i+1}, \ldots, v_{n'}\}$ (since the intersection of any two closed sets yields a closed set). Thus, a_Z satisfies ϕ by inductive assumption. It remains to show that the fact that $a_{D(v_i)}$, $a_{D(W)}$ and $a_Z = a_{D(v_i) \cap D(W)}$ all satisfy ϕ, while $a_U = a_{D(v_i) \cup D(W)}$ violates ϕ, gives a contradiction to $\mathcal{F}$ avoiding NAND_2.

To this end, let C be a constraint violated by a_U and note that $C = f(x_{i_1}, \ldots, x_{i_r})$ for some $f \in \mathcal{F}$ and $i_1, \ldots, i_r \in [n]$. Note that we can view f as $f : \{0,1\}^{V_c} \to \{0,1\}$ for some appropriate variable set V_C. We show how to obtain NAND_2 as a restriction of f by partitioning V_C into $X' \coloneqq (D(v_i) \setminus Z) \cap V_C, Y' \coloneqq (D(W) \setminus Z) \cap V_C, Z_1 \coloneqq Z \cap V_C, Z_0 \coloneqq V_C \setminus (X' \cup Y' \cup Z_1)$ and observing that

$$
\begin{array}{llllcll}
 & \overbrace{}^{X'} & \overbrace{}^{Y'} & \overbrace{}^{Z_0} & \overbrace{}^{Z_1} & & \\
f(0\ldots0, & 0\ldots0, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_Z \text{ satisfies } C\text{]} \\
f(1\ldots1, & 0\ldots0, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_{D(v_i)} \text{ satisfies } C\text{]} \\
f(0\ldots0, & 1\ldots1, & 0\ldots0, & 1\ldots1) & = & 1, & \text{[since } a_{D(W)} \text{ satisfies } C\text{]} \\
\hline
f(1\ldots1, & 1\ldots1, & 0\ldots0, & 1\ldots1) & = & 0. & \text{[since } a_{D(W)\cup D(v_i)} \text{ violates } C\text{]}
\end{array}
$$

◀

It remains to give the proof of Theorem 4.1.

Proof of Theorem 4.1. By Corollary 4.3, we may assume without loss of generality that $\mathcal{F}$ is 0-valid. We repeat the following process $g(k)$ many times: We use the above algorithm to generate an Implications instance G, and return YES if G contains a closed set of size k, which we determine using an optimal Implications algorithm. If none of the $g(k)$ iterations were successful, we return NO. Note that this approach can be implemented in time $g(k)\mathcal{O}(g(k)n^{c_{\mathcal{F}}} + T_{\text{IMPL}}(n,k))$ by (P1), and correctly decides the instance with probability at least $1-(1-1/g(k))^{g(k)} \geq 1-1/e$ by (P2) and (P3).

The algorithm described above can be derandomized using the standard technique of Color Coding [4]. In each iteration when vertex v is considered, a random vertex x is selected from a set X of at most $K = \mathcal{O}(kr^k)$ vertices. As each vertex is considered at most k times, we can represent the random choices by a function $r : V \to [K]^k$, with the meaning that $r(v)$ is the vector of choices made when considering vertex v. As discussed in the proof of Lemma 4.4, when considering vertices $v \in S$, these random choices need to be consistent with S to ensure that S is a closed set in the resulting graph. That is, for each $v \in S$ there is a vector $c(v) \in [K]^k$ such that if the random choice satisfies $r(v) = c(v)$ for every $v \in S$, then S is a closed set.

We say that a family $\mathcal{H}$ of functions $h : [n] \to [k]$ is a (n,k)-*perfect family of hash functions* if for every $S \subseteq V$ of size k, there is an $h \in \mathcal{H}$ that is injective on S, i.e., assigns different values to different elements of S. It is known that a (n,k)-perfect family of size $2^{O(k)}\log n$ can be computed in time $2^{O(k)}n\log n$ [4]. The derandomized algorithm would first compute such a family $\mathcal{H}$ over V and would iteratively go through every $h \in \mathcal{H}$ and function $q : [k] \to [K]^k$. For a given choice of h and q, we define the function $r(v) = q(h(v))$ and run the randomized algorithm using this function r instead of the random choices. It is easy to see that the definition of (n,k)-perfect hash functions implies that there is at least one choice of h and q where $r(v)$ is exactly the prescribed value $c(v)$ for every $v \in S$ and therefore the randomized algorithm correctly finds the solution S. As we are considering at most $|\mathcal{H}| = 2^{O(k)}\log n$ functions h and K^{k^2} different functions q, there is a function $f(k)$ such that the total running time is at most $f(k)\log n$ times a single run of the randomized algorithm. ◀

5 Algorithms for NAND-representing $\mathcal{F}$: Reduction to Clique

In this section, we develop algorithm for constraint families that might represent NAND_2, but avoid NAND_d for some $d \geq 3$. To this end, we give a reduction to $(d-1)$-uniform Hyper-Clique for NAND_d-avoiding families, giving in particular a $f(k)n^{(\omega/3)k+c_{\mathcal{F}}}$-time algorithm for NAND_3-avoiding families.

We first start with a natural reduction of $\mathsf{SAT}(\mathcal{F})$ for any $\mathcal{F}$ with arity bounded by r to r-uniform HyperClique, based on color-coding. To this end, let $T_{d\text{-HC}}(n,k)$ denote the optimal running time of finding a k-clique in a d-uniform hypergraph.

▶ **Proposition 5.1.** *Let $\mathcal{F}$ be a constraint family of arity at most r. Then $\mathsf{SAT}(\mathcal{F})$ can be solved in time $f(k)(n^{2r} + T_{r\text{-}HC}(n,k))\log n$.*

Proof. Let ϕ be an arbitrary $\mathsf{SAT}(\mathcal{F})$ formula. Observe that any constraint C of ϕ depends only on a set $\mathrm{vars}(C) \subseteq \{x_1, \ldots, x_n\}$ of at most r variables. For an assignment a, we let $C(a) \in \{0,1\}$ denote whether C is satisfied by a.

We first show how to determine, given a partition of $x_1, \ldots, x_n$ into k sets $X_1, \ldots, X_k$, whether there is a solution that sets precisely one variable in each X_i to true. To this end, we construct a hypergraph G with vertex set $X_1 \cup X_2 \cup \cdots \cup X_k$ and the following set of hyperedges: we include each possible hyperedge $e = \{x_{j_1}, \ldots, x_{j_r}\}$ with $x_{j_1} \in X_{j_1}, \ldots, x_{j_r} \in X_{j_r}$ and distinct $j_1, \ldots, j_r \in [k]$ *unless* there exists a clause C with $\mathrm{vars}(C) \subseteq X_{j_1} \cup \cdots \cup X_{j_r}$ which is violated by the assignment that sets precisely the variables $e = \{x_{j_1}, \ldots, x_{j_r}\}$ to 1, i.e., $C(a_e) = 0$.

We claim that $H := \{x_{i_1}, \ldots, x_{i_k}\}$ with $x_{i_1} \in X_1, \ldots, x_{i_k} \in X_k$ yields a k-clique in G if and only if the assignment a_H satisfies ϕ. Indeed, assume that there is a clause C violated by a_H. Note that as C has arity at most r, we have $\mathrm{vars}(C) \subseteq X_{j_1} \cup \cdots \cup X_{j_r}$ for some distinct $i_1, \ldots, i_r \in [k]$ (if C involves variables of less than r sets, we may use arbitrary additional sets). Thus, $e := \{x_{j_1}, \ldots, x_{j_r}\}$ cannot be an edge in G, since a_H violates C, a_e and a_H agree on $\mathrm{vars}(C)$, and thus also a_e violates C. Conversely, if there is some $e := \{x_{i_1}, \ldots, x_{i_r}\}$ with distinct $i_1, \ldots, i_r \in [k]$ such that e is not an edge in G, then there exists some clause C with $\mathrm{vars}(C) \subseteq X_{i_1} \cup \cdots \cup X_{i_r}$ which is violated by a_e. Since a_H and a_e agree on $\mathrm{vars}(C)$, we conclude that also a_H violates C and thus ϕ.

To create the desired k-partition of variables, we use a (deterministic) color-coding scheme: Let $\mathcal{H}$ be a (n,k)-*perfect family of hash functions* $h : [n] \to [k]$ – recall that this means that for any $S = \{s_1, \ldots, s_k\} \subseteq [n]$, there exists some $h \in \mathcal{H}$ such that $\{h(s_1), \ldots, h(s_k)\} = \{1, \ldots, k\}$. Known efficient constructions [32, 4] produce such assignments with $\ell = 2^{\mathcal{O}(k)} \log(n)$ in time $2^{\mathcal{O}(k)} n \log n$. Given this family, we create for each $h \in \mathcal{H}$ the k-partition $X_1^{(h)}, \ldots, X_k^{(h)}$ with $X_j^{(h)} = \{x_s \mid h(s) = j\}$ and solve the corresponding r-uniform HyperClique instance in time $T_{r\text{-HC}}(n,k)$. If any of these instances returns a solution, then indeed ϕ has a satisfiable assignment of weight k. Conversely, if a_S is a weight-k satisfying assignment for ϕ, then by construction, there exists a hash function $h \in \mathcal{H}$ such that $|S \cap X_j^{(h)}| = 1$ for $j = 1, \ldots, k$, and thus the corresponding r-uniform HyperClique instance indeed contains a solution. For each of the $2^{\mathcal{O}(k)} \log(n)$ hash functions, the time to construct and solve the d-uniform HyperClique instance is bounded by $\mathcal{O}(n^{2r} + T_{r\text{-HC}}(n,k))$, concluding the claim. ◀

The main result in this section is the following reduction from NAND_{d+1}-avoiding constraint families to d-uniform HyperClique.

▶ **Theorem 5.2.** *Let $d \geq 2$ and $\mathcal{F}$ be an NAND_{d+1}-avoiding constraint family. If there are constants $\gamma \geq d/(d+1)$ and c, and a computable $g(k)$ such that d-uniform HyperClique can be solved in time $g(k)n^{\gamma k + c}$, then there is a constant c' and computable $g'(k)$ such that $\mathsf{SAT}(\mathcal{F})$ can be solved in time $g'(k)n^{\gamma k + c'}$.*

In particular, since we can find k-cliques in graphs in time $\mathcal{O}(n^{\frac{\omega}{3}k+1})$, we obtain an $g(k)n^{\frac{\omega}{3}k+c'}$-time algorithm for solving $\mathsf{SAT}(\mathcal{F})$ for all NAND_3-avoiding constraint families. Similarly, if for $d \geq 3$ the d-uniform HyperClique conjecture is refuted by exhibiting a $g(k)n^{(1-\epsilon)k+c}$-time algorithm for some constants $0 < \epsilon < 1/(d+1)$ and c, we would obtain a $g'(k)n^{(1-\epsilon)k+c'}$-time algorithm for $\mathsf{SAT}(\mathcal{F})$ for NAND_{d+1}-avoiding families $\mathcal{F}$.

In the remainder of the section, we give the proof of Theorem 5.2. The main task of the algorithm is to detect *robust* assignments, defined as follows.

▶ **Definition 5.3.** *Let $a : [n] \to \{0,1\}$ be a weight-k assignment that satisfies ϕ. We say that a is* d-robust *if there is no assignment $a' \leq a$ of weight at most d that violates ϕ.*

The first step of the algorithm is the easier task of detecting satisfying assignments that are *not* d-robust (if there exists any): Intuitively, an assignment that is not d-robust offers an advantage to find it: Assume we correctly guess an assignment $a' \leq a$ of weight $w \leq d$ such that some clause C is violated by a', then to extend a' to the satisfying assignment a, we know that at least one additional variable in C must be set to true. By bruteforcing over the at most $r - w \leq r$ many possibilities, we gain an advantage. Specifically, by enumerating $O(n^w r) = \mathcal{O}(n^w)$ many possibilities, we can fix $w + 1$ true variables in our solution.

Let $T(n, k)$ denote the time our algorithms takes to solve an arbitrary $\mathsf{SAT}(\mathcal{F})$ instance for a NAND_{d+1}-avoiding family $\mathcal{F}$. In a preprocessing step, we first enumerate all assignments a' of weight at most d. If there exists a clause $C_{a'}$ that is violated by a', then we enumerate all variables $x \in \mathrm{vars}(C_{a'}) \setminus \mathrm{ones}(a')$ (recall that $\mathrm{vars}(C)$ is the set of variables involved in C and $\mathrm{ones}(a)$ denotes the set of variables set to 1 under a). We recursively determine satisfiability of the formula $\phi_{a',x}$ obtained by restricting all variables in $\mathrm{ones}(a') \cup \{x\}$ to true. Disregarding the time to determine existence of violated clauses $C_{a'}$, this step takes time

$$\sum_{w=0}^{d} \sum_{\substack{\text{weight-}w \\ \text{assignment } a'}} \sum_{x \in \mathrm{vars}(C_{a'}) \setminus \mathrm{ones}(a')} T(n, k-(w+1)) \leq \sum_{w=0}^{d} O(n^w) T(n, k-(w+1)). \quad (5)$$

To determine a violated clause $C_{a'}$ (if it exists) for all weight-($\leq d$) assignments a', we simply traverse each clause C, determine the at most $\sum_{w=0}^{d} \binom{r}{w} = O(1)$ weight-($\leq d$) assignments violating C and store C as violated for each of these assignments (if no other clause is already stored). This step takes time $O(m) = O(n^r)$ in the beginning.

After this preprocessing, it remains to consider d-robust assignments. To determine whether a d-robust assignment satisfies ϕ, we define a formula ϕ_d that is satisfied only by satisfying assignments of ϕ, and particularly by all d-robust satisfying assignments of ϕ. To this end, let F_d contain all assignments of weight at most d that violate some clause C of ϕ, and define

$$\phi_d := \bigwedge_{a \in F_d} \mathrm{NAND}(\mathrm{ones}(a)).$$

▶ **Lemma 5.4.** *The constructed formula ϕ_d has the following properties:*

(P1) *If $\mathcal{F}$ is NAND_{d+1}-avoiding, then any satisfying assignment a of ϕ_d is a satisfying assignment of ϕ.*

(P2) *If a is a d-robust satisfying assignment of ϕ, then a satisfies ϕ_d.*

Proof. To prove (P1), we will make use of the following property.

▷ Proposition 5.5. Let $\mathcal{F}$ be a NAND_{d+1}-avoiding family. Then if an assignment a violates some clause C (chosen from $\mathcal{F}$), there is an assignment $a' \leq a$ of weight at most d that violates C.

Proof. We prove the claim via induction on the weight w of the clause C under a. If $w \leq d$, the claim trivially holds. To prove the inductive step, we may assume for contradiction that there is an assignment a of weight $w \geq d+1$ violating some clause $C = f(\bar{x})$, but no assignment $a' \leq a$ of weight at most $w - 1$ violates C. We will show that NAND_{d+1} can be obtained as a restriction of f. To this end, choose some set $S \subseteq \mathrm{ones}(a) \cap \mathrm{vars}(C)$ of size $d+1$ (which is possible as $w \geq d+1$), and partition $\mathrm{vars}(C)$ into S, $Z_1 := (\mathrm{ones}(a) \cap \mathrm{vars}(C)) \setminus S$ and $Z_0 := \mathrm{vars}(C) \setminus (S \cup Z_1)$. Observe that we have

$$\begin{array}{lllll} f(\overbrace{y_1 \dots y_{d+1}}^{S}, & \overbrace{0 \dots 0}^{Z_0}, & \overbrace{1 \dots 1}^{Z_1}) & = 1, & \text{if } (y_1, \dots, y_{d+1}) \neq (1, \dots, 1) \\ \hline f(\ 1 \dots 1\ \ , & 0 \dots 0, & 1 \dots 1) & = 0. & [\text{since } a \text{ violates } C] \end{array}$$

where the first line follows since no assignment $a' \leq a$ of weight at most $w-1$ violates C, yielding a contradiction. ◁

To prove (P1), assume that an assignment a violates some clause C of ϕ. Since $\mathcal{F}$ is NAND_{d+1}-avoiding, by Proposition 5.5 there exists an assignment $a' \leq a$ of weight at most d such that a' violates C. Thus, ϕ_d contains a clause $\text{NAND}(\text{ones}(a'))$, which is violated by a, as $a' \leq a$.

To prove (P2), assume for contradiction that a d-robust assignment a satisfies ϕ but not ϕ_d. Then there is some $a' \in F_d$ such that $\text{NAND}(\text{ones}(a'))$ is violated by a, i.e., $a' \leq a$. As $a' \in F_d$, there must be a clause C of ϕ that is violated by $a' \leq a$, which proves that a is not d-robust and thus yields a contradiction. ◀

Note that ϕ_d is a $\mathsf{SAT}(\mathcal{F}')$ formula with constraint family $\mathcal{F}' = \{\text{NAND}_j \mid 2 \leq j \leq d\}$ of arity d. Thus, by Proposition 5.1, we can determine satisfiability of ϕ_d in time $f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n$. We obtain the following recurrence by combining (5), the $O(m)$-time preprocessing to determine violated classes $C_{a'}$, and $f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n$ to solve ϕ_d:

$$T(n,k) = \mathcal{O}(m) + f(k)(n^{2d} + T_{d\text{-HC}}(n,k)) \log n + \sum_{w=0}^{d} O(n^w) T(n, k-(w+1)) \tag{6}$$

Assume that there are $\gamma \geq d/(d+1)$ and c such that $T_{d\text{-HC}}(n,k) \leq g(k) n^{\gamma k + c}$. We will show that $T(n,k) = g'(k)\mathcal{O}(n^{\gamma k + c'})$ for any $c' > \max\{c, 2r\}$ and $g'(k) = f(k)g(k)$.

We prove the claim via induction on k. The base case is $k < c'$, in which case we can solve $\mathsf{SAT}(\mathcal{F})$ in time $f(k)(n^{2r} + T_{r\text{-HC}}(n,k)) \log n = f(k)\mathcal{O}((n^{2r} + n^k) \log n) \leq \mathcal{O}(n^{c'})$, satisfying the claim. Thus, let us assume that $k \geq c'$ and that the claim holds for all $k' \leq k-1$. Using (6), we obtain

$$\begin{aligned} T(n,k) &\leq \mathcal{O}(m) + f(k)(n^{2d} + g(k) n^{\gamma k + c}) \log n + g'(k) \left(\sum_{w=0}^{d} O(n^w) n^{\gamma(k-w+1)+c'} \right) \\ &\leq g'(k) \log n \cdot \mathcal{O}\left(n^{2r} + n^{\gamma k + c} + \sum_{w=0}^{d} n^{w + \gamma(k-(w+1))+c'} \right) \\ &\leq g'(k) \log n \cdot \mathcal{O}\left(n^{2r} + n^{\gamma k + c} + n^{\gamma k + c'} \right) = g'(k)\mathcal{O}(n^{\gamma k + c'}), \end{aligned}$$

where in the second line, we used that $g'(k) = f(k)g(k)$, and in the last line we used that $\gamma(w+1) \geq w$ as $\gamma \geq d/(d+1) \geq w/(w+1)$ for $w \leq d$, as well as our choice of c' which satisfies $c' > c$ and $\gamma k + c' \geq c' > 2r$.

6 Hardness Results

In this section, we give our hardness results. To this end, we first consider Implications = $\mathsf{SAT}(\text{IMPL})$ and give a $f(k) n^{(\omega/6 - o(1))\sqrt[3]{k}}$-lower bound under the k-clique conjecture. Afterwards, we handle the case of NAND_d- or IMPL-representing families, by reducing from d-uniform (Hyper)Clique or Implications, respectively.

6.1 Hardness for Implications

▶ **Theorem 6.1.** *If* Implications *can be solved in time* $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$ *for some* $\epsilon > 0, c$ *and* $f(k)$*, then the* k*-Clique conjecture fails.*

Proof. Let $G = (V, E)$ be an undirected graph. We construct an Weighted DAG Implications instance $G' = (V', E', w')$ with parameter $k' = k \cdot K + \binom{k}{2}$ with $K \coloneqq \binom{k}{2} + 1$ as follows. The vertex set V' is the disjoint union of *vertex nodes* $V'_V \coloneqq \{v_u \mid u \in V\}$ and *edge nodes* $V'_E \coloneqq \{v_e \mid e \in E\}$. For every $e = \{u, w\} \in E$, we introduce the edges $(v_e, v_u), (v_e, v_w)$ to E'. Furthermore, we set the weights of vertex nodes to K, and the weights of edge nodes to 1.

▷ Claim 6.2. There is a closed set X of weight k' in G' if and only if there is a k-clique in G.

Proof. Let $C = \{v_1, \ldots, v_k\}$ be a k-clique in G. Observe that $X = \{v_u \mid u \in C\} \cup \{v_e \mid e \in \binom{C}{2}\}$ is a closed set in G' of weight $|C|K + \binom{|C|}{2} = k \cdot K + \binom{k}{2} = k'$.

For the converse, assume that X is a closed set in G' of weight k'. Setting $X_V \coloneqq X \cap V'_V$ and $X_E \coloneqq X \cap V'_E$, we show the following sequence of facts:

1) $X_E \subseteq \binom{X_V}{2}$: note that X is only closed if for all $v_{\{u,w\}} \in X_E$, we have $v_u, v_w \in X_V$.
2) $|X_V| = k$ and $|X_E| = \binom{k}{2}$: note that if $|X_V| < k$, then $|X_E| \leq \binom{k-1}{2}$ by 1) and thus the weight of X is $|X_V|K + |X_E| \leq (k-1)K + \binom{k-1}{2} < kK + \binom{k}{2} = k'$. Furthermore, if $|X_V| > k$, then the weight of X is at least $|X_V|K \geq (k+1)K = kK + \binom{k}{2} + 1 > k'$. Thus, we have $|X_V| = k$, and hence we must have $|X_E| = \binom{k}{2}$ for $|X_V|K + |X_E| = k'$ to hold.
3) X_V forms a k-clique in G: Facts 1) and 2) require that $X_E = \binom{X_V}{2}$, which implies that E contains all edges between vertices of X_V.

The last statement concludes the proof of the claim. ◁

Assume that for some c and $\epsilon > 0$, there is an Implications algorithm running in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$. Given a k-clique instance G, we run the above reduction to create a Weighted DAG Implications instance G' with parameter $k' \leq (k+1)(\binom{k}{2}+1) = (k^3+k+2)/2 \leq k^3$ for $k \geq 2$. Observe that G' has $\mathcal{O}(n^2)$ nodes and can be converted to an equivalent Implications instance G'' with the same parameter k' and $\mathcal{O}(k^2n^2)$ nodes by simulating each node weight w by a cycle of w nodes. Now, we determine whether G'' has a closed set of weight $k' \leq k^3$ using the Implications algorithm and thus decide k-clique in time $f(k^3)\mathcal{O}((k^2n^2)^{(\omega/6-\epsilon)k+c}) = f(k^3)k^{\mathcal{O}(k)}n^{(\omega/3-2\epsilon)k+2c}$, refuting the k-Clique conjecture. ◀

6.2 Hardness for SAT($\mathcal{F}$)

In this section, we give our hardness results for general constraint families $\mathcal{F}$ by reducing from (d-uniform Hyper-)Clique either via the independent set problem or via Implications.

To obtain these results, we frequently have to plug-in constant 0s or 1s to obtain our desired constraints. Technically, this is a non-trivial step, as we need to enforce some variables to be assigned fixed values without blowing up the number of variables or the weight of the desired solution. To facilitate our proofs, we first formalize the problem variant that allows us to plug-in constants freely.

▶ **Definition 6.3.** *Let* $\mathcal{F}$ *be an arbitrary constraint family and* $\Sigma \subseteq \{0, 1\}$*. The problem* $\mathsf{SAT}_\Sigma(\mathcal{F})$ *asks to determine whether a given formula* ϕ *with Boolean variables* $x_1, \ldots, x_n$ *has a satisfying assignment of weight* k*, where* ϕ *is a conjunction of* m *constraints of the form* $f(\mathbf{x})$*, where* $f : \{0,1\}^r \to \{0,1\}$ *is a constraint function in* $\mathcal{F}$ *and* $\mathbf{x}$ *is an* r*-tuple over* $\{x_1, \ldots, x_n\} \cup \Sigma$ *(any variable or constant* $c \in \Sigma$ *may be used repeatedly). Note that* $\mathsf{SAT}_\emptyset(\mathcal{F}) = \mathsf{SAT}(\mathcal{F})$*.*

Ideally, we would like to show that $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ is equivalent to $\mathsf{SAT}(\mathcal{F})$. More specifically, we would like to employ reductions of the following form.

▶ **Definition 6.4.** *Let $\mathcal{F}$ be an arbitrary constraint family, and $\Sigma, \Sigma' \subseteq \{0,1\}$ be disjoint. We say that* $\mathsf{SAT}_\Sigma(\mathcal{F})$ expresses Σ'*, if there is a constant c such that the following holds: For any formula ϕ of $\mathsf{SAT}_{\Sigma\cup\Sigma'}(\mathcal{F})$ and parameter k, we can compute, in linear time, a formula ϕ' of $\mathsf{SAT}_\Sigma(\mathcal{F})$ with parameter $k' := k + c$ such that ϕ has a satisfying assignment of weight k if and only if ϕ' has a satisfying assignment of weight k'.*

Indeed, for 0-invalid $\mathcal{F}$, we can show that $\mathsf{SAT}(\mathcal{F})$ expresses $\{0,1\}$ (this is straightforward and was already shown in [28]). For 0-valid $\mathcal{F}$, however, expressing the constant 1 in general appears impossible. To still give tight hardness results for $\mathcal{F}$ whenever it represents a hard function g, we make use of a stronger notion that captures whether we can obtain g already as a restriction that avoids the constant 1. Formally, let $f : \{0,1\}^r \to \{0,1\}, g : \{0,1\}^s \to \{0,1\}$ be arbitrary Boolean functions. We say that a function f *contains g as a 0-restriction* if g is obtained from f by replacing each argument of f either by an argument of g or the constant 0, i.e., we can partition $[r]$ into $X_1, \ldots, X_s, Z_0$ such that

$$g(x_1, \ldots, x_s) = f(\overbrace{x_1 \ldots x_1}^{X_1}, \ldots, \overbrace{x_s \ldots x_s}^{X_s}, \overbrace{0 \ldots 0}^{Z_0}).$$

Using careful constructions, we can prove the following central technical lemma.

▶ **Lemma 6.5.** *Let $\mathcal{F}$ be an arbitrary constraint family and let g be* IMPL *or* NAND_d *for some $d \geq 2$. If some $f \in \mathcal{F}$ contains g as a restriction, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* $\{0,1\}$*, or* $\mathsf{SAT}(\mathcal{F})$ *expresses* 0 *and f contains g already as a 0-restriction.*

Postponing the proof of the above lemma to the Sections 6.3 and 6.4, we can give the proof of our hardness results.

▶ **Theorem 6.6** (Hardness for $\mathsf{SAT}(\mathcal{F})$)**.** *Let $\mathcal{F}$ be a constraint family.*

1. *If $\mathcal{F}$ represents* IMPL*, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(\omega/6-\epsilon)\sqrt[3]{k}+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the k-Clique conjecture fails.*
2. *If $\mathcal{F}$ represents* NAND_2*, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(\omega/3-\epsilon)k+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the k-Clique conjecture fails.*
3. *If $\mathcal{F}$ represents* NAND_d *with $d \geq 3$, then* $\mathsf{SAT}(\mathcal{F})$ *cannot be solved in time $f(k)n^{(1-\epsilon)k+c}$ for any computable $f(k)$ and constants $c, \epsilon > 0$, unless the d-uniform HyperClique conjecture fails.*

Proof. First, we observe that Implications reduces to SAT(IMPL) such that

$$T_{\text{Implications}}(n,k) \leq O(T_{\mathsf{SAT}(\text{IMPL})}(n,k)). \tag{7}$$

Indeed, given any directed graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, we define the formula ϕ with variables $x_1, \ldots, x_n$ and the set of constraints obtained by including $x_i \Rightarrow x_j$ for all $(v_i, v_j) \in E$. Note that for any $S \subseteq [n]$, $\{v_i\}_{i\in S}$ is a valid set in G iff a_S is a satisfying assignment of ϕ, yielding (7).

Similarly, we observe that the d-uniform HyperClique problem reduces to $\mathsf{SAT}(\mathrm{NAND}_d)$ such that

$$T_{d\text{-HC}}(n,k) \leq O(T_{\mathsf{SAT}(\mathrm{NAND}_d)}(n,k)). \tag{8}$$

Indeed, given any d-uniform hypergraph $G = (V, E)$ with $|V| = n$, we define the formula ϕ with variables $x_1, \ldots, x_n$ and the constraints obtained by including, for each distinct $v_{i_1}, \ldots, v_{i_d} \in V$ such that $(v_{i_1}, \ldots, v_{i_d}) \notin E$, the constraint $\mathrm{NAND}_d(x_{i_1}, \ldots, x_{i_d})$. Observe that $(v_{i_1}, \ldots, v_{i_k}) \in V^k$ is a hyperclique in G iff the weight-k assignment with $x_{i_\ell} = 1$ for all $\ell \in [k]$ satisfies ϕ, yielding (8).

It remains to show that whenever some $f \in \mathcal{F}$ contains $g \in \{\mathrm{IMPL}\} \cup \{\mathrm{NAND}_d \mid d \geq 2\}$ as a restriction, then there is a computable $f'(k)$ and constant c' such that

$$T_{\mathsf{SAT}(g)}(n, k) \leq f'(k) \cdot T_{SAT(\mathcal{F})}(n, k + c'). \tag{9}$$

Indeed, if $\mathsf{SAT}(\mathcal{F})$ expresses $\{0, 1\}$, then

$$T_{\mathsf{SAT}(g)}(n, k) \leq \mathcal{O}(T_{\mathsf{SAT}_{\{0,1\}}(\mathcal{F})}(n, k)) \leq f'(k)\mathcal{O}(T_{\mathsf{SAT}(\mathcal{F})}(n, k + c')).$$

Here the first inequality follows by replacing each occurrence of a constraint $g(x_{i_1}, \ldots, x_{i_d})$ of $\mathsf{SAT}(g)$ by the corresponding restriction $f(g_1(x_{i_1}, \ldots, x_{i_d}), \ldots, g_r(x_{i_1}, \ldots, x_{i_d}))$ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$. The second inequality follows from the definition of $\mathsf{SAT}(\mathcal{F})$ expressing $\{0, 1\}$.

In the other case, $\mathsf{SAT}(\mathcal{F})$ expresses only 0, but f contains g already as a 0-restriction. Then we have

$$T_{\mathsf{SAT}(g)}(n, k) \leq \mathcal{O}(T_{\mathsf{SAT}_{\{0\}}(\mathcal{F})}(n, k)) \leq f'(k)\mathcal{O}(T_{\mathsf{SAT}(\mathcal{F})}(n, k + c')),$$

as replacing each occurrence of a constraint $g(x_{i_1}, \ldots, x_{i_d})$ of $\mathsf{SAT}(g)$ by the corresponding restriction $f(g_1(x_{i_1}, \ldots, x_{i_d}), \ldots, g_r(x_{i_1}, \ldots, x_{i_d}))$ does not require the use of the constant 1. The second inequality again follows from the definition of $\mathsf{SAT}(\mathcal{F})$ expressing 0.

As a consequence, by (7) and (9), a $f(k) \cdot O(n^{(\omega/6-\epsilon)\sqrt[3]{k}+c})$ $\mathsf{SAT}(\mathcal{F})$ algorithm for an IMPL-representing family $\mathcal{F}$ would then give an Implications algorithm running in time

$$f(k)f'(k)\mathcal{O}(n^{(\omega/6-\epsilon)\sqrt[3]{k+c'}+c}) = f''(k)\mathcal{O}(n^{(\omega/6-\epsilon)\sqrt[3]{k}+c''}),$$

where $f''(k) = f(k)f'(k)$ and $c'' \leq c + \sqrt[3]{c'}$. This would refute the k-Clique conjecture by Theorem 6.1, concluding 1.

Similarly, a $f(k) \cdot O(n^{\gamma k+c})$ $\mathsf{SAT}(\mathcal{F})$ algorithm for an NAND_d-representing family $\mathcal{F}$ would give a d-uniform HyperClique algorithm running in time

$$f(k)f'(k)\mathcal{O}(n^{\gamma k+c+c'}) = f''(k)\mathcal{O}(n^{\gamma k+c''}),$$

where $f''(k) = f(k)f'(k)$ and $c'' = c + c'$. This yields 2. and 3. by the k-Clique or d-uniform HyperClique conjecture, respectively. ◀

In the remainder of the section, we prove Lemma 6.5. We split the proof in two cases, depending on whether f is 0-invalid (Lemma 6.7) or 0-valid (Corollary 6.14).

6.3 Proof of Lemma 6.5: 0-invalid case

Let f be such that we can obtain IMPL or NAND_d for $d \geq 2$ as a restriction. Note that if it contains $\mathrm{NAND}_d, d > 2$ then it also must contain NAND_2 as a restriction.

In this section, we consider the case that $f(y_1, \ldots, y_r)$ is not 0-valid, i.e., the all-zeroes assignment $u_1 = \cdots = u_r = 0$ does not satisfy f.

▶ **Lemma 6.7.** *If f contains* IMPL *or* NAND_2 *as a restriction and f is 0-invalid, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* $\{0, 1\}$.

The above result in fact follows from the following claim.

▷ Claim 6.8. Let f be as above. Given a parameter k', we can compute, in time $\mathcal{O}(k')$, a formula $\phi_{0,1}$ of $\mathsf{SAT}(\mathcal{F})$ with variables $y, z_1, \ldots, z_{k'+1}$ such that the only satisfying assignment of weight at most k' is $y = 1, z_1 = \cdots = z_{k'+1} = 0$.

Indeed, let us assume the above claim, and take any formula ϕ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ with parameter k. We construct $\phi_{0,1}$ with parameter $k' := k + 1$ and define the formula ϕ' on variable set $x_1, \ldots, x_n, y, z_1, \ldots, z_{k'+1}$ where we include all constraints of $\phi_{0,1}$ and all constraints of ϕ, *replacing each use of the constant* 0 *by* z_1 *and each use of the constant* 1 *by* y. This yields a formula of $\mathsf{SAT}(\mathcal{F})$ with the property that for any weight-k solution $x_1, \ldots, x_n$ of ϕ, the corresponding assignment that sets $y = 1$ and $z_1 = \cdots = z_{k'+1} = 0$ is a weight-$(k+1)$ solution of ϕ'. Conversely, any $(k+1)$-weight solution of ϕ' must set $y = 1$ and $z_1 = 0$ by the above claim, and hence the assignment to $x_1, \ldots, x_n$ must also satisfy ϕ. Observe that this proves Lemma 6.7.

Proof of Claim 6.8. We first give a set of constraints that enforces $y = 1$. To this end, let $S \subseteq [r]$ be such that a_S satisfies f; observe that S exists and is non-empty (otherwise f contains neither IMPL nor NAND_2 as a restriction). For each $j = 1, \ldots, k' + 1$, define the constraint C_j obtained by plugging in y for each u_i with $i \in S$ (i.e., all arguments set to 1 under a_S), and z_j for all other values. We claim that any weight-$(\leq k')$ assignment satisfying $\bigwedge_{j=1}^{k'+1} C_j$ sets $y = 1$: by the weight restriction, at least one of $z_1, \ldots, z_{k'+1}$ must be equal to 0, say z_{j^*}. Then setting $y = 0$ would falsify C_{j^*}, as then all its arguments are 0. Note, however, that the desired assignment $y = 1, z_1 = \cdots = z_{k'+1} = 0$ satisfies $\bigwedge_{j=1}^{k'+1} C_j$.

It remains to give additional constraints enforcing that $z_j = 0$ for *all* $j \in [k' + 1]$. As a first step, we find $S \subsetneq T$ such that $f(a_S) = 1$ but $f(a_T) = 0$: Since f represents IMPL or NAND_2, there is a partition of $[r]$ into X, Y, Z_0, Z_1 such that one of the following set of equalities hold:

	X	Y	Z_0	Z_1				X	Y	Z_0	Z_1		
$f($	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{1\ldots1}$)	$=$	1,	$f($	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{0\ldots0}$,	$\overbrace{1\ldots1}$)	$=$	1,
$f($	$0\ldots0$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,	$f($	$1\ldots1$,	$0\ldots0$,	$0\ldots0$,	$1\ldots1$)	$=$	1,
$f($	$1\ldots1$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,	$f($	$0\ldots0$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	1,
$f($	$1\ldots1$,	$0\ldots0$,	$0\ldots0$,	$1\ldots1$)	$=$	0.	$f($	$1\ldots1$,	$1\ldots1$,	$0\ldots0$,	$1\ldots1$)	$=$	0.

In both cases, the first and fourth line yield sets $S \subsetneq T$ with $f(a_S) = 1$ and $f(a_T) = 0$ (specifically, for $S = Z_1$ and $T = X \cup Z_1$ or for $S = Z_1$ and $T = X \cup Y \cup Z_1$).

Given such S, T, for each $j, j' \in \binom{[r]}{2}$, we define the constraint $C'_{j,j'}$ obtained from $f(u_1, \ldots, u_r)$ by plugging-in y for all u_i with $i \in S$, z_j for all $i \in T \setminus S$ and $z_{j'}$ for all other i. Note that any satisfying assignment of weight at most k sets at least one of $z_1, \ldots, z_{k'+1}$ to 0, say z_{j^*}. Observe that the constraint C'_{j,j^*} is satisfied iff $z_j = 0$, as setting z_j to 0 or 1 corresponds to the assignments a_S (satisfying) or a_T (unsatisfying), respectively. Furthermore, observe that setting $y = 1$ and $z_1 = \cdots = z_{k'+1} = 0$ indeed satisfies all $C'_{j,j'}$. This concludes the claim that the only satisfying assignment of weight at most k' is $y = 1, z_1 = \cdots = z_{k'+1} = 0$. ◁

6.4 Proof of Lemma 6.5: 0-valid case

In this section, we consider the case that $f(y_1, \ldots, y_r)$ is 0-valid, i.e., the all-zeroes assignment $u_1 = \cdots = u_r = 0$ satisfies f. We first observe that we can still express at least the constant 0.

▶ **Lemma 6.9.** *If some* $f \in \mathcal{F}$ *contains* IMPL *or* NAND_2 *as a restriction and* f *is 0-valid, then* $\mathsf{SAT}(\mathcal{F})$ *expresses* 0.

Proof. Observe that it suffices to show how to construct, given a parameter k, a formula on variables $z_1, \dots, z_{k+1}$ such that the only satisfying assignment of weight at most k sets $z_1 = \cdots = z_{k+1} = 0$.

To this end, assume first that f is not satisfied by the all-ones assignment. Then, the formula $\bigwedge_{i=1}^{k+1} f(z_i, \dots, z_i)$ is trivially only satisfied by the assignment $z_1 = \cdots = z_{k+1} = 0$.

Otherwise, observe that there must be a non-empty set $S \subsetneq [r]$ such that a_S does not satisfy f (otherwise f would be a trivial constraint and could contain neither of IMPL and NAND_2). For each $i, i' \in [k+1]$, we define the constraint $C_{i,i'}$ obtained by using z_i for all arguments in S, and $z_{i'}$ for all arguments not in S. Observe that $C_{i,i'} \wedge C_{i',i}$ forces $z_i = z_{i'}$, and thus $z_1 = \cdots = z_{k+1}$, which is satisfied by an assignment of weight at most k if and only if the common value is 0. ◀

Interestingly, for 0-valid f, containing IMPL as a restriction is equivalent to containing IMPL already as a 0-restriction.

▶ **Lemma 6.10.** *If f contains* IMPL *as a restriction and is* 0*-valid, then f contains* IMPL *already as a* 0*-restriction.*

Proof. Since $f : \{0,1\}^r \to \{0,1\}$ contains IMPL as a restriction, we can partition $[r]$ into sets X, Y, Z_0, Z_1 and write

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X}, & \overbrace{0\dots0}^{Y}, & \overbrace{0\dots0}^{Z_0}, & \overbrace{1\dots1}^{Z_1}) & = & 1, \\ f(0\dots0, & 1\dots1, & 0\dots0, & 1\dots1) & = & 1, \\ f(1\dots1, & 1\dots1, & 0\dots0, & 1\dots1) & = & 1, \\ \hline f(1\dots1, & 0\dots0, & 0\dots0, & 1\dots1) & = & 0. \end{array} \tag{10}$$

Assume first that

$$f(\overbrace{0\dots0}^{X}, \overbrace{1\dots1}^{Y}, \overbrace{0\dots0}^{Z_0}, \overbrace{0\dots0}^{Z_1}) = 0. \tag{11}$$

Then, we obtain IMPL as a 0-restriction by setting $X' \coloneqq Y, Y' \coloneqq Z_1, Z' \coloneqq X \cup Z_0$ and observing that

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X'=Y}, & \overbrace{0\dots0}^{Y'=Z_1}, & \overbrace{0\dots0}^{Z'=X\cup Z_0}) & = & 1, & [f \text{ is 0-valid}] \\ f(0\dots0, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ f(1\dots1, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ \hline f(1\dots1, & 0\dots0, & 0\dots0) & = & 0. & [\text{by (11)}] \end{array}$$

Otherwise, if (11) does not hold, then we obtain IMPL as a 0-restriction by setting $X' \coloneqq X \cup Z_1, Y' \coloneqq Y, Z' \coloneqq Z_0$ and observing that

$$\begin{array}{llllll} f(\overbrace{0\dots0}^{X'=X\cup Z_1}, & \overbrace{0\dots0}^{Y'=Y}, & \overbrace{0\dots0}^{Z'=Z_0}) & = & 1, & [f \text{ is 0-valid}] \\ f(0\dots0, & 1\dots1, & 0\dots0) & = & 1, & [\text{by } \neg(11)] \\ f(1\dots1, & 1\dots1, & 0\dots0) & = & 1, & [\text{by (10)}] \\ \hline f(1\dots1, & 0\dots0, & 0\dots0) & = & 0. & [\text{by (10)}] \end{array}$$

◀

It remains to handle the case that f contains NAND_d as a restriction. We first observe that if f contains IMPL as a 0-restriction, then $\mathsf{SAT}_0(\mathcal{F})$ even expresses the constant 1. (Thus, afterwards, we may assume that f does not contain IMPL as a 0-restriction.)

▶ **Lemma 6.11.** *If some $f \in \mathcal{F}$ contains* IMPL *as a 0-restriction, then* $\mathsf{SAT}_0(\mathcal{F})$ *expresses* 1.

Proof. Given any formula ϕ of $\mathsf{SAT}_{\{0,1\}}(\mathcal{F})$ on variables $x_1, \dots, x_n$, we construct a formula ϕ' on variables $x_1, \dots, x_n, y$ as follows: Since some $f \in \mathcal{F}$ contains IMPL as a 0-restriction, we can express, for any variables v, v', the implication $v \Rightarrow v'$ by a corresponding constraint of $\mathsf{SAT}_0(\mathcal{F})$. We construct n such constraints to enforce $\bigwedge_{j=1}^{n}(x_j \Rightarrow y)$. Subsequently, we may use y to replace any use of the constant 1 to convert the constraints of ϕ to constraints of the $\mathsf{SAT}_0(\mathcal{F})$-formula ϕ'.

To argue correctness, note that any satisfying weight-k assignment of ϕ yields a satisfying weight-$(k+1)$ assignment of ϕ' by setting $y = 1$. Conversely, note that any weight-$(k+1)$-assignment of ϕ' must set $y = 1$ (since $k \geq 1$ implies that at least one variable x_i is set to one, which enforces $y = 1$ by the corresponding implication $x_i \Rightarrow y$) and thus corresponds to a weight-k assignment to $x_1, \dots, x_n$ satisfying ϕ. ◀

In the remainder of this section, we assume that f contains NAND_d as a restriction, but does not contain IMPL as a 0-restriction, and the aim is to find NAND_d already as a 0-restriction.

▶ **Lemma 6.12.** *For any 0-valid f, if f does not contain* IMPL *as a 0-restriction, then whenever $f(a_S) = f(a_T) = 1$ with $S \subseteq T$, then $f(a_{T \setminus S}) = 1$.*

Proof. If $S = T$, there is nothing to show, so let $S \subsetneq T$ and assume for contradiction that $f(a_{T \setminus S}) = 0$. We obtain IMPL as a 0-restriction as follows:

$f(\overbrace{0\dots0}^{X=T\setminus S},$	$\overbrace{0\dots0}^{Y=S},$	$\overbrace{0\dots0}^{Z=[r]\setminus T})$	$=$ $1,$	$[f$ is 0-valid$]$
$f(0\dots0,$	$1\dots1,$	$0\dots0)$	$=$ $1,$	$[f(a_S) = 1]$
$f(1\dots1,$	$1\dots1,$	$0\dots0)$	$=$ $1,$	$[f(a_T) = 1]$
$f(1\dots1,$	$0\dots0,$	$0\dots0)$	$=$ $0.$	[by assumption]

This yields the claim. ◀

We can finally obtain NAND_d as a 0-restriction.

▶ **Lemma 6.13.** *If f contains NAND_d as a restriction, does not contain* IMPL *as a 0-restriction and is 0-valid, then f contains NAND_d already as a 0-restriction.*

Proof. Since $f : \{0,1\}^r \to \{0,1\}$ contains NAND_d as a restriction, we can partition $[r]$ into sets $X_1, \dots, X_d, Z_0, Z_1$ such that $X_I \coloneqq \bigcup_{i \in I} X_i$ with $I \subseteq [d]$ satisfies:

$$f(a_{X_I \cup Z_1}) = \begin{cases} 0 & \text{if } I = [d], \\ 1 & \text{if } I \subsetneq [d]. \end{cases} \tag{12}$$

We claim that the partition $X'_i \coloneqq X_i$ for $i < d$, $X'_d \coloneqq X_d \cup Z_1$, $Z' \coloneqq Z_0$ provides NAND_d as a 0-restriction: Letting $X'_I \coloneqq \bigcup_{i \in I} X'_i$, this follows from

$$f(a_{X'_I}) = \begin{cases} 0 & \text{if } I = [d], \\ 1 & \text{if } I \subsetneq [d]. \end{cases} \tag{13}$$

To verify (13), note first that $f(a_{X'_{[d]}}) = f(a_{X_{[d]} \cup Z_1}) = 0$ by (12). Second, let $I \subsetneq [d]$. If $d \in I$, then $f(a_{X'_I}) = f(a_{X_I \cup Z_1}) = 1$ by (12). Otherwise, if $d \notin I$, then we have $f(a_{X'_I}) = f(a_{X_I}) = 1$ by Lemma 6.12 (for this, note that f does not contain IMPL as 0-restriction and that $f(a_{X_I \cup Z_1}) = f(a_{Z_1}) = 1$). This concludes the claim. ◀

The proof of this section is summarized in the following corollary.

▶ **Corollary 6.14.** *If f contains $g \in \{\text{IMPL}\} \cup \bigcup_{d\geq 2}\{\text{NAND}_d\}$ and f is 0-valid, then $\mathsf{SAT}(f)$ expresses $\{0,1\}$, or $\mathsf{SAT}(f)$ expresses 0 and contains g as a 0-restriction.*

Proof. If $g = \text{IMPL}$, then f contains g already as a 0-restriction by Lemma 6.10 and $\mathsf{SAT}(f)$ expresses $\{0,1\}$ by Lemmas 6.9 and 6.11.

If $g = \text{NAND}_d$, then either f also contains IMPL as a 0-restriction, in which case $\mathsf{SAT}(f)$ expresses $\{0,1\}$ by Lemmas 6.9 and 6.11, or it does not contain IMPL as a 0-restriction, and thus f contains g as a 0-restriction by Lemma 6.13 and $\mathsf{SAT}(f)$ expresses 0 by Lemma 6.9. ◀

References

1 Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over Decompress-and-Solve. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 192–203, 2017. doi:10.1109/FOCS.2017.26.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. *SIAM J. Comput.*, 47(6):2527–2555, 2018. doi:10.1137/16M1061771.

3 Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the Orthogonal Vectors conjecture. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 253–266, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188938.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.

5 Enric Boix-Adserà, Matthew Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *Proc. 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 1256–1280, 2019. doi:10.1109/FOCS.2019.00078.

6 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-Parameter Approximability of Boolean MinCSPs. In Piotr Sankowski and Christos Zaroliagis, editors, *Proc. 24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2016.18.

7 Alfred Brauer. On a problem of partitions. *American Journal of Mathematics*, 64(1):299–312, 1942.

8 Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1073–1084, 2017. doi:10.1137/1.9781611974782.69.

9 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of Schaefer's theorem in P: dichotomy of $\exists^k\forall$-quantified first-order graph properties. In *Proc. 34th Computational Complexity Conference (CCC 2019)*, pages 31:1–31:27, 2019. doi:10.4230/LIPIcs.CCC.2019.31.

10 Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In Chris Umans, editor, *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 307–318. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.36.

11 Andrei A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 319–330, 2017. doi:10.1109/FOCS.2017.37.

12 Andrei A. Bulatov and Dániel Marx. Constraint satisfaction parameterized by solution size. *SIAM J. Comput.*, 43(2):573–616, 2014. doi:10.1137/120882160.

13 Nofar Carmeli and Markus Kröll. On the enumeration complexity of unions of conjunctive queries. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proc. 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2019)*, pages 134–148. ACM, 2019. doi:10.1145/3294052.3319700.

14 Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Inf. Comput.*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

15 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.*, 72(8):1346–1367, 2006. doi:10.1016/j.jcss.2006.04.007.

16 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

17 Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995. doi:10.1006/jcss.1995.1087.

18 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM, 2001. doi:10.1137/1.9780898718546.

19 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

20 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

21 Jean Gallier. The Frobenius coin problem. Upper bounds on the Frobenius number, 2014.

22 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000. doi:10.1137/S0097539799349948.

23 Konstantinos Koiliaris and Chao Xu. Faster pseudopolynomial time algorithms for subset sum. *ACM Trans. Algorithms*, 15(3):40:1–40:20, 2019. doi:10.1145/3329863.

24 Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. *TOCT*, 8(1):1:1–1:28, 2016. doi:10.1145/2858787.

25 Andrei A. Krokhin and Dániel Marx. On the hardness of losing weight. *ACM Trans. Algorithms*, 8(2):19:1–19:18, 2012. doi:10.1145/2151171.2151182.

26 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

27 Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 1236–1252, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.

28 Dániel Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005. doi:10.1007/s00037-005-0195-9.

29 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985.

30 Jorge L. Ramirez Alfonsin. *The diophantine Frobenius problem*. Oxford University Press, Oxford, 2005.

31 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory of Computing (STOC 1978)*, pages 216–226, 1978. doi:10.1145/800133.804350.

32 Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, 1990. doi:10.1137/0219054.

33 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

34 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *Proc. 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 331–342, 2017. doi:10.1109/FOCS.2017.38.

35 Dmitriy Zhuk and Barnaby Martin. QCSP monsters and the demise of the Chen conjecture. *CoRR*, abs/1907.00239, 2019. arXiv:1907.00239.

Exponential Resolution Lower Bounds for Weak Pigeonhole Principle and Perfect Matching Formulas over Sparse Graphs

Susanna F. de Rezende
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Jakob Nordström
University of Copenhagen, Denmark
Lund University, Sweden

Kilian Risse
KTH Royal Institute of Technology, Stockholm, Sweden

Dmitry Sokolov
St. Petersburg State University, Russia
St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia

Abstract

We show exponential lower bounds on resolution proof length for pigeonhole principle (PHP) formulas and perfect matching formulas over highly unbalanced, sparse expander graphs, thus answering the challenge to establish strong lower bounds in the regime between balanced constant-degree expanders as in [Ben-Sasson and Wigderson '01] and highly unbalanced, dense graphs as in [Raz '04] and [Razborov '03, '04]. We obtain our results by revisiting Razborov's pseudo-width method for PHP formulas over dense graphs and extending it to sparse graphs. This further demonstrates the power of the pseudo-width method, and we believe it could potentially be useful for attacking also other longstanding open problems for resolution and other proof systems.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases proof complexity, resolution, weak pigeonhole principle, perfect matching, sparse graphs

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.28

Related Version A full version of the paper is available at `https://arxiv.org/abs/1912.00534` and `https://eccc.weizmann.ac.il/report/2019/174/`.

Funding The authors were funded by the Knut and Alice Wallenberg grant KAW 2016.0066. In addition, the second author was supported by the Swedish Research Council grant 2016-00782 and the Independent Research Fund Denmark (DFF) grant 9040-00389B, and the fourth author by the Knut and Alice Wallenberg grant KAW 2016.0433. Part of this work was carried out while visiting the Simons Institute for the Theory of Computing in association with the DIMACS/Simons Collaboration on Lower Bounds in Computational Complexity, which is conducted with support from the National Science Foundation.

Acknowledgements First and foremost, we are most grateful to Alexander Razborov for many discussions about pseudo-width, graph closure, and other mysteries of the universe. We also thank Paul Beame and Johan Håstad for useful discussions, and Jonah Brown-Cohen for helpful references on expander graphs. Finally, we gratefully acknowledge the feedback from participants of the Dagstuhl workshop 19121 *Computational Complexity of Discrete Problems* in March 2019.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 28; pp. 28:1–28:24

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

1 Introduction

In one sentence, proof complexity is the study of efficient certificates of unsatisfiability for formulas in conjunctive normal form (CNF). In its most general form, this is the question of whether coNP can be separated from NP or not, and as such appears out of reach for current techniques. However, if one instead focuses on concrete proof systems, which can be thought of as restricted models of nondeterministic computation, this opens up the view to a rich landscape of results.

One line of research in proof complexity has been to prove superpolynomial lower bounds for stronger and stronger proof systems, as a way of approaching the distant goal of establishing $\mathsf{NP} \neq \mathsf{coNP}$. A perhaps even more fruitful direction, however, has been to study different combinatorial principles and investigate what kind of reasoning is needed to efficiently establish the validity of these principles. In this way, one can quantify the "depth" of different mathematical truths, measured in terms of how strong a proof system is required to prove them.

In this paper, we consider the proof system *resolution* [10], in which one derives new disjunctive clauses from the formula until an explicit contradiction is reached. This is arguably the most well-studied proof system in proof complexity, for which numerous exponential lower bounds on proof size have been shown (starting with [19, 31, 13]). Yet many basic questions about resolution remain stubbornly open. One such set of questions concerns the *pigeonhole principle (PHP)* stating that there is no injective mapping of m pigeons into n holes if $m > n$. This is one of the simplest, and yet most useful, combinatorial principles in mathematics, and it has been topic of extensive study in proof complexity.

When studying the pigeonhole principle, it is convenient to think of it in terms of a bipartite graph $G = (U \dot\cup V, E)$ with pigeons $U = [m]$ and holes $V = [n]$ for $m \geq n + 1$. Every pigeon i can fly to its neighbouring pigeonholes $N(i)$ as specified by G, which for now we fix to be the complete bipartite graph $K_{m,n}$ with $N(i) = [n]$ for all $i \in [m]$. Since we wish to study unsatisfiable formulas, we encode the claim that there does in fact exist an injective mapping of pigeons to holes as a CNF formula consisting of *pigeon axioms*

$$P^i = \bigvee_{j \in N(i)} x_{ij} \qquad \text{for } i \in [m] \tag{1a}$$

and *hole axioms*

$$H_j^{i,i'} = (\overline{x}_{ij} \vee \overline{x}_{i'j}) \qquad \text{for } i \neq i' \in [m], j \in N(i) \cap N(i') \tag{1b}$$

(where the intended meaning of the variables is that $x_{i,j}$ is true if pigeon i flies to hole j). To rule out multi-valued mappings one can also add *functionality axioms*

$$F_{j,j'}^i = (\overline{x}_{ij} \vee \overline{x}_{ij'}) \qquad \text{for } i \in [m], j \neq j' \in N(i) \ , \tag{1c}$$

and a further restriction is to include *surjectivity* or *onto axioms*

$$S_j = \bigvee_{i \in N(j)} x_{ij} \qquad \text{for } j \in [n] \tag{1d}$$

requiring that every hole should get a pigeon. Clearly, the "basic" *pigeonhole principle (PHP) formulas* with clauses (1a) and (1b) are the least constrained. As one adds clauses (1c) to obtain the *functional pigeonhole principle (FPHP)* and also clauses (1d) to get the *onto functional pigeonhole principle (onto-FPHP)*, the formulas become more overconstrained and thus (potentially) easier to disprove, meaning that establishing lower bounds becomes harder.

A moment of reflection reveals that onto-FPHP formulas are just saying that complete bipartite graphs with m left vertices and n right vertices have perfect matchings, and so these formulas are also referred to as *perfect matching formulas*.

Another way of varying the hardness of PHP formulas is by letting the number of pigeons m grow larger as a function of the number of holes n. What this means is that it is not necessary to count exactly to refute the formulas. Instead, it is sufficient to provide a precise enough estimate to show that $m > n$ must hold (where the hardness of this task depends on how much larger m is than n). Studying the hardness of such so-called *weak PHP formulas* gives a way of measuring how good different proof systems are at approximate counting. A second application of lower bounds for weak PHP formulas is that they can be used to show that proof systems cannot produce efficient proofs of the claim that $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ [24, 28].

Yet another version of more constrained formulas is obtained by restricting what choices the pigeons have for flying into holes, by defining the formulas not over $K_{m,n}$ but sparse bipartite graphs with bounded left degree – such instances are usually called *graph PHP formulas*. Again, this makes the formulas easier to disprove in the sense that pigeons are more constrained, and it also removes the symmetry in the formulas that plays an essential role in many lower bound proofs.

Our work focuses on the most challenging setting in terms of lower bounds, when all of these restrictions apply: the PHP formulas contain both functionality and onto axioms, the number of pigeons m is very large compared to the number of holes n, and the choices of holes are restricted by a sparse graph. But before discussing our contributions, let us review what has been known about resolution and pigeonhole principle formulas. We emphasize that what will follow is a brief and selective overview focusing on resolution only – see Razborov's beautiful survey paper [26] for a discussion of upper and lower bounds on PHP formulas in other proof systems.

1.1 Previous Work

In a breakthrough result, which served as a strong impetus for further developments in proof complexity, Haken [19] proved a lower bound $\exp(\Omega(n))$ on resolution proof length for $m = n + 1$ pigeons. Haken's proof was for the basic PHP formulas, but easily extends to onto-FPHP formulas. This result was simplified and improved in a sequence of works [12, 7, 8, 32] to a lower bound of the form $\exp\big(n^2/m\big)$, which, unfortunately, does not yield anything nontrivial for $m = \Omega\big(n^2\big)$ pigeons.

Buss and Pitassi [11] showed that the pigeonhole principle does in fact get easier for resolution when m becomes sufficiently large: namely, for $m = \exp\big(\Omega\big(\sqrt{n \log n}\big)\big)$ PHP formulas can be refuted in length $\exp\big(\mathrm{O}\big(\sqrt{n \log n}\big)\big)$. This is in contrast to what holds for the weaker subsystem *tree-like resolution*, for which the formulas remain equally hard as the number of pigeons increases, and where the complexity was even sharpened in [11, 15, 17, 9] to an $\exp(\Omega(n \log n))$ length lower bound.

Obtaining lower bounds beyond $m = n^2$ pigeons for non-tree-like resolution turned out to be quite challenging. Haken's bottleneck counting method fundamentally breaks down when the number of pigeons is quadratic in the number of holes, and the same holds for the celebrated length-width lower bound in [8]. Some progress was made for restricted forms of resolution in [30] and [22], leading up to an $\exp\big(n^{\varepsilon}\big)$ lower bound for so-called *regular resolution*. In a technical tour de force, Raz [23] finally proved that general, unrestricted resolution requires length $\exp\big(n^{\varepsilon}\big)$ to refute the basic PHP formulas even with arbitrary many pigeons. Razborov followed up on this in three papers where he first simplified and slightly strengthened Raz's result in [25], then extended it to FPHP formulas in [27] and lastly established an analogous lower bound for onto-FPHP formulas in [28].

More precisely, what Razborov showed is that for any version of the PHP formula with m pigeons and n holes, the minimal proof length required in resolution is $\exp\bigl(\Omega\bigl(n/\log^2 m\bigr)\bigr)$. It is easy to see that this implies a lower bound $\exp\bigl(\Omega\bigl(\sqrt[3]{n}\bigr)\bigr)$ for any number of pigeons – for $m = \exp\bigl(\mathrm{O}\bigl(\sqrt[3]{n}\bigr)\bigr)$ we can appeal directly to the bound above, and if a resolution proof would use $\exp\bigl(\Omega\bigl(\sqrt[3]{n}\bigr)\bigr)$ pigeons, then just mentioning all these different pigeons already requires $\exp\bigl(\Omega\bigl(\sqrt[3]{n}\bigr)\bigr)$ distinct clauses. It is also clear that considering complexity in terms of the number of holes n is the right measure. Since any formula contains a basic PHP subformula with $n+1$ pigeons that can be refuted in length $\exp(\mathrm{O}(n))$, we can never hope for exponential lower bounds in terms of formula size as the number of pigeons m grows to exponential.

So far we have stated results only for the standard PHP formulas over $K_{m,n}$, where any pigeon can fly to any hole. However, the way Ben-Sasson and Wigderson [8] obtained their result was by considering graph PHP formulas over balanced bipartite expander graphs of constant left degree, from which the lower bound for $K_{m,n}$ easily follows by a restriction argument. It was shown in [20] that an analogous bound holds for onto-FPHP formulas, i.e., perfect matching formulas, on bipartite expanders. In this context is is also relevant to mention the exponential lower bounds in [1, 16] on *mutilated chessboard formulas*, which can be viewed as perfect matching formulas on balanced, sparse bipartite graphs with very bad expansion. At the other end of the spectrum, Razborov's PHP lower bound in [28] for highly unbalanced bipartite graphs also applies in a more general setting than $K_{m,n}$: namely, for any graph where the minimal degree of any left vertex is δ, the minimal length of any resolution proof is $\exp\bigl(\Omega\bigl(\delta/\log^2 m\bigr)\bigr)$. Thus, for graph PHP formulas we have exponential lower bounds on the one hand [8] for $m \ll n^2$ pigeons, where each pigeon is adjacent to a constant number of holes, and on the other hand [28] for any number of pigeons given that each pigeon is adjacent to a polynomial $n^{\Omega(1)}$ number of holes, but nothing has been known in between these extremes. In [28], Razborov asks whether a *"common generalization"* of the techniques in [8] and [27, 28] can be found *"that would uniformly cover both cases?"* Urquhart [33] also discusses Razborov's lower bound technique, but notes that *"the search for a yet more general point of view remains a topic for further research."*

1.2 Our Results

In this work, we give an answer to the questions raised in [28, 33] by presenting a general technique that applies for any number of pigeons m all the way from linear to weakly exponential, and that establishes exponential lower bounds on resolution proof length for all flavours of graph PHP formulas (including perfect matching formulas) even over sparse graphs.

Let us state below three examples of the kind of lower bounds we obtain – the full, formal statements will follow in later sections. Our first theorem is an average-case lower bound for onto-FPHP formulas with slightly superpolynomial number of pigeons.

▶ **Theorem 1** (Informal). *Let G be a randomly sampled bipartite graph with n right vertices, $m = n^{\mathrm{o}(\log n)}$ left vertices, and left degree $\Theta\bigl(\log^3 m\bigr)$. Then refuting the onto-FPHP formula (a.k.a. perfect matching formula) over G in resolution requires length $\exp\bigl(\Omega\bigl(n^{1-\mathrm{o}(1)}\bigr)\bigr)$ asymptotically almost surely.*

Note that as the number of pigeons grow larger, it is clear that the left degree also has to grow – otherwise we will get a small number of pigeons constrained to fly to a small number of holes by a birthday paradox argument, yielding a small unsatisfiable subformula that can easily be refuted by brute force.

If the number of pigeons increases further to weakly exponential, then randomly sampled graphs no longer have good enough expansion for our technique to work, but there are explicit constructions of unbalanced expanders for which we can still get lower bounds.

▶ **Theorem 2** (Informal). *There are explicitly constructible bipartite graphs G with n right vertices, $m = \exp\big(\mathrm{O}\big(n^{1/16}\big)\big)$ left vertices, and left degree $\Theta\big(\log^4 m\big)$ such that refuting the perfect matching formula over G requires length $\exp\big(\Omega\big(n^{1/8-\varepsilon}\big)\big)$ in resolution.*

Finally, for functional pigeonhole principle formulas we can also prove an exponential lower bound for *constant* left degree even if the number of pigeons is a large polynomial.

▶ **Theorem 3** (Informal). *Let G be a randomly sampled bipartite graph with n right vertices, $m = n^k$ left vertices, and left degree $\Theta\big((k/\varepsilon)^2\big)$. Then refuting the functional pigeonhole principle formula over G in resolution requires length $\exp\big(\Omega\big(n^{1-\varepsilon}\big)\big)$ asymptotically almost surely.*

1.3 Techniques

At a very high level, what we do in terms of techniques is to revisit the pseudo-width method introduced by Razborov for functional PHP formulas in [27]. We strengthen this method to work in the setting of sparse graphs by combining it with the closure operation on expander graphs in [4, 3], which is a way to restore expansion after a small set of (potentially adversarially chosen) vertices have been removed. To extend the results further to perfect matching formulas, we apply a "preprocessing step" on the formulas as in [28]. In what remains of this section, we focus on graph FPHP formulas and give an informal overview of the lower bound proof in this setting, which already contains most of the interesting ideas (although the extension to onto-FPHP also raises significant additional challenges).

Let $\mathit{FPHP}(G)$ denote the functional pigeonhole principle formula over the graph G consisting of clauses (1a)–(1c). A first, quite naive (and incorrect), description of the proof structure is that we start by defining a *pseudo-width* measure on clauses C that counts pigeons i that appear in C in many variables x_{ij} for distinct j. We then show that any short resolution refutation of $\mathit{FPHP}(G)$ can be transformed into a refutation where all clauses have small pseudo-width. By a separate argument, we establish that any refutation of $\mathit{FPHP}(G)$ requires large pseudo-width. Hence, no short refutations can exist, which is precisely what we were aiming to prove.

To fill in the details (and correct) this argument, let us start by making clear what we mean by pseudo-width. Suppose that the graph G has left degree Δ. In what follows, we identify a mapping of pigeon i to a neighbouring hole j with the partial assignment ρ such that $\rho(x_{i,j}) = 1$ and $\rho(x_{i,j'}) = 0$ for all $j' \in N(i) \setminus \{j\}$. We denote by $d_i(C)$ the number of mappings of pigeon i that satisfy C. Note that if C contains at least one negated literal $\overline{x}_{i,j}$, then $d_i(C) \geq \Delta - 1$, and otherwise $d_i(C)$ is the number of positive literals $x_{i,j}$ for $j \in N(i)$. Given a judiciously chosen "filter vector" $\vec{d} = (d_1, \ldots, d_m)$ for $d_i \approx \Delta$ and a "slack" $\delta \approx \Delta/\log m$, we say that pigeon i is *heavy* in C if $d_i(C) \geq d_i - \delta$ and *super-heavy* if $d_i(C) \geq d_i$. We define the *pseudo-width* of a clause C to be the number of heavy pigeons in C.

With these definitions in hand, we can give a description of the actual proof:

1. Given any resolution refutation π of $\mathit{FPHP}(G)$ in small length L, we argue that all clauses can be classified as having either low or high pseudo-width, where an important additional guarantee is that the high-width clauses not only have many heavy pigeons but actually many super-heavy pigeons.

2. We replace all clauses C with many super-heavy pigeons with "fake axioms" $C' \subseteq C$ obtained by throwing away literals from C until we have nothing left but a medium number of super-heavy pigeons. By construction, the set $\mathcal{A}$ of such fake axioms is of size $|\mathcal{A}| \leq L$, and after making the replacement we have a resolution refutation π' of $FPHP(G) \cup \mathcal{A}$ in low pseudo-width.
3. However, since $\mathcal{A}$ is not too large, we are able to show that any resolution refutation of $FPHP(G) \cup \mathcal{A}$ must still require large pseudo-width. Hence, L cannot be small, and the lower bound follows.

Part 1 is similar to [27], but with a slight twist. We show that if the length of π is $L < 2^{w_0}$ and if we choose $\delta \leq \varepsilon\Delta \log n / \log m$, then there exists a vector $\vec{d} = (d_1, \ldots, d_m)$ such that for all clauses in π either the number of super-heavy pigeons is at least w_0 or else the number of heavy pigeons is at most $\mathrm{O}\big(w_0 \cdot n^{\varepsilon}\big)$. The proof of this is by sampling the coordinates d_i independently from a suitable probability distribution and then applying a union bound argument. Once this has been established, part 2 follows easily: we just replace all clauses with at least w_0 super-heavy pigeons by (stronger) fake axioms. Including all fake axioms $\mathcal{A}$ yields a refutation π' of $FPHP(G) \cup \mathcal{A}$ (since we can add a weakening rule deriving C from $C' \subseteq C$ to resolution without loss of generality) and clearly all clauses in π' have pseudo-width $\mathrm{O}\big(w_0 \cdot n^{\varepsilon}\big)$.

Part 3 is where most of the hard work is. Suppose that G is an excellent expander graph, so that for some value r all left vertex sets U' of size $\big|U'\big| \leq r$ have at least $(1 - \varepsilon \log n / \log m)\Delta|U'|$ unique neighbours on the right-hand side. We show that, under the assumptions above, refuting $FPHP(G) \cup \mathcal{A}$ requires pseudo-width $\Omega\big(r \cdot \log n / \log m\big)$. Tuning the parameters appropriately, this yields a contradiction with part 2.

Before outlining how the proof of part 3 goes, we remark that the requirements we place on the expansion of G are quite severe. Clearly, any left vertex set U can have at most $\Delta|U'|$ neighbours in total, and we are asking for all except a vanishingly small fraction of these neighbours to be unique. This is why we can etablish Theorem 1 but not Theorem 2 for randomly sampled graphs. We see no reason to believe that the latter theorem would not hold also for random graphs, but the expansion properties required for our proof are so stringent that they are not satisfied in this parameter regime. This seems to be a fundamental shortcoming of our technique, and it appears that new ideas would be required to circumvent this problem.

In order to argue that refuting $FPHP(G) \cup \mathcal{A}$ in resolution requires large pseudo-width, we want to estimate how much progress the resolution derivation has made up to the point when it derives some clause C. Following Razborov's lead, we measure this by looking at what fraction of partial matchings of all the heavy pigeons in C do not satisfy C (meaning, intuitively, that the derivation has managed to rule out this part of the search space). It is immediate by inspection that all pigeons mentioned in the real axiom clauses (1a)–(1c) are heavy, and any matching of such pigeons satisfies the clauses. Thus, the original axioms in $FPHP(G)$ do not rule out any matchings. Also, it is easy to show that fake axioms rule out only an exponentially small fraction of matchings, since they contain many super-heavy pigeons and it is hard to match all of these pigeons without satisfying the clause. However, the contradictory empty clause $\bot$ rules out 100% of partial matchings, since it contains no heavy pigeons to match in the first place.

What we would like to prove now is that for any derivation in small pseudo-width it holds that the derived clause cannot rule out any matching other than those already eliminated by the clauses used to derive it. This means that the fake axioms together need to rule out all partial matchings, but since every fake axiom contributes only an exponentially small fraction they are too few to achieve this. Hence, it is not possible to derive contradiction in small pseudo-width, which completes part 3 of our proof outline.

There is one problem, however: the last claim above is not true, and so what is outlined above is only a fake proof. While we have to defer the discussion of what the full proof actually looks like in detail, we conclude this section by attempting to hint at a couple of technical issues and how to resolve them.

Firstly, it does not hold that a derived clause C eliminates only those matchings that are also forbidden by one of the predecessor clauses used to derive C. The issue is that a pigeon i that is heavy in both predecessors might cease to be heavy in C – for instance, if C was derived by a resolution step over a variable $x_{i,j}$. If this is so, then we would need to show that any matching of the heavy pigeons in C can be extended to match also pigeon i to any of its neighbouring holes without satisfying both predecessor clauses. But this will not be true, because a non-heavy pigeon can still have some variable $x_{i,j}$ occurring in both predecessors. The solution to this, introduced in [27], is to do a "lossy counting" of matchings by associating each partial matching with a linear subspace of some suitable vector space, and then to consider the span of all matchings ruled out by C. When we accumulate a "large enough" number of matchings for a pigeon i, then the whole subspace associated to i is spanned and we can stop counting.

But this leads to a second problem: when studying matchings of the heavy pigeons in C we might already have assigned pigeons $i'_1, \ldots, i'_w$ that occupy holes where pigeon i might want to fly. For standard PHP formulas over complete bipartite graphs this is not a problem, since at least $n - w$ holes are still available and this number is "large enough" in the sense described above. But for a sparse graph it will typically be the case that $w \gg \Delta$, and so it might well be the case that pigeons $i'_1, \ldots, i'_w$ are already occupying all the Δ holes available for pigeon i according to G. Although it is perhaps hard to see from our (admittedly somewhat informal) discussion, this turns out to be a very serious problem, and indeed it is one of the main technical challenges we need to overcome.

To address this problem we consider not only the heavy pigeons in C, but also any other pigeons in G that risk becoming far too constrained when the heavy pigeons of C are matched. Inspired by [4, 3], we define the *closure* to be a superset S of the heavy pigeons such that when S and the neighbouring holes of S are removed it holds that the residual graph is still guaranteed to be a good expander. Provided that G is an excellent expander to begin with, and that the number of heavy pigeons in C is not too large, it can then be shown that an analogue of the original argument outlined above goes through.

1.4 Outline of This Paper

We review the necessary preliminaries in Section 2 and introduce two crucial technical tools in Section 3. The lower bounds for weak graph FPHP formulas are then presented in Section 4, after which the perfect matching lower bounds follow in Section 5. We conclude with a discussion of questions for future research in Section 6. We refer to the full-length version of this paper for any details missing in this extended abstract.

2 Preliminaries

A *literal* over a Boolean variable x is either the variable x itself (a *positive literal*) or its negation $\overline{x}$ (a *negative literal*). A *clause* $C = \ell_1 \vee \cdots \vee \ell_w$ is a disjunction of literals. We write $\bot$ to denote the empty clause without any literals. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We think of clauses and CNF formulas as sets: order is irrelevant and there are no repetitions. We let $\mathit{Vars}(F)$ denote the set of variables of F.

A *resolution refutation* π of an unsatisfiable CNF formula F, or *resolution proof* for (the unsatisfiability of) F, is an ordered sequence of clauses $\pi = (D_1, \ldots, D_L)$ such that $D_L = \bot$ and for each $i \in [L]$ either D_i is a clause in F (an *axiom*) or there exist $j < i$ and $k < i$ such that D_i is derived from D_j and D_k by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \ . \tag{2}$$

We refer to $B \vee C$ as the *resolvent* of $B \vee x$ and $C \vee \overline{x}$ over x, and to x as the *resolved variable*. For technical reasons it is sometimes convenient to also allow clauses to be derived by the *weakening* rule

$$\frac{C}{D} \ \ [C \subseteq D] \tag{3}$$

(and for two clauses $C \subseteq D$ we will sometimes refer to C as a *strengthening* of D).

The *length* $L(\pi)$ of a refutation $\pi = (D_1, \ldots, D_L)$ is L. The length of refuting F is $\min_{\pi : F \vdash \bot}\{L(\pi)\}$, where the minimum is taken over all resolution refutations π of F. It is easy to show that removing the weakening rule (3) does not increase the refutation length.

A *partial assignment* or a *restriction* on a formula F is a partial function $\rho : \mathit{Vars}(F) \to \{0, 1\}$. The clause C *restricted by* ρ, denoted $C\restriction_\rho$, is the trivial 1-clause if any of the literals in C is satisfied by ρ and otherwise it is C with all falsified literals removed. We extend this definition to CNF formulas in the obvious way by taking unions. For a variable $x \in \mathit{Vars}(F)$ we write $\rho(x) = *$ if $x \notin \mathrm{dom}(\rho)$, i.e., if ρ does not assign a value to x.

We write $G = (V, E)$ to denote a graph with vertices V and edges E, where G is always undirected and without loops or multiple edges. Moreover, for bipartite graphs we write $G = (U \dot\cup V, E)$, where edges in E have one endpoint in the left vertex set U and the other in the right vertex set V. A *partial matching* φ in G is a subset of edges that are vertex-disjoint. Let $V(\varphi) = \{v \mid \exists e \in \varphi : v \in e\}$ be the vertices of φ and for $v \in V(\varphi)$ denote by φ_v the unique vertex u such that $\{u, v\} \in \varphi$. A vertex v is *covered* by φ if $v \in V(\varphi)$. If φ is a partial matching in a bipartite graph $G = (U \dot\cup V, E)$, we identify it with a partial mapping of U to V. When referring to the pigeonhole formula, this mapping will also be identified with an assignment ρ_φ to the variables defined by

$$\rho_\varphi(x_{i,j}) = \begin{cases} * & \text{if } i \notin \mathrm{dom}(\varphi), \\ 0 & \text{if } i \in \mathrm{dom}(\varphi) \text{ and } \varphi(i) \neq j, \\ 1 & \text{if } i \in \mathrm{dom}(\varphi) \text{ and } \varphi(i) = j. \end{cases} \tag{4}$$

Given a vertex $v \in V(G)$, we write $N_G(v)$ to denote the set of *neighbours of* v in the graph G and $\Delta_G(v) = |N_G(v)|$ to denote the degree of v. We extend this notion to sets and denote by $N_G(S) = \{v \mid \exists\, (u, v) \in E \text{ for } u \in S\}$ the *neighbourhood* of a set of vertices $S \subseteq V$. The *boundary*, or *unique neighbourhood*, $\partial_G(S) = \{v \in V \setminus S : |N_G(v) \cap S| = 1\}$ of a set of vertices $S \subseteq V$ contains all vertices in $V \setminus S$ that have a single neighbour in S. We will sometimes drop the subscript G when the graph is clear from context. We denote by $G \setminus U$ the subgraph of G induced by the vertex set $V \setminus U$.

A graph $G = (V, E)$ is an (r, Δ, c)-*expander* if all vertives $v \in V$ have degree at most Δ and for all sets $S \subseteq V$, $|S| \leq r$, it holds that $|N(S) \setminus S| \geq c \cdot |S|$. Similarly, $G = (V, E)$ is an (r, Δ, c)-*boundary expander* if all vertices $v \in V$ have degree at most Δ and for all sets $S \subseteq V$, $|S| \leq r$, it holds that $|\partial(S)| \geq c \cdot |S|$. For bipartite graphs, the degree and expansion requirements only apply to the left vertex set: $G = (U \dot\cup V, E)$ is an (r, Δ, c)-*bipartite expander* if all vertices $u \in U$ have degree at most Δ and for all sets $S \subseteq U$, $|S| \leq r$, it holds that

$|N(S)| \geq c \cdot |S|$, and an (r, Δ, c)-*bipartite boundary expander* if for all sets $S \subseteq U$, $|S| \leq r$, it holds that $|\partial(S)| \geq c \cdot |S|$. For bipartite graphs we will only ever be interested in bipartite notions of expansions, and so which kind of expansion is meant will always be clear from context. A simple but useful observation is that

$$|N(S) \setminus S| \leq |\partial(S)| + \frac{\Delta|S| - |\partial(S)|}{2} = \frac{\Delta|S| + |\partial(S)|}{2} \ , \tag{5}$$

since all non-unique neighbours in $N(S) \setminus S$ have at least two incident edges. This implies that if a graph G is an $(r, \Delta, (1-\xi)\Delta)$-expander then it is also an $(r, \Delta, (1-2\xi)\Delta)$-boundary expander.

We often denote random variables in boldface and write $\boldsymbol{X} \sim \mathcal{D}$ to denote that $\boldsymbol{X}$ is sampled from the distribution $\mathcal{D}$.

For $n, m, \Delta \in \mathbb{N}$, we denote by $\mathcal{G}(m, n, \Delta)$ the distribution over bipartite graphs with disjoint vertex sets $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$ where the neighbourhood of a vertex $u \in U$ is chosen by sampling a subset of size Δ uniformly at random from V. A property is said to hold *asymptotically almost surely* on $\mathcal{G}(f(n), n, \Delta)$ if it holds with probability that approaches 1 as n approaches infinity.

For the right parameters, a randomly sampled graph $G \sim \mathcal{G}(m, n, \Delta)$ is asymptotically almost surely a good boundary expander as stated next.

▶ **Lemma 4.** *Let m, n and Δ be large enough integers such that $m > n \geq \Delta$. Let $\xi, \chi \in \mathbb{R}^+$ be such that $\xi < 1/2$, $\xi \ln \chi \geq 2$ and $\xi\Delta \ln \chi \geq 4 \ln m$. Then for $r = n/(\Delta \cdot \chi)$ and $c = (1-2\xi)\Delta$ it holds asymptotically almost surely for a randomly sampled graph $G \sim \mathcal{G}(m, n, \Delta)$ that G is an (r, Δ, c)-boundary expander.*

We will also consider some parameter settings where randomly sampled graphs do not have strong enough expansion for our purposes, but where we can resort to explicit constructions as follows.

▶ **Theorem 5** ([18]). *For all positive integers m, $r \leq m$, all $\xi > 0$, and all constant $\nu > 0$, there is an explicit $(r, \Delta, (1-\xi)\Delta)$-expander $G = (U \dot\cup V, E)$, with $|U| = m$, $|V| = n$, $\Delta = \mathrm{O}\left(((\log m)(\log r)/\xi)^{1+1/\nu}\right)$ and $n \leq \Delta^2 \cdot r^{1+\nu}$.*

▶ **Corollary 6.** *Let κ, ε, ν be positive constants, $\kappa < \frac{1}{8}$, and let n be a large enough integer. Then there is an explicit graph $G = (U \dot\cup V, E)$, with $|U| = m = 2^{\Omega(n^\kappa)}$ and $|V| \leq n$, that is an $(n^{\frac{1}{1+\nu} - \frac{4\kappa}{\nu}}, \Delta, (1-2\xi)\Delta)$-boundary expander for $\xi = \frac{\varepsilon \log n}{\log m}$ and $\Delta = O(\log^{2(1+1/\nu)} m)$.*

3 Two Key Technical Tools

In this section we review two crucial technical ingredients of the resolution lower bound proofs.

3.1 Pigeon Filtering

The following lemma is a generalization of [27, Lemma 6]. The difference is that we have an additional parameter α (which is implicitly fixed to $\alpha = 2$ in [27]) that allows us to get a better upper bound on the numbers r_i. This turns out to be crucial for us – we discuss this in more detail in Section 4.

▶ **Lemma 7** (Filter lemma). *Let $m, L \in \mathbb{N}^+$ and suppose that $w_0, \alpha \in [m]$ are such that $w_0 > \ln L$ and $w_0 \geq \alpha^2 \geq 4$. Further, let $\vec{r}(1), \ldots, \vec{r}(L)$ be integer vectors, each of the form $\vec{r}(\ell) = (r_1(\ell), \ldots, r_m(\ell))$. Then there exists a vector $\vec{r} = (r_1, \ldots, r_m)$ of positive integers $r_i \leq \left\lfloor \frac{\log m}{\log \alpha} \right\rfloor - 1$ such that for all $\ell \in [L]$ at least one of the following holds:*
1. $\left|\{i \in [m] : r_i(\ell) \leq r_i\}\right| \geq w_0$,
2. $\left|\{i \in [m] : r_i(\ell) \leq r_i + 1\}\right| \leq O(\alpha \cdot w_0)$.

Proof sketch. We first define a weight function $W(\vec{r})$ for vectors $\vec{r} = (r_1, \ldots, r_m)$ as

$$W(\vec{r}) = \sum_{i \in [m]} \alpha^{-r_i} \ . \tag{6}$$

In order to establish the lemma, it is sufficient to show that there exist constants γ and γ' and a vector $r = (r_1, \ldots, r_m)$ such that for all $\ell \in [L]$ the implications

$$W(\vec{r}(\ell)) \geq \frac{\gamma' w_0}{\alpha} \ \Rightarrow \ |\{i \in [m] \mid r_i(\ell) \leq r_i\}| \geq w_0 \ , \tag{7a}$$

$$W(\vec{r}(\ell)) \leq \frac{\gamma' w_0}{\alpha} \ \Rightarrow \ |\{i \in [m] \mid r_i(\ell) \leq r_i + 1\}| \leq \gamma \alpha w_0 \tag{7b}$$

hold. Let $t = \left\lfloor \frac{\log m}{\log \alpha} \right\rfloor - 1$ and let μ be a probability distribution on $[t]$ given by $\Pr[\boldsymbol{r} = i] = \beta \cdot \alpha^{-i}$ for all $i \in [t]$, where $\beta = \frac{\alpha - 1}{1 - \alpha^{-t}}$. Let us write $\vec{\boldsymbol{r}} = (\boldsymbol{r_1}, \ldots, \boldsymbol{r_m})$ to denote a random vector with coordinates sampled independently according to μ. We claim that for every $\ell \in [L]$ the implications (7a) and (7b) are true asymptotically almost surely. The proof of this fact follows by applying Chernoff bounds as in [27]. A union bound argument over all vectors in $\{\vec{r}(\ell) : \ell \in [L]\}$ for both cases shows that for $\gamma' \geq 13$ and $\gamma \geq 5\gamma'$ there exists a choice of $\vec{r} = (r_1, \ldots, r_m)$ such that both implications (7a) and (7b) hold. ◀

3.2 Graph Closure

A key concept in our work will be that of a *closure* of a vertex set, which seems to have originated in [4, 3]. Intuitively, for an expander graph G, the closure of $T \subseteq V(G)$ is a suitably small set S that contains T such that $G \setminus S$ is an expander. In order to have a definition that makes sense for both expanders and bipartite expanders, we define $V_{\mathsf{exp}}(G)$ to be the set of vertices of G that expand, that is, if $G = (V, E)$ is an expander then $V_{\mathsf{exp}}(G) = V$, and if $G = (U \dot\cup V, E)$ is a bipartite expander then $V_{\mathsf{exp}}(G) = U$.

▶ **Definition 8** (Closure). *For an expander graph G and vertex sets $S \subseteq V_{\mathsf{exp}}(G)$ and $U \subseteq V(G)$, we say that the set S is (U, r, ν)-*contained *if $|S| \leq r$ and $\left|\partial(S) \setminus U\right| < \nu \cdot |S|$.*

For any expander graph G and any set $T \subseteq V_{\mathsf{exp}}(G)$ of size $|T| \leq r$, we will let $\mathsf{closure}_{r,\nu}(T)$ denote an arbitrary but fixed maximal set such that $T \subseteq \mathsf{closure}_{r,\nu}(T) \subseteq V_{\mathsf{exp}}(G)$ and $\mathsf{closure}_{r,\nu}(T)$ is $(N(T), r, \nu)$-contained.

Note that the closure of any set T of size $|T| \leq r$ as defined above does indeed exist, since T itself is $(N(T), r, \nu)$-contained.

▶ **Lemma 9.** *Suppose that G is an (r, Δ, c)-boundary expander and that $T \subseteq V_{\mathsf{exp}}(G)$ has size $|T| \leq k \leq r$. Then $|\mathsf{closure}_{r,\nu}(T)| < \frac{k\Delta}{c - \nu}$.*

Proof. By definition we have that $\left|\partial(\mathsf{closure}_{r,\nu}(T)) \setminus N(T)\right| < \nu \cdot |\mathsf{closure}_{r,\nu}(T)|$. Furthermore, since $|\mathsf{closure}_{r,\nu}(T)| \leq r$ by definition, we can use the expansion property of the graph to derive the inequality $\left|\partial(\mathsf{closure}_{r,\nu}(T)) \setminus N(T)\right| \geq |\partial(\mathsf{closure}_{r,\nu}(T))| - |N(T)| \geq c \cdot |\mathsf{closure}_{r,\nu}(T)| - k\Delta$. Note that we also use the fact that the neighbourhood of T is of size at most $k\Delta$. The conclusion follows by combining both statements. ◀

Suppose G is an excellent boundary expander and that $T \subseteq V_{\mathsf{exp}}(G)$ is not too large. Then Lemma 9 shows that the closure of T is not much larger. And if the closure is not too large, then after removing the closure and its neighbourhood from the graph we are still left with a decent expander, a fact which will play a key role in the technical arguments in later sections. The following lemma makes this intuition precise.

▶ **Lemma 10.** *For G an (r, Δ, c)-boundary expander, let $T \subseteq V_{\mathsf{exp}}(G)$ be such that $|T| \leq r$ and $|\mathsf{closure}_{r,\nu}(T)| \leq r/2$, let $G' = G \setminus \big(\mathsf{closure}_{r,\nu}(T) \cup N(\mathsf{closure}_{r,\nu}(T))\big)$ and $V_{\mathsf{exp}}(G') = V_{\mathsf{exp}}(G) \cap V(G')$. Then any set $S \subseteq V_{\mathsf{exp}}(G')$ of size $|S| \leq r/2$ satisfies $|\partial_{G'}(S)| \geq \nu|S|$.*

Proof. Suppose the set $S \subseteq V_{\mathsf{exp}}(G')$ is of size $|S| \leq r/2$ and does not satisfy $|\partial_{G'}(S)| \geq \nu|S|$. Since $\mathsf{closure}_{r,\nu}(T)$ is also of size at most $r/2$, we have that the set $(\mathsf{closure}_{r,\nu}(T) \cup S)$ is $(N(T), r, \nu)$-contained in G. But this contradicts the maximality of $\mathsf{closure}_{r,\nu}(T)$. ◀

4 Lower Bounds for Weak Graph FPHP Formulas

We now proceed to establish lower bounds on the length of resolution refutations of functional pigeonhole principle formulas defined over bipartite graphs. We write $G = (V_P \dot\cup V_H, E)$ to denote the graph over which the formulas are defined and $\mathcal{M}$ to denote the set of partial matchings on G (also viewed as partial mappings of V_P to V_H). Let us start by making more precise some of the technical notions discussed in the introduction (which were originally defined in [25]).

For a clause C and a pigeon i we denote the set of holes j with the property that C is satisfied if i is matched to j by

$$N_C(i) = \{j \in V_H \mid e = \{i, j\} \in E \text{ and } \rho_{\{e\}}(C) = 1\} \tag{8}$$

and we define the *ith pigeon degree* $\Delta_C(i)$ *of* C as $\Delta_C(i) = |N_C(i)|$. We think of a pigeon i with large $\Delta_C(i)$ as a pigeon on which the derivation has not made any significant progress up to the point of deriving C, since the clause rules out very few holes. The pigeons with high enough pigeon degree in a clause are the *heavy pigeons* of the clause as defined next.

▶ **Definition 11** (Pigeon weight, pseudo-width and $(w_0, \vec{d})$-axioms)**.** *Let C be a clause and let $\vec{d} = (d_1, \ldots, d_m)$ and $\vec{\delta} = (\delta_1, \ldots, \delta_m)$ be two vectors of positive integers such that $\vec{d}$ is elementwise greater than $\vec{\delta}$. We say that pigeon i is* $\vec{d}$-super-heavy for C *if $\Delta_C(i) \geq d_i$ and that pigeon i is* $(\vec{d}, \vec{\delta})$-heavy for C *if $\Delta_C(i) \geq d_i - \delta_i$. When $\vec{d}$ and $\vec{\delta}$ are understood from context, which is most often the case, we omit the parameters and just refer to* super-heavy *and* heavy *pigeons. Pigeons that are not heavy are referred to as* light pigeons*. The set of pigeons that are super-heavy for C is denoted by*

$$P_{\vec{d}}(C) = \{i \in [m] \mid \Delta_C(i) \geq d_i\}$$

and the set of pigeons that are heavy for C is denoted by

$$P_{\vec{d},\vec{\delta}}(C) = \{i \in [m] \mid \Delta_C(i) \geq d_i - \delta_i\} \ .$$

The pseudo-width *of C is the number of heavy pigeons in C and the pseudo-width of a resolution refutation π, denoted by $w_{\vec{d},\vec{\delta}}(\pi)$, is $\max_{C \in \pi} w_{\vec{d},\vec{\delta}}(C)$. Finally, we will refer to clauses C with precisely w_0 super-heavy pigeons, i.e., such that $|P_{\vec{d}}(C)| = w_0$, as $(w_0, \vec{d})$-*axioms.

Note that according to Definition 11 super-heavy pigeons are also heavy. Making the connection back to the introduction, the "fake axioms" mentioned there are nothing other than $\left(w_0, \vec{d}\right)$-axioms.

Now that we have all the notions needed, let us give a detailed proof outline. Given a short resolution refutation π of the formula $FPHP(G)$, we use the Filter lemma (Lemma 7) to get a filter vector $\vec{d} = (d_1, \ldots, d_m)$ such that each clause either has many super-heavy pigeons or there are not too many heavy pigeons (for an appropriately chosen vector $\vec{\delta}$). Clearly, clauses that fall into the second case of the filter lemma have bounded pseudo-width. On the other hand, clauses in the first case may have very large pseudo-width. In order to obtain a proof of low pseudo-width, these clauses are strengthened to $\left(w_0, \vec{d}\right)$-axioms and added to a special set $\mathcal{A}$. This then gives a refutation π' that refutes the formula $FPHP(G) \cup \mathcal{A}$ in bounded pseudo-width. The following lemma summarizes the upper bound on pseudo-width that we obtain.

▶ **Lemma 12.** *Let $G = (V_P \dot\cup V_H, E)$ be a bipartite graph with $|V_P| = m$ and $|V_H| = n$; let π be a resolution refutation of $FPHP(G)$; let $w_0, \alpha \in [m]$ be such that $w_0 > \log L(\pi)$ and $w_0 \geq \alpha^2 \geq 4$, and let $\vec{\delta} = (\delta_1, \ldots, \delta_m)$ be defined by $\delta_i = \frac{\Delta_G(i) \log \alpha}{\log m}$. Then there exists an integer vector $\vec{d} = (d_1, \ldots, d_m)$, with $\delta_i < d_i \leq \Delta_G(i)$ for all $i \in V_P$, a set of $\left(w_0, \vec{d}\right)$-axioms $\mathcal{A}$ with $|\mathcal{A}| \leq L(\pi)$, and a resolution refutation π' of $FPHP(G) \cup \mathcal{A}$ such that $w_{\vec{d},\vec{\delta}}(\pi') = \mathrm{O}(\alpha \cdot w_0)$.*

As mentioned above, this upper bound is a straightforward application of Lemma 7. We defer the formal proof to a later point in this section. What we will need from Lemma 12 is that a resolution refutation of $FPHP(G)$ in length less than 2^{w_0} can be transformed into a refutation of $FPHP(G) \cup \mathcal{A}$ in pseudo-width at most $\mathrm{O}(\alpha \cdot w_0)$.

The second step in the proof is to show that any resolution refutation π of $FPHP(G) \cup \mathcal{A}$ requires large pseudo-width. The high-level idea is to define a progress measure on clauses $C \in \pi$ by counting the number of matchings on $P_{\vec{d},\vec{\delta}}(C)$ that do not satisfy C. We then show that in order to increase this progress measure we need large pseudo-width. The following lemma states the pseudo-width lower bound.

▶ **Lemma 13.** *Let $\xi \leq 1/4$ and $m, n, r, \Delta \in \mathbb{N}$; let $G = (V_P \dot\cup V_H, E)$ with $|V_P| = m$ and $|V_H| = n$ be an $(r, \Delta, (1-2\xi)\Delta)$-boundary expander, and let $\vec{\delta} = (\delta_1, \ldots, \delta_m)$ be defined by $\delta_i = 4\Delta_G(i)\xi$. Suppose that $\vec{d} = (d_1, \ldots, d_m)$ is an integer vector such that $\delta_i < d_i \leq \Delta_G(i)$ for all $i \in V_P$. Let w_0 be an arbitrary parameter and $\mathcal{A}$ be an arbitrary set of $\left(w_0, \vec{d}\right)$-axioms with $|\mathcal{A}| \leq (1+\xi)^{w_0}$. Then every resolution refutation π of $FPHP(G) \cup \mathcal{A}$ must satisfy $w_{\vec{d},\vec{\delta}}(\pi) \geq r\xi/4$.*

In one sentence, the lemma states that if the set of "fake axioms" $\mathcal{A}$ is not too large, then resolution requires large pseudo-width to refute $FPHP(G) \cup \mathcal{A}$. Note that this lemma holds for any filter vector and not just for the one obtained from Lemma 12.

In order to prove Lemma 13, we wish to define a progress measure on clauses that indicates how close the derivation is to refuting the formula (i.e., it should be small for axiom clauses but large for contradiction). A first attempt would be to define the progress of a clause C as the number of ruled-out matchings (i.e., matchings that do not satisfy C) on the pigeons mentioned by C. This definition does not quite work, but we can refine it by counting matchings less carefully. Namely, if for a pigeon i there are more than $\Delta_G(i) - d_i + \delta_i/4$ holes to which it can be mapped without satisfying C, then we think of C as ruling out *all holes* for this pigeon. Since the pigeon degree of a light pigeon i is at most $d_i - \delta_i$, such a pigeon will certainly have at least $\Delta_G(i) - d_i + \delta_i \geq \Delta_G(i) - d_i + \delta_i/4$ holes to which it can be mapped, and the "lossy counting" will ensure that all holes are considered as ruled out.

We realize this "lossy counting" through a linear space Λ, in which each partial matching φ is associated with a subspace $\lambda(\varphi)$. Roughly speaking, the progress $\lambda(C)$ of a clause C is then defined to be the span of all partial matchings that are ruled out by C. We design the association between matchings and subspaces so that the contradictory empty clause $\bot$ has $\lambda(\bot) = \Lambda$ but so that the span of all the axioms $\text{span}(\{\lambda(A) \mid A \in FPHP(G) \cup \mathcal{A}\})$ is a proper subspace of Λ. This implies that in a refutation π of $FPHP(G) \cup \mathcal{A}$ there must exist a resolution step deriving a clause C from clauses C_0 and C_1 such that the linear space of the resolvent $\lambda(C)$ is not contained in $\text{span}(\lambda(C_0), \lambda(C_1))$. But the main technical lemma of this section (Lemma 20) says that for any derivation in low pseudo-width the linear space of the resolvent is contained in the span of the linear spaces of the clauses being resolved. Hence, in order for π to be a refutation it must contain a clause with large pseudo-width, and this establishes Lemma 13.

So far our argument follows that of Razborov very closely, but it turns out we cannot realize this proof idea if we only keep track of heavy and light pigeons. Let us attempt a proof of the claim in Lemma 20 that low-width resolution steps cannot increase the span to illustrate what the problem is. The interesting case is when there is a pigeon i that is heavy for C_0 or C_1 but not for their resolvent C. Then, following Razborov, for any matching φ on the heavy pigeons of C that fails to satisfy C, we need to be able to extend φ in at least $\Delta_G(i) - d_i + \delta_i/4$ different ways to a matching including also pigeon i that falsifies either C_0 or C_1. If this can be done, then we think of C_0 and C_1 as together ruling out (essentially) all holes for i, and the linear space associated with C will be contained in the span of the spaces for C_0 and C_1. The problem, though, is that φ may send all heavy pigeons to the neighbourhood of pigeon i. In this scenario, there might be very few holes, or even no holes, to which i can be mapped when extending φ, and even our lossy counting will not be able to pick up enough holes for the argument to go through. We resolve this problem by not only considering the heavy pigeons but a larger set of *relevant* pigeons including all pigeons i' that can become overly constrained when some matching on the heavy pigeons shrinks the neighbourhood of i' too much. Formally, the *closure* of the set of heavy pigeons, as defined in Definition 8, is the notion that we need.

4.1 Formal Statements of Graph FPHP Formula Lower Bounds

Deferring the proofs of all technical lemmas for now, let us state our lower bounds for graph FPHP formulas and see how they follow from Lemmas 12 and 13 above.

▶ **Theorem 14.** *Let $m = |U|$ and $n = |V|$ and suppose that $G = (U \dot\cup V, E)$ is an $\left(r, \Delta, \left(1 - \frac{\log \alpha}{2 \log m}\right)\Delta\right)$-boundary expander for $\alpha \in [m]$ such that $8 \leq \frac{\alpha^3}{\log \alpha} = o\left(\frac{r}{\log m}\right)$. Then resolution requires length $\exp\left(\Omega\left(\frac{r \log^2 \alpha}{\alpha \log^2 m}\right)\right)$ to refute $FPHP(G)$.*

Note that, on the one hand, the larger α is, the more relaxed we can be with respect to the expansion requirements, and hence the set of formulas to which the lower bound applies becomes larger. On the other hand, the strength of the lower bound deterioratesith α. Hence, we need to choose α carefully to find a good compromise between these two concerns.

Proof of Theorem 14. Let $\xi = \frac{\log \alpha}{4 \log m}$ and let $w_0 = \frac{\varepsilon_0 r \xi}{\alpha}$ for some small enough $\varepsilon_0 > 0$. We note that the choice of parameters and the condition on α ensure that $4 \leq \alpha^2 \leq w_0$. Furthermore, in terms of ξ, the graph G is an $(r, \Delta, (1 - 2\xi)\Delta)$-boundary expander.

We proceed by contradiction. Suppose π is a resolution refutation with $L(\pi) < 2^{\varepsilon' w_0 \xi}$ for a small enough constant $\varepsilon' > 0$. Applying Lemma 12 we get a set of $\left(w_0, \vec{d}\right)$-axioms $\mathcal{A}$ with $|\mathcal{A}| \leq L(\pi)$ and a resolution refutation π' of $FPHP(G) \cup \mathcal{A}$ such that $w_{\vec{d},\vec{\delta}}(\pi') \leq K \alpha w_0$ for some large enough constant K.

Note that $|\mathcal{A}| \leq L(\pi) < 2^{\varepsilon' w_0 \xi} \leq (1+\xi)^{w_0}$ for $\varepsilon' < 1/2$. Applying Lemma 13 to π' yields a pseudo-width lower bound of $r\xi/4$. We conclude that

$$r\xi/4 \leq w_{\vec{d},\vec{\delta}}(\pi') \leq K\alpha w_0 = \varepsilon_0 K r \xi \ . \tag{9}$$

Choosing $\varepsilon_0 < \frac{1}{4K}$ yields a contradiction. ◀

The following corollary summarizes our claims for random graphs.

▶ **Corollary 15.** *Let m and n be positive integers and let $\Delta : \mathbb{N}^+ \to \mathbb{N}^+$ and $\varepsilon : \mathbb{N}^+ \to [0,1]$ be any monotone functions of n such that $n < m \leq n^{(\varepsilon/16)^2 \log n}$ and $n \geq \Delta \geq \left(\frac{16 \log m}{\varepsilon \log n}\right)^2$. Then asymptotically almost surely resolution requires length $\exp\left(\Omega\left(n^{1-\varepsilon}\right)\right)$ to refute FPHP(G) for $G \sim \mathcal{G}\left(m, n, \Delta\right)$.*

Proof sketch. We first note that it is sufficient to prove the claim for $m = n^{(\varepsilon/16)^2 \log n}$ and $\Delta = \left((16 \log m)/(\varepsilon \log n)\right)^2$. By applying Lemma 4 for $\chi = \alpha = n^{\varepsilon/4}$ and $\xi = \frac{\log \alpha}{4 \log m}$, we conclude that asymptotically almost surely, $G \sim \mathcal{G}\left(m, n, \Delta\right)$ is an $\left(n^{1-\varepsilon/2}, \Delta, (1-2\xi)\Delta\right)$-boundary expander. Theorem 14 then gives a length lower bound of $\exp\left(\Omega\left(n^{1-\varepsilon}\right)\right)$. ◀

The following two corollaries are simple consequences of Corollary 15, optimizing for different parameters. The first corollary gives the strongest lower bounds, while the second minimizes the degree.

▶ **Corollary 16.** *Let m, n be such that $m \leq n^{\mathrm{o}(\log n)}$. Then asymptotically almost surely resolution requires length $\exp\left(\Omega\left(n^{1-\mathrm{o}(1)}\right)\right)$ to refute FPHP(G) for $G \sim \mathcal{G}\left(m, n, \log m\right)$.*

Proof. Let $m = n^{f(n)}$, where $f(n) = \mathrm{o}(\log n)$. Applying Corollary 15 for $\varepsilon = 16\sqrt{\frac{f(n)}{\log n}} = \mathrm{o}(1)$ we get the desired statement. ◀

▶ **Corollary 17** (Restatement of Theorem 3)**.** *Let k and n be positive integers and let $m = n^k$ and $\varepsilon \in \mathbb{R}^+$. Then asymptotically almost surely resolution requires length $\exp\left(\Omega\left(n^{1-\varepsilon}\right)\right)$ to refute FPHP(G) for $G \sim \mathcal{G}\left(m, n, \left(\frac{16k}{\varepsilon}\right)^2\right)$.*

Proof. We appeal to Corollary 15 with $\Delta = \left(\frac{16k}{\varepsilon}\right)^2$, $m = n^k$ and ε constant. A short calculation shows that all conditions are met. ◀

Our final corollary shows that we can get meaningful lower bounds even for a weakly exponential number of pigeons. Unfortunately, the statement does not hold for random graphs.

▶ **Corollary 18.** *Let $\kappa < 3/2 - \sqrt{2}$ and $\varepsilon > 0$ be constant and n be integer. Then there is a family of explicitly constructible graphs G with $m = 2^{\Omega(n^\kappa)}$ and left degree $\mathrm{O}\left(\log^{1/\sqrt{\kappa}}(m)\right)$ such that resolution requires length $\exp\left(\Omega\left(n^{1-2\sqrt{\kappa}(2-\sqrt{\kappa})-\varepsilon}\right)\right)$ to refute FPHP(G).*

Proof. Let G be the graph from Corollary 6 with $\nu = \frac{2\sqrt{\kappa}}{1-2\sqrt{\kappa}}$. An appeal to Theorem 14 using the graph G yields the desired lower bound. ◀

4.2 A Pseudo-Width Upper Bound for Graph FPHP Formulas with Extra Axioms

Let us now prove Lemma 12. For this proof, let us identify V_P with $[m]$. For every clause C in the refutation π, let $\vec{r}(C) = (r_1(C), \ldots, r_m(C))$ be the vector where each coordinate is given by

$$r_i(C) = \left\lfloor \frac{\Delta_G(i) - \Delta_C(i)}{\delta_i} \right\rfloor + 1 \ . \tag{10}$$

We apply the filter lemma (Lemma 7) to the set of vectors $\{\vec{r}(C) \mid C \in \pi\}$. Denote by $\vec{r} = (r_1, \ldots, r_m)$ a vector as guaranteed to exist by Lemma 7. Let

$$d_i = \Delta_G(i) - \lceil \delta_i r_i \rceil + 1 \ . \tag{11}$$

A short calculation establishes that d_i is the smallest integer such that $\left\lfloor \frac{\Delta_G(i) - d_i}{\delta_i} \right\rfloor + 1 \leq r_i$.

Note that every pigeon $i \in [m]$ such that $r_i(C) \leq r_i$ is super-heavy for C. Also, every heavy pigeon of a clause C satisfies that $r_i(C) \leq r_i + 1$.

To obtain a refutation π' that satisfies the conclusions of the lemma, we consider every clause $C \in \pi$ and either add a strengthening of C to the $(w_0, \vec{d})$-axiom set $\mathcal{A}$ or conclude that the pseudo-width of C is small enough that the clause can stay in π'. More concretely, we make a case distinction whether $\vec{r}(C)$ satisfies case 1 of Lemma 7 or only case 2. In one case C can be strengthened to a $(w_0, \vec{d})$-axiom, while in the other the pseudo-width of C is bounded:

1. C satisfies $\left|\{i \in [m] \mid r_i(C) \leq r_i\}\right| \geq w_0$: As every pigeon $i \in [m]$ with $r_i(C) \leq r_i$ also satisfies $\Delta_C(i) \geq d_i$, we can strengthen this clause to a $(w_0, \vec{d})$-axiom and add it to $\mathcal{A}$. This reduces the pseudo-width of this clause to w_0.
2. C satisfies $\left|\{i \in [m] \mid r_i(C) \leq r_i + 1\}\right| \leq O(\alpha \cdot w_0)$: As every heavy pigeon always satisfies $r_i(C) \leq r_i + 1$, the pseudo-width of C is $\mathrm{O}(\alpha \cdot w_0)$.

This concludes the proof as $|\mathcal{A}| \leq L(\pi)$ and the pseudo-width of π' is $\mathrm{O}(\alpha \cdot w_0)$ by construction.

4.3 A Pseudo-Width Lower Bound for Graph FPHP Formulas with Extra Axioms

We continue to the proof of Lemma 13. Using Definition 8, we define the set of *relevant* pigeons of a clause C as

$$\mathsf{closure}(C) = \mathsf{closure}_{r,(1-3\xi)\Delta}(P_{\vec{d},\vec{\delta}}(C)) \ , \tag{12}$$

where $P_{\vec{d},\vec{\delta}}(C)$ denotes the set of $(\vec{d}, \vec{\delta})$-heavy pigeons for C as defined in Definition 11. By definition, the closure of a set T contains T itself but is only defined if $|T| \leq r$. However, if $\left|P_{\vec{d},\vec{\delta}}(C)\right| \geq r \geq r\xi/4$ then we already have the lower bound claimed in the lemma, and so we may assume that the closure is well defined for all clauses in the refutation π. This implies, in particular, that for every clause $C \in \pi$ we have $P_{\vec{d},\vec{\delta}}(C) \subseteq \mathsf{closure}(C)$.

Let us next construct the linear space Λ and describe how matchings are mapped into it. Fix a field $\mathbb{F}$ of characteristic 0 and for each pigeon $i \in V_P$ let Λ_i be a linear space over $\mathbb{F}$ of dimension $\Delta_G(i) - d_i + \delta_i/4$. Let Λ be the tensor product $\Lambda = \bigotimes_{i \in V_P} \Lambda_i$ and denote by $\lambda_i : V_H \mapsto \Lambda_i$ a function with the property that any subset of holes $J \subseteq V_H$ of size at least $\dim(\Lambda_i)$ spans Λ_i. In other words, for J as above we have that $\Lambda_i = \mathrm{span}(\lambda_i(j) : j \in J)$. This is how we will realize the idea of "lossy counting." For $J \subseteq V_H$ such that $|J| \leq \dim(\Lambda_i)$ we have exact counting $\dim(\mathrm{span}(\{\lambda_i(j) \mid j \in J\})) = |J|$, but when $|J| > \dim(\Lambda_i)$ gets large enough we have $\dim(\mathrm{span}(\{\lambda_i(j) \mid j \in J\})) = \dim(\Lambda_i)$.

In order to map functions $V_P \mapsto V_H$ into Λ, we define $\lambda : V_H^{V_P} \mapsto \Lambda$ by $\lambda(j_1, \ldots, j_m) = \bigotimes_{i \in V_P} \lambda_i(j_i)$, where will we abuse notions slightly in that we identify a vector with the 1-dimensional space spanned by this vector. For a partial function $\varphi : V_P \mapsto V_H$, we let $\lambda(\varphi)$ be the span of all total extensions of φ (not necessarily matchings), or equivalently

$$\lambda(\varphi) = \bigotimes_{i \in \mathrm{dom}(\varphi)} \lambda_i(\varphi_i) \otimes \bigotimes_{i \notin \mathrm{dom}(\varphi)} \Lambda_i \ . \tag{13}$$

Recall that $\mathcal{M}$ is the set of all partial matchings on the graph G and that we interchangeably think of partial matchings as partial functions $\varphi : V_P \to V_H$ or as Boolean assignments ρ_φ as defined in (4). For each clause C, we are interested in the partial matchings $\varphi \in \mathcal{M}$ with domain $\mathrm{dom}(\varphi) = \mathsf{closure}(C)$ such that ρ_φ does not satisfy C. We refer to the set of such matchings as the *zero space* of C and denote it by

$$Z(C) = \{\varphi \in \mathcal{M} \mid \mathrm{dom}(\varphi) = \mathsf{closure}(C) \wedge \rho_\varphi(C) \neq 1\} \ . \tag{14}$$

We associate C with the linear space

$$\lambda(C) = \mathrm{span}(\{\lambda(\varphi) \mid \varphi \in Z(C)\}) \ . \tag{15}$$

Note that contradiction is mapped to Λ, i.e., $\lambda(\bot) = \Lambda$.

We assert that the span of the axioms $\mathrm{span}(\{\lambda(A) \mid A \in \mathit{FPHP}(G) \cup \mathcal{A}\})$ is a proper subspace of Λ.

▶ **Lemma 19.** *If* $|\mathcal{A}| \leq (1+\xi)^{w_0}$*, then* $\mathrm{span}(\{\lambda(A) \mid A \in \mathit{FPHP}(G) \cup \mathcal{A}\}) \subsetneq \Lambda$.

Accepting this claim without proof for now, this implies that in π there is some resolution step deriving C from C_0 and C_1 where the subspace of the resolvent is not contained in the span of the subspaces of the premises, or in other words $\lambda(C) \not\subseteq \mathrm{span}(\lambda(C_0), \lambda(C_1))$. Our next lemma, which is the heart of the argument, says that this cannot happen as long as the closures of the clauses are small.

▶ **Lemma 20.** *Let* C *be a clause derived from clauses* C_0 *and* C_1*. If it is the case that* $\max\{|\mathsf{closure}(C_0)|, |\mathsf{closure}(C_1)|, |\mathsf{closure}(C)|\} \leq r/4$*, then* $\lambda(C) \subseteq \mathrm{span}(\lambda(C_0), \lambda(C_1))$.

Since contradiction cannot be derived while the closure is of size at most $r/4$, any refutation π must contain a clause C with $|\mathsf{closure}(C)| > r/4$. But then Lemma 9 implies that C has pseudo-width at least $r\xi/4$, and Lemma 13 follows. All that remains for us is to establish Lemmas 19 and 20.

Proof of Lemma 19. We need to show that the axioms $\mathit{FPHP}(G) \cup \mathcal{A}$ do not span all of Λ. We start with the axioms in $\mathit{FPHP}(G)$.

Let A be pigeon axiom P^i as in (1a) or a functionality axiom $F^i_{j,j'}$ as in (1c). Note that i is a heavy pigeon for A. Clearly, there are no pigeon-to-hole assignments for pigeon i that do not satisfy A. Thus there are no matchings on $\mathsf{closure}(A)$ that do not satisfy A. We conclude that $\lambda(A) = \emptyset$. If instead A is a hole axiom $H^{i,i'}_j$ as in (1b), then we can observe that $\Delta_G(i) - 1 \geq d_i - \delta_i$ since $\delta_i = 4\xi\Delta_G(i) \geq 2\xi\Delta \geq 1$ (by boundary expansion). This implies that A has two heavy pigeons. Observe that there are no matchings on these two pigeons that do not satisfy A. Thus $Z(A) = \emptyset$ and we conclude that $\lambda(A) = \emptyset$.

Now consider the $(w_0, \vec{d})$-axioms in $\mathcal{A}$. We wish to show that any $A \in \mathcal{A}$ can only span a very small fraction of Λ. We can estimate the the number of dimensions $\lambda(A)$ spans by

$$\dim \lambda(A) \leq \prod_{i \notin P_{\vec{d}}(A)} \dim \Lambda_i \cdot \prod_{i \in P_{\vec{d}}(A)} (\Delta_G(i) - d_i) \ . \tag{16}$$

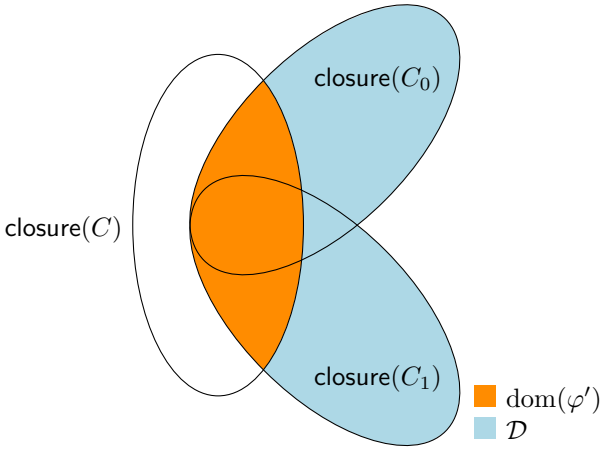


Figure 1 Depiction of relations between $\mathsf{closure}(C), \mathsf{closure}(C_i), i = 1, 2, \mathrm{dom}(\varphi')$ and $\mathcal{D}$ in proof of Lemma 20.

Hence the fraction of the space Λ that A may span is bounded by

$$\frac{\dim \lambda(A)}{\dim \Lambda} \leq \prod_{i \in P_{\vec{d}}(A)} \frac{\Delta_G(i) - d_i}{\Delta_G(i) - d_i + \delta_i/4} \leq (1-\xi)^{w_0} \quad . \tag{17}$$

As $|\mathcal{A}| \leq (1+\xi)^{w_0}$ we can conclude that not all of Λ is spanned by the axioms. ◀

Proof of Lemma 20. For conciseness of notation, let us write $S_{01} = \mathsf{closure}(C_0) \cup \mathsf{closure}(C_1)$ and $S = \mathsf{closure}(C)$. In order to establish the lemma, we need to show for all $\varphi \in Z(C)$ that

$$\lambda(\varphi) \subseteq \mathrm{span}(\lambda(C_0), \lambda(C_1)) \quad . \tag{18}$$

To comprehend the argument that will follow below, it might be helpful to refer to the illustration in Figure 1.

Denote by φ' the restriction of φ to the domain $S \cap S_{01}$ and note that C is not satisfied under $\rho_{\varphi'}$. Also, observe that if a matching η extends a matching η', then $\lambda(\eta)$ is contained in $\lambda(\eta')$. This is so since for any pigeon $i \in \mathrm{dom}(\eta) \setminus \mathrm{dom}(\eta')$ we have from (13) that η' picks up the whole subspace Λ_i while η only gets a single vector. Thus, if we can show that $\lambda(\varphi') \subseteq \mathrm{span}(\lambda(C_0), \lambda(C_1))$, then we are done as φ extends φ' and hence $\lambda(\varphi) \subseteq \lambda(\varphi')$.

Let $\mathcal{D} = S_{01} \setminus S$ and denote by $\mathcal{M}_{\mathcal{D}}$ the set of matchings that extend φ' to the domain $\mathcal{D}$ and do not satisfy C. Since each matching $\psi \in \mathcal{M}_{\mathcal{D}}$ fails to satisfy C, by the soundness of the resolution rule we have that it also fails to satisfy either C_0 or C_1. Assume without loss of generality that ψ does not satisfy C_0 and denote by ψ' the restriction of ψ to the domain of $\mathsf{closure}(C_0)$. From (14) we see that $\psi' \in Z(C_0)$ and therefore $\lambda(\psi) \subseteq \lambda(\psi') \subseteq \lambda(C_0)$.

So far we have argued that for all $\psi \in \mathcal{M}_{\mathcal{D}}$ it holds that $\lambda(\psi) \subseteq \mathrm{span}(\lambda(C_0), \lambda(C_1))$. Let $\lambda(\mathcal{M}_{\mathcal{D}}) = \mathrm{span}(\lambda(\psi) \mid \psi \in \mathcal{M}_{\mathcal{D}})$. If we can show that the set of matchings $\mathcal{M}_{\mathcal{D}}$ is large enough for $\lambda(\mathcal{M}_{\mathcal{D}}) = \lambda(\varphi')$ to hold, then the lemma follows. In other words, we want to show that $\lambda(\mathcal{M}_{\mathcal{D}})$ projected to $\Lambda_{\mathcal{D}} = \bigotimes_{i \in \mathcal{D}} \Lambda_i$ spans all of the space $\Lambda_{\mathcal{D}}$.

To argue this, note first that $\mathcal{D}$ is completely outside the $\mathsf{closure}(C)$. Furthermore, by assumption we have $|\mathsf{closure}(C)| \leq r/4$ and $|\mathcal{D}| \leq |S_{01}| \leq r/2$. An application of Lemma 10 now tells us that

$$|\partial_{G \setminus (\mathsf{closure}(C) \cup N(\mathsf{closure}(C)))}(\mathcal{D})| \geq (1 - 3\xi)\Delta|\mathcal{D}| \quad . \tag{19}$$

By an averaging argument, there must exist a pigeon $i_1 \in \mathcal{D}$ that has more than $(1-3\xi)\Delta$ unique neighbours in $\partial_{G\setminus(\mathsf{closure}(C)\cup N(\mathsf{closure}(C)))}(\mathcal{D})$. The same argument applied to $\mathcal{D}\setminus\{i_1\}$ show that some pigeon i_2 has more than $(1-3\xi)\Delta$ unique neighbours on top of the neighbours reserved for pigeon i_1. Iterating this argument, we derive by induction that for each pigeon $i \in \mathcal{D}$ we can find $(1-3\xi)\Delta$ distinct holes in $N(\mathcal{D})$. Since all pigeons in $\mathcal{D}$ are light in C, it follows that at most $d_i - \delta_i$ mappings of pigeon i can satisfy the clause C. Hence, there are at least

$$(1-3\xi)\Delta - (d_i - \delta_i) \;\geq\; (1-3\xi)\Delta_G(i) - d_i + 4\xi\Delta_G(i) \;\geq\; \Delta_G(i) - d_i + \delta_i/4 \tag{20}$$

many holes to which each pigeon in $\mathcal{D}$ can be sent, independently of all other pigeons in $\mathcal{D}$, without satisfying C. As we have that $\dim(\Lambda_i) = \Delta_G(i) - d_i + \delta_i/4$, we conclude that $\lambda(\mathcal{M}_\mathcal{D})$ projected to $\Lambda_\mathcal{D}$ spans the whole space. This concludes the proof of the lemma. ◀

5 Lower Bounds for Perfect Matching Principle Formulas

In this section, we show that the perfect matching principle formulas defined over even highly unbalanced bipartite graphs require exponentially long resolution refutations if the graphs are expanding enough.

Just as in [28], our proof is by an indirect reduction to the FPHP lower bound, and therefore there is a significant overlap in concepts and notation with Section 4. However, since there are also quite a few subtle shifts in meaning, we restate all definitions in full below to make the exposition in this section self-contained and unambiguous.

We first review some useful notions from [25]. Let $G = (V, E)$ denote the graph over which the formulas are defined. For a clause C and a vertex $v \in V(G)$, let the *clause-neighbourhood of v in C*, denoted by $N_C(v)$, be the vertices $u \in V(G)$ with the property that C is satisfied if v is matched to u, that is,

$$N_C(v) = \{u \in V \mid e = \{u, v\} \in E \text{ and } \rho_{\{e\}}(C) = 1\} \;. \tag{21}$$

For a set $V \subseteq V(G)$ let $N_C(V)$ be the union of the clause-neighbourhoods of the vertices in V, i.e., $N_C(V) = \bigcup_{v\in V} N_C(v)$ and let the *vth vertex degree of C* be

$$\Delta_C(v) = |N_C(v)| \;. \tag{22}$$

We think of a vertex v with large degree $\Delta_C(v)$ as a vertex on which the derivation has not made any progress up to the point of deriving C, since the clause rules out very few neighbours. The vertices with high enough vertex degree in a clause are the *heavy vertices* of the clause as defined next.

▶ **Definition 21** (Vertex weight, pseudo-width and $(w_0, \vec{d})$-axioms). *Let $\vec{d} = (d_1, \ldots, d_{m+n})$ and $\vec{\delta} = (\delta_1, \ldots, \delta_{m+n})$ be two vectors such that $\vec{d}$ is elementwise greater than $\vec{\delta}$. We say that a vertex v is* $\vec{d}$-super-heavy for C *if $\Delta_C(v) \geq d_v$ and that vertex v is* $(\vec{d}, \vec{\delta})$-heavy for C *if $\Delta_C(v) \geq d_v - \delta_v$. When $\vec{d}$ and $\vec{\delta}$ are understood from context we omit the parameters and just refer to* super-heavy *and* heavy *vertices. Vertices that are not heavy are referred to as* light vertices. *The set of vertices that are super-heavy for C is denoted by*

$$V_{\vec{d}}(C) = \{v \in V \mid \Delta_C(v) \geq d_v\} \tag{23}$$

and the set of heavy vertices for C is denoted by

$$V_{\vec{d},\vec{\delta}}(C) = \{v \in V \mid \Delta_C(v) \geq d_v - \delta_v\} \;. \tag{24}$$

The pseudo-width $w_{\vec{d},\vec{\delta}}(C) = |V_{\vec{d},\vec{\delta}}(C)|$ *of a clause* C *is the number of heavy vertices in it, and the pseudo-width of a resolution refutation* π *is* $w_{\vec{d},\vec{\delta}}(\pi) = \max_{C\in\pi} w_{\vec{d},\vec{\delta}}(C)$*. We refer to clauses* C *with precisely* w_0 *super-heavy vertices as* $(w_0, \vec{d})$-axioms.

To a large extent, the proof of the lower bounds for perfect matching formulas follows the general idea of the proof of Theorem 14: given a short refutation we first apply the filter lemma to obtain a refutation of small pseudo-width; we then prove that in small pseudo-width contradiction cannot be derived and can thus conclude that no short refutation exists. In more detail, given a short resolution refutation π, we use the filter lemma (Lemma 7) to get a filter vector $\vec{d} = (d_1, \ldots, d_{m+n})$ such that each clause either has many super-heavy vertices or not too many heavy vertices (for an appropriately chosen vector $\vec{\delta}$). Clearly, clauses that fall into the second case of the filter lemma have bounded pseudo-width. Clauses in the first case, however, may have very large pseudo-width. In order to obtain a proof of low pseudo-width, these latter clauses are strengthened to $(w_0, \vec{d})$-axioms and added to a special set $\mathcal{A}$. This then gives a refutation π' that refutes the formula $PM(G) \cup \mathcal{A}$ in bounded pseudo-width as stated in the next lemma.

▶ **Lemma 22.** *Let* $G = (V_L \dot\cup V_R, E)$ *be a bipartite graph with* $|V_L| = m$ *and* $|V_R| = n$*; let* π *be a resolution refutation of* $PM(G)$*; let* $w_0, \alpha \in [m+n]$ *be such that* $w_0 > \log L(\pi)$ *and* $w_0 \geq \alpha^2 \geq 4$*, and let* $\vec{\delta} = (\delta_1, \ldots, \delta_{m+n})$ *be defined by* $\delta_v = \frac{\Delta_G(v)\log\alpha}{\log(m+n)}$ *for* $v \in V(G)$*. Then there exists an integer vector* $\vec{d} = (d_1, \ldots, d_{m+n})$*, with* $\delta_v < d_v \leq \Delta_G(v)$ *for all* $v \in V(G)$*, a set of* $(w_0, \vec{d})$*-axioms* $\mathcal{A}$ *with* $|\mathcal{A}| \leq L(\pi)$*, and a resolution refutation* π' *of* $PM(G) \cup \mathcal{A}$ *such that* $L(\pi') \leq L(\pi)$ *and* $w_{\vec{d},\vec{\delta}}(\pi') \leq \mathrm{O}(\alpha \cdot w_0)$*.*

The proof of the above lemma is omitted as it is syntactically equivalent to the proof of Lemma 12. Until this point, we have almost mimicked the proof of Theorem 14. The main differences will appear in the proof of the counterpart to Lemma 22, which states a pseudo-width lower bound.

▶ **Lemma 23.** *Assume for* $\xi \leq 1/64$ *and* $m, n, r, \Delta \in \mathbb{N}$ *that* $G = (V_L \dot\cup V_R, E)$ *is an* $(r, \Delta, (1-2\xi)\,\Delta)$*-boundary expander with* $|V_L| = m$*,* $|V_R| = n$*,* $\Delta \geq \log m/\xi^2$*, and* $\min\{\Delta_G(v) : v \in V_R\} \geq r/\xi$*. Let* $\vec{\delta} = (\delta_v \mid v \in V(G))$ *be defined by* $\delta_v = 64\Delta_G(v)\xi$ *and suppose that* $\vec{d} = (d_v \mid v \in V(G))$ *is an integer vector such that* $\delta_v < d_v \leq \Delta_G(v)$ *for all* $v \in V(G)$*. Fix* w_0 *such that* $64 \leq w_0 \leq r\xi - \log n$ *and let* $\mathcal{A}$ *be an arbitrary set of* $(w_0, \vec{d})$*-axioms with* $|\mathcal{A}| \leq (1+16\xi)^{w_0/8}$*. Then every resolution refutation* π *of* $PM(G) \cup \mathcal{A}$ *has either length* $L(\pi) \geq 2^{w_0/32}$ *or pseudo-width* $w_{\vec{d},\vec{\delta}}(\pi) \geq r\xi$*.*

The proof of the above lemma is based on a sort of reduction to the $FPHP(G)$ case. The idea, due to Razborov [28], is to first pick a partition of the vertices of G that looks random to every clause in the refutation and then simulate the $FPHP(G)$ lower bound on this partition. In our setting, however, this process gets quite involved. Already implementing the partition idea of Razborov is non-trivial: for a fixed clause C some vertices that are light may be super-heavy with respect to the partition, and we do not have an upper bound on the pseudo-width any longer. The insight needed to solve this issue is to show that by expansion there are not too many such vertices per clause, and then adapt the closure definition to take these vertices into account.

Another issue we run into is that the span argument from Section 4 cannot be applied to all the vertices in the graph. Instead, for the vertices in V_R, we need to resort to the span argument from [27]. Moreover, vertices in the neighbourhood of $\mathcal{D}$ (as defined in the proof of Lemma 20) may already be matched and we are hence unable to attain enough

matchings. Our solution is to consider a "lazy" edge removal procedure from the original matching, which with a careful analysis can be shown to circumvent the problem. We refer to the full-length version of this paper for the proof of Lemma 23.

5.1 Formal Statements of Perfect Matching Formula Lower Bounds

Let us state the lower bounds we obtain for the perfect matching formulas.

▶ **Theorem 24.** *Let $G = (U \dot\cup V, E)$ be a bipartite graph with $m = |U|$ and $n = |V|$. Suppose that G is an $(r, \Delta, (1-2\xi)\Delta)$-boundary expander for $\Delta \geq \frac{\log(m+n)}{\xi^2}$ and $\xi = \frac{\log \alpha}{64 \log(m+n)}$ where $\alpha \geq 2$ and $\frac{\alpha^3}{\log \alpha} = \mathrm{o}\left(\frac{r}{\log(m+n)}\right)$, which furthermore satisfies the degree requirement $\min\{\Delta_G(v) : v \in V\} \geq r/\xi$. Then resolution requires length $\exp\left(\Omega\left(\frac{r \log^2 \alpha}{\alpha \log^2(m+n)}\right)\right)$ to refute the perfect matching formula $PM(G)$ defined over G.*

We remark that this theorem also holds if we replace the minimum degree constraint of V with an expansion guarantee from V to U. We state the theorem in the above form as we want to apply it to the graphs from [18] for which we have no expansion guarantee from V to U.

Proof of Theorem 24. Let $w_0 = \frac{\varepsilon_0 r \xi}{\alpha}$, for some small enough $\varepsilon_0 > 0$. Suppose for the sake of contradiction that π is a resolution refutation of $PM(G)$ such that $L(\pi) < (1+16\xi)^{w_0/8}$. Since $w_0 > \log L(\pi)$, by Lemma 22 we have that there exists an integer vector $\vec{d} = (d_1, \ldots, d_{m+n})$, with $\delta_v < d_v \leq \Delta_G(v)$, a set of $(w_0, \vec{d})$-axioms $\mathcal{A}$ with $|\mathcal{A}| \leq L(\pi) < (1+16\xi)^{w_0/8}$, and a resolution refutation π' of $PM(G) \cup \mathcal{A}$ such that $L(\pi') \leq L(\pi)$ and $w_{\vec{d},\vec{\delta}}(\pi') \leq K\alpha w_0$ for some large enough constant K. Since $L(\pi') < (1+16\xi)^{w_0/8} \leq 2^{w_0/32}$, by Lemma 23, we have that $w_{\vec{d},\vec{\delta}}(\pi') \geq r\xi \geq \alpha w_0/\varepsilon_0$. Choosing $\varepsilon_0 < 1/K$, we get a contradiction and, thus, $L(\pi) \geq (1+16\xi)^{w_0/8} = \exp\left(\Omega\left(\frac{r\xi^2}{\alpha}\right)\right)$. ◀

As in Section 4, we have a general statement for random graphs.

▶ **Corollary 25.** *Let m and n be positive integers, let $\Delta : \mathbb{N}^+ \to \mathbb{N}^+$ and $\varepsilon : \mathbb{N}^+ \to [0,1]$ be any monotone functions of n such that $n^3 < m \leq n^{(\varepsilon/128)^2 \log n}$ and $n \geq \Delta \geq \log(m+n)\left(\frac{128\log(m+n)}{\varepsilon \log n}\right)^2$. Then asymptotically almost surely resolution requires length $\exp\left(\Omega\left(n^{1-\varepsilon}\right)\right)$ to refute $PM(G)$ for $G \sim \mathcal{G}\left(m, n, \Delta\right)$.*

Proof sketch. It suffices to prove the claim for $m = n^{(\varepsilon/128)^2 \log n}$ and $\Delta = \log(m+n) \cdot \left((128\log(m+n))/(\varepsilon \log n)\right)^2$. By applying Lemma 4 for $\chi = \alpha = n^{\varepsilon/4}$ and $\xi = \frac{\log \alpha}{64 \log m}$, we conclude that asymptotically almost surely, $G \sim \mathcal{G}\left(m, n, \Delta\right)$ is an $\left(n^{1-\varepsilon/2}, \Delta, (1-2\xi)\Delta\right)$-boundary expander. Furthermore, by the Chernoff inequality asymptotically almost surely all right vertices have degree at least $n \cdot \frac{64\log(m+n)}{\varepsilon \log n}$. Thus, Theorem 24 gives a length lower bound of $\exp\left(\Omega\left(n^{1-\varepsilon}\right)\right)$ as claimed. ◀

The following corollary is a simple consequence of Corollary 25, optimizing for the strongest lower bounds.

▶ **Corollary 26** (Restatement of Theorem 1)**.** *Let m, n be such that $m \leq n^{\mathrm{o}(\log n)}$. Then asymptotically almost surely resolution requires length $\exp\left(\Omega\left(n^{1-\mathrm{o}(1)}\right)\right)$ to refute $PM(G)$ for $G \sim \mathcal{G}\left(m, n, 8\log^2 m\right)$.*

Proof. Let $m = n^{f(n)}$, where $f(n) = o(\log n)$. Applying Corollary 25 for $\varepsilon = 128\sqrt{\frac{f(n)}{\log n}} = o(1)$, we get the desired statement. ◀

Our final corollary shows that we even get meaningful lower bounds for highly unbalanced bipartite graphs. As was the case for $FPHP(G)$, the required expansion is too strong to hold for random graphs with such large imbalance, but does hold for explicitly constructed graphs from [18].

▶ **Corollary 27** (Restatement of Theorem 2). *Let $\kappa < 3/2 - \sqrt{2}$ and $\varepsilon > 0$ be constants, and let n be an integer. Then there is a family of (explicitly constructible) graphs G with $m = 2^{\Omega(n^\kappa)}$ and left degree $O(\log^{1/\sqrt{\kappa}}(m))$, such that resolution requires length $\exp(\Omega(n^{1-2\sqrt{\kappa}(2-\sqrt{\kappa})-\varepsilon}))$ to refute $PM(G)$.*

Proof. Let G be the graph from Corollary 6 with $\nu = \frac{2\sqrt{\kappa}}{1-2\sqrt{\kappa}}$. In order to apply Theorem 24 we need to satisfy the minimum right degree constraint. A simple way of doing this is by adding n^2 edges to G such that each vertex on the right has exactly n incident edges added while each vertex on the left has at most one incident edge added. This will leave us with a graph which has large enough right degree while each left degree increased by at most one. The additional edges may reduce the boundary expansion a bit, but a short calculation shows that by choosing $\xi = \frac{\log \alpha}{128 \log(m+n)}$ in Corollary 6, we can still guarantee the needed boundary expansion for Theorem 24. The corollary bound follows. ◀

6 Concluding Remarks

In this work, we extend the pseudo-width method developed by Razborov [27, 28] for proving lower bounds on severely overconstrained CNF formulas in resolution. In particular, we establish that pigeonhole principle formulas and perfect matching formulas over highly unbalanced bipartite graphs remain exponentially hard for resolution even when these graphs are sparse. This resolves an open problem in [28].

The main technical difference in our work compared to [27, 28] goes right to the heart of the proof, where one wants to argue that resolution in small pseudo-width cannot make progress towards a derivation of contradiction. Here Razborov uses the global symmetry properties of the formula, whereas we resort to a local argument based on graph expansion. This argument needs to be carefully combined with a graph closure operation as in [4, 3] to ensure that the residual graph always remains expanding as matched pigeons and their neighbouring holes are removed. It is this change of perspective that allows us to prove lower bounds for sparse bipartite graphs with the size m of the left-hand side (i.e., the number of pigeons) varying all the way from linear to exponential in the size n of the right-hand size (i.e., the number of pigeonholes), thus covering the full range between [8] on the one hand and [23, 27, 28] on the other.

One shortcoming of our approach is that the sparse expander graphs are required to have very good expansion – for graphs of left degree Δ, the size of the set of unique neighbours of any not too large left vertex set has to scale like $(1 - o(1))\Delta$. We would like to prove that graph PHP formulas are hard also for graphs with constant expansion $(1 - \varepsilon)\Delta$ for some $\varepsilon > 0$, but there appear to be fundamental barriers to extending our lower bound proof to this setting.

Another intriguing problem left over from [28] is to determine the true resolution complexity of weak PHP formulas over complete bipartite graphs $K_{m,n}$ as $m \to \infty$. The best known upper bound from [11] is $\exp\big(O\big(\sqrt{n \log n}\big)\big)$, whereas the lower bound in [27, 28] is

$\exp\bigl(\Omega\bigl(\sqrt[3]{n}\bigr)\bigr)$. It does not seem unreasonable to hypothesize that $\exp\bigl(\Omega\bigl(\sqrt[2]{n}\bigr)\bigr)$ should be the correct lower bound (ignoring lower-order terms), but establishing such a lower bound again appears to require substantial new ideas.

We believe that one of the main contributions of our work is that it again demonstrates the power of Razborov's pseudo-width method, and we are currently optimistic that it could be useful for solving other open problems for resolution and other proof systems.

For resolution, an interesting question mentioned in [28] is whether pseudo-width can be useful to prove lower bounds for formulas that encode the Nisan–Wigderson generator [3, 29]. Since the clauses in such formulas encode local constraints, we hope that techniques from our paper could be helpful. Another long-standing open problem is to prove lower bounds on proofs in resolution that k-clique free sparse graph do not contain k-cliques, where the expected length lower bound would be $n^{\Omega(k)}$. Here we only know weakly exponential lower bounds for quite dense random graphs [6, 21], although an asymptotically optimal $n^{\Omega(k)}$ lower bound has been established in the sparse regime for the restricted subsystem of regular resolution [5].

Finally, we want to highlight that for the stronger proof system *polynomial calculus* [2, 14] no lower bounds on proof size are known for PHP formulas with $m \geq n^2$ pigeons. It would be very interesting if some kind of "pseudo-degree" method could be developed that would finally lead to progress on this problem.

References

1 Michael Alekhnovich. Mutilated chessboard problem is exponentially hard for resolution. *Theoretical Computer Science*, 310(1–3):513–525, January 2004.

2 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version in *STOC '00*.

3 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004. Preliminary version in *FOCS '00*.

4 Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at `http://people.cs.uchicago.edu/~razborov/files/misha.pdf`. Preliminary version in *FOCS '01*.

5 Albert Atserias, Ilario Bonacina, Susanna F. de Rezende, Massimo Lauria, Jakob Nordström, and Alexander Razborov. Clique is hard on average for regular resolution. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC '18)*, pages 866–877, June 2018.

6 Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. The resolution complexity of independent sets and vertex covers in random graphs. *Computational Complexity*, 16(3):245–297, October 2007. Preliminary version in *CCC '01*.

7 Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS '96)*, pages 274–282, October 1996.

8 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

9 Olaf Beyersdorff, Nicola Galesi, and Massimo Lauria. A lower bound for the pigeonhole principle in tree-like resolution by asymmetric prover-delayer games. *Information Processing Letters*, 110(23):1074–1077, 2010. `doi:10.1016/j.ipl.2010.09.007`.

10 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

11 Samuel R. Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In *11th International Workshop on Computer Science Logic (CSL '97), Selected Papers*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer, August 1997.

12 Samuel R. Buss and Győrgy Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62(3):311–317, December 1988.

13 Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

14 Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, May 1996.

15 Stefan S. Dantchev. Resolution width-size trade-offs for the pigeon-hole principle. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity (CCC '02)*, pages 39–43, May 2002. doi:10.1109/CCC.2002.1004337.

16 Stefan S. Dantchev and Søren Riis. "Planar" tautologies hard for resolution. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 220–229, October 2001.

17 Stefan S. Dantchev and Søren Riis. Tree resolution proofs of the weak pigeon-hole principle. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity (CCC '01)*, pages 69–75, June 2001. doi:10.1109/CCC.2001.933873.

18 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. *Journal of the ACM*, 56(4):20:1–20:34, July 2009. Preliminary version in *CCC '07*.

19 Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

20 Dmitry Itsykson, Vsevolod Oparin, Mikhail Slabodkin, and Dmitry Sokolov. Tight lower bounds on the resolution complexity of perfect matching principles. *Fundamenta Informaticae*, 145(3):229–242, August 2016. doi:10.3233/FI-2016-1358.

21 Shuo Pang. Large clique is hard on average for resolution. Technical Report TR19-068, Electronic Colloquium on Computational Complexity (ECCC), April 2019.

22 Toniann Pitassi and Ran Raz. Regular resolution lower bounds for the weak pigeonhole principle. *Combinatorica*, 24(3):503–524, 2004. Preliminary version in *STOC '01*. doi:10.1007/s00493-004-0030-y.

23 Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *Journal of the ACM*, 51(2):115–138, March 2004. Preliminary version in *STOC '02*.

24 Alexander A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, December 1998.

25 Alexander A. Razborov. Improved resolution lower bounds for the weak pigeonhole principle. Technical Report TR01-055, Electronic Colloquium on Computational Complexity (ECCC), July 2001.

26 Alexander A. Razborov. Proof complexity of pigeonhole principles. In *5th International Conference on Developments in Language Theory, (DLT '01), Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*, pages 100–116. Springer, July 2002.

27 Alexander A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theoretical Computer Science*, 1(303):233–243, June 2003.

28 Alexander A. Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69(1):3–27, August 2004. Preliminary version in *CCC '02*.

29 Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181(2):415–472, March 2015.

30 Alexander A. Razborov, Avi Wigderson, and Andrew Chi-Chih Yao. Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. *Combinatorica*, 22(4):555–574, 2002. Preliminary version in *STOC '97*. doi:10.1007/s00493-002-0007-7.

31 Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

32 Alasdair Urquhart. Resolution proofs of matching principles. *Annals of Mathematics and Artificial Intelligence*, 37(3):241–250, March 2003. doi:10.1023/A:1021231610627.

33 Alasdair Urquhart. Width versus size in resolution proofs. *Theoretical Computer Science*, 384(1):104–110, September 2007.

Groups with ALOGTIME-Hard Word Problems and PSPACE-Complete Circuit Value Problems

Laurent Bartholdi
ENS Lyon, Unité de Mathématiques Pures et Appliquées, France
Universität Göttingen, Mathematisches Institut, Germany
laurent.bartholdi@gmail.com

Michael Figelius
Universität Siegen, Germany
figelius@eti.uni-siegen.de

Markus Lohrey
Universität Siegen, Germany
lohrey@eti.uni-siegen.de

Armin Weiß
Universität Stuttgart, Institut für Formale Methoden der Informatik (FMI), Germany
armin.weiss@fmi.uni-stuttgart.de

Abstract

We give lower bounds on the complexity of the word problem of certain non-solvable groups: for a large class of non-solvable infinite groups, including in particular free groups, Grigorchuk's group and Thompson's groups, we prove that their word problem is ALOGTIME-hard. For some of these groups (including Grigorchuk's group and Thompson's groups) we prove that the circuit value problem (which is equivalent to the circuit evaluation problem) is PSPACE-complete.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Circuit complexity; Mathematics of computing → Combinatorics

Keywords and phrases NC^1-hardness, word problem, G-programs, straight-line programs, non-solvable groups, self-similar groups, Thompson's groups, Grigorchuk's group

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.29

Related Version For the full version see `https://arxiv.org/abs/1909.13781`.

Funding *Michael Figelius*: Funded by DFG project LO 748/12-1.
Markus Lohrey: Funded by DFG project LO 748/12-1.
Armin Weiß: Funded by DFG project DI 435/7-1.

Acknowledgements The authors are grateful to Schloss Dagstuhl and the organizers of Seminar 19131 for the invitation, where this work began.

1 Introduction

Groups and word problems. The *word problem* of a finitely generated group G is the most fundamental algorithmic problem in group theory [28, 42]. Recall that a group G with identity element 1 is finitely generated (f.g. for short) if there is a finite set $\Sigma \subseteq G$ such that every element of G can be written as a product of elements of Σ; this product can be formally written as a word from Σ^*. For technical reasons we assume that $1 \in \Sigma$ (which is needed for padding reasons) and that for every $a \in \Sigma$ also the inverse a^{-1} belongs to Σ; such a generating set Σ is called *standard*. We have a natural involution on Σ^* defined by $(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1}$ for $a_i \in \Sigma$ (which is the same as forming inverses in the group). For words $u, v \in \Sigma^*$ we write $u =_G v$ if u and v are equal in G; sometimes we just say $u = v$

in G. The word problem for G, $\mathrm{WP}(G)$ for short, is the question whether $u =_G 1$ holds for a given word $u \in \Sigma^*$. In this formulation the word problem depends on the generating set Σ, but it is well-known that the complexity/decidability status of the word problem does not depend on Σ.

The original motivation for the word problem came from topology and group theory [14] within Hilbert's "Entscheidungsproblem". Nevertheless, it also played a role in early computer science when Novikov and Boone constructed finitely presented groups with an undecidable word problem [10, 40]. Still, in many classes of groups it is (efficiently) decidable, a prominent example being the class of linear groups: Lipton and Zalcstein [36] (for linear groups over a field of characteristic zero) and Simon [44] (for linear groups over a field of prime characteristic) showed that their word problem is in LOGSPACE. A striking connection between the word problem for groups and complexity theory was established by Barrington [3]: for every finite non-solvable group G, the word problem of G is complete for ALOGTIME, which is the same as DLOGTIME-uniform NC^1. Moreover, the reduction is as simple as it could be: every output bit depends on only one input bit. Thus, one can say that ALOGTIME is completely characterized via group theory. Moreover, this idea has been extended to characterize ACC^0 by solvable monoids [4]. On the other hand, the word problem of a finite p-group is in $\mathsf{ACC}^0[p]$, so Smolensky's lower bound [45] implies that it is strictly easier than the word problem of a finite non-solvable group.

Barrington's construction is based on the observation that an and-gate can be simulated by a commutator. This explains the connection to non-solvability. In this light, it seems natural that the word problem of finite p-groups is not ALOGTIME-hard: they are all nilpotent, so iterated commutators eventually become trivial. For infinite groups, a construction similar to Barrington's was used by Robinson [41] to show that the word problem of a non-abelian free group is ALOGTIME-hard. Since by [36] the word problem of a free group is in LOGSPACE, the complexity is narrowed down quite precisely (although no completeness is known).

Strongly efficiently non-solvable groups and ALOGTIME. The first contribution of this paper is to identify the essence of Barrington's and Robinson's constructions. For this we introduce a strengthened condition of non-solvability. Here $[h, g] = h^{-1}g^{-1}hg$ denotes the *commutator* of h and g.

▶ **Definition 1.** *We call a group G with the finite standard generating set Σ* uniformly strongly efficiently non-solvable (uniformly SENS) *if there is a constant $\mu \in \mathbb{N}$ and words $g_{d,v} \in \Sigma^*$ for all $d \in \mathbb{N}$, $v \in \{0,1\}^{\leq d}$ such that*

(a) *$|g_{d,v}| = 2^{\mu d}$ for all $v \in \{0,1\}^d$,*

(b) *$g_{d,v} = \left[g_{d,v0},\, g_{d,v1}\right]$ for all $v \in \{0,1\}^{<d}$ (here we take the commutator of words),*

(c) *$g_{d,\varepsilon} \neq 1$ in G, and*

(d) *given $v \in \{0,1\}^d$, a positive integer i encoded in binary with μd bits, and $a \in \Sigma$ one can decide in* DLINTIME *(see Section 3 for a definition of* DLINTIME*) whether the i-th letter of $g_{d,v}$ is a.*

If G is required to only satisfy (a)–(c), then G is called SENS.

In a SENS group G, non-solvability is witnessed by efficiently computable balanced nested commutators of arbitrary depth that are non-trivial in G. The class of (uniformly) SENS groups enjoys several nice properties: in particular, the definition is independent of the choice of the generating set, it is inherited from subquotients (quotients of subgroups) and it is preserved under forming the quotient by the center of a group (see Lemmas 12–14). By following Barrington's arguments we show:

▶ **Theorem 2.** *Let G be uniformly SENS. Then WP(G) is hard for* ALOGTIME *under* DLOGTIME*-reductions (and also* DLOGTIME*-uniform projection reductions or* AC^0*-reductions).*

That means that for every non-solvable group G, the word problem for G is ALOGTIME-hard, unless the word length of the G-elements witnessing the non-solvability grows very fast (in the full version [5] we give an example of a non-solvable group where the latter happens) or these elements cannot be computed efficiently. For Theorem 2 the padding letter 1 in the generating set for G is important; otherwise, we only get a TC^0-many-one reduction.

Examples of SENS groups. Finite non-solvable groups and non-abelian free groups are easily seen to be uniformly SENS; this was essentially shown by Barrington (for finite non-solvable groups) and Robinson (for non-abelian free groups) in their ALOGTIME-hardness proofs for the word problem. We go beyond these classes and present in the full version [5] a general criterion that implies the uniform SENS-condition. As an application, we can show ALOGTIME-hardness of the word problems for several famous groups:

▶ **Corollary 3.** *The word problems for the following groups are hard for* ALOGTIME*:*

- *Thompson's groups,*
- *weakly branched self-similar groups with a finitely generated branching subgroup.*

Thompson's groups $F < T < V$ (introduced in 1965) belong due to their unusual properties to the most intensively studied infinite groups. From a computational perspective it is interesting to note that all three Thompson's groups are co-context-free (i.e., the set of all non-trivial words over any set of generators is a context-free language) [33]. This implies that the word problems for Thompson's groups are in LOGCFL. To the best of our knowledge no better upper complexity bound is known. Weakly branched groups form an important subclass of the self-similar groups [39], containing several celebrated groups like the Grigorchuk group (the first example of a group with intermediate word growth) and the Gupta-Sidki groups. We also show that the word problem for so-called contracting self-similar groups is in LOGSPACE. This result is well-known, but to the best of our knowledge no proof has appeared in the literature. The Grigorchuk group as well as the Gupta-Sidki groups are known to be contracting and have finitely generated branching subgroups, so Corollary 3 leaves only a small range for the complexity of their word problems.

The proof of the general result implying Corollary 3 is deferred to the full version [5] (we give a sketch in Section 4). Nevertheless, in this work we present direct proofs that Thompson's groups F and the Grigorchuk group are SENS, which yields Corollary 3 for these special cases.

In [31, Theorem 7], König and the third author showed that the word problem of f.g. solvable linear group is in TC^0. They asked the question whether there is a dichotomy in the sense that the word problem of a linear group either is in TC^0 or ALOGTIME-hard. As another application of the SENS condition, we can answer this question affirmatively using the famous Tits' alternative [46]:

▶ **Corollary 4.** *For every f.g. linear group the word problem either is in* DLOGTIME*-uniform* TC^0 *or the word problem is* ALOGTIME*-hard.*

Circuit value problems for groups. In the second part of the paper we study the *circuit value problem* for a finitely generated group G, CVP(G) for short. Fix a standard generating set Σ for G. The input for CVP(G) is a circuit in which input gates are labelled with generators

from Σ and every non-input gate computes the group-product of its two predecessor gates (we have to distinguish the left and right predecessor gate since G is in general not commutative). There is a distinguished output gate and the question is whether it evaluates to 1. In the group-theoretic literature, the circuit value problem for G is usually called the *compressed word problem* [38]. The reason for this is that one can evaluate a circuit $\mathcal{G}$ of the above form also in the free monoid Σ^*; it then corresponds to a context-free grammar that generates a single word w. The circuit $\mathcal{G}$ can be seen as a compressed representation of this word w. The circuit value problem is the succinct version of the word problem, where the input word is represented by a circuit.

Circuit value problems for finite groups have been studied in [8]. It was shown that $\mathrm{CVP}(G)$ is P-complete for finite non-solvable groups (by a Barrington style argument) and in NC^2 for finite solvable groups. The circuit value problem for linear groups is tightly related to PIT (polynomial identity testing, i.e., the question whether a circuit over a polynomial ring evaluates to the zero-polynomial; see e.g. [43]): For every f.g. linear group the circuit value problem reduces in polynomial time to PIT for $\mathbb{Z}[x]$ or $\mathbb{F}[x]$ and hence belongs to coRP, the complement of randomized polynomial time [38, Theorem 4.15]. Moreover, the circuit value problem for the group $\mathrm{SL}_3(\mathbb{Z})$ is equivalent to PIT for $\mathbb{Z}[x]$ with respect to polynomial time reductions [38, Theorem 4.16].

From a group theoretic viewpoint, the circuit value problem is interesting not only because it is a natural succinct version of the word problem, but also because several classical word problems efficiently reduce to circuit value problems. For instance, the word problem for a finitely generated subgroup of $\mathrm{Aut}(G)$ reduces in polynomial time to the circuit value problem for G [38, Theorem 4.6]. Similar statements hold for certain group extensions [38, Theorems 4.8 and 4.9]. This motivates the search for groups in which the circuit value problem can be solved in polynomial time. This applies to finitely generated nilpotent groups [31] (for which the circuit value problem can be even solved in NC^2), hyperbolic groups [27] and virtually special groups [38]. The latter are defined as finite extensions of subgroups of right-angled Artin groups and form a very rich class of groups containing for instance Coxeter groups [21], fully residually free groups [51] and fundamental groups of hyperbolic 3-manifolds [1].

Recently, Wächter and the fourth author constructed an automaton group (a finitely generated group of tree automorphism, where the action of generators is defined by a Mealy automaton) with a PSPACE-complete word problem and $\mathsf{EXPSPACE}$-complete circuit value problem [49]. The group arises by encoding a Turing machine into a group; in particular, one cannot call this group natural. In this paper, we exhibit several natural groups (that were intensively studied in other parts of mathematics) with a PSPACE-complete circuit value problem (and a word problem in $\mathsf{LOGSPACE}$). The two main ingredients for our construction is the uniform SENS-property defined above and the wreath product construction.

Circuit value problems for wreath products. The wreath product $G \wr H$ is a fundamental construction in group theory and semigroup theory; important applications are the Krasner–Kaloujnine embedding theorem in group theory and the Krohn-Rhodes decomposition theorem in semigroup theory. The formal definition of wreath products can be found in Section 2. We are interested in the circuit value problem for wreath products of the form $G \wr \mathbb{Z}$ (for G f.g.). Such groups are also called lamplighter groups (the classical lamplighter group is $(\mathbb{Z}/2) \wr \mathbb{Z}$). The following result was shown in [38] (for G non-abelian) and [32] (for G abelian via a reduction to polynomial identity testing).

▶ **Theorem 5** (c.f. [32, 38]). *If G is a f.g. group, then*
- *$CVP(G \wr \mathbb{Z})$ is coNP-hard if G is non-abelian and*
- *$CVP(G \wr \mathbb{Z})$ belongs to coRP (co-randomized polynomial time) if G is abelian.*

Our main result for the circuit value problem in wreath products pinpoints the exact complexity of $\mathrm{CVP}(G \wr \mathbb{Z})$ for the case that G has a trivial center (recall that the *center* $Z(G)$ of the group G is the normal subgroup consisting of all elements $g \in G$ that commute with every element from G). Theorem 6 below uses the concept of leaf languages [11, 23, 25, 26, 29], which is formally defined in Appendix A. For a language $K \subseteq \Gamma^*$ over a finite alphabet Γ one considers nondeterministic polynomial time machines M that after termination print a symbol from Γ on every computation path. Moreover, one fixes a linear ordering on the transition tuples of M. As a consequence the computation tree $T(x)$ for a machine input x becomes a finite ordered tree. The corresponding leaf string $\mathrm{leaf}(M, x)$ is obtained by listing symbols from Γ that are printed in the leafs of $T(x)$ from left to right. The class $\mathsf{LEAF}(K)$ consists of all languages L for which there exists a nondeterministic polynomial time machines as described above such that $x \in L$ if and only if $\mathrm{leaf}(M, x) \in K$. As a prototypical example note that $\mathsf{NP} = \mathsf{LEAF}(\{0,1\}^*1\{0,1\}^*)$. Here, we are interested in leaf language classes where K is the word problem for a f.g. group. For this we identify the word problem with the language $\mathrm{WP}(G, \Sigma) = \{w \in \Sigma^* \mid w =_G 1\}$. One can easily show that the generating set Σ has no influence on the class $\mathsf{LEAF}(\mathrm{WP}(G, \Sigma))$ (see Lemma 30 in the appendix). Hence, we simply write $\mathsf{LEAF}(\mathrm{WP}(G))$. We are actually interested in the class $\forall\mathsf{LEAF}(\mathrm{WP}(G))$, where for a complexity class C we denote by $\forall\mathsf{C}$ the class of all languages L such that there exists a polynomial $p(n)$ and a language $K \in \mathsf{C}$ with $L = \{u \mid \forall v \in \{0,1\}^{p(|u|)} : u\#v \in K\}$ (hence, for instance $\forall\mathsf{P} = \mathsf{coNP}$ and $\forall\mathsf{PSPACE} = \mathsf{PSPACE}$). Our main result for the circuit value problem in wreath products is:

▶ **Theorem 6.** *Let G be a f.g. non-trivial group with center $Z = Z(G)$.*

- *$CVP(G \wr \mathbb{Z})$ belongs to $\forall\mathsf{LEAF}(WP(G))$.*
- *$CVP(G \wr \mathbb{Z})$ is hard for the class $\forall\mathsf{LEAF}(WP(G/Z))$.*

In particular, if $Z = 1$, then $CVP(G \wr \mathbb{Z})$ is complete for $\forall\mathsf{LEAF}(WP(G))$.

PSPACE-complete circuit value problems. From Theorem 6 we derive PSPACE-completeness of the circuit value problem for some interesting groups:

▶ **Corollary 7.** *The circuit value problem for the following groups is* PSPACE*-complete:*

(i) *wreath products $G \wr \mathbb{Z}$ where G is finite non-solvable or free of rank at least two,*

(ii) *Thompson's groups,*

(iii) *the Grigorchuk group, and*

(iv) *all Gupta-Sidki groups.*

In order to derive this corollary from Theorem 6 we also need a kind of padded version of Theorem 2 saying that PSPACE is contained in $\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$ (this yields PSPACE-hardness of $\mathrm{CVP}(G \wr \mathbb{Z})$ for every SENS group G). For Thompson's groups, the Grigorchuk group, and the Gupta-Sidki groups we also use a certain self-embedding property: for all these groups G a wreath product $G \wr A$ embeds into G for some $A \neq 1$. Thompson's group F has this property for $A = \mathbb{Z}$ [19]. For the Grigorchuk group and the Gupta-Sidki groups (and, more generally, weakly branched groups whose branching subgroup is finitely generated and has elements of finite order) we show that one can take $A = \mathbb{Z}/p$ for some $p \geq 2$.

Some of the proofs can be found only in the full version [5] of this paper. The proof of Theorem 6 can be found in the appendix.

2 Background from group theory

For group elements $g, h \in G$ or words $g, h \in \Sigma^*$ we write g^h for the *conjugate* $h^{-1}gh$ and $[h, g]$ for the *commutator* $h^{-1}g^{-1}hg$. We call g a *d-fold nested commutator*, if $d = 0$ or $g = [h_1, h_2]$ for $(d-1)$-fold nested commutators h_1, h_2. A *subquotient* of a group G is a quotient of a subgroup of G.

Wreath products. We consider groups G that act on a set X on the left or right. For $g \in G$ and $x \in X$ we write $x^g \in X$ (resp., ${}^g x$) for the result of a right (resp., left) action. An important case arises when $G = \mathrm{Sym}(X)$ is the symmetric group on a set X, which acts on X on the right.

A fundamental group construction that we shall use is the *wreath product*: given groups G and H acting on the right on sets X and Y respectively, their *wreath product* $G \wr H$ is a group acting on $X \times Y$. We start with the restricted direct product $G^{(Y)}$ (the base group) of all mappings $f : Y \to G$ having finite support $\mathrm{supp}(f) = \{y \mid f(y) \neq 1\}$ with the operation of pointwise multiplication. The group H has a natural left action on $G^{(Y)}$: for $f \in G^{(Y)}$ and $h \in H$, we define ${}^h f \in G^{(Y)}$ by $({}^h f)(y) = f(y^h)$. The corresponding semidirect product $G^{(Y)} \rtimes H$ is the *wreath product* $G \wr H$. In other words:

- Elements of $G \wr H$ are pairs $(f, h) \in G^{(Y)} \times H$; we simply write fh for this pair.
- The multiplication in $G \wr H$ is defined as follows: Let $f_1 h_1, f_2 h_2 \in G \wr H$. Then $f_1 h_1 f_2 h_2 = f_1 {}^{h_1} f_2 h_1 h_2$, where the product $f_1 {}^{h_1} f_2 \colon y \mapsto f_1(y) f_2(y^{h_1})$ is the pointwise product.

The wreath product $G \wr H$ acts on $X \times Y$ by $(x, y)^{fh} = (x^{f(y)}, y^h)$. The wreath product defined above is also called the *(restricted) permutational wreath product*. There is also the variant where $G = X$, $H = Y$ and both groups act on themselves by right-multiplication, which is called the *(restricted) regular wreath product* (or *standard wreath product*). A subtle point is that the permutational wreath product is an associative operation whereas the regular wreath product is in general not. The term "restricted" refers to the fact that the base group is $G^{(Y)}$, i.e., only finitely supported mappings are taken into account. If $G^{(Y)}$ is replaced by G^Y (i.e., the set of all mappings from Y to G with pointwise multiplication), then one speaks of an unrestricted wreath product. For Y finite this makes of course no difference. We will only deal with restricted wreath products. The action of G on X is usually not important for us, but it is nice to have an associative operation. The right group H will be either a symmetric group $\mathrm{Sym}(Y)$ acting on the right on Y or a (finite or infinite) cyclic group acting on itself by $g^h = g + h$. Thus, if H is cyclic, the permutational wreath product and the regular wreath product (both denoted by $G \wr H$) coincide. Nevertheless, be aware that $G \wr (H \wr H) = (G \wr H) \wr H$ holds only for the permutational wreath product even if H is cyclic. Note that if G is generated by Σ and H is generated by Γ then $G \wr H$ is generated by $\Sigma \cup \Gamma$.

Richard Thompson's groups. In 1965 Richard Thompson introduced three finitely presented groups $F < T < V$ acting on the unit-interval, the unit-circle and the Cantor set, respectively. Of these three groups, F received most attention (the reader should not confuse F with a free group). This is mainly due to the still open conjecture that F is not amenable, which would imply that F is another counterexample to a famous conjecture of von Neumann (a counterexample was found by Ol'shanskii). A standard reference for Thompson's groups is [12]. The group F consists of all homeomorphisms of the unit interval that are piecewise affine, with slopes a power of 2 and dyadic breakpoints. Famously, F has a finite presentation with two generators: $F = \langle x_0, x_1 \mid [x_0 x_1^{-1}, x_0^{-1} x_1 x_0], [x_0 x_1^{-1}, x_0^{-2} x_1 x_0^2] \rangle$. Very convenient is

also the following infinite presentation: $F = \langle x_0, x_1, x_2, \dots \mid x_k^{x_i} = x_{k+1} (i < k) \rangle$. The group F is orderable (so in particular torsion-free), its derived subgroup $[F, F]$ is simple and the center of F is trivial. Important for us is the following fact:

▶ **Lemma 8** ([19, Lemma 20]). *The group F contains a subgroup isomorphic to $F \wr \mathbb{Z}$.*

Hence, the limit group $H_\infty = \bigcup_{i \geq 0} H_i$, where $H_0 = \mathbb{Z}$ and $H_{i+1} = H_i \wr \mathbb{Z}$, is contained in F.

Weakly branched groups. We continue our list of examples with an important class of groups acting on rooted trees. For more details, [6, 39] serve as good references. Let X be a finite set. The free monoid X^* serves as the vertex set of a regular rooted tree with an edge between v and vx for all $v \in X^*$ and all $x \in X$. The group W of automorphisms of this tree naturally acts on the set X of level-1 vertices, and permutes the subtrees hanging from them. Exploiting the bijection $X^+ = X^* \times X$, we thus have an isomorphism

$$\varphi \colon W \to W \wr \mathrm{Sym}(X) = W^X \rtimes \mathrm{Sym}(X), \tag{1}$$

mapping $g \in W$ to elements $f \in W^X$ and $\pi \in \mathrm{Sym}(X)$ as follows: π is the restriction of g to $X \subseteq X^*$, and f is uniquely defined by $(xv)^g = x^\pi v^{f(x)}$. We always write $g@x$ for $f(x)$ and call it the *state (or coordinate) of g at x*. If $X = \{0, \dots, k\}$ we write $g = \langle\!\langle g@0, \dots, g@k \rangle\!\rangle \pi$.

▶ **Definition 9.** *A subgroup $G \leq W$ is* self-similar *if $\varphi(G) \leq G \wr \mathrm{Sym}(X)$. In other words: the actions on subtrees xX^* are given by elements of G itself. A self-similar group G is* weakly branched *if there exists a non-trivial subgroup $K \leq G$ with $\varphi(K) \geq K^X$. In other words: for every $k \in K$ and every $x \in X$ the element acting as k on the subtree xX^* and trivially elsewhere belongs to K. A subgroup K as above is called a* branching subgroup.

Note that we are weakening the usual definition of "weakly branched": indeed it is usually additionally required that G act transitively on X^n for all $n \in \mathbb{N}$. This extra property is not necessary for our purposes, so we elect to simply ignore it. In fact, all the results concerning branched groups that we shall use will be proven directly from Definition 9.

Note also that the join $\langle K_1 \cup K_2 \rangle$ of two branching subgroups K_1 and K_2 is again a branching subgroup. Hence, there exists a maximal branching subgroup. It immediately follows from the definition that, if G is weakly branched, then for every $v \in X^*$ there is in G a copy of its branching subgroup K whose action is concentrated on the subtree vX^*.

There exist important examples of f.g. self-similar weakly branched groups, notably the *Grigorchuk group* G, see [17]. It may be described as a self-similar group in the following manner: it is a group generated by $\{a, b, c, d\}$, and acts on the rooted tree X^* for $X = \{0, 1\}$. The action, and therefore the whole group, are defined by the restriction of φ to G's generators: $\varphi(a) = (0, 1)$, $\varphi(b) = \langle\!\langle a, c \rangle\!\rangle$, $\varphi(c) = \langle\!\langle a, d \rangle\!\rangle$, and $\varphi(d) = \langle\!\langle 1, b \rangle\!\rangle$, where we use the notation $(0, 1)$ for the non-trivial element of $\mathrm{Sym}(X)$ (that permutes 0 and 1) and $\langle\!\langle w_0, w_1 \rangle\!\rangle$ for a tuple in $G^{\{0,1\}} \cong G \times G$. We record some classical facts:

▶ **Lemma 10.** *The Grigorchuk group G is infinite, torsion, weakly branched, and all its finite subquotients are 2-groups (so in particular nilpotent). It has a f.g. branching subgroup.*

Other examples of f.g. self-similar weakly branched groups with a f.g. branching subgroup include the Gupta-Sidki groups [20], the Hanoi tower groups [18], and all iterated monodromy groups of degree-2 complex polynomials [7] except z^2 and $z^2 - 2$.

Contracting self-similar groups. Recall the notation $g@x$ for the coordinates of $\varphi(g)$. We iteratively define $g@v = g@x_1 \cdots @x_n$ for any word $v = x_1 \cdots x_n \in X^*$. A self-similar group G is called *contracting* if there is a finite subset $N \subseteq G$ such that, for all $g \in G$, we have $g@v \in N$ whenever v is long enough (depending on g), see also [39, Definition 2.11.1].

If G is a f.g. contracting group with word norm $\|\cdot\|$ (i.e., for $g \in G$, $\|g\|$ is the length of a shortest word over a fixed generating set of G that represents g), then a more quantitative property holds: there are constants $0 < \lambda < 1$, $h \geq 1$ and $k \geq 0$ such that for all $g \in G$ we have $\|g@v\| \leq \lambda\|g\| + k$ for all $v \in X^h$; see e.g. [28, Proposition 9.3.11]. Then, for $c = -h/\log\lambda$ and a possibly larger k we have $g@v \in N$ whenever $|v| \geq c\log\|g\| + k$. It is well-known and easy to check that the Grigorchuk group, the Gupta-Sidki groups and the Hanoi tower group for three pegs are contracting. The following result has been quoted numerous times, but has never appeared in print. We give a proof in the full version [5]. A proof for the Grigorchuk group may be found in [16]:

▶ **Proposition 11.** *Let G be a f.g. contracting self-similar group. Then $WP(G)$ can be solved in* LOGSPACE *(deterministic logarithmic space).*

For the proof of Proposition 11 one shows that if an element g of a contracting self-similar group G acts as the identity on all words $v \in X^*$ of length $\mathcal{O}(\log\|g\|)$, then $g = 1$.

3 Complexity theory

Since we also deal with sublinear time complexity classes, we use Turing machines with *random access*. Such a machine has an additional index tape and some special query states. Whenever the Turing machine enters a query state, the following transition depends on the input symbol at the position which is currently written on the index tape in binary notation. We use the abbreviations DTM/NTM/ATM for deterministic/non-deterministic/alternating Turing machine. We define the following complexity classes:

- DLINTIME: the class of languages that can be accepted by a DTM in linear time.
- DLOGTIME: the class of languages that can be accepted by a DTM in logarithmic time.
- ALOGTIME: the class of languages that can be accepted by an ATM in logarithmic time. It is well-known that ALOGTIME = DLOGTIME-uniform NC^1, see [47] for details.
- APTIME: the class of languages that can be accepted by an ATM in polynomial time. We have APTIME = PSPACE.

A function $f\colon \Gamma^* \to \Sigma^*$ is DLOGTIME-computable if there is some polynomial p with $|f(x)| \leq p(|x|)$ for all $x \in \Gamma^*$ and the set $L_f = \{(x, a, i) \mid x \in \Gamma^*$ and the i-th letter of $f(x)$ is $a\}$ belongs to DLOGTIME. Here i is a binary encoded integer. A DLOGTIME-reduction is a DLOGTIME-computable many-one reduction.

The class AC^0 (resp. TC^0) is defined as the class of languages (respectively functions) accepted (respectively computed) by circuits of constant depth and polynomial size with not-gates and unbounded fan-in and- and or-gates (resp. unbounded fan-in threshold-gates).

4 Efficiently non-solvable groups and ALOGTIME

Recall the definition of a SENS (strongly efficiently non-solvable) group from Definition 1; (a)–(d) refer to this definition in the following. We start with some observations:

- A SENS group is clearly non-solvable, so the terminology makes sense. In the full version [5] we give an example of a f.g. group that is non-solvable, has decidable word problem, but is not SENS. The construction is inspired from [50].

- If one can find suitable $g_{d,v}$ of length at most $2^{\mu d}$, then these words can always be padded to length $2^{\mu d}$ thanks to the padding letter 1.
- It suffices to specify $g_{d,v}$ for $v \in \{0,1\}^d$; the other $g_{d,v}$ are then defined by Condition (b).
- We have $|g_{d,v}| = 2^{\mu d + 2(d-|v|)}$ for all $v \in \{0,1\}^{\leq d}$. Thus, all $g_{d,v}$ have length $2^{\mathcal{O}(d)}$.
- Equivalently to Condition (d), we can require that given $v \in \{0,1\}^d$ and a binary encoded number i with μd bits, one can compute the i-th letter of $g_{d,v}$ in DLINTIME.

Proof sketch for Theorem 2. The proof of Theorem 2 essentially follows Barrington's proof that the word problem of finite non-solvable groups is ALOGTIME-hard [3]. The entire proof can be found in the full version [5], where we also state a non-uniform version of Theorem 2. Since the proof for the uniform case is not difficult but tedious, in this sketch we primarily focus on the non-uniform version, which only gives us hardness via (non-uniform) AC^0-reductions instead of DLOGTIME-reductions.

Like in Barrington's proof, we start with an ALOGTIME-machine (resp. NC^1-circuit) and construct a family of so-called G-programs. Since we are dealing with finitely generated, but infinite groups, we have to adapt the definition of G-programs slightly.

Fix a finite standard generating set Σ of G. A G-program P of length m and input length n is a sequence of *instructions* $\langle i_j, b_j, c_j\rangle$ for $0 \leq j \leq m-1$ where $i_j \in [1..n]$ and $b_j, c_j \in \Sigma$. On input of a word $x = x_1 \cdots x_n \in \{0,1\}^*$, an instruction $\langle i_j, b_j, c_j\rangle$ evaluates to b_j if $x_{i_j} = 1$ and to c_j otherwise. The evaluation of a G-program is the product (in the specified order) of the evaluations of its instructions, and is denoted with $P[x] \in \Sigma^*$.

Let M be an ALOGTIME-machine in input normal form [47, Lemma 2.41], i. e., every computation path queries at most one input bit and M halts immediately after the query. For every input size n, the computation tree of M translates immediately into a Boolean circuit of depth $d \in \mathcal{O}(\log n)$. Moreover, M can be normalized such that this circuit is a fully balanced binary tree meaning that the gates of the circuit are indexed by the set $\{0,1\}^{\leq d}$, where $\{0,1\}^{<d}$ are the inner gates (where ε is the output gate, which counts here as an inner gate) and $\{0,1\}^d$ are the leaves (input gates). We can assume that all inner gates are nand-gates (where the Boolean function $\mathsf{nand} : \{0,1\}^2 \to \{0,1\}$ is defined by $\mathsf{nand}(0,0) = \mathsf{nand}(0,1) = \mathsf{nand}(1,0) = 1$ and $\mathsf{nand}(1,1) = 0$) and each leaf is labelled by a possibly negated input variable or constant via an input mapping $q_n \colon \{0,1\}^d \to [1..n] \times \{0,1\} \times \{0,1\}$. The meaning of this mapping is as follows: if the input to the circuit is the bit string $x_1 x_2 \cdots x_n$ and $q_n(v) = \langle i, a, b\rangle$, then the input gate $v \in \{0,1\}^d$ evaluates to a (resp., b) if $x_i = 1$ (resp., $x_i = 0$).

The family of circuits obtained this way can be shown to be DLOGTIME-uniform in an even stronger sense than the usual definition (see e. g. [47]). For the sake of a simpler description, we fix the input length n and write C for the n-input circuit of depth $d \in \mathcal{O}(\log n)$. W. l. o. g. for every $x \in \{0,1\}^n$, we have $x \in L(M)$ if and only if the output gate of C evaluates to 0 on input x.

For each gate $v \in \{0,1\}^{\leq d}$, let $g_v = g_{d,v}$ as in Definition 1. We construct two G-programs P_v and P_v^{-1} (both of input length n) such that for every input $x \in \{0,1\}^n$ we have

$$P_v[x] =_G \begin{cases} g_v & \text{if gate } v \text{ evaluates to 1,} \\ 1 & \text{if gate } v \text{ evaluates to 0,} \end{cases} \tag{2}$$

and $P_v^{-1}[x] = P_v[x]^{-1}$ in G. Notice that $g_v P_v^{-1}[x] = g_v$ if v evaluates to 0 and $g_v P_v^{-1}[x] = 1$, otherwise. Thus, $g_v P_v^{-1}$ is a G-program for the "negation" of P_v. Moreover, by Equation (2), P_ε evaluates to 1 on input x if and only if the output gate evaluates to 0 which by our assumption is the case if and only if $x \in L$.

The construction of the P_v and P_v^{-1} is straightforward: For an input gate $v \in \{0,1\}^d$ with $q_n(v) = \langle i, a, b\rangle$ we define P_v to be a G-program evaluating to g_v or 1 depending on the i-th input bit. More precisely, write $g_v = a_1 \cdots a_m$ with $a_i \in \Sigma$. If $q_n(v) = \langle i, a, b\rangle$ for $i \in [1..n]$ and $a, b \in \{0,1\}$, we set $P_v = \langle i, a_1^a, a_1^b\rangle \cdots \langle i, a_m^a, a_m^b\rangle$ and $P_v^{-1} = \langle i, a_m^{-a}, a_m^{-b}\rangle \cdots \langle i, a_1^{-a}, a_1^{-b}\rangle$. For a nand-gate v with inputs from $v0$ and $v1$, we define

$$\begin{aligned} P_v &= g_v[P_{v1}, P_{v0}] = g_v P_{v1}^{-1} P_{v0}^{-1} P_{v1} P_{v0}, \\ P_v^{-1} &= [P_{v0}, P_{v1}] g_v^{-1} = P_{v0}^{-1} P_{v1}^{-1} P_{v1} P_{v1} g_v^{-1}, \end{aligned}$$

where the g_v and g_v^{-1} represent constant G-programs evaluating to g_v and g_v^{-1}, respectively, irrespective of the actual input (such constant G-programs consist of triples of the form $\langle 1, a, a\rangle$ for $a \in \Sigma$). These constant G-programs are defined via the commutator identities $g_v = \left[g_{v0}, g_{v1}\right]$ for $v \in \{0,1\}^{<d}$ in Definition 1.

Clearly, by induction we have $P_v[x]^{-1} = P_v^{-1}[x]$ in G (for every input x). Let us show that Equation (2) holds: For an input gate $v \in \{0,1\}^d$, Equation (2) holds by definition. Now, let $v \in \{0,1\}^{<d}$. Then, by induction, we have the following equalities in G:

$$\begin{aligned} P_v[x] = g_v[P_{v1}[x], P_{v0}[x]] &= \begin{cases} g_v & \text{if } v0 \text{ or } v1 \text{ evaluates to 0,} \\ g_v[g_{v1}, g_{v0}] & \text{if } v0 \text{ and } v1 \text{ evaluate to 1,} \end{cases} \\ &= \begin{cases} g_v & \text{if } v \text{ evaluates to 1,} \\ 1 & \text{if } v \text{ evaluates to 0.} \end{cases} \end{aligned}$$

Note that $[g_{v1}, g_{v0}] = [g_{v0}, g_{v1}]^{-1} = g_v^{-1}$ for the last equality. Thus, P_v satisfies Equation (2). For P_v^{-1} the analogous statement can be shown with the same calculation. For a leaf $v \in \{0,1\}^d$, we have $|g_v| \in 2^{\mathcal{O}(d)} = n^{\mathcal{O}(1)}$ by Condition (a) from Definition 1 (recall that $d \in \mathcal{O}(\log n)$). Hence, P_v^{-1} and P_v have polynomial length in n. Finally, also P_ε has polynomial length in n.

This gives us a non-uniform AC^0-reduction (more precisely, a projection reduction) of $L(M)$ to $\mathrm{WP}(G)$. In order to obtain a DLOGTIME-reduction, we apply essentially the same construction. However, we need to introduce some padding so that the function mapping an index i encoded in binary to the i-th instruction of P_ε can be computed in DLINTIME. In order to show the last point, we need condition (d) of the uniform SENS definition (Definition 1). ◀

Let us next state some algebraic properties of SENS groups. The proofs of the following three lemmas are straightforward.

▶ **Lemma 12.** *The property of being (uniformly) SENS is independent of the choice of the standard generating set.*

▶ **Lemma 13.** *If $Q = H/K$ is a f.g. subquotient of a f.g. group G and Q is (uniformly) SENS, then G is also (uniformly) SENS.*

▶ **Lemma 14.** *If G is (uniformly) SENS, then $G/Z(G)$ is (uniformly) SENS.*

The following result is, for $G = A_5$, the heart of Barrington's argument:

▶ **Lemma 15.** *If G is a finite non-solvable group, then G is uniformly SENS.*

Proof. Let us first show the statement for a non-abelian finite simple group G. By the proof of Ore's conjecture [34], every element of G is a commutator. This means that we may choose $g_\varepsilon \neq 1$ at will, and given g_v we define g_{v0}, g_{v1} by table lookup, having chosen once and for all for each element of G a representation of it as a commutator. Computing g_v requires $|v|$ steps and bounded memory.

If G is finite non-solvable, then any composition series of G contains a non-abelian simple composition factor G_i/G_{i+1}. Hence, we can apply Lemma 13. ◀

We remark that a direct proof of Lemma 15 without using the deep result [34] is also not difficult, but requires some more work.

By Lemmas 13 and 15, every group having a finite non-solvable subquotient is uniformly SENS. Since every free group of rank $n \geq 2$ projects to a finite non-solvable group, we get:

▶ **Corollary 16.** *A f.g. free group of rank $n \geq 2$ is uniformly SENS.*

This result was essentially shown by Robinson [41], who showed that the word problem of a free group of rank two is ALOGTIME-hard. He used a similar commutator approach as Barrington. One can prove Corollary 16 also directly by exhibiting a free subgroup of infinite rank whose generators are easily computable. For example, in the free group $F_2 = \langle x_0, x_1 \rangle$ take $g_{d,v} = x_0^{-v} x_1 x_0^v$ for $v \in \{0,1\}^d$ viewing the string v as a binary encoded number (the other $g_{d,v}$ for $v \in \{0,1\}^{<d}$ are then defined by the commutator identity in Definition 1), and appropriately padding with 1's. It is even possible to take the $g_{d,v}$ of constant length: consider a free group $F = \langle x_0, x_1, x_2 \rangle$ of rank 3 and the elements $g_{d,v} = x_{v \bmod 3}$ with v read as the binary representation of an integer. It is easy to see that the nested commutator $g_{d,\varepsilon}$ is non-trivial.

A dichotomy for linear groups. Instead of Corollary 4, we prove a slightly more detailed result. Recall that a group G is called C_1-by-C_2 for group classes C_1 and C_2 if G has a normal subgroup $K \in \mathsf{C}_1$ such that $G/K \in \mathsf{C}_2$.

▶ **Theorem 17.** *For every f.g. linear group the word problem is in* DLOGTIME*-uniform* TC^0 *or* ALOGTIME*-hard. More precisely: let G be a f.g. linear group.*

- *If G is finite solvable, then $WP(G)$ belongs to* DLOGTIME*-uniform* ACC^0*.*
- *If G is infinite solvable, then $WP(G)$ is complete for* DLOGTIME*-uniform* TC^0 *(via uniform* AC^0*-Turing-reductions).*
- *If G is solvable-by-(finite non-solvable), then $WP(G)$ is complete for* ALOGTIME *(via* DLOGTIME*-reductions).*
- *In all other cases, $WP(G)$ is* ALOGTIME*-hard and in* LOGSPACE*.*

Note that we can obtain a similar dichotomy for hyperbolic groups: they are either virtually abelian or contain a non-abelian free subgroup. In the first case, the word problem is in DLOGTIME-uniform TC^0, in the second case it is ALOGTIME-hard.

Proof. Let G be f.g. linear. First of all, by [36, 44], $\mathrm{WP}(G)$ belongs to LOGSPACE. By Tits alternative [46], G either contains a free subgroup of rank 2 or is virtually solvable. In the former case, $\mathrm{WP}(G)$ is ALOGTIME-hard by Corollary 16 and Theorem 2. Let us now assume that G is virtually solvable. Let K be a solvable subgroup of G of finite index. By taking the intersection of all conjugates of K in G, we can assume that K is a normal subgroup of G. If also G/K is solvable, then G is solvable. Hence, $\mathrm{WP}(G)$ is in DLOGTIME-uniform ACC^0 (if G is finite) or, by [31], complete for DLOGTIME-uniform TC^0 (if G is infinite). Finally, assume that the finite group G/K is non-solvable (thus, G is solvable-by-(finite non-solvable).

By Lemmas 13 and 15, G is uniformly SENS, and Theorem 2 implies that $\mathrm{WP}(G)$ is $\mathsf{ALOGTIME}$-hard. Moreover, by [41, Theorem 5.2], $\mathrm{WP}(G)$ is AC^0-reducible to $\mathrm{WP}(K)$ and $\mathrm{WP}(G/K)$. The latter belongs to $\mathsf{ALOGTIME}$ and $\mathrm{WP}(K)$ belongs to $\mathsf{DLOGTIME}$-uniform ACC^0 if K is finite and to $\mathsf{DLOGTIME}$-uniform TC^0 if K is infinite (note that K as a finite index subgroup of G is f.g. linear too). In all cases, $\mathrm{WP}(G)$ belongs to $\mathsf{ALOGTIME}$. ◀

New examples of SENS groups. In order to get new examples of (uniformly) SENS groups, we use the following result on groups with a certain self-embedding property with respect to wreath products.

▶ **Theorem 18.** *Let G be a finitely generated group with $G \wr H \leq G$ for some non-trivial group H. Then G is uniformly SENS.*

As an immediate consequence of this theorem and Lemma 8, we obtain:

▶ **Corollary 19.** *Thompson's groups $F < T < V$ are uniformly SENS.*

By the following result, Grigorchuk's group and the Gupta-Sidki groups are uniformly SENS.

▶ **Theorem 20.** *Let G be a weakly branched self-similar group, and assume that it admits a f.g. branching subgroup K. Then K and hence G are uniformly SENS.*

In the long version [5] we derive Theorems 18 and 20 from a single result. Roughly speaking, it states that a f.g. group G is uniformly SENS if G contains a subgroup $\langle h_0, h_1, h_2, \ldots \rangle$ that acts on a tree X^* (where X can be also infinite) in such a way that there exist $x_{-1}, x, x_1 \in X$ with the following properties for all $k \geq 0$:

(i) h_k only acts non-trivially on the subtree below x^k and

(ii) the three tree nodes $x^k x_{-1}$, $x^k x$, and $x^k x_1$ are consecutive in the orbit of h_k.

In addition, h_k must be of word length $2^{\mathcal{O}(k)}$ (with respect to the generators of G) and the symbol in h_k at a certain position must be computable in linear time.

Theorem 18 can be derived from this statement as follows: we can use the embedding $G \wr H \leq G$ in order to find in G a subgroup $\langle h_0, h_1, \ldots \rangle \cong (\cdots \wr \mathbb{Z}) \wr \mathbb{Z}$ or $\langle h_0, h_1, \ldots \rangle \cong (\cdots \wr (\mathbb{Z}/p)) \wr (\mathbb{Z}/p)$. This subgroup acts in a canonical way on the tree X^* for $X = \mathbb{Z}$ or $X = \mathbb{Z}/p$. Let the element h_k be a generator of the $(k+1)$-st cyclic factor from the right. Then h_0 cyclically permutes the children of the root ε, and, more generally, h_k cyclically permutes the children of the node 0^k and stabilizes all other nodes. Using an appropriate padding, the symbols of the h_k are computable in linear time, so we can apply the above mentioned result from the long version [5]. For weakly branched self-similar groups, after overcoming some minor technical difficulties, the proof follows the same outline.

Direct proofs for Thompson's and Grigorchuk's groups. One can also prove the uniform SENS property for Thompson's group F directly. Recall the infinite presentation $F = \langle x_0, x_1, x_2, \ldots \mid x_k^{x_i} = x_{k+1} (i < k) \rangle$.

▶ **Proposition 21.** *Let $g = x_3 x_2^{-1} \in F$ and define c_v for $v \in \{0,1\}^*$ inductively via*

$$c_\varepsilon = \varepsilon, \quad c_{v0} = x_1 c_v, \text{ and } c_{v1} = x_0^{-1} x_1 c_v.$$

Finally, for $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ let $g_{d,v} = g^{c_v}$. Then $g_{d,v} = [g_{d,v0}, g_{d,v1}]$ for all $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ and $g_\varepsilon = g \neq 1$ in F. In particular, G is uniformly SENS.

Proof. Obviously, we have $g_\varepsilon = g$ in F. Moreover, since $x_2 \neq x_3$ (which follows directly from the normal form theorem for F; see e.g. [12, Corollary 2.7]), we have $g \neq 1$ in F (Condition (c) of Definition 1). The identity $g = [g, g^{x_0^{-1}}]^{x_1} = [g^{x_1}, g^{x_0^{-1}x_1}]$ is straightforward to check. Thus, we obtain Condition (b):

$$g_{d,v} = g^{c_v} = [g^{x_1 c_v}, g^{x_0^{-1}x_1 c_v}] = [g^{c_{v0}}, g^{c_{v1}}] = [g_{d,v0}, g_{d,v1}].$$

Moreover, since $|g_{d,v}| = |g| + 2|c_v| \leq |g| + 4d$, we can pad with the identity symbol writing $g_{v,d}$ as a word of length $2^{\mu d}$ for some proper constant $\mu \in \mathbb{N}$ in order to meet Condition (a) (be aware that the lengths need to be computed over a finite standard generating set Σ, e. g. $\Sigma = \{1, x_0, x_0^{-1}, x_1, x_1^{-1}\}$). This shows that F is SENS.

Condition (d) of Definition 1 is also straightforward to check by introducing some more padding: we slightly change the definition of the c_v by setting $c_\varepsilon = \varepsilon, c_{v0} = 1x_1c_v$, and $c_{v1} = x_0^{-1}x_1c_v$ where $1 \in \Sigma$. This new c_v represents the same group element as the old c_v, but now we have $|c_v| = 2|v|$ for all $v \in \{0,1\}^*$ and all letters at even positions are x_1, while letters at odd positions are either 1 or x_0^{-1} (the $(2j+1)$-st letter of c_v depends on the $(|v| - j)$-th bit of v). Notice that we have $|g_{d,v}| = |g| + 2|c_v| = |g| + 4d$.

On input of $v \in \{0,1\}^d$, $i \in \mathbb{N}$ (encoded with μd bits), and $a \in \Sigma$, first decide whether $1 \leq i \leq 2d$, or $2d < i \leq 2d + |g|$, or $2d + |g| < i \leq 4d + |g|$, or $i > 4d + |g|$. This clearly can be done in DLINTIME (recall that g is a constant). Now assume that $1 \leq i \leq 2d$ (i. e., i points into the leading c_v^{-1} of $g_{d,v} = c_v^{-1}gc_v$). If i is odd, one accepts if and only if $a = x_1^{-1}$; if i is even, one accepts if and only if the $i/2$-th bit of v is zero and $a = 1$ or if the $i/2$-th bit of v is one and $a = x_0$. The other cases are similar. ◀

For the special case of Grigorchuk's group we give below an alternative proof for the uniform SENS property, where the $g_{d,v}$ are of constant length.

▶ **Proposition 22.** *Consider in the Grigorchuk group $G = \langle a, b, c, d\rangle$ the elements $x = (abad)^2$ and $y = x^b = bababadabac$. Define inductively $z_v \in \{x, y, x^{-1}, y^{-1}\}$ for $v \in \{0,1\}^*$: $z_\varepsilon = x$ and if z_v is defined, then we define z_{v0} and z_{v1} according to the following table:*

z_v	z_{v0}	z_{v1}
x	x^{-1}	y^{-1}
x^{-1}	y^{-1}	x^{-1}
y	y	x
y^{-1}	x	y

For every $d \in \mathbb{N}$ and $v \in \{0,1\}^{\leq d}$ let $g_{d,v} = z_v$ for $|v| = d$ and $g_{d,v} = [g_{d,v0}, g_{d,v1}]$ for $|v| < d$. We then have $g_{d,\varepsilon} \neq 1$ in G. In particular, G is uniformly SENS.

Proof. That $x \neq 1 \neq y$ is easy to check by computing the action of x and y on the third level of the tree. Now the following equations are easy to check in G:

$$[x, y] = \langle\!\langle 1, \langle\!\langle 1, y^{-1}\rangle\!\rangle\rangle\!\rangle \qquad [x^{-1}, y^{-1}] = \langle\!\langle 1, \langle\!\langle 1, x\rangle\!\rangle\rangle\!\rangle$$
$$[y, x] = \langle\!\langle 1, \langle\!\langle 1, y\rangle\!\rangle\rangle\!\rangle \qquad [y^{-1}, x^{-1}] = \langle\!\langle 1, \langle\!\langle 1, x^{-1}\rangle\!\rangle\rangle\!\rangle$$

Hence, $[z_{v0}, z_{v1}] = \langle\!\langle 1, \langle\!\langle 1, z_v\rangle\!\rangle\rangle\!\rangle$. The checks are tedious to compute by hand, but easy in the GAP package FR (note that vertices are numbered from 1 in GAP and from 0 here):

```
gap> LoadPackage("fr");
gap> AssignGeneratorVariables(GrigorchukGroup);
gap> x := (a*b*a*d)^2; y := x^b;
```

```
gap> Assert(0,Comm(x,y) = VertexElement([2,2],y^-1));
gap> Assert(0,Comm(x^-1,y^-1) = VertexElement([2,2],x));
gap> Assert(0,Comm(y,x) = VertexElement([2,2],y));
gap> Assert(0,Comm(y^-1,x^-1) = VertexElement([2,2],x^-1));
```

We claim that $g_{d,\varepsilon} \neq 1$ in G. The equation $[z_{v0}, z_{v1}] = \langle\!\langle 1, \langle\!\langle 1, z_v \rangle\!\rangle \rangle\!\rangle$ immediately implies that $g_{d,v}$ acts as z_v on the subtree below vertex $1^{2(d-|v|)}$ and trivially elsewhere. In particular, $g_{d,\varepsilon}$ acts as $z_\varepsilon = x \neq 1$ on the subtree below vertex 1^{2d} and is non-trivial. With this definition, the $g_{d,v}$ satisfy the definition of a SENS group. Moreover, given some $v \in \{0,1\}^d$, $g_{d,v}$ can be computed in time $\mathcal{O}(d)$ by a deterministic finite automaton with state set $\{x^{\pm 1}, y^{\pm 1}\}$. ◀

5 Circuit value problems for wreath products

This section provides further details regarding Theorem 6. We start with two applications for Mod-classes (applications for PSPACE can be found in the next section). For a complexity class C we define the class $\mathsf{Mod}_m\mathsf{C}$ by $L \in \mathsf{Mod}_m\mathsf{C}$ if there exists a polynomial $p(n)$ and a language $K \in \mathsf{C}$ such that $L = \{u \mid |\{v \in \{0,1\}^{p(|u|)} : u\#v \in K\}| \not\equiv 0 \mod m\}$.

▶ **Example 23.** If G is a finite non-abelian p-group, then $\mathsf{LEAF}(\mathrm{WP}(G)) \subseteq \mathsf{Mod}_p \cdots \mathsf{Mod}_p\mathsf{P} = \mathsf{Mod}_p\mathsf{P} \subseteq \mathsf{LEAF}(\mathrm{WP}(G))$ by [22, Satz 4.32], [9, Theorem 6.7], and [24, Theorem 2.2] and likewise $\mathsf{LEAF}(\mathrm{WP}(G/Z(G))) = \mathsf{Mod}_p\mathsf{P}$. Hence, in this case $\mathrm{CVP}(G \wr \mathbb{Z})$ is complete for $\forall\mathsf{Mod}_p\mathsf{P}$.

▶ **Example 24.** Consider the symmetric group on three elements S_3. By [24, Example 2.5] we have $\mathsf{LEAF}(\mathrm{WP}(S_3)) = \mathsf{Mod}_3\mathsf{Mod}_2\mathsf{P}$ (also written as $\mathsf{Mod}_3{\oplus}\mathsf{P}$). Since S_3 has trivial center, it follows that $\mathrm{CVP}(S_3 \wr \mathbb{Z})$ is complete for $\forall\mathsf{Mod}_3{\oplus}\mathsf{P}$.

In the rest of the section, we outline the proof of Theorem 6; full details can be found in the appendix. The first statement of Theorem 6 (that $\mathrm{CVP}(G \wr \mathbb{Z})$ belongs to $\forall\mathsf{LEAF}(\mathrm{WP}(G))$) is the easy part: Let Σ be a standard generating set for G and fix a generator t for $\mathbb{Z}$. Then $\Gamma = \Sigma \cup \{t, t^{-1}\}$ is a standard generating set for $G \wr \mathbb{Z}$. Consider a circuit over the wreath product $G \wr \mathbb{Z}$ whose input gates are labelled with generators from Γ. We can evaluate this circuit also in the free monoid Γ^* and obtain a word $w \in \Gamma^*$ as the evaluation of the output gate. We have to verify that $w = 1$ in $G \wr \mathbb{Z}$. One first checks in polynomial time whether the exponent sum of t in w is zero. If not, the algorithm rejects, otherwise the word w represents in $G \wr \mathbb{Z}$ a function $f\colon \mathbb{Z} \to G$ with finite support. One can easily compute in polynomial time two binary encoded integers i, j such that $\mathrm{supp}(f)$ is contained in $[i..j]$ (the integer interval from i to j). It remains to verify that $\forall x \in [i..j] : f(x) = 1$. The $\forall$-quantifier over $[i..j]$ corresponds to the $\forall$-part in $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. Finally, for a specific number $x \in [i..j]$ the machine then produces a leaf string $w_x \in \Sigma^*$ such that w_x represents the group element $f(x) \in G$. Basically, the machine branches to all positions p in the word w and prints the symbol a at that position p, if (i) $a \in \Sigma$ and (ii) the exponent sum of all t's in the prefix up to position $p-1$ in w is x (this can be checked in polynomial time). Otherwise, the machine prints the padding letter 1.

The hard part of Theorem 6 is showing hardness for $\forall\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$. The proof for this uses some of the techniques from the paper [37], where a connection between leaf strings and string compression was established. Instead of going into the details (which can be found in Appendix C) we want to explain another perspective on the theorem. Let us restrict to the case that the center of G is trivial; hence the circuit value problem for $G \wr \mathbb{Z}$ is complete for $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. Also fix a standard generating set Γ for G. Recall that a circuit over

the group G can be seen also as a succinct representation of a string over the alphabet Γ, which is obtained by evaluating the circuit in the free monoid Γ^*. The circuit value problem for G then asks whether this word belongs to $\mathrm{WP}(G)$. Leaf languages correspond to an even more succinct form of string compression by Boolean circuits. A Boolean circuit $\mathcal{C}$ with n inputs represents the binary string of length 2^n, where the i-th symbol is 1 if and only if $\mathcal{C}$ evaluates to true under the i-th truth assignment. In order to represent an arbitrary string $w \in \Gamma^*$ by a Boolean circuit one has to (i) fix an encoding of the symbols from Γ by binary strings and (ii) specify in addition to the circuit also the length of w. It is then well-known that the question whether a string specified by a Boolean circuit belongs to a fixed language K is complete for $\mathsf{LEAF}(K)$ (strictly speaking this is only true if $\mathsf{LEAF}(K)$ is replaced by the corresponding balanced leaf language class, but for $K = \mathrm{WP}(G)$ this makes no difference due to the padding letter 1; see Appendix A). Compression by Boolean circuits is much more succinct than compression by group circuits. For instance, for a finite non-solvable group G, $\mathsf{LEAF}(\mathrm{WP}(G)) = \mathsf{PSPACE}$ but $\mathrm{CVP}(G)$ is P-complete. Roughly speaking, Theorem 6 says that compression by group circuits over the wreath product $G \wr \mathbb{Z}$ has the same power as compression by Boolean circuits over the group G.

6 PSPACE-complete circuit value problems

In this section we apply Theorem 6 to SENS groups. The results are summarized in Corollary 7 from the introduction. The proofs of this section can be found in the full version [5].

The following result can be derived from Theorem 2 using a padding argument (recall that $\mathsf{PSPACE} = \mathsf{APTIME}$). Another point of view is that Lemma 25 generalizes the inclusion $\mathsf{PSPACE} \subseteq \mathsf{LEAF}(\mathrm{WP}(G))$ for a finite simple group (in which case equality holds) [25]. Note that, by Lemma 14, $G/Z(G)$ is uniformly SENS if G is uniformly SENS.

▶ **Lemma 25.** *If G is uniformly SENS, then $\mathsf{PSPACE} \subseteq \mathsf{LEAF}(WP(G/Z(G)))$.*

From Theorem 6 and Lemma 25 we get:

▶ **Corollary 26.** *If G is uniformly SENS, then $CVP(G \wr \mathbb{Z})$ is PSPACE-hard.*

We can apply Corollary 26 to wreath products $G \wr \mathbb{Z}$ where G is finite non-solvable or free of rank $n \geq 2$ (statement i in Corollary 7 from the introduction). In this case, the word problem for $G \wr \mathbb{Z}$ can be solved in $\mathsf{LOGSPACE}$ (which follows from a transfer theorem for wreath products [48] and the fact the word problem for finite groups and free groups can be solved in $\mathsf{LOGSPACE}$ [36]). This in turn implies that $\mathrm{CVP}(G \wr \mathbb{Z})$ belongs to PSPACE.

For Thompson's group F we have $F \wr \mathbb{Z} \leq F$ (Lemma 8). Moreover, F is uniformly SENS (Corollary 19). Hence, Corollary 26 shows that $\mathrm{CVP}(F)$ is PSPACE-hard. Furthermore, $\mathrm{CVP}(F)$ belongs to PSPACE. This follows from the fact that F is co-context-free, i.e., the complement of the word problem of F is a context-free language [33] (this is independent of the finite generating set). We obtain statement ii in Corollary 7.

Finally, statements iii and iv from Corollary 7 are consequences of the following result:

▶ **Corollary 27.** *If G is a weakly branched torsion group whose branching subgroup is f.g., then $CVP(G)$ is PSPACE-hard. If, in addition, G is contracting, then $CVP(G)$ is PSPACE-complete.*

Proof sketch. The second statement follows easily from the first statement since for a contracting group the word problem belongs to $\mathsf{LOGSPACE}$ (Proposition 11), which implies that the circuit value problem belongs to PSPACE (see Lemma 34 in the appendix). For the

first statement, the main observation is that the hardness proof for the second statement of Theorem 6 (see Appendix C) uses only a subinterval of the integers whose length is exponentially bounded in the input length. This means that it suffices to find a copy of $G \wr (\mathbb{Z}/p^n)$ in G for each n and some fixed $p \geq 2$. Moreover, the embedding $\varphi\colon G \wr (\mathbb{Z}/p^n) \to G$ must be circuit-efficient in the sense that for every generator a of $G \wr (\mathbb{Z}/p^n)$ one can compute from n (given in unary notation) a circuit $\mathcal{G}_{n,a}$ for $\varphi(a)$. For the case of a weakly branched torsion group G whose branching subgroup K is finitely generated we cannot show the existence of such a circuit-efficient embedding for G itself, but we can prove it for K (in fact, it suffices to show an embedding $K \wr (\mathbb{Z}/p) \leq K$, which can then be iterated n-times in order to get the embedding $K \wr (\mathbb{Z}/p^n) \leq K$). This implies that the circuit value problem for K (and hence for G) is PSPACE-hard. ◀

7 Conclusion and open problems

We have added an algorithmic constraint (uniformly SENS) to the algebraic notion of being a non-solvable group, which implies that the word problem is ALOGTIME-hard. Using this, we produced several new examples of non-solvable groups with an ALOGTIME-hard word problem. However, the question remains open whether every non-solvable group has an ALOGTIME-hard word problem, even if it is not SENS. We showed that for every contracting self-similar group the word problem belongs to LOGSPACE. Here, the question remains whether there exists a contracting self-similar group with a LOGSPACE-complete word problem. In particular, is the word problem for the Grigorchuk group LOGSPACE-complete? (We proved that it is ALOGTIME-hard.) Also the precise complexity of the word problem for Thompson's group F is open. It is ALOGTIME-hard and belongs to LOGCFL; the latter follows from [33]. In fact, from the proof in [33] one can deduce that the word problem for F belongs to LOGDCFL (the closure of the deterministic context-free languages with respect to LOGSPACE-reductions).

References

1 Ian Agol. The virtual Haken conjecture. *Documenta Mathematica*, 18:1045–1087, 2013. With an appendix by Ian Agol, Daniel Groves, and Jason Manning.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.

3 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. doi:10.1016/0022-0000(89)90037-8.

4 David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35:941–952, 1988. doi:10.1145/48014.63138.

5 Laurent Bartholdi, Michael Figelius, Markus Lohrey, and Armin Weiß. Groups with ALOGTIME-hard word problems and PSPACE-complete compressed word problems. Technical report, arXiv.org, 2020. arXiv:1909.13781.

6 Laurent Bartholdi, Rostislav I. Grigorchuk, and Zoran Šuniḱ. Branch groups. In *Handbook of algebra, Volume 3*, pages 989–1112. Elsevier/North-Holland, Amsterdam, 2003. doi:10.1016/S1570-7954(03)80078-5.

7 Laurent Bartholdi and Volodymyr V. Nekrashevych. Iterated monodromy groups of quadratic polynomials. I. *Groups, Geometry, and Dynamics*, 2(3):309–336, 2008. doi:10.4171/GGD/42.

8 Martin Beaudry, Pierre McKenzie, Pierre Péladeau, and Denis Thérien. Finite monoids: From word to circuit evaluation. *SIAM Journal on Computing*, 26(1):138–152, 1997. doi:10.1137/S0097539793249530.

9 Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103(1):3–23, 1992. doi:10.1016/0304-3975(92)90084-S.

10 William W. Boone. The Word Problem. *Annals of Mathematics*, 70(2):207–265, 1959. URL: www.jstor.org/stable/1970103.

11 Daniel P. Bovet, Pierluigi Crescenzi, and Riccardo Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104(2):263–283, 1992. doi:10.1016/0304-3975(92)90125-Y.

12 John W. Cannon, William J. Floyd, and Walter R. Parry. Introductory notes on Richard Thompson's groups. *L'Enseignement Mathématique*, 42(3):215–256, 1996.

13 Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005. doi:10.1109/TIT.2005.850116.

14 Max Dehn. Über unendliche diskontinuierliche Gruppen. *Mathematische Annalen*, 71(1):116–144, 1911. doi:10.1007/BF01456932.

15 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–completeness*. Freeman, 1979.

16 Max Garzon and Yechezkel Zalcstein. The complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88(1):83–98, 1991. doi:10.1016/0304-3975(91)90074-C.

17 Rostislav I. Grigorchuk. Burnside's problem on periodic groups. *Functional Analysis and Its Applications*, 14:41–43, 1980. doi:10.1007/BF01078416.

18 Rostislav I. Grigorchuk and Zoran Šunić. Asymptotic aspects of Schreier graphs and Hanoi Towers groups. *C. R. Math. Acad. Sci. Paris*, 342(8):545–550, 2006. doi:10.1016/j.crma.2006.02.001.

19 Victor S. Guba and Mark V. Sapir. On subgroups of the R. Thompson group F and other diagram groups. *Matematicheskii Sbornik*, 190(8):3–60, 1999. doi:10.1070/SM1999v190n08ABEH000419.

20 Narain Gupta and Saïd Sidki. On the Burnside problem for periodic groups. *Mathematische Zeitschrift*, 182(3):385–388, 1983. doi:10.1007/BF01179757.

21 Frédéric Haglund and Daniel T. Wise. Coxeter groups are virtually special. *Advances in Mathematics*, 224(5):1890–1903, 2010. doi:10.1016/j.aim.2010.01.011.

22 Ulrich Hertrampf. *Über Komplexitätsklassen, die mit Hilfe von k-wertigen Funktionen definiert werden*. Habilitationsschrift, Universität Würzburg, 1994.

23 Ulrich Hertrampf. The shapes of trees. In *Proceedings of COCOON 1997*, volume 1276 of *Lecture Notes in Computer Science*, pages 412–421. Springer, 1997. doi:10.1007/BFb0045108.

24 Ulrich Hertrampf. Algebraic acceptance mechanisms for polynomial time machines. *SIGACT News*, 31(2):22–33, 2000. doi:10.1145/348210.348215.

25 Ulrich Hertrampf, Clemens Lautemann, Thomas Schwentick, Heribert Vollmer, and Klaus W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 200–207. IEEE Computer Society Press, 1993. doi:10.1109/SCT.1993.336526.

26 Ulrich Hertrampf, Heribert Vollmer, and Klaus Wagner. On balanced versus unbalanced computation trees. *Mathematical Systems Theory*, 29(4):411–421, 1996. doi:10.1007/BF01192696.

27 Derek F. Holt, Markus Lohrey, and Saul Schleimer. Compressed decision problems in hyperbolic groups. In *Proceedings of STACS 2019*, volume 126 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: http://www.dagstuhl.de/dagpub/978-3-95977-100-9.

28 Derek F. Holt, Sarah Rees, and Claas E. Röver. *Groups, Languages and Automata*, volume 88 of *London Mathematical Society Student Texts*. Cambridge University Press, 2017. doi:10.1017/9781316588246.

29 Birgit Jenner, Pierre McKenzie, and Denis Thérien. Logspace and logtime leaf languages. *Information and Computation*, 129(1):21–33, 1996. doi:10.1006/inco.1996.0071.

30 Howard J. Karloff and Walter L. Ruzzo. The iterated mod problem. *Information and Computation*, 80(3):193–204, 1989. doi:10.1016/0890-5401(89)90008-4.

31 Daniel König and Markus Lohrey. Evaluation of circuits over nilpotent and polycyclic groups. *Algorithmica*, 80(5):1459–1492, 2018. doi:10.1007/s00453-017-0343-z.

32 Daniel König and Markus Lohrey. Parallel identity testing for skew circuits with big powers and applications. *International Journal of Algebra and Computation*, 28(6):979–1004, 2018. doi:10.1142/S0218196718500431.

33 Jörg Lehnert and Pascal Schweitzer. The co-word problem for the Higman-Thompson group is context-free. *Bulletin of the London Mathematical Society*, 39(2):235–241, February 2007. doi:10.1112/blms/bdl043.

34 Martin W. Liebeck, Eamonn A. O'Brien, Aner Shalev, and Pham Huu Tiep. The Ore conjecture. *Journal of the European Mathematical Society*, 12(4):939–1008, 2010. doi:10.4171/JEMS/220.

35 Yury Lifshits and Markus Lohrey. Querying and embedding compressed texts. In *Proceedings of MFCS 2006*, volume 4162 of *Lecture Notes in Computer Science*, pages 681–692. Springer, 2006. doi:10.1007/11821069_59.

36 Richard J. Lipton and Yechezkel Zalcstein. Word problems solvable in logspace. *Journal of the Association for Computing Machinery*, 24(3):522–526, 1977. doi:10.1145/322017.322031.

37 Markus Lohrey. Leaf languages and string compression. *Information and Computation*, 209(6):951–965, 2011. doi:10.1016/j.ic.2011.01.009.

38 Markus Lohrey. *The Compressed Word Problem for Groups*. Springer Briefs in Mathematics. Springer, 2014. doi:10.1007/978-1-4939-0748-9.

39 Volodymyr Nekrashevych. *Self-similar groups*, volume 117 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 2005. doi:10.1090/surv/117.

40 Piotr S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, pages 1–143, 1955. In Russian. URL: http://mi.mathnet.ru/eng/tm1180.

41 David Robinson. *Parallel Algorithms for Group Word Problems*. PhD thesis, University of California, San Diego, 1993.

42 Joseph J. Rotman. *An Introduction to the Theory of Groups (fourth edition)*. Springer, 1995.

43 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

44 Hans-Ulrich Simon. Word problems for groups and contextfree recognition. In *Proceedings of FCT 1979*, pages 417–422. Akademie-Verlag, 1979.

45 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of STOC 1987*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

46 Jacques Tits. Free subgroups in linear groups. *Journal of Algebra*, 20(2):250–270, 1972. doi:10.1016/0021-8693(72)90058-0.

47 Heribert Vollmer. *Introduction to Circuit Complexity*. Springer, Berlin, 1999. doi:10.1007/978-3-662-03927-4.

48 Stephan Waack. The parallel complexity of some constructions in combinatorial group theory. *Journal of Information Processing and Cybernetics EIK*, 26:265–281, 1990. doi:10.1007/BFb0029647.

49 Jan Philipp Wächter and Armin Weiß. An automaton group with pspace-complete word problem. In *Proceedings of STACS 2020*, volume 154 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://www.dagstuhl.de/dagpub/978-3-95977-140-5.

50 John S. Wilson. Embedding theorems for residually finite groups. *Mathematische Zeitschrift*, 174(2):149–157, 1980. doi:10.1007/BF01293535.

51 Daniel T. Wise. Research announcement: the structure of groups with a quasiconvex hierarchy. *Electronic Research Announcements in Mathematical Sciences*, 16:44–55, 2009. URL: http://aimsciences.org//article/id/8d3fc128-8d02-4fee-af67-38001ca1d0ac.

A Leaf languages

In the following, we introduce more details concerning leaf languages that were briefly explained in the introduction. An NTM M with input alphabet Γ is called *adequate*, if (i) for every input $x \in \Gamma^*$, M does not have an infinite computation on input x, (ii) the finite set of transition tuples of M is linearly ordered, and (iii) when terminating M prints a symbol $\alpha(q)$ from a finite alphabet Σ, where q is the current state of M. For an input $x \in \Gamma^*$, we define the computation tree by unfolding the configuration graph of M from the initial configuration. By condition (i) and (ii), the computation tree can be identified with a finite ordered tree $T(x) \subseteq \mathbb{N}^*$. For $u \in T(x)$ let $q(u)$ be the M-state of the configuration that is associated with the tree node u. Then, the leaf string $\mathrm{leaf}(M, x)$ is the string $\alpha(q(v_1)) \cdots \alpha(q(v_k)) \in \Sigma^+$, where $v_1, \ldots, v_k$ are all leaves of $T(x)$ listed in lexicographic order.

A *complete binary tree* is a rooted tree, where every non-leaf node has a left and a right child, and every path from the root to a leaf has the same length. An adequate NTM M is called *balanced*, if for every input $x \in \Gamma^*$, the computation $T(x)$ (produced by M on input x) is a complete binary tree. In Section 3 we defined the complexity class $\mathsf{LEAF}(K) = \{\mathsf{LEAF}(M, K) \mid M \text{ is an adequate polynomial time NTM}\}$ for a language $K \subseteq \Sigma^*$. Let us define the subclass

$$\mathsf{bLEAF}(K) = \{\mathsf{LEAF}(M, K) \mid M \text{ is a balanced polynomial time NTM}\}.$$

Both classes $\mathsf{LEAF}(K)$ and $\mathsf{bLEAF}(K)$ are closed under polynomial time reductions. We clearly have $\mathsf{bLEAF}(K) \subseteq \mathsf{LEAF}(K)$. The following result was shown in [29] by padding computation trees to complete binary trees.

▶ **Lemma 28.** *Assume that $K \subseteq \Sigma^*$ is a language such that Σ contains a symbol 1 with the following property: if $uv \in K$ for $u, v \in \Sigma^*$ then $u1v \in K$. Then $\mathsf{LEAF}(K) = \mathsf{bLEAF}(K)$.*

In particular, we obtain the following lemma:

▶ **Lemma 29.** *Let G be a finitely generated group and Σ a finite standard generating set for G. Then $\mathsf{LEAF}(WP(G, \Sigma)) = \mathsf{bLEAF}(WP(G, \Sigma))$.*

Moreover, we have:

▶ **Lemma 30.** *Let G be finitely generated group and Σ, Γ finite standard generating sets for G. Then $\mathsf{LEAF}(WP(G, \Sigma)) = \mathsf{LEAF}(WP(G, \Gamma))$.*

Proof. Consider a language $L \in \mathsf{LEAF}(\mathrm{WP}(G, \Sigma))$. Thus, there exists an adequate polynomial time NTM M such that $L = \mathsf{LEAF}(M, \mathrm{WP}(G, \Sigma))$. We modify M as follows: If M terminates and prints the symbol $a \in \Sigma$, it enters a small nondeterministic subcomputation that produces the leaf string w_a, where $w_a \in \Gamma^*$ is a word that evaluates to the same group element as a. Let M' be the resulting adequate polynomial time NTM. It follows that $\mathsf{LEAF}(M, \mathrm{WP}(G, \Sigma)) = \mathsf{LEAF}(M', \mathrm{WP}(G, \Gamma))$. ◀

As remarked in the main part, Lemma 29 allows to just write $\mathsf{LEAF}(\mathrm{WP}(G))$ (as well as $\mathsf{bLEAF}(\mathrm{WP}(G))$). In [25] it was shown that $\mathsf{PSPACE} = \mathsf{LEAF}(\mathrm{WP}(G))$ for every finite non-solvable group.

B Compressed words and the circuit value problem

We mentioned in the introduction that the circuit value problem for a f.g. group G can be seen as a succinct version of the word problem, where the input word is succinctly represented by a context-free grammar that produces exactly one word. Such a context-free grammar is

also called a straight-line program in the literature on string compression. This alternative viewpoint on the circuit value problem turns out to be convenient for our proofs. In the following we will elaborate this viewpoint.

A *straight-line program* (SLP for short) over the alphabet Σ is a triple $\mathcal{G} = (V, \rho, S)$, where V is a finite set of variables such that $V \cap \Sigma = \emptyset$, $S \in V$ is the start variable, and $\rho : V \to (V \cup \Sigma)^*$ is a mapping such that the relation $\{(A, B) \in V \times V : B \text{ occurs in } \rho(A)\}$ is acyclic. For the reader familiar with context free grammars, it might be helpful to view the SLP $\mathcal{G} = (V, \rho, S)$ as the context-free grammar (V, Σ, P, S), where P contains all productions $A \to \rho(A)$ for $A \in V$. The definition of an SLP implies that this context-free grammar derives exactly on terminal word, which will be denoted by $\mathrm{val}(\mathcal{G})$. Formally, one can extend ρ to a morphism $\rho : (V \cup \Sigma)^* \to (V \cup \Sigma)^*$ by setting $\rho(a) = a$ for all $a \in \Sigma$. The above acyclicity condition on ρ implies that for $m = |V|$ we have $\rho^m(w) \in \Sigma^*$ for all $w \in (V \cup \Sigma)^*$. We then define $\mathrm{val}_{\mathcal{G}}(w) = \rho^m(w)$ (the string derived from the sentential form w) and $\mathrm{val}(\mathcal{G}) = \mathrm{val}_{\mathcal{G}}(S)$. We define the size of the SLP $\mathcal{G}$ as the total length of all right-hand sides: $|\mathcal{G}| = \sum_{A \in V} |\rho(A)|$. SLPs offer a succinct representation of words that contain many repeated substrings. For instance, the word $(ab)^{2^n}$ can be produced by the SLP $\mathcal{G} = (\{A_0, \ldots, A_n\}, \rho, A_n)$ with $\rho(A_0) = ab$ and $\rho(A_{i+1}) = A_i A_i$ for $0 \leq i \leq n-1$.

The word $\rho(A)$ is also called the *right-hand side* of A. Quite often, it is convenient to assume that all right-hand sides are of the form $a \in \Sigma$ or BC with $B, C \in V$. This corresponds to the well-known Chomsky normal form for context-free grammars. There is a simple linear time algorithm that transforms an SLP $\mathcal{G}$ with $\mathrm{val}(\mathcal{G}) \neq \varepsilon$ into an SLP $\mathcal{G}'$ in Chomsky normal form with $\mathrm{val}(\mathcal{G}) = \mathrm{val}(\mathcal{G}')$, see e.g. [38, Proposition 3.8].

The circuit value problem $\mathrm{CVP}(G)$ for a f.g. group G (as defined in the introduction) is equivalent to the following problem, where Σ is a finite standard generating set for G:
Input: an SLP $\mathcal{G}$ over the alphabet Σ.
Question: does $\mathrm{val}(\mathcal{G}) = 1$ hold in G?
It is an easy observation that the computational complexity of the circuit value problem for G does not depend on the chosen generating set Σ: More precisely, if we denote for a moment with $\mathrm{CVP}(G, \Sigma)$ the circuit value problem for the specific generating set Σ, then for all generating sets Σ and Σ' for G, $\mathrm{CVP}(G, \Sigma)$ is LOGSPACE-reducible to $\mathrm{CVP}(G, \Sigma')$ [38, Lemma 4.2]. The circuit value problem for G is also known as the compressed word problem for G [38].

We need a couple of known results for SLPs. For this we use the following notations on strings (which will be also needed in Appendix C). Take a word $w = a_0 \cdots a_{n-1} \in \Sigma^*$ over the alphabet Σ ($n \geq 0$, $a_0, \ldots, a_{n-1} \in \Sigma$). For $0 \leq i < n$ let $w[i] = a_i$ and for $0 \leq i \leq j < n$ let $w[i : j] = a_i a_{i+1} \cdots a_j$. Moreover $w[: i] = w[0 : i]$. Note that in the notations $w[i]$ and $w[i : j]$ we take 0 as the first position in w. This will be convenient in Appendix C. Finally, with $|w|_a = |\{i \mid a_i = a\}|$ we denote the number of occurrences of a in w.

▶ **Lemma 31** (c.f. [13]). *For every SLP $\mathcal{G}$ we have $|\mathrm{val}(\mathcal{G})| \leq 3^{|\mathcal{G}|/3}$.*

▶ **Lemma 32** ([38, Chapter 3]). *The following problems can be solved in polynomial time, where $\mathcal{G}$ is an SLP over a terminal alphabet Σ, $a \in \Sigma$, and $p, q \in \mathbb{N}$ are numbers given in binary notation:*

- *Given $\mathcal{G}$, compute the length $|\mathrm{val}(\mathcal{G})|$ of the word $\mathrm{val}(\mathcal{G})$.*
- *Given $\mathcal{G}$ and a, compute the number $|\mathrm{val}(\mathcal{G})|_a$ of occurrences of a in $\mathrm{val}(\mathcal{G})$.*
- *Given $\mathcal{G}$ and p, compute the symbol $\mathrm{val}(\mathcal{G})[p] \in \Sigma$ (in case $0 \leq p < |\mathrm{val}(\mathcal{G})|$ does not hold, the algorithm outputs a special symbol).*
- *Given $\mathcal{G}$ and p, q, compute an SLP for the string $\mathrm{val}(\mathcal{G})[p : q]$ (in case $0 \leq p \leq q < |\mathrm{val}(\mathcal{G})|$ does not hold, the algorithm outputs a special symbol).*

▶ **Lemma 33** (c.f. [38, Lemma 3.12]). *Given a symbol $a_0 \in \Sigma$ and a sequence of morphisms $\varphi_1, \ldots, \varphi_n : \Sigma^* \to \Sigma^*$, where every φ_i is given by a list of the words $\varphi_i(a)$ for $a \in \Sigma$, one can compute in* LOGSPACE *an SLP for the word $\varphi_1(\varphi_2(\cdots \varphi_n(a_0) \cdots))$.*

The next lemma follows easily by evaluating a given SLP and then running an algorithm for the "ordinary" word problem:

▶ **Lemma 34.** *If G is a finitely generated group such that $WP(G)$ can be solved in polylogarithmic space, then $CVP(G)$ belongs to* PSPACE.

C Additional details for Section 5

This section presents a detailed proof of Theorem 6. Instead of circuits over groups we will use the equivalent framework of SLPs from Appendix B. The proof of the lower bound in Theorem 6 uses some of the techniques from the paper [37], where a connection between leaf strings and SLPs was established. In Sections C.1–C.3 we will introduce these techniques. The proof of Theorem 6 will be given in Appendix C.4.

Let us first fix some general notations. For integers $i \leq j$ we write $[i..j]$ for the interval $\{z \in \mathbb{Z} \mid i \leq z \leq j\}$. In the following, we will identify a bit string $\alpha = a_1 a_2 \cdots a_n$ $(a_1, \ldots, a_n \in \{0,1\})$ with the vector $(a_1, a_2, \ldots, a_n)$. In particular, for another vector $\overline{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{N}^n$ we will write $\alpha \cdot \overline{s} = \sum_{i=1}^{n} a_i \cdot s_i$ for the scalar product. Moreover, we write $\sum \overline{s}$ for the sum $s_1 + s_2 + \cdots + s_n$.

C.1 Subsetsum problems

A sequence $(s_1, \ldots, s_n)$ of natural numbers is *super-decreasing* if $s_i > s_{i+1} + \cdots + s_n$ for all $1 \leq i \leq n$. For example, $(s_1, \ldots, s_n)$ with $s_i = 2^{n-i}$ is super-decreasing. An instance of the *subsetsum problem* is a tuple $(t, s_1, \ldots, s_k)$ of binary encoded natural numbers. It is a positive instance if there are $a_1, \ldots, a_k \in \{0,1\}$ such that $t = a_1 s_1 + \cdots + a_k s_k$. Subsetsum is a classical NP-complete problem, see e.g. [15]. The *super-decreasing subsetsum* problem is the restriction of subsetsum to instances $(t, s_1, \ldots, s_k)$, where $(s_1, \ldots, s_k)$ is super-decreasing. In [30] it was shown that super-decreasing subsetsum is P-complete.[1] We need a slightly generalized version of the construction showing P-hardness that we discuss in Appendix C.2.

C.2 From Boolean circuits to super-decreasing subsetsum

For this section, we have to fix some details on Boolean circuits. Let us consider a Boolean circuit C with input gates $x_1, \ldots, x_m$ and output gates $y_0, \ldots, y_{n-1}$.[2] We view C as a directed acyclic graph with multi-edges (there can be two edges between two nodes); the nodes are the gates of the circuit. The number of incoming edges of a gate is called its *fan-in* and the number of outgoing edges is the *fan-out*. Every input gate x_i has fan-in zero and every output gate y_i has fan-out zero. Besides the input gates there are two more gates c_0 and c_1 of fan-in zero, where c_i carries the constant truth value $i \in \{0,1\}$. Besides $x_1, \ldots, x_m, c_0, c_1$ every other gate has fan-in two and computes the nand of its two input gates. Moreover, we assume that every output gate y_i is a nand-gate. For a bit string $\alpha = b_1 \cdots b_m$ $(b_1, \ldots, b_m \in \{0,1\})$ and $0 \leq i \leq n-1$ we denote with $C(\alpha)_i$ the value of the output gate y_i when every input gate x_j $(1 \leq j \leq m)$ is set to b_j. Thus, C defines a map $\{0,1\}^m \to \{0,1\}^n$.

[1] In fact, [30] deals with the *super-increasing* subsetsum problem. But this is only a nonessential detail. For our purpose, super-decreasing sequences are more convenient.

[2] It will be convenient for us to number the input gates from 1 and the output gates from 0.

We assume now that C is a Boolean circuit as above with the following additional property that will be satisfied later: For all input bit strings $\alpha \in \{0,1\}^m$ there is exactly one $i \in [0..n-1]$ such that $C(\alpha)_i = 1$. Using a refinement of the construction from [30] we compute in LOGSPACE $q_0, \ldots, q_{n-1} \in \mathbb{N}$ and two super-decreasing sequences $\overline{r} = (r_1, \ldots r_m)$ and $\overline{s} = (s_1, \ldots, s_k)$ for some k (all numbers are represented in binary notation) with the following properties:

- The $r_1, \ldots, r_m$ are pairwise distinct powers of 4.
- For all $0 \leq i \leq n-1$ and all $\alpha \in \{0,1\}^m$: $C(\alpha)_i = 1$ if and only if there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = q_i + \alpha \cdot \overline{r}$.

Let us first add for every input gate x_i two new nand-gates $\bar{x}_i$ and $\bar{\bar{x}}_i$, where $\bar{\bar{x}}_i$ has the same outgoing edges as x_i. Moreover we remove the old outgoing edges of x_i and replace them by the edges $(x_i, \bar{x}_i)$, $(c_1, \bar{x}_i)$ and two edges from $\bar{x}_i$ to $\bar{\bar{x}}_i$. This has the effect that every input gate x_i has a unique outgoing edge. Clearly, the new circuit computes the same Boolean function (basically, we introduce two negation gates for every input gate). Let $g_1, \ldots, g_p$ be the nand-gates of the circuit enumerated in reverse topological order, i.e., if there is an edge from gate g_i to gate g_j then $i > j$. We denote the two edges entering gate g_i with e_{2i+n-2} and e_{2i+n-1}. Moreover, we write e_i $(0 \leq i \leq n-1)$ for an imaginary edge that leaves the output gate y_i and whose target gate is unspecified. Thus, the edges of the circuit are $e_0, \ldots, e_{2p+n-1}$. We now define the natural numbers $q_0, \ldots, q_{n-1}, r_1, \ldots r_m, s_1, \ldots, s_k$ with $k = 3p$:

- Let $I = \{j \mid e_j \text{ is an outgoing edge of the constant gate } c_1 \text{ or a nand-gate}\}$. For $0 \leq i \leq n-1$ we define the number q_i as

$$q_i = \sum_{j \in I \setminus \{i\}} 4^j.$$

  Recall that e_i is the unique outgoing edge of the output gate y_i.
- If e_j is the unique outgoing edge of the input gate x_i then we set $r_i = 4^j$. We can choose the reverse topological sorting of the nand-gates in such a way that $r_1 > r_2 > \cdots > r_m$ (we only have to ensure that the target gates $\overline{x}_1, \ldots, \overline{x}_m$ of the input gates appear in the order $\overline{x}_m, \ldots, \overline{x}_1$ in the reverse topological sorting of the nand-gates).
- To define the numbers $s_1, \ldots, s_k$ we first define for every nand-gate g_i three numbers t_{3i}, t_{3i-1} and t_{3i-2} as follows, where $I_i = \{j \mid e_j \text{ is an outgoing edge of gate } g_i\}$:

$$\begin{aligned} t_{3i} &= 4^{2i+n-1} + 4^{2i+n-2} + \sum_{j \in I_i} 4^j \\ t_{3i-1} &= 4^{2i+n-1} - 4^{2i+n-2} = 3 \cdot 4^{2i+n-2} \\ t_{3i-2} &= 4^{2i+n-2} \end{aligned}$$

  Then, the tuple $(s_1, \ldots, s_k)$ is $(t_{3p}, t_{3p-1}, t_{3p-2}, \ldots, t_3, t_2, t_1)$, which is indeed super-decreasing (see also [30]). In fact, we have $s_i - (s_{i+1} + \cdots + s_k) \geq 4^{n-1}$ for all $i \in [1..k]$. To see this, note that the sets $I_{i+1}, \ldots, I_k$ are pairwise disjoint. This implies that the $n-1$ low-order digits in the base-4 expansion of $s_{i+1} + \cdots + s_k$ are zero or one.

In order to understand this construction, one should think of the edges of the circuit carrying truth values. Recall that there are $2p+n$ edges in the circuit (including the imaginary outgoing edges of the output gates $y_0, \ldots, y_{n-1}$). A number in base-4 representation with $2p+n$ digits that are either 0 or 1 represents a truth assignment to the $2p+n$ edges, where a 1-digit represents the truth value 1 and a 0-digit represents the truth value 0. Consider an input string $\alpha = b_1 \cdots b_m \in \{0,1\}^m$ and consider an output gate y_i, $i \in [0..n-1]$. Then the number $N := 4^i + q_i + b_1 r_1 + \cdots + b_m r_m$ encodes the truth assignment for the circuit edges, where:

- all outgoing edges of the constant gate c_1 carry the truth value 1,
- all outgoing edges of the constant gate c_0 carry the truth value 0,
- the unique outgoing edge of an input gate x_i carries the truth value b_i,
- all outgoing edges of nand-gates carry the truth value 1.

We have to show that $C(\alpha)_i = 1$ if and only if there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = N - 4^i$. For this we apply the canonical algorithm for super-decreasing subsetsum with input $(N, s_1, \ldots, s_k)$. This algorithm initializes a counter A to N and then goes over the sequence $s_1, \ldots, s_k$ in that order. In the j-th step $(1 \leq j \leq k)$ we set A to $A - s_j$ if $A \geq s_j$. If $A < s_j$ then we do not modify A. After that we proceed with s_{j+1}. The point is that this process simulates the evaluation of the circuit on the input values $b_1, \ldots, b_m$. Thereby the nand-gates are evaluated in the topological order $g_p, g_{p-1}, \ldots, g_1$. Assume that g_j is the gate that we want to evaluate next. In the above algorithm for super-decreasing subsetsum the evaluation of g_j is simulated by the three numbers t_{3j}, t_{3j-1}, and t_{3j-2}. At the point where the algorithm checks whether t_{3j} can be subtracted from A, the base-4 digits at positions $2j+n, \ldots, 2p+n-1$ in the counter value A have been already set to zero. If the digits at the next two high-order positions $2j+n-1$ and $2j+n-2$ are still 1 (i.e., the input edges e_{2j+n-2} and e_{2j+n-1} for gate g_j carry the truth value 1), then we can subtract t_{3j} from A. Thereby we subtract all powers 4^{2j+n-1}, 4^{2j+n-2} and 4^h, where e_h is an outgoing edge for gate g_j. Since gate g_j evaluates to zero (both input edges carry 1), this subtraction correctly simulates the evaluation of gate g_j: all outgoing edges e_h of g_j (that were initially set to the truth value 1) are set to the truth value 0. On the other hand, if one of the two digits at positions $2j+n-1$ and $2j+n-2$ in A is 0 (which means that gate g_j evaluates to 1), then we cannot subtract t_{3j} from A. If both digits at positions $2j+n-1$ and $2j+n-2$ in A are 0, then also t_{3j-1} and t_{3j-2} cannot be subtracted. On the other hand, if exactly one of the two digits at positions $2j+n-1$ and $2j+n-2$ is 1, then with t_{3j-1} and t_{3j-2} we can set these two digits to 0 (thereby digits at positions $< 2j+n-2$ are not modified).

Assume now that y_j $(j \in [0..n-1])$ is the unique output gate that evaluates to 1, i.e., all output gates $y_{j'}$ with $j' \neq j$ evaluate to zero. Then after processing all weights $s_1, \ldots, s_k$ we have $A = 4^j$ (we will never subtract 4^j). We have shown that there exists $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} + 4^j = N$. Hence, if $i = j$ (i.e., $C(\alpha)_i = 1$) then $\delta \cdot \overline{s} = N - 4^i$. Now assume that $i \neq j$. In order to get a contradiction, assume that there is $\delta' \in \{0,1\}^k$ such that $\delta' \cdot \overline{s} + 4^i = N$. We have $\delta \neq \delta'$ and $\delta \cdot \overline{s} + 4^j = \delta' \cdot \overline{s} + 4^i$, i.e, $\delta \cdot \overline{s} - \delta' \cdot \overline{s} = 4^i - 4^j$. Since $i, j \in [0..n-1]$, we get $|\delta \cdot \overline{s} - \delta' \cdot \overline{s}| < 4^{n-1}$. But $s_i - (s_{i+1} + \cdots s_k) \geq 4^{n-1}$ for all $i \in [1..k]$ implies that $|\delta \cdot \overline{s} - \delta' \cdot \overline{s}| \geq 4^{n-1}$.

C.3 From super-decreasing subsetsum to straight-line programs

In [35] a super-decreasing sequence $\overline{t} = (t_1, \ldots, t_k)$ of natural numbers is encoded by the string $S(\overline{t}) \in \{0,1\}^*$ of length $\sum \overline{t} + 1$ such that for all $0 \leq p \leq \sum \overline{t}$:

$$S(\overline{t})[p] = \begin{cases} 1 & \text{if } p = \alpha \cdot \overline{t} \text{ for some } \alpha \in \{0,1\}^k, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Note that in the first case α is unique. Since $\overline{t}$ is a super-decreasing sequence, the number of 1's in the string $S(\overline{t})$ is 2^k. Also note that $S(\overline{t})$ starts and ends with 1. In [35] it was shown that from a super-decreasing sequence $\overline{t}$ of binary encoded numbers one can construct in LOGSPACE an SLP for the word $S(\overline{t})$.

C.4 Proof of Theorem 6

Let us fix a regular wreath product of the form $G \wr \mathbb{Z}$ for a finitely generated group G. Such groups are also known as generalized lamplighter groups (the lamplighter group arises for $G = \mathbb{Z}_2$). Throughout this section, we fix a set of standard generators Σ for G and let $\tau = 1$ be the generator for $\mathbb{Z}$. Then $\Sigma \cup \{\tau, \tau^{-1}\}$ is a standard generating set for the wreath product $G \wr \mathbb{Z}$. In $G \wr \mathbb{Z}$ the G-generator $a \in \Sigma$ represents the mapping $f_a \in G^{(\mathbb{Z})}$ with $f_a(0) = a$ and $f_a(z) = 1$ for $z \neq 0$. For a word $w \in (\Sigma \cup \{\tau, \tau^{-1}\})^*$ we define $\eta(w) := |w|_\tau - |w|_{\tau^{-1}}$. Thus, the element of $G \wr \mathbb{Z}$ represented by w is of the form $f\tau^{\eta(w)}$ for some $f \in G^{(\mathbb{Z})}$. Recall the definition of the left action of $\mathbb{Z}$ on $G^{(\mathbb{Z})}$ from Section 2 (where we take $H = Y = \mathbb{Z}$). For better readability, we write $c \circ f$ for ${}^c f$ ($c \in \mathbb{Z}$, $f \in G^{(\mathbb{Z})}$). Hence, we have $(c \circ f)(z) = f(z+c)$. If one thinks of f as a bi-infinite word over the alphabet G, then $c \circ f$ is the same word but shifted by $-c$.

The following intuition might be helpful: Consider a word $w \in (\Sigma \cup \{\tau, \tau^{-1}\})^*$. In $G \wr \mathbb{Z}$ we can simplify w to a word of the form $\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k}$ (with $z_j \in \mathbb{Z}$, $a_j \in \Sigma$), which in $G \wr \mathbb{Z}$ can be rewritten as

$$\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k} = \Big(\prod_{j=1}^{k} \tau^{z_0+\cdots+z_{j-1}} a_j \tau^{-(z_0+\cdots+z_{j-1})} \Big)\, \tau^{z_0+\cdots+z_k}.$$

Hence, the word w represents the group element

$$\Big(\prod_{j=1}^{k} (z_0 + \cdots + z_{j-1}) \circ f_{a_j} \Big)\, \tau^{z_0+\cdots+z_k}.$$

This gives the following intuition for evaluating $\tau^{z_0} a_1 \tau^{z_1} a_2 \cdots \tau^{z_{k-1}} a_k \tau^{z_k}$: In the beginning, every $\mathbb{Z}$-position carries the G-value 1. First, go to the $\mathbb{Z}$-position $-z_0$ and multiply the G-element at this position with a_1 (on the right), then go to the $\mathbb{Z}$-position $-z_0 - z_1$ and multiply the G-element at this position with a_2, and so on.

Proof of Theorem 6. The easy part is to show that the circuit value problem for $G \wr \mathbb{Z}$ belongs to $\forall\mathsf{LEAF}(\mathrm{WP}(G))$. In the following, we make use of the statements from Lemma 32. Let $\mathcal{G}$ be an SLP over the alphabet $\Sigma \cup \{\tau, \tau^{-1}\}$ and let $f\tau^{\eta(\mathrm{val}(\mathcal{G}))} \in G \wr \mathbb{Z}$ be the group element represented by $\mathrm{val}(\mathcal{G})$. By Lemma 32 we can compute $\eta(\mathrm{val}(\mathcal{G}))$ in polynomial time. If $\eta(\mathrm{val}(\mathcal{G})) \neq 0$ then the Turing-machine rejects by printing a non-trivial generator of G (here we need the assumption that G is non-trivial). So, let us assume that $\eta(\mathrm{val}(\mathcal{G})) = 0$. We can also compute in polynomial time two integers $b, c \in \mathbb{Z}$ such that $\mathrm{supp}(f) \subseteq [b..c]$. We can take for instance $b = -|\mathrm{val}(\mathcal{G})|$ and $c = |\mathrm{val}(\mathcal{G})|$. It suffices to check whether for all $x \in [b..c]$ we have $f(x) = 1$. For this, the Turing-machine branches universally to all binary encoded integers $x \in [b..c]$ (this yields the $\forall$-part in $\forall\mathsf{LEAF}(\mathrm{WP}(G))$). Consider a specific branch that leads to the integer $x \in [b..c]$. From x and the input SLP $\mathcal{G}$ the Turing-machine then produces a leaf string over the standard generating set Σ of G such that this leaf string represents the group element $f(x) \in G$. For this, the machine branches to all positions $p \in [0..|\mathrm{val}(\mathcal{G})|-1]$ (if $p < q < |\mathrm{val}(\mathcal{G})|$ then the branch for p is to the left of the branch for q). For a specific position p, the machine computes in polynomial time the symbol $a = \mathrm{val}(\mathcal{G})[p]$. If a is τ or τ^{-1} then the machine prints $1 \in \Sigma$. On the other hand, if $a \in \Sigma$ then the machine computes in polynomial time $d = \eta(\mathrm{val}(\mathcal{G})[:p])$. This is possible by first computing an SLP for the prefix $\mathrm{val}(\mathcal{G})[:p]$. If $d = -x$ then the machine prints the symbol a, otherwise the machine prints the trivial generator 1. It is easy to observe that the leaf string produced in this way represents the group element $f(x)$.

Let us now prove hardness for $\forall\mathsf{LEAF}(\mathrm{WP}(G/Z(G)))$ with respect to $\mathsf{LOGSPACE}$-reductions. By Lemma 29 it suffices to show that $\mathrm{CVP}(G \wr \mathbb{Z})$ is hard for the balanced class $\forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$. Let $a_0, \ldots, a_{n-1}$ be an arbitrary enumeration of the standard generators in Σ. Fix a language $L \in \forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$. From the definition of the class $\forall\mathsf{bLEAF}(\mathrm{WP}(G/Z(G)))$ it follows that there exist two polynomials p_1 and p_2 and a balanced polynomial time NTM M running in time $p_1 + p_2$ that outputs a symbol from Σ after termination and such that the following holds: Consider an input word z for M and let $m_1 = p_1(|z|)$, $m_2 = p_2(|z|)$, $m = m_1 + m_2$, and $T(z)$ be the corresponding computation tree of M on input z. It is a complete binary tree of height m. Its node set can be identified with the bit strings in $\{0,1\}^{\leq m}$, where $v0$ (resp., $v1$) is the left (resp., right) child of $v \in \{0,1\}^{<m}$. For every leaf $\alpha \in \{0,1\}^m$ let us denote with $\lambda(\alpha)$ the symbol from Σ that M prints when reaching the leaf α. Then we have: $z \in L$ if and only if for all $\beta \in \{0,1\}^{m_1}$ the string

$$\lambda_\beta := \prod_{\gamma \in \{0,1\}^{m_2}} \lambda(\beta\gamma) \tag{4}$$

represents a group element from the center $Z(G)$. Here (and in the following), the product in the right-hand side of (4) goes over all bit strings of length m_2 in lexicographic order. Our construction consists of five steps:

Step 1. Note that given a bit string $\alpha \in \{0,1\}^m$, we can compute in polynomial time the symbol $\lambda(\alpha) \in \Sigma$ by following the computation path specified by α. Using the classical Cook-Levin construction (see e.g. [2]), we can compute from the input z and $a \in \Sigma$ in $\mathsf{LOGSPACE}$ a Boolean circuit $C_{z,a}$ with m input gates $x_1, \ldots, x_m$ and a single output gate y_0 such that for all $\alpha \in \{0,1\}^m$: $C_{z,a}(\alpha)_0 = 1$ if and only if $\lambda(\alpha) = a$. By taking the disjoint union of these circuits and merging the input gates, we can build a single circuit C_z with m input gates $x_1, \ldots, x_m$ and $n = |\Sigma|$ output gates $y_0, \ldots, y_{n-1}$. For every $\alpha \in \{0,1\}^m$ and every $0 \leq i \leq n-1$ the following holds: $C_z(\alpha)_i = 1$ if and only if $\lambda(\alpha) = a_i$.

Step 2. Using the construction from Appendix C.2 we can compute from the circuit C_z in $\mathsf{LOGSPACE}$ numbers $q_0, \ldots, q_{n-1} \in \mathbb{N}$ and two super-decreasing sequences $\overline{r} = (r_1, \ldots, r_m)$ and $\overline{s} = (s_1, \ldots, s_k)$ with the following properties:

- The $r_1, \ldots, r_m$ are pairwise distinct powers of 4.
- For all $0 \leq i \leq n-1$ and all $\alpha \in \{0,1\}^m$ we have: $\lambda(\alpha) = a_i$ if and only if $C_z(\alpha)_i = 1$ if and only if there is $\delta \in \{0,1\}^k$ such that $\delta \cdot \overline{s} = q_i + \alpha \cdot \overline{r}$.

Note that for all $\alpha \in \{0,1\}^m$ there is a unique i such that $C_z(\alpha)_i = 1$. Hence, for all $\alpha \in \{0,1\}^m$ there is a unique i such that $q_i + \alpha \cdot \overline{r}$ is of the form $\delta \cdot \overline{s}$ for some $\delta \in \{0,1\}^k$. For this unique i we have $\lambda(\alpha) = a_i$.

We split the super-decreasing sequence $\overline{r} = (r_1, \ldots, r_m)$ into the two sequences $\overline{r}_1 = (r_1, \ldots, r_{m_1})$ and $\overline{r}_2 = (r_{m_1+1}, \ldots, r_m)$. For the following consideration, we need the following numbers:

$$\ell = \max\left\{ \textstyle\sum \overline{r}_1 + \max\{q_0, \ldots, q_{n-1}\} + 1,\ \sum \overline{s} - \sum \overline{r}_2 - \min\{q_0, \ldots, q_{n-1}\} + 1 \right\} \tag{5}$$

$$\pi = \ell + \textstyle\sum \overline{r}_2 \tag{6}$$

The binary encodings of these numbers can be computed in $\mathsf{LOGSPACE}$ (since iterated addition, max, and min can be computed in $\mathsf{LOGSPACE}$). The precise value of ℓ will be only relevant at the end of step 4.

Step 3. By the result from [35] (see Appendix C.3) we can construct in LOGSPACE from the three super-decreasing sequences $\overline{r}_1, \overline{r}_2$ and $\overline{s}$ three SLPs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{H}$ over the alphabet $\{0,1\}$ such that $\mathrm{val}(\mathcal{G}_1) = S(\overline{r}_1)$, $\mathrm{val}(\mathcal{G}_2) = S(\overline{r}_2)$ and $\mathrm{val}(\mathcal{H}) = S(\overline{s})$ (see (3)). For all positions $p \geq 0$ (in the suitable range) we have:

$$\begin{aligned}
\mathrm{val}(\mathcal{G}_1)[p] = 1 &\iff \exists \beta \in \{0,1\}^{m_1} : p = \beta \cdot \overline{r}_1 \\
\mathrm{val}(\mathcal{G}_2)[p] = 1 &\iff \exists \gamma \in \{0,1\}^{m_2} : p = \gamma \cdot \overline{r}_2 \\
\mathrm{val}(\mathcal{H})[p] = 1 &\iff \exists \delta \in \{0,1\}^{k} : p = \delta \cdot \overline{s}
\end{aligned}$$

Note that $|\mathrm{val}(\mathcal{G}_1)| = \sum \overline{r}_1 + 1$, $|\mathrm{val}(\mathcal{G}_2)| = \sum \overline{r}_2 + 1$, and $|\mathrm{val}(\mathcal{H})| = \sum \overline{s} + 1$.

Step 4. We build in LOGSPACE for every $0 \leq i \leq n-1$ an SLP $\mathcal{H}_i$ from the SLP $\mathcal{H}$ by replacing in every right-hand side of $\mathcal{H}$ every occurrence of 0 by τ^{-1} and every occurrence of 1 by $a_i\tau^{-1}$. Let T_i be the start variable of $\mathcal{H}_i$, let S_1 be the start variable of $\mathcal{G}_1$, and let S_2 be the start variable of $\mathcal{G}_2$. We can assume that the variable sets of the SLPs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_0, \ldots, \mathcal{H}_{n-1}$ are pairwise disjoint. We next combine these SLPs into a single SLP $\mathcal{I}$. The variables of $\mathcal{I}$ are the variables of the SLPs $\mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_0, \ldots, \mathcal{H}_{n-1}$ plus a fresh variable S which is the start variable of $\mathcal{I}$. The right-hand sides for the variables are defined below. In the right-hand sides we write powers τ^p for integers p whose binary encodings can be computed in LOGSPACE. Such powers can be produced by small subSLPs that can be constructed in LOGSPACE too.

- In all right-hand sides of $\mathcal{G}_1$ and $\mathcal{G}_2$ we replace all occurrences of the terminal symbol 0 by the $\mathbb{Z}$-generator τ.
- We replace every occurrence of the terminal symbol 1 in a right-hand side of $\mathcal{G}_1$ by $S_2\tau^{\ell}$, where ℓ is from (5).
- We replace every occurrence of the terminal symbol 1 in a right-hand side of $\mathcal{G}_2$ by $\sigma\tau$, where

$$\sigma = \tau^{q_0} T_0 \tau^{h-q_0} \tau^{q_1} T_1 \tau^{h-q_1} \cdots \tau^{q_{n-1}} T_{n-1} \tau^{h-q_{n-1}} \tag{7}$$

  and $h = \sum \overline{s} + 1$ is the length of the word $\mathrm{val}(\mathcal{H})$ (which is $-\eta(\mathrm{val}_{\mathcal{I}}(T_i))$ for every $i \in [0..n-1]$). Note that $\eta(\mathrm{val}_{\mathcal{I}}(\sigma)) = 0$.
- Finally, the right-hand side of the start variable S is $S_1\tau^{-d}$ where $d := \sum \overline{r}_1 + 1 + 2^{m_1} \cdot \pi$. (note that $d = \eta(\mathrm{val}_{\mathcal{I}}(S_1))$).

Before we explain this construction, let us first introduce some notations.

- Let $u := \mathrm{val}_{\mathcal{I}}(S_2)$. We have $\eta(u) = |\mathrm{val}(\mathcal{G}_2)|$. Hence, the group element represented by u can be written as $f_u\tau^{|\mathrm{val}(\mathcal{G}_2)|}$ for a mapping $f_u \in G^{(\mathbb{Z})}$.
- Let $v := \mathrm{val}_{\mathcal{I}}(\sigma)$, where σ is from (7). Note that $\eta(v) = 0$. Hence, the group element represented by v is a mapping $f_v \in G^{(\mathbb{Z})}$. Its support is a subset of the interval from position $-\max\{q_0, \ldots, q_{n-1}\}$ to position $\sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\}$.
- For $\beta \in \{0,1\}^{m_1}$ let $\mathsf{bin}(\beta)$ be the number represented by β in binary notation (thus, $\mathsf{bin}(0^{m_1}) = 0$, $\mathsf{bin}(0^{m_1-1}1) = 1$, …, $\mathsf{bin}(1^{m_1}) = 2^{m_1} - 1$). Moreover, let

$$p_\beta := -\mathsf{bin}(\beta) \cdot \pi.$$

First, note that $\eta(\mathrm{val}(\mathcal{I})) = 0$. This is due to the factor τ^{-d} in the right-hand side of the start variable S of $\mathcal{I}$. Hence, the group element represented by $\mathrm{val}(\mathcal{I})$ is a mapping $f \in G^{(\mathbb{Z})}$. The crucial claim is the following:

▷ Claim. For every $\beta \in \{0,1\}^{m_1}$, $f(p_\beta)$ is the group element represented by the leaf string λ_β from (4).

Proof of the claim. In the following, we compute in the restricted direct product $G^{(\mathbb{Z})}$. Recall that the multiplication in this group is defined by the pointwise multiplication of mappings.

Since we replaced in $\mathcal{G}_1$ every 1 in a right-hand side by $S_2\tau^\ell$, which produces $u\tau^\ell$ in $\mathcal{I}$ (which evaluates to $f_u\tau^{\pi+1}$), the mapping f is a product (in the restricted direct product $G^{(\mathbb{Z})}$) of shifted copies of f_u. More precisely, for every $\beta' \in \{0,1\}^{m_1}$ we get the shifted copy

$$\big(\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi\big) \circ f_u \tag{8}$$

of f_u. The shift distance $\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi$ can be explained as follows: The 1 in $\mathrm{val}(\mathcal{G}_1)$ that corresponds to $\beta' \in \{0,1\}^{m_1}$ occurs at position $\beta' \cdot \overline{r}_1$ (the first position is 0) and to the left of this position we find $\mathsf{bin}(\beta')$ many 1's and $\beta' \cdot \overline{r}_1 - \mathsf{bin}(\beta')$ many 0's in $\mathrm{val}(\mathcal{G}_1)$. Moreover, every 0 in $\mathrm{val}(\mathcal{G}_1)$ was replaced by τ (shift by 1) and every 1 in $\mathrm{val}(\mathcal{G}_1)$ was replaced by $u\tau^\ell$ (shift by $\ell + |\mathrm{val}(\mathcal{G}_2)| = \pi + 1$). Hence, the total shift distance is indeed (8). Also note that if $\beta' \in \{0,1\}^{m_1}$ is lexicographically smaller than $\beta'' \in \{0,1\}^{m_1}$ then $\beta' \cdot \overline{r}_1 < \beta'' \cdot \overline{r}_1$. This implies that

$$f = \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 + \mathsf{bin}(\beta') \cdot \pi\big) \circ f_u = \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ f_u.$$

Let us now compute the mapping f_u. Recall that we replaced in $\mathcal{G}_2$ every occurrence of 1 by $\sigma\tau$, where σ is from (7) and derives to v. The 1's in $\mathrm{val}(\mathcal{G}_2)$ occur at positions of the form $\gamma \cdot \overline{r}_2$ for $\gamma \in \{0,1\}^{m_2}$ and if $\gamma \in \{0,1\}^{m_2}$ is lexicographically smaller than $\gamma' \in \{0,1\}^{m_2}$ then $\gamma \cdot \overline{r}_2 < \gamma' \cdot \overline{r}_2$. We therefore get

$$f_u = \prod_{\gamma \in \{0,1\}^{m_2}} (\gamma \cdot \overline{r}_2) \circ f_v.$$

We obtain

$$\begin{aligned} f &= \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ f_u \\ &= \prod_{\beta' \in \{0,1\}^{m_1}} \big(\beta' \cdot \overline{r}_1 - p_{\beta'}\big) \circ \prod_{\gamma \in \{0,1\}^{m_2}} (\gamma \cdot \overline{r}_2 \circ f_v) \\ &= \prod_{\beta' \in \{0,1\}^{m_1}} \prod_{\gamma \in \{0,1\}^{m_2}} \big(\beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 - p_{\beta'}\big) \circ f_v \end{aligned}$$

and hence

$$f(p_\beta) = \prod_{\beta' \in \{0,1\}^{m_1}} \prod_{\gamma \in \{0,1\}^{m_2}} f_v(p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2).$$

We claim that for all $\beta \neq \beta'$ and all $\gamma \in \{0,1\}^{m_2}$ we have

$$f_v(p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2) = 1. \tag{9}$$

Let us postpone the proof of this for a moment. From (9) we get

$$f(p_\beta) = \prod_{\gamma \in \{0,1\}^{m_2}} f_v(\beta \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2).$$

Consider a specific $\gamma \in \{0,1\}^{m_2}$ and let $\alpha = \beta\gamma$ and $p = \beta \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 = \alpha \cdot \overline{r}$. From the definition of $v = \mathrm{val}_{\mathcal{I}}(\sigma)$ it follows that for all $x \in \mathbb{Z}$, $f_v(x)$ is a product of those group generators a_i such that $x = -q_i + \delta \cdot \overline{s}$ for some $\delta \in \{0,1\}^k$. For the position p this means

that $q_i + \alpha \cdot \overline{r} = \delta \cdot \overline{s}$. By our previous remarks, there is a unique such $i \in [0..n-1]$ and for this i we have $\lambda(\alpha) = a_i$. Hence, we obtain $f_v(p) = \lambda(\alpha) = \lambda(\beta\gamma)$ and thus

$$f(p_\beta) = \prod_{\gamma \in \{0,1\}^{m_2}} \lambda(\beta\gamma) = \lambda_\beta.$$

It remains to show (9). To get this identity, we need the precise value of ℓ from (5) (so far, the value of ℓ was not relevant). Assume now that $\beta \neq \beta'$, which implies

$$|p_\beta - p_{\beta'}| \geq \pi = \ell + \sum \overline{r}_2.$$

Hence, we either have

$$\begin{aligned} p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 &\geq \ell + \sum \overline{r}_2 + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 \\ &\geq \ell + \sum \overline{r}_2 \\ &> \sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\} \end{aligned}$$

or

$$\begin{aligned} p_\beta - p_{\beta'} + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 &\leq -\ell - \sum \overline{r}_2 + \beta' \cdot \overline{r}_1 + \gamma \cdot \overline{r}_2 \\ &\leq -\ell + \sum \overline{r}_1 \\ &< -\max\{q_0, \ldots, q_{n-1}\}, \end{aligned}$$

where the strict inequalities follow from our choice of ℓ. Recall that the support of the mapping f_v is contained in $[-\max\{q_0, \ldots, q_{n-1}\}.. \sum \overline{s} - \min\{q_0, \ldots, q_{n-1}\}]$. This shows (9) and hence the claim. ◁

Step 5. By the above claim, we have $f(p_\beta) \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$ if and only if $\lambda_\beta \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$, which is equivalent to $z \in L$. The only remaining problem is that the word $\mathrm{val}(\mathcal{I})$ produces some "garbage" group elements $f(x)$ on positions x that are not of the form p_β. Note that for every $g \in G \setminus Z(G)$, there is a generator $a_i \in \Sigma$ such that the commutator $[g, a_i]$ is non-trivial. We now produce from $\mathcal{I}$ an SLP $\mathcal{I}^{-1}$ such that $\mathrm{val}(\mathcal{I}^{-1})$ represents the inverse element of $f \in G^{(\mathbb{Z})}$, which is the mapping g with $g(x) = f(x)^{-1}$ for all $x \in \mathbb{Z}$. To construct $\mathcal{I}^{-1}$, we have to reverse every right-hand side of $\mathcal{I}$ and replace every occurrence of a symbol $a_0, \ldots, a_{n-1}, \tau, \tau^{-1}$ by its inverse.

It is easy to compute in LOGSPACE for every $i \in [0..n-1]$ an SLP for the word

$$w_i := \left(a_i \tau^\pi\right)^{2^{m_1}} \tau^{-2^{m_1} \cdot \pi}.$$

Then the group element represented by w_i is the mapping $f_i \in G^{(\mathbb{Z})}$ whose support is the set of positions p_β for $\beta \in \{0,1\}^{m_1}$ and $f_i(p_\beta) = a_i$ for all $\beta \in \{0,1\}^{m_1}$. We can also compute in LOGSPACE an SLP for the word w_i^{-1}. We then built in LOGSPACE SLPs $\mathcal{J}_0, \ldots, \mathcal{J}_{n-1}$ such that $\mathrm{val}(\mathcal{J}_i) = \mathrm{val}(\mathcal{I}^{-1}) w_i^{-1} \mathrm{val}(\mathcal{I}) w_i$. Hence, the word $\mathrm{val}(\mathcal{J}_i)$ represents the group element $g_i \in G^{(\mathbb{Z})}$, where $g_i(x) = 1$ for all $x \in \mathbb{Z} \setminus \{p_\beta \mid \beta \in \{0,1\}^{m_1}\}$ and $g_i(p_\beta) = f(p_\beta)^{-1} a_i^{-1} f(p_\beta) a_i = [f(p_\beta), a_i]$.

Finally, we construct in LOGSPACE an SLP $\mathcal{J}$ such that

$$\mathrm{val}(\mathcal{J}) = \mathrm{val}(\mathcal{J}_0)\, \tau\, \mathrm{val}(\mathcal{J}_1)\, \tau\, \mathrm{val}(\mathcal{J}_2) \cdots \tau\, \mathrm{val}(\mathcal{J}_{n-1})\, \tau^{-n+1}.$$

We can assume that $n \leq \ell + \sum \overline{r}_2 = \pi$ (n is a constant and we can always make ℓ bigger). Then $\mathrm{val}(\mathcal{J})$ evaluates to the group element $g \in G^{(\mathbb{Z})}$ with $g(x) = 1$ for $x \in \mathbb{Z} \setminus \{p_\beta - i \mid \beta \in \{0,1\}^{m_1}, 0 \leq i \leq n-1\}$ and $g(p_\beta - i) = g_i(p_\beta) = [f(p_\beta), a_i]$ for $0 \leq i \leq n-1$. Hence, if $f(p_\beta) \in Z(G)$ for all $\beta \in \{0,1\}^{m_1}$ then $\mathrm{val}(\mathcal{J}) = 1$ in $G \wr \mathbb{Z}$. On the other hand, if there is a $\beta \in \{0,1\}^{m_1}$ such that $f(p_\beta) \in G \setminus Z(G)$ then there is an a_i such that $[f(p_\beta), a_i] \neq 1$. Hence $g(p_\beta - i) \neq 1$ and $\mathrm{val}(\mathcal{J}) \neq 1$ in $G \wr \mathbb{Z}$. This concludes the proof of Theorem 6. ◀

Optimal Lower Bounds for Matching and Vertex Cover in Dynamic Graph Streams

Jacques Dark
Department of Computer Science, University of Warwick, Coventry, UK
j.dark@warwick.ac.uk

Christian Konrad
Department of Computer Science, University of Bristol, UK
http://people.cs.bris.ac.uk/~konrad/
christian.konrad@bristol.ac.uk

Abstract

In this paper, we give simple optimal lower bounds on the one-way two-party communication complexity of approximate Maximum Matching and Minimum Vertex Cover with deletions. In our model, Alice holds a set of edges and sends a single message to Bob. Bob holds a set of edge deletions, which form a subset of Alice's edges, and needs to report a large matching or a small vertex cover in the graph spanned by the edges that are not deleted. Our results imply optimal space lower bounds for insertion-deletion streaming algorithms for Maximum Matching and Minimum Vertex Cover.

Previously, Assadi et al. [SODA 2016] gave an optimal space lower bound for insertion-deletion streaming algorithms for Maximum Matching via the simultaneous model of communication. Our lower bound is simpler and stronger in several aspects: The lower bound of Assadi et al. only holds for algorithms that (1) are able to process streams that contain a triple exponential number of deletions in n, the number of vertices of the input graph; (2) are able to process multi-graphs; and (3) never output edges that do not exist in the input graph when the randomized algorithm errs. In contrast, our lower bound even holds for algorithms that (1) rely on short ($O(n^2)$-length) input streams; (2) are only able to process simple graphs; and (3) may output non-existing edges when the algorithm errs.

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Communication complexity; Theory of computation → Lower bounds and information complexity

Keywords and phrases Maximum matching, Minimum vertex cover, Dynamic graph streams, Communication complexity, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.30

Funding *Jacques Dark*: Jacques Dark was supported by an EMEA Microsoft Research studentship and European Research Council grant ERC-2014-CoG 647557.

Acknowledgements The authors are grateful for the numerous excellent comments and suggestions from an anonymous reviewer.

1 Introduction

Streaming algorithms for processing massive graphs have been studied for two decades [16]. In the most traditional setting, the *insertion-only model*, an algorithm receives a sequence of the edges of the input graph in arbitrary order, and the objective is to solve a graph problem using as little space as possible. The insertion-only model has received significant attention, and many problems, such as matchings (e.g. [26, 13, 21, 27, 18, 24, 31, 12]), independent sets (e.g. [15, 14, 9, 10]), and subgraph counting (e.g. [20, 11, 7]), have since been studied in this model. See [29] for an excellent survey.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 30; pp. 30:1–30:14

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

COMPUTATIONAL COMPLEXITY CONFERENCE

In 2012, Ahn et al. [1] introduced the first techniques for addressing *insertion-deletion* graph streams, where the input stream consists of a sequence of edge insertions and deletions. They showed that many problems, such as Connectivity and Bipartiteness, can be solved using the same amount of space as in insertion-only streams up to poly-logarithmic factors. Various other works subsequently gave results of a similar flavor and presented insertion-deletion streaming algorithms with similar space complexity as their insertion-only counterparts for problems including Spectral Sparsification [22] and $\Delta + 1$-coloring [3]. Konrad [23] and Assadi et al. [5] were the first to give a separation result between the insertion-only graph stream model and the insertion-deletion graph stream model: While it is known that a 2-approximation to Maximum Matching can be computed using space $\mathrm{O}(n \log n)$ in insertion-only streams, Konrad showed that space $\Omega(n^{\frac{3}{2}-4\epsilon})$ is required for an n^{ϵ}-approximation in insertion-deletion streams, and Assadi et al. gave a lower bound of $n^{2-3\epsilon-o(1)}$ for such an approximation. Assadi et al. also presented an $\tilde{\mathrm{O}}(n^{2-3\epsilon})$ space algorithm that matches their lower bound up to lower order terms, which establishes that their lower bound is optimal (a different algorithm that matches this lower bound is given by Chitnis et al. [8]).

Both Konrad and Assadi et al. exploit an elegant connection between insertion-deletion streaming algorithms and linear sketches. Ai et al. [2], building on the work of Yi et al. [28], showed that insertion-deletion graph streaming algorithms can be characterized as algorithms that essentially solely rely on the computation of linear sketches of the input stream. A consequence of this result is that space lower bounds for insertion-deletion streaming algorithms can also be proved in the *simultaneous model of communication*, since linear sketches can be implemented in this model. This provides an alternative to the more common approach of proving streaming lower bounds in the one-way model of communication. In particular, the lower bounds by Konrad and Assadi et al. are proved in the simultaneous model of communication.

From a technical perspective, this model has various attractive features, however, it comes with a major disadvantage: The characterization of Ai et al. only holds for insertion-deletion streaming algorithms that (1) are able to process "very long" input streams, i.e., input streams of triple exponential length in n, the number of vertices of the input graph, and (2) are able to process multi-graphs. In particular, this characterization does not hold for insertion-deletion streaming algorithms that rely on the assumption that input streams are short and the graph described by the input stream is always simple. Consequently, the lower bounds of Konrad and Assadi et al. do not hold for such algorithms.

Our Results. In this work, we prove an optimal space lower bound for Maximum Matching in insertion-deletion streams via the one-way two-party model of communication. Our lower bound construction yields insertion-deletion streams of length $\mathrm{O}(n^2)$ and does not involve multi-edges. Our lower bound therefore also holds for streaming algorithms that are designed for short input streams and simple graphs for which the characterization by Ai et al. does not hold. Furthermore, the optimal lower bound by Assadi et al. [5] only holds for streaming algorithms that never output non-existing edges when the (randomized) algorithms fail. We do not require this restriction.

Our lower bound method is simple and more widely applicable. Using the same method, we also give an optimal lower bound for Minimum Vertex Cover, showing that computing a n^{ϵ}-approximation requires $\Omega(n^{2-2\epsilon})$ space. Assadi and Khanna mention in [4] that the $n^{2-3\epsilon-o(1)}$ space lower bound for Maximum Matching given in [5] also applies to Minimum Vertex Cover. Our lower bound therefore improves on this result by a factor of $n^{\epsilon+o(1)}$. Furthermore, we show that our lower bound is optimal up to a factor of $\log n$: We give a very simple deterministic insertion-deletion streaming algorithm for Minimum Vertex Cover that uses space $\mathrm{O}(n^{2-2\epsilon} \log n)$.

While the main application of our lower bounds in the one-way two-party communication model are lower bounds for insertion-deletion graph streaming algorithms, we believe that our lower bounds are of independent interest. Indeed, the one-way two-party communication complexity of Maximum Matching without deletions has been addressed in [13], and our result can therefore also be understood as a generalization of their model to incorporate deletions.

The Simultaneous Model of Communication. The lower bounds by Konrad [23] and Assadi et al. [5] are proved in the simultaneous model of communication. In this model, a typically large number of parties k hold not necessarily disjoint subsets of the edges of the input graph. Each party P_i sends a message M_i to a referee, who then outputs the result of the protocol. The connection between insertion-deletion streaming algorithms and linear sketches by Ai et al. [2] then implies that a lower bound on the size of any message M_i yields a lower bound on the space requirements of any insertion-deletion streaming algorithm.

In the lower bound of Assadi et al. [5] for Maximum Matching, each party P_i holds the edges E_i of a dense subgraph, which itself constitutes a *Ruzsa-Szemerédi graph*, i.e., a graph whose edge set can be partitioned into large disjoint induced matchings. All previous streaming lower bounds for approximate Maximum Matching rely on realizations of Ruzsa-Szemerédi graphs [13, 23, 5]. Their construction is so that only a single induced matching of every party P_i is useful for the construction of a global large matching. Due to symmetry of the construction, the parties are unable to identify the important induced matching and therefore need to send large messages that contain information about most of the induced matchings to the referee for them to be able to compute a large global matching. Interestingly, none of the parties hold edge deletions in their construction.

The One-way Model of Communication. In this paper, we give a lower bound in the one-way two-party model of communication. In this model, Alice holds a set of edges E of the input graph and sends a message M to Bob. Bob holds a set of edge deletions $D \subseteq E$ and outputs a large matching in the graph spanned by the edges $E \setminus D$. A standard reduction shows that a lower bound on the size of message M also constitutes a lower bound on the space requirements of an insertion-deletion streaming algorithm. The two models are illustrated in Figure 1.

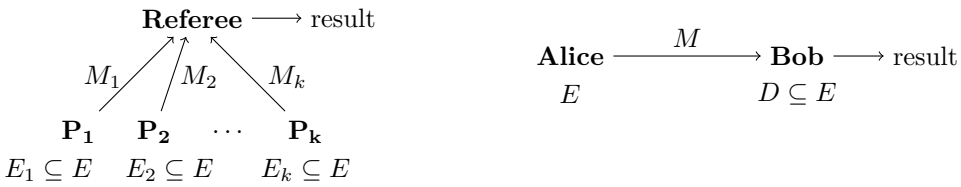


Figure 1 The simultaneous (left) and the one-way two-party (right) models of communication.

Our Techniques. To prove our lower bound, we identify that an insertion-deletion streaming algorithm for Maximum Matching or Minimum Vertex Cover can be used to obtain a one-way two-party communication protocol for a two-dimensional variant of the well-known Augmented Index problem that we denote by Augmented Bi-Index, or BInd in short. In an instance of BInd, Alice holds an n-by-n binary matrix $A \in \{0,1\}^{n \times n}$. Bob is given a position $(x, y) \in [n-k]^2$ and needs to output the bit $A_{x,y}$. Besides (x, y), he also knows the k-by-k submatrix of A with upper left corner at position (x, y), however with the bit at position (x, y) missing – we

will denote this k-by-k submatrix with (x, y) missing by $A_{S(x,y)}$. We show that this problem has a one-way communication complexity of $\Omega((n-k)^2)$ by giving a reduction from the Augmented Index problem.

To obtain a lower bound for Maximum Matching, we show that Alice and Bob can construct a protocol for BInd given an insertion-deletion streaming algorithm for Maximum Matching. In our reduction, we will consider instances with $k = n - \Theta(n^{1-\epsilon})$, for some $\epsilon > 0$. Consider the following attempt: Suppose that the input matrix A is a uniform random binary matrix and that $A_{x,y} = 1$ (we will get rid of these assumptions later). Alice and Bob interpret the matrix A as the incidence matrix of a bipartite graph G. Bob interprets the "1" entries in the submatrix $A_{S(x,y)}$ outside the diagonal, i.e., all "1" entries except those in positions $\{(x+j, y+j) \,:\, 0 \leq j < k\}$, as edge deletions F. The graph $G-F$ has a large matching: Since the diagonal of $A_{S(x,y)}$ is not deleted, and each entry in the diagonal is 1 with probability $1/2$, we expect that half of all potential edges in the diagonal of $S(x, y)$ are contained in $G-F$ and thus form a matching of size $\Theta(k) = \Theta(n - n^{1-\epsilon})$. An n^ϵ-approximation algorithm for Maximum Matching would therefore report $\Omega(n^{1-\epsilon})$ of these edges. Suppose that the algorithm reported $\Omega(n^{1-\epsilon})$ uniform random edges from the diagonal in $A_{S(x,y)}$ (we will also get rid of this assumption). Then, by repeating this scheme $\Theta(n^\epsilon)$ times in parallel, with large constant probability the edge corresponding to $A_{x,y}$ is reported at least once, which allows us to solve BInd. This reduction yields an optimal $\Omega(n^{2-3\epsilon})$ space lower bound for insertion-deletion streaming algorithm for Maximum Matching, since $\Theta(n^\epsilon)$ parallel executions are used to solve a problem that has a lower bound of $\Omega((n-k)^2) = \Omega(n^{2-2\epsilon})$.

In the description above, we assumed that (1) A is a uniform random binary matrix; (2) $A_{x,y} = 1$; and (3) the algorithm outputs uniform random positions from the diagonal of $A_{S(x,y)}$. To eliminate (1) and (2), Alice and Bob first sample a uniform random binary matrix $X \in \{0,1\}^{n \times n}$ from public randomness and consider the matrix obtained by computing the entry-wise XOR between A and X, i.e., matrix $A \oplus X$, instead. Observe that $A \oplus X$ is a uniform random binary matrix (independently of A), and with probability $\frac{1}{2}$, property (2), i.e., $(A \oplus X)_{x,y} = 1$, holds. Regarding assumption (3), besides computing the XOR $A \oplus X$, Alice and Bob also sample two random permutations $\sigma_1, \sigma_2 : [n] \to [n]$ from public randomness. Alice and Bob permute the rows and columns of $A \oplus X$ using σ_1 and σ_2, respectively. Then, no matter which elements from the permuted relevant diagonal of $A \oplus X$ are reported by the algorithm, due to the random permutations, these elements could have originated from any other position in this diagonal. This in turn makes every element along the diagonal equally likely to be reported, including the position (x, y) (in the unpermuted) matrix that we are interested in.

Our reduction for Minimum Vertex Cover is similar but simpler. We show that only a constant number of parallel executions of an insertion-deletion streaming are required.

Further Related Work. Hosseini et al. [17] were able to improve on the "triple exponential length" requirement of the input streams for a characterization of insertion-deletion streaming algorithms in terms of linear sketches by Li et al. [28] and Ai et al. [2]. They showed that in the case of XOR-streams and 0/1-output functions, input streams of length $O(n^2)$ are enough.

Very recently, Kallaugher and Price [19] showed that if either the stream length or the maximum value of the stream (e.g. the maximum multiplicity of an edge in a graph stream) are substantially restricted, then the characterization of turnstile streams as linear sketches cannot hold. For these situations they discuss problems where linear sketching is exponentially harder than turnstile streaming.

Besides the Maximum Matching problem, the only other separation result between the insertion-only and the insertion-deletion graph stream models that we are aware of is a recent result by Konrad [25], who showed that approximating large stars is significantly harder in insertion-deletion streams.

Outline. We give a lower bound on the communication complexity of Augmented Bi-Index in Section 2. Then, in Section 3, we show that a one-way two-party communication protocol for Maximum Matching can be used to solve Augmented Bi-Index, which yield an optimal space lower bound for Maximum Matching in insertion-deletion streams. We conclude with a similar reduction for Minimum Vertex Cover in Section 4, which also implies an optimal space lower bound for Minimum Vertex Cover in insertion-deletion streams.

2 Augmented Bi-Index

In this section, we define the one-way two-party communication problem Augmented Bi-Index and prove a lower bound on its communication complexity.

▶ **Problem 1** (Augmented Bi-Index). *In an instance of* Augmented Bi-Index $\mathsf{BInd}_\delta^{n,k}$ *we have two players denoted Alice and Bob. Alice holds a binary matrix* $A \in \{0,1\}^{n\times n}$*. Bob holds indices* $x, y \in [n-k]$ *and the incomplete*[1] *binary matrix* $A_{S(x,y)}$ *where*

$$S(x,y) = \{(i,j) \in [n]^2 \mid (x \le i < x+k) \text{ and } (y \le j < y+k)\} \setminus \{(x,y)\} .$$

Alice sends a single message M *to Bob who must output* $A_{x,y}$ *with probability at least* $1-\delta$*.*

Our lower bound proof consists of a reduction from the well-known Augmented Index problem, which is known to have large communication complexity.

▶ **Problem 2** (Augmented Index). *In an instance of* Augmented Index Ind_δ^n *we have two players denoted Alice and Bob. Alice holds a binary vector* $V \in \{0,1\}^n$*. Bob holds an index* $\ell \in [n]$ *and the vector suffix* $V_{>\ell} = (V_{\ell+1}, V_{\ell+2}, \cdots, V_n)$*. Alice sends a single message* M *to Bob who must output* V_ℓ *with probability at least* $1-\delta$*.*

As a consequence of Lemma 13 in [30], we can see that this problem has linear communication complexity (see also Lemma 2 in [6] for a more direct proof technique).

▶ **Theorem 3** (e.g. [30]). *For* $\delta < 1/3$*, any randomised one-way communication protocol which solves* Ind_δ^n *must communicate* $\Omega(n)$ *bits.*

We are now ready to prove our lower bound for Augmented Bi-Index.

▶ **Theorem 4.** *For* $\delta < 1/3$*, any randomised one-way communication protocol which solves* $\mathsf{BInd}_\delta^{n,k}$ *must communicate* $\Omega((n-k)^2)$ *bits.*

Proof. Let $\mathcal{P}$ be a communication protocol for $\mathsf{BInd}_\delta^{n,k}$ that uses messages of length at most $S(n,k)$ bits. We will show how $\mathcal{P}$ can be used to solve $\mathsf{Ind}_\delta^{(n-k)^2}$ with the same message size.

Let V, ℓ be any instance of $\mathsf{Ind}_\delta^{(n-k)^2}$. Alice builds the matrix $A \in \{0,1\}^{n\times n}$ by placing the bits of V in lexicographical order in the top-left $(n-k)$-by-$(n-k)$ region:

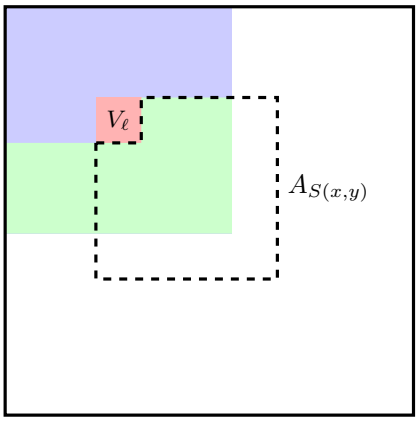
$$A_{i,j} = \begin{cases} V_{j+(n-k)(i-1)} & \text{for } i,j \in [n-k] \\ 0 & \text{otherwise} \end{cases} .$$

This packing is illustrated in Figure 2(a).

[1] We use A_S to refer to the collection of entries indexed by the set S, so $A_S = (A_{i,j})_{(i,j)\in S}$.

V_1	V_2	V_3	V_4	V_5	0	0	0	0
V_6	V_7	V_8	V_9	V_{10}	0	0	0	0
V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	0	0	0	0
V_{16}	V_{17}	V_{18}	V_{19}	V_{20}	0	0	0	0
V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(a) Example packing of the bits of V into matrix A with $n = 9$ and $k = 4$.



(b) Bob can construct the area $A_{S(x,y)}$ given $V_{>\ell}$, which is part of his input.

Figure 2 The construction of A and $A_{S(x,y)}$ in Theorem 4.

Alice runs protocol $\mathcal{P}$ on A and sends the resulting message M to Bob. Now, Bob has the message M, the index $\ell \in [(n-k)^2]$ and the suffix $V_{>\ell}$. Let $x, y \in [n-k]$ be the unique pair of integers such that $\ell = y + (n-k)(x-1)$. Observe that $A_{x,y} = V_\ell$.

For Bob to be able to complete protocol $\mathcal{P}$ he needs to provide $A_{S(x,y)}$. Because of the way we packed the entries of V onto A, the overlap between V and $A_{S(x,y)}$ is a subset of the entries of $V_{>\ell}$ (see Figure 2(b) for an illustration). Therefore Bob can complete the protocol and determine $A_{x,y} = V_\ell$ with probability at least $1 - \delta$. By Theorem 3, it must be that $S(n,k) = \Omega((n-k)^2)$. ◀

3 Maximum Matching

Let $\mathbf{A}$ be a C-approximation insertion-deletion streaming algorithm for Maximum Matching that errs with probability at most $1/10$. We will now show that $\mathbf{A}$ can be used to solve $\mathsf{BInd}_\delta^{n,k}$.

3.1 Reduction

Let $A \in \{0,1\}^{n \times n}, x \in [n-k]$ and $y \in [n-k]$ be an instance of $\mathsf{BInd}_\delta^{n,k}$. Alice and Bob first sample a uniform random binary matrix $X \in \{0,1\}^{n \times n}$ and random permutations $\sigma_1, \sigma_2 : [n] \to [n]$ from public randomness. Alice then computes matrix A' which is obtained by first computing the entry-wise XOR of A and X, denoted by $A \oplus X$, and then by permuting the rows and columns of the resulting matrix by σ_1 and σ_2, respectively. Next, Alice interprets A' as the incidence matrix of a bipartite graph $G(A')$. Alice runs algorithm $\mathbf{A}$ on a random ordering of the edges of $G(A')$ and sends the resulting memory state to Bob.

Next, Bob also computes the entry-wise XOR between the part of the matrix A that he knows about, $A_{S(x,y)}$, and X, followed by applying the permutations σ_1 and σ_2. In doing so, Bob knows the matrix entries of A' at positions $(\sigma_1(i), \sigma_2(j))$ for every $(i,j) \in S(x,y)$. He can therefore compute the subset E_S of the edges of $G(A')$ with

$$E_S = \{(\sigma_1(i), \sigma_2(j)) \in [n]^2 \mid (i,j) \in S(x,y) \text{ and } A'(\sigma_1(i), \sigma_2(j)) = 1\} .$$

Furthermore, let $E_{diag} \subseteq E_S$ be the set of edges $(\sigma_1(i), \sigma_2(j))$ so that (i,j) lies on the same diagonal in A as (x,y), or, in other words, there exists an integer $1 \leq q \leq k-1$ such that $(x+q, y+q) = (i,j)$. Then, let $E_{del} = E_S \setminus E_{diag}$. Bob continues the execution of algorithm **A**, as follows: for every edge $e \in E_{del}$, Bob introduces an edge deletion of e, in random order.

Let M' be the matching returned by **A**. From M' Bob computes the matching M as follows: If $|M' \leq 0.99\frac{k}{2C}|$ then Bob sets $M = \varnothing$. Otherwise, Bob sets M to be a uniform random subset of M' of size exactly $0.99\frac{k}{2C}$.

Parallel Executions. Alice and Bob execute the previous process $\ell = 100 \cdot C$ times in parallel. Let M^i, X^i, σ_1^i and σ_2^i be M, X, σ_1 and σ_2 that are used in run i, respectively. Let Q_i be the indicator random variable that is 1 iff M^i contains the edge $(\sigma_1^i(x), \sigma_2^i(y))$. We also define $p = \sum_i Q_i$ to be the total number of times the edges $(\sigma_1^i(x), \sigma_2^i(y))$ are reported. Whenever the edge $(\sigma_1^i(x), \sigma_2^i(y))$ is reported, we interpret this to be a claim that $A_{x,y} = \neg X^i_{x,y}$. So depending on the value of $X^i_{x,y}$, this acts as a claim that $A_{x,y} = 0$ or $A_{x,y} = 1$. We define $p_0 = \sum_{i:Q_i=1} X^i_{x,y}$ (which counts how often $A_{x,y} = 0$ was claimed) and let $p_1 = p - p_0$ (the number of times $A_{x,y} = 1$ was claimed). Bob outputs 1 as his estimator for $A_{x,y}$ if $p_1 \geq p_0$ and 0 otherwise.

3.2 Analysis

Let G be the bipartite graph with incidence matrix $A \oplus X$, and let

$$F = \{(i,j) \in S(x,y) \mid (A \oplus X)_{i,j} = 1 \text{ and } \nexists\ q \text{ s.t. } (i,j) = (x+q, y+q)\}\ .$$

Then the graph $G - F$ is isomorphic to the graph $G(A') - E_{del}$. In particular, $G(A') - E_{del}$ is obtained from $G - F$ by relabeling the vertex sets of the two bipartitions using the permutations σ_1 and σ_2.

We will first bound the maximum matching size in $G(A') - E_{del}$. To this end, we will bound the maximum matching size in $G - F$, which is easier to do:

▶ **Lemma 5.** *With probability* $1 - \frac{1}{k^{10}}$*, the graph* $G(A') - E_{del}$ *is such that:*

$$0.99\frac{k}{2} \leq \mu(G(A') - E_{del}) \leq 1.01\frac{k}{2} + 2(n-k)\ ,$$

where $\mu(G)$ *denotes the* matching number *of* G*, i.e., the size of a maximum matching.*

Proof. We will consider the graph $G - F$ instead, since it is isomorphic to $G(A') - E_{del}$ and has the same maximum matching size.

First, observe that G is a random bipartite graph where every edge is included with probability $\frac{1}{2}$. Let U and V denote the bipartitions in G, and consider the subsets $U' = [x, x+k)$ and $V' = [y, y+k)$. Observe that in the vertex induced subgraph $G[U' \cup V']$ all edges are deleted in F except those that connect the vertices $x+i$ and $y+i$, for every $0 \leq i \leq k-1$. By a Chernoff bound, the number of edges and thus the maximum matching size in $G[U' \cup V']$ is bounded by:

$$0.99 \cdot \frac{k}{2} \leq \mu(G[U' \cup V']) \leq 1.01 \cdot \frac{k}{2}\ ,$$

with probability $1 - \frac{1}{k^{10}}$.

Observe that, with probability $1 - \frac{1}{k^{10}}$, the neighborhood $\Gamma(U')$ is such that

$$0.99 \cdot \frac{k}{2} \leq |\Gamma(U')| \leq 1.01 \cdot \frac{k}{2} + (n-k) \ .$$

The set U' can therefore be matched to at most $1.01 \cdot \frac{k}{2} + (n-k)$ vertices in V. We thus obtain

$$\mu(G - F) \leq 1.01 \cdot \frac{k}{2} + 2(n-k) \ ,$$

since we may also be able to match all $n-k$ vertices of $U \setminus U'$. ◀

▶ **Lemma 6.** *Suppose that $M_i \neq \varnothing$. Then:*

$$\frac{0.99}{2C} - \frac{2(n-k)}{k} \quad \leq \quad \mathbb{P}\left[Q_i = 1\right] \leq \frac{0.99}{2C} \ .$$

Proof. First, by construction of our reduction, since $M_i \neq \varnothing$ we have $|M_i| = 0.99\frac{k}{2C}$. Let

$$U'_i = \sigma_1^i([x, x+k)) \text{ and } V'_i = \sigma_2^i([y, y+k)) \ .$$

Let $\tilde{M}_i$ be the set of edges of M_i connecting vertices in U'_i to V'_i. Observe that there are $2(n-k)$ vertices in the graph outside the set $U'_i \cup V'_i$. We thus have

$$|M_i| - 2(n-k) \leq |\tilde{M}_i| \leq |M_i| \ .$$

Next, since the permutations σ_1^i, σ_2^i are chosen uniformly at random, any edge of $\tilde{M}_i$ may have originated from any of the diagonal entries in $A_{S(x,y)}$. Hence, $\tilde{M}_i$ claims the bits of at least $|M_i| - 2(n-k)$ and at most $|M_i|$ uniform random positions in the diagonal of $A_{S(x,y)}$. Every entry in the diagonal of $A_{S(x,y)}$ is thus claimed with the same probability. Since the diagonal of $A_{S(x,y)}$ is of length k, this probability is at least

$$\frac{|M_i| - 2(n-k)}{k} = \frac{0.99\frac{k}{2C} - 2(n-k)}{k} = \frac{0.99}{2C} - \frac{2(n-k)}{k} \ ,$$

and at most

$$\frac{|M_i|}{k} = \frac{0.99\frac{k}{2C}}{k} = \frac{0.99}{2C} \ .$$ ◀

▶ **Theorem 7.** *Let $\mathbf{A}$ be a n^ϵ-approximation insertion-deletion streaming algorithm for Maximum Matching that errs with probability at most $1/10$ and uses space s. Then there exists a communication protocol for $\mathsf{BInd}_{0.05}^{n, n-\frac{1}{40}n^{1-\epsilon}}$ that communication $\mathrm{O}(n^\epsilon \cdot s)$ bits.*

Proof. Let $C = n^\epsilon$ and let $k = n - \frac{1}{40}n^{1-\epsilon}$. First, by Lemma 5, with probability $1 - \frac{1}{k^{10}}$, the graph $G(A') - E_{del}$ contains a matching of size at least $0.99k/2$. By a union bound, the probability that this graph is of at least this size in each of the ℓ iterations is at least $1 - \frac{\ell}{k^{10}}$. Suppose from now on that this event happens.

Let ℓ_1 be the number of times the algorithm $\mathbf{A}$ succeeds, and let ℓ_0 be the number of times $\mathbf{A}$ errs. Then, $\ell = \ell_0 + \ell_1$. Whenever $\mathbf{A}$ succeeds, since $\mathbf{A}$ is a C-approximation algorithm, the matching M'_i is of size $0.99\frac{k}{2C}$, which further implies that M_i is of size exactly $0.99\frac{k}{2C}$. Since the algorithm must return a correct matching, every time we have a claim (i.e. $Q_i = 1$), the claimed bit value must be correct. Thus, by Lemma 6, we get a correct claim on $A_{x,y}$ with probability at least

$$\frac{0.99}{2C} - \frac{2(n-k)}{k} \quad = \quad \frac{0.99}{2n^\epsilon} - \frac{2(\frac{1}{40}n^{1-\epsilon})}{n - \frac{1}{40}n^{1-\epsilon}} \geq \frac{0.99}{2n^\epsilon} - \frac{\frac{1}{40}n^{1-\epsilon}}{n} = \frac{0.99}{2n^\epsilon} - \frac{1}{40n^\epsilon} \geq \frac{2}{5n^\epsilon} \ ,$$

where we used the inequality $\frac{2x}{y-x} \geq \frac{x}{y}$, which holds for every $y > x$. We thus expect to see the correct bit claimed at least $\ell_1 \cdot \frac{2}{5n^\epsilon}$ times in total. On the other hand, incorrect claims of the bit value can only occur when the algorithm errs. In the worst case, $\mathbf{A}$ will make as many false claims as possible - so we assume the algorithm never results in $M_i = \emptyset$ when it errs. Lemma 6 also allows us to bound the probability of an incorrect claim for this bad algorithm by $\frac{0.99}{2n^\epsilon}$. We thus expect to see the wrong bit value claimed at most $\ell_0 \cdot \frac{0.99}{2C} \leq \frac{\ell_0}{2n^\epsilon}$ times.

Recall that $\ell = 100n^\epsilon$. Then, by standard concentration bounds, the probability that $\ell_0 \geq 2 \cdot \frac{\ell}{10}$ is at most $\frac{1}{100}$ (recall that the error probability of $\mathbf{A}$ is at most $\frac{1}{10}$). Suppose now that $\ell_0 \leq \frac{1}{5}\ell$ holds, which also implies that $\ell_1 \geq \frac{4}{5}\ell$. We thus expect to learn the correct bit at least

$$\frac{4}{5}100n^\epsilon \cdot \frac{2}{5n^\epsilon} = 32$$

times, and using a Chernoff bound, it can be seen that the probability that we learn the correct bit less than 21 times is at most 0.02. Similarly, we expect to learn the incorrect bit at most

$$\frac{1}{5}100n^\epsilon \cdot \frac{1}{2n^\epsilon} = 10$$

times, and by a Chernoff bound, it can be seen that the probability that we learn the incorrect bit at least 20 times is at most 0.01. Our algorithm therefore succeeds if all these events happen. Taking a union bound over all failure probabilities that occurred in this proof, we see that our algorithm succeeds with probability

$$1 - \frac{100n^\epsilon}{k^{10}} - 0.01 - 0.02 - 0.01 \geq 0.95 \ . \qquad \blacktriangleleft$$

Since by Theorem 4, $\mathsf{BInd}_{0.05}^{n,n-\frac{1}{40}n^{1-\epsilon}}$ has randomized one-way communication complexity $\Omega(n^{2-2\epsilon})$, by Theorem 7 we obtain our main result of this section:

▶ **Corollary 8.** *Every insertion-deletion n^ϵ-approximation streaming algorithm for Maximum Matching that errs with probability at most $\frac{1}{10}$ requires space $\Omega(n^{2-3\epsilon})$.*

4 Minimum Vertex Cover

Let $\mathbf{B}$ be a C-approximation insertion-deletion streaming algorithm for Minimum Vertex Cover that succeeds with probability $1 - 1/400$. Similar to the previous section, we will now show how $\mathbf{B}$ can be used to solve $\mathsf{BInd}_\delta^{n,k}$.

4.1 Reduction

Let $A \in \{0,1\}^{n\times n}, x \in [n-k]$ and $y \in [n-k]$ be an instance of $\mathsf{BInd}_\delta^{n,k}$. The reduction for Minimum Vertex Cover is very similar to the reduction for Maximum Matching presented in the previous section. Alice's behaviour is in fact identical:

First, Alice and Bob sample a uniform random binary matrix $X \in \{0,1\}^{n\times n}$ and random permutations $\sigma_1, \sigma_2 : [n] \to [n]$ from public randomness. Alice then computes matrix A' which is obtained by first computing $A \oplus X$ and then permuting the rows and then the columns of the resulting matrix by σ_1 and σ_2, respectively. Alice interprets A' as the incidence matrix of a bipartite graph $G(A')$. Alice then runs algorithm $\mathbf{B}$ on a random ordering of the edges of $G(A')$ and sends the resulting memory state to Bob.

Next, Bob also computes the entry-wise XOR between the part of the matrix A that he knows about and X, followed by applying the permutations σ_1 and σ_2. In doing so, Bob knows the matrix entries of A' at positions $(\sigma_1(i), \sigma_2(j))$ for every $(i,j) \in S(x,y)$. He can therefore compute the subset E_S of the edges of $G(A')$ with

$$E_S = \{(\sigma_1(i), \sigma_2(j)) \in [n]^2 \mid (i,j) \in S(x,y) \text{ and } A'(\sigma_1(i), \sigma_2(j)) = 1\} .$$

Next, Bob continues the execution of $\mathbf{B}$ and introduces deletions *for all edges in* E_S in random order. Observe that this step is different to the reduction for Maximum Matching. Let I be the vertex cover produced by $\mathbf{B}$.

Parallel Executions. Alice and Bob run the procedure above 40 times in parallel. Denote by I^i, X^i, E_S^i, A'^i, σ_1^i, and σ_2^i the variables I, X, E_S, A', σ_1 and σ_2 used in iteration i. Furthermore, let Q_i be the indicator variable that is 1 iff $\{\sigma_1^i(x), \sigma_2^i(y)\} \cap I_i \neq \varnothing$, i.e., the potential edge $(\sigma_1^i(x), \sigma_2^i(y))$ is covered by the vertex cover.

If there exists a run j with $Q_j = 0$, then Bob predicts $A_{x,y} = X_{x,y}$ (if there are multiple such runs then Bob breaks ties arbitrarily). Otherwise, Bob returns `fail` and the algorithm errs.

4.2 Analysis

The first lemma applies to every parallel run j. For simplicity of notation, we will omit the superscripts that indicate the parallel run in our random variables.

We first show an upper bound on the size of a minimum vertex cover in $G(A') - E_S$.

▶ **Lemma 9.** *The size of a minimum vertex cover in $G(A') - E_S$ is at most $2(n-k)+1$.*

Proof. Let U, V be the bipartitions of the graph $G(A') - E_S$, let $U' = \{\sigma_1(a) \ : \ a \in [x, x+k)\}$ and let $V' = \{\sigma_2(b) \ : \ b \in [y, y+k)\}$. Observe that $(G(A') - E_S)[U' \cup V']$ contains at most one edge: The potential edge between $\sigma_1(x)$ and $\sigma_2(y)$. A valid vertex cover of $G(A') - E_S$ is therefore $(U \setminus U') \cup (V \setminus V') + \sigma_1(x)$, which is of size $2(n-k)+1$. ◀

Next, we prove the key property of our reduction: We show that if $A'_{\sigma_1(x),\sigma_2(y)} = 0$ (or equivalently, $A_{x,y} \oplus X_{x,y} = 0$) then neither $\sigma_1(x)$ nor $\sigma_2(y)$ is in the output vertex cover with large probability.

▶ **Lemma 10.** *Assume that algorithm $\mathbf{B}$ does not err in run j. Suppose that $A'^j_{\sigma_1^j(x),\sigma_2^j(y)} = 0$. Then the probability that $Q_j = 1$ is at most*

$$\frac{3C \cdot (2(n-k)+1)}{k} .$$

Proof. Consider the set $D = \{(\sigma_1^j(x+i), \sigma_2^j(y+i)) \mid 0 \leq i \leq k-1\}$, i.e., the positions of the diagonal of $S(x,y) \cup \{x,y\}$ permuted by σ_1^j and σ_2^j. Then, since A'^j is a uniform random matrix, with probability at least $1 - \frac{1}{k^{10}}$, the "permuted diagonal" A'^j_D contains at least $0.99k/2$ entries with value 0, or, in other words, graph $G(A'^j) - E_S^j$ contains at least $0.99k/2$ non-edges in the positions of the permuted diagonal D. By Lemma 9, the size of a minimum vertex cover in $G(A'^j) - E_S^j$ is at most $2(n-k)+1$, and since $\mathbf{B}$ has an approximation factor of C, the vertex cover I_j is of size at most $C \cdot (2(n-k)+1)$. Hence, at most $C \cdot (2(n-k)+1)$ non-edges in D can be covered in I_j. However, since the permutations are random, the probability that the non-edge $(\sigma_1^j(x), \sigma_2^j(y))$ is covered, which is identical to the event $Q_j = 1$, is therefore at most

$$\frac{C \cdot (2(n-k)+1)}{0.99k/2} \leq \frac{3C \cdot (2(n-k)+1)}{k} .$$ ◀

▶ **Theorem 11.** *Let* $\mathbf{B}$ *be a* n^ϵ*-approximation insertion-deletion streaming algorithm for Minimum Vertex Cover that uses space* s *and errs with probability at most* $1/400$*. Then, there exists a communication protocol for* $\mathsf{BInd}_{\frac{1}{3}}^{n,n-\frac{1}{20}n^{1-\epsilon}}$ *that communicates* $\mathrm{O}(s)$ *bits.*

Proof. Let $k = n - \frac{1}{40}n^{1-\epsilon}$ and let $C = n^\epsilon$. Consider the reduction given in the previous subsection. First, observe that since $\mathbf{B}$ errs with probability at most $1/400$, by the union bound the probability that $\mathbf{B}$ errs at least once in the 40 parallel executions of our reduction is at most $\frac{1}{10}$. We assume from now on that the algorithm never errs.

Observe that the matrices A'^j are random matrices. Hence, the probability that there exists at least one run i with $A'^i_{\sigma_1^i(x),\sigma_2^i(y)} = 0$ is at least $1 - (\frac{1}{2})^{40}$. Suppose that this event happens. Let run i be so that $A'^i_{\sigma_1^i(x),\sigma_2^i(y)} = 0$. Then, by Lemma 10, the probability that the non-edge $(\sigma_1^i(x), \sigma_2^i(y))$ is covered by I_i, or in other words, the probability that $Q_i = 1$, is at most

$$\frac{3C \cdot (2(n-k)+1)}{k} = \frac{3n^\epsilon \cdot (\frac{1}{20}n^{1-\epsilon}+1)}{n - \frac{1}{40}n^{1-\epsilon}} = \frac{\frac{3}{20}n + 3n^\epsilon}{n - \frac{1}{40}n^{1-\epsilon}} = \frac{3}{20} + o(1) \ .$$

Observe that whenever $Q_i = 0$, the algorithm outputs $X^i_{x,y}$ as a predictor for $A_{x,y}$. Since the algorithm $\mathbf{B}$ does not err, we have $A_{x,y} \oplus X^i_{x,y} = 0$. This implies that $A_{x,y} = X^i_{x,y}$, which establishes correctness.

Last, we need to bound the error probability of our algorithm. First, the probability that at least one of the 40 runs fails is at most $\frac{1}{10}$. Next, the probability that none of the runs are such that $A'^j_{\sigma_1^j(x),\sigma_2^j(y)} = 0$ is at most $(\frac{1}{2})^{40}$. Furthermore, the probability that $Q_i = 1$ when $A'^i_{\sigma_1^i(x),\sigma_2^i(y)} = 0$ is at most $\frac{3}{20} + o(1)$. Applying the union bound, we see that the overall error probability of our algorithm is at most

$$\frac{1}{10} + (\frac{1}{2})^{40} + \frac{3}{20} + o(1) \leq \frac{1}{3} \ ,$$

for large enough n. ◀

Since by Theorem 4, $\mathsf{BInd}_{\frac{1}{3}}^{n,n-\frac{1}{40}n^{1-\epsilon}}$ has a communication complexity of $\Omega(n^{2-2\epsilon})$, we obtain the following result:

▶ **Corollary 12.** *Every insertion-deletion* n^ϵ*-approximation streaming algorithm for Minimum Vertex Cover with error probability at most* $\frac{1}{400}$ *requires space* $\Omega(n^{2-2\epsilon})$*.*

4.3 Insertion-deletion Streaming Algorithm for Minimum Vertex Cover

We now sketch a simple deterministic n^ϵ-approximation insertion-deletion streaming algorithm for Minimum Vertex Cover on general graphs that uses space $\mathrm{O}(n^{2-2\epsilon}\log n)$. Let $G = (V, E)$ be the graph described by the input stream. The algorithm proceeds as follows:

Algorithm 1 A simple deterministic n^ϵ-approximation insertion-deletion streaming algorithm for Minimum Vertex Cover.

1. Arbitrarily partition V into subsets $V_1, V_2, \ldots, V_{n^{1-\epsilon}}$, each of size n^ϵ.
2. Consider the multi-graph G' obtained from G by contracting the sets V_i into vertices.
3. **While processing the stream:** For each pair of vertices V_i, V_j in G' deterministically maintain the number of edges connecting V_i to V_j.
4. **Post-processing:** Compute a minimum vertex cover I' in the multi-graph G'.
5. **Return** $I = \cup_{V_j \in I'} V_j$ as the vertex cover in G.

Analysis: Regarding space, the dominating space requirement is the maintenance of the number of edges between every pair V_i, V_j. Since there are $n^{2-2\epsilon}$ such pairs, this requires space $\mathrm{O}(n^{2-2\epsilon} \cdot \log n)$.

Concerning the approximation factor, let I^* be a minimum vertex cover in G. Recall that I' is an optimal cover in G' and hence $|I'| \leq |I^*|$ (edge contractions cannot increase the size of a minimum vertex cover). Since every set V_j is of size n^ϵ, the computed vertex cover I is of size at most $|I'| \cdot n^\epsilon \leq |I^*| n^\epsilon$, which proves the approximation factor. By construction of the algorithm, every edge is covered.

▶ **Theorem 13.** *There is a deterministic n^ϵ-approximation insertion-deletion streaming algorithm for Minimum Vertex Cover that uses space $\mathrm{O}(n^{2-2\epsilon} \log n)$.*

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, page 459–467, USA, 2012. Society for Industrial and Applied Mathematics.

2 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 20:1–20:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.20.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786. SIAM, 2019. doi:10.1137/1.9781611975482.48.

4 Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '17, page 3–12, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3087556.3087581.

5 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364. SIAM, 2016. doi:10.1137/1.9781611974331.ch93.

6 Ziv Bar-Yossef, Thathachar S Jayram, Robert Krauthgamer, and Ravi Kumar. The sketching complexity of pattern matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 261–272. Springer, 2004.

7 Suman K. Bera and Amit Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.11.

8 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344. SIAM, 2016. doi:10.1137/1.9781611974331.ch92.

9 Graham Cormode, Jacques Dark, and Christian Konrad. Approximating the caro-wei bound for independent sets in graph streams. In Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub, editors, *Combinatorial Optimization - 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers*, volume 10856 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2018. doi:10.1007/978-3-319-96151-4_9.

10 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.45.

11 Graham Cormode and Hossein Jowhari. A second look at counting triangles in graph streams (corrected). *Theor. Comput. Sci.*, 683:22–30, 2017. doi:10.1016/j.tcs.2016.06.020.

12 Alireza Farhadi, MohammadTaghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 1773–1785, USA, 2020. Society for Industrial and Applied Mathematics.

13 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

14 Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. Streaming algorithms for independent sets in sparse hypergraphs. *Algorithmica*, 76(2):490–501, 2016. doi:10.1007/s00453-015-0051-5.

15 Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. Streaming and communication complexity of clique approximation. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2012. doi:10.1007/978-3-642-31594-7_38.

16 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In James M. Abello and Jeffrey Scott Vitter, editors, *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, volume 50 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 107–118. DIMACS/AMS, 1998. doi:10.1090/dimacs/050/05.

17 Kaave Hosseini, Shachar Lovett, and Grigory Yaroslavtsev. Optimality of Linear Sketching Under Modular Updates. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC 2019)*, volume 137 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2019.13.

18 Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.15.

19 John Kallaugher and Eric Price. Separations and equivalences between turnstile streaming and linear sketching. In *Symposium on Theory of Computing, STOC 2020*, 2020. to appear.

20 Daniel M. Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 598–609. Springer, 2012. doi:10.1007/978-3-642-31585-5_53.

21 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751. SIAM, 2014. doi:10.1137/1.9781611973402.55.

22 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1814–1833. SIAM, 2020. doi:10.1137/1.9781611975994.111.

23 Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 840–852. Springer, 2015. doi:10.1007/978-3-662-48350-3_70.

24 Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 74:1–74:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.74.

25 Christian Konrad. Streaming frequent items with timestamps and detecting large neighborhoods in graph streams. *CoRR*, abs/1911.08832, 2019. arXiv:1911.08832.

26 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. doi:10.1007/978-3-642-32512-0_20.

27 Christian Konrad and Adi Rosén. Approximating semi-matchings in streaming and in two-party communication. *ACM Trans. Algorithms*, 12(3):32:1–32:21, 2016. doi:10.1145/2898960.

28 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 174–183. ACM, 2014. doi:10.1145/2591796.2591812.

29 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

30 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.

31 Ami Paz and Gregory Schwartzman. A (2+ϵ)-approximation for maximum weight matching in the semi-streaming model. *ACM Trans. Algorithms*, 15(2):18:1–18:15, 2019. doi:10.1145/3274668.

Connecting Perebor Conjectures: Towards a Search to Decision Reduction for Minimizing Formulas

Rahul Ilango
Massachusetts Institute of Technology, Cambridge, MA, USA
rilango@mit.edu

Abstract

A longstanding open question is whether there is an equivalence between the computational task of determining the minimum size of any circuit computing a given function and the task of producing a minimum-sized circuit for a given function. While it is widely conjectured that both tasks require "perebor," or brute-force search, researchers have not yet ruled out the possibility that the search problem requires exponential time but the decision problem has a linear time algorithm.

In this paper, we make progress in connecting the search and decision complexity of minimizing *formulas*. Let MFSP denote the problem that takes as input the truth table of a Boolean function f and an integer size parameter s and decides whether there is a formula for f of size at most s. Let Search-MFSP denote the corresponding search problem where one has to output some optimal formula for computing f.

Our main result is that given an oracle to MFSP, one can solve Search-MFSP in time polynomial in the length N of the truth table of f and the number t of "near-optimal" formulas for f, in particular $O(N^6t^2)$-time. While the quantity t is not well understood, we use this result (and some extensions) to prove that given an oracle to MFSP:

- there is a deterministic $2^{O(\frac{N}{\log\log N})}$-time oracle algorithm for solving Search-MFSP on all but a $o(1)$-fraction of instances, and
- there is a randomized $O(2^{.67N})$-time oracle algorithm for solving Search-MFSP on *all* instances.

Intriguingly, the main idea behind our algorithms is in some sense a "reverse application" of the gate elimination technique.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases minimum circuit size problem, minimum formula size problem, gate elimination, search to decision reduction, self-reducibility

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.31

Funding During this work the author was supported by an Akamai Presdential fellowship.

Acknowledgements I want to thank Eric Allender, Mathew Katzman, Aditya Potukuchi, Michael Saks, Rahul Santhanam, and Ryan Williams for many helpful discussions regarding this work.

1 Introduction

In his fascinating historical account, Trakhtenbrot [20] describes the early developments of the Russian cybernetics program. Beginning in the 1950s, this program was largely driven by a desire to understand the necessity of "perebor," or brute-force, in solving various problems related to complexity minimization. What Trakhtenbrot calls "Task 1" in [20] is an analogue[1] of what is now commonly referred to as the Minimum Circuit Size Problem, MCSP. In his article, Trakhtenbrot delineates two versions of "Task 1": an "existential version," where

[1] We say analogue since Task 1 was defined in the slightly different model of switching circuits.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 31; pp. 31:1–31:35

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


given a Boolean function f one must compute the minimum number of gates needed in a circuit computing f, corresponding to MCSP and a "constructive version," where one must produce such an optimal circuit for f, corresponding to Search-MCSP.

Both versions were conjectured to require "perebor," or brute-force to solve. However, while it is clear that if perebor is required for MCSP then perebor must also be required for Search-MCSP, it is a longstanding open question (since at least 1999 [10]) to prove a reverse implication: that is, to show that if Search-MCSP requires brute-force to solve, then MCSP requires brute-force.

Indeed, this question is closely related to another major open question surrounding MCSP: is MCSP NP-complete? Despite being an open problem since the discovery of NP-completeness[2] in the 1970s and numerous fascinating papers studying MCSP, we still know little about the computational complexity of MCSP. The problem is known to lie in NP, but even formal evidence supporting or opposing the NP-completeness of MCSP is lacking. This is in contrast to other prominent problems that are believed to be intractable yet are not known to be NP-complete (such as integer factorization or the discrete logarithm[3]).

However, a remarkable line of research demonstrates that a proof that MCSP is NP-complete would have significant ramifications. For example, Murray and Williams [14] show that it would imply the breakthrough complexity separation EXP $\neq$ ZPP, and Hirahara [8] shows that it implies a worst-case to average case reduction for NP (if the hardness holds for an approximate version of MCSP).

An NP-completeness proof for MCSP would also resolve the "search versus decision" question mentioned at the beginning of this paper. In particular, since SAT is known to have a polynomial-time search to decision reduction, MCSP being NP-complete would imply that MCSP would also have a polynomial-time search to decision reduction. Hence, the time complexity of computing MCSP and Search-MCSP would be equivalent up to a polynomial.

Because of this, Kabanets and Cai observed that finding a search to decision reduction for MCSP is, in fact, a *necessary* step to showing that MCSP is NP-complete, and left finding such a reduction as an open question. Indeed, it is a bit unnerving (at least to the author) that researchers have not yet ruled out the possibility that MCSP has a linear-time algorithm but solving Search-MCSP requires exponential-time! The present work was born out of a motivation to (at least partially) mediate this large gap.

Alas, while we fail to improve the status of this question for MCSP, we make considerable progress in connecting the search and decision complexity of the analogous *Formula* Minimization Problem, MFSP.

1.1 Prior Work

In light of the numerous research papers studying MCSP and its variants, we do not attempt to survey the full body of literature but rather concentrate on those works related to search to decision reductions and MFSP. We point a reader interested in a more detailed overview to Allender's excellent new survey [1] and the references therein.

Search to decision reductions for MCSP. There are two main prior works for search to decision reductions for MCSP-like problems. Both provide algorithms that find approximately optimal circuits that are efficient as long as MCSP has efficient algorithms. Interestingly,

[2] [4] cites a personal communication from Levin that he delayed publishing his initial NP-completeness results in hopes of showing MCSP is NP-complete.

[3] Intriguingly, it is known [18, 2] that both of these problems reduce to MCSP under randomized reductions!

both algorithms require that MCSP actually has efficient algorithms and seemingly fail if they are "only" provided oracle access to MCSP (the reason is that the approximately optimal circuit that these algorithms output actually include a small MCSP circuit within them).

The first prior work is a celebrated paper by Carmosino, Impagliazzo, Kabanets, and Kolokolova [6] that establishes connections between algorithms for MCSP-like problems and PAC-learning of circuits. In their paper, they show the following theorem.

▶ **Theorem 1** (Carmosino, Impagliazzo, Kabanets, and Kolokolova [6]). *Suppose* MCSP $\in$ BPP. *Then there is a randomized polynomial-time algorithm that, given the truth table of a function f with n-bit inputs, outputs a circuit C of size at most* poly(s) *such that $C(x) = f(x)$ for all but a $\frac{1}{\text{poly}(n)}$ fraction of inputs x, where s is the minimum size of any circuit computing f.*

Building on [6], Hirahara [8] proved a breakthrough worst-case to average-case reduction for an approximation version of MCSP. In said paper, Hirahara shows the following theorem.

▶ **Theorem 2** (Hirahara [8]). *Suppose for some $\epsilon > 0$ that one can approximate* MCSP *to within a factor of $N^{1-\epsilon}$ in randomized polynomial-time (where N is the length of the truth table). Then there is some $\epsilon' > 0$ such that, given a length-N truth table for computing f, one can, in randomized polynomial-time, output a circuit for computing f (exactly) whose size is within a $N^{1-\epsilon'}$ factor of being optimal.*

Using similar ideas, Santhanam [19] independently obtained a comparable search-to-decision reduction (with somewhat better parameters than Theorem 2) for AveMCSP, a natural variant of MCSP where one asks for the smallest circuit computing a function on a 0.9-fraction of the inputs.

We find it interesting that "approximate" search to decision reductions for MCSP have been a building block in these celebrated results. It seems to suggest that further exploring the interplay between the search and decision versions of MCSP could be a fruitful direction.

Hardness of MFSP. As with MCSP, we have good reason to believe that MFSP is intractable, since it is in some sense "hard" for cryptography computable in NC^1.

▶ **Theorem 3** (Razborov and Rudich [17], Kabanets and Cai [10]). *If* MFSP $\in$ P*, then there are no pseudorandom function generators computable in* NC^1.

Allender, Koucký, Ronneburger, and Roy [4] build on this connection to show that MFSP is hard to approximate if factoring Blum integers is intractable.

Despite the strength of this cryptographic hardness connection, we know very little about the complexity of MFSP unconditionally. Indeed, part of the difficulty is that it seems difficult to design reductions that make use of an MFSP (or MCSP) oracle, since we do not understand the model of formulas (or circuits) very well. Until very recently [7], it was even open whether MFSP was in $\mathsf{AC}^0[2]$!

One reason for focusing on MFSP is that one might expect it to be an easier problem to analyze than MCSP since formulas are somewhat better understood than circuits. In support of this intuition, we know that the formula minimization problem for DNFs and DNF $\circ$ XOR formulas are NP-complete [13, 9] and that the natural Σ_2 variant of MFSP is complete for Σ_2 [5].

However, counter to this intuition, there are some cases in which it has been more difficult to prove hardness for MFSP than for MCSP. While it is known that MCSP is hard for SZK under randomized reductions [3], it remains open to prove such a result for MFSP. We take this as further evidence of the subtleties involved in designing reductions for MFSP.

1.2 Our Results

In contrast to prior results, we examine the case of having to *exactly* solve Search-MFSP, that is, producing an exactly optimal (instead of approximately optimal) formula.

We define MFSP over the model of DeMorgan formulas (formulas with AND and OR gates) where the size of a formula is the number of leaf nodes in its binary tree. Our main results are robust to changes in the model however. In particular, unless otherwise stated, all our results also extend to the case when gates are from the full binary basis $\mathbb{B}_2$ and to the case when the notion of size is the number of wires or the number of gates.

Our main result is to show that one can efficiently find an optimal formula for a given function f using an oracle to MFSP when f has a small number of "near-optimal formulas" (we say what this means after our theorem statement).

▶ **Theorem 4** (also Theorem 33). *There is an algorithm solving* Search-MFSP *using an oracle to* MFSP *that given a length-N truth table of a function f runs in time $O(N^6t^2)$ where t is the number of "near-optimal" formulas computing f.*

Defining "near-optimal" formulas. We now define what we mean by "near-optimal" formulas. Let $\mathsf{L}(f)$ denote the minimum size of any formula computing f. We say a formula φ is a *near-optimal formula* for $f : \{0,1\}^n \to \{0,1\}$ if φ has size at most $\mathsf{L}(f) + n + 1$.

Furthermore, in counting the number of near-optimal formulas, we consider formulas that are isomorphic as labelled binary trees to be the same formula. This avoids counting many trivially equivalent formulas as distinct near-optimal formulas. See Section 2.2 for a precise definition.

Bounding the number of near-optimal formulas. Unfortunately, we do not understand the quantity t in Theorem 4 very well. However, using the nearly tight upper bounds by Lozhkin [11] on the maximum formula size required to compute an n-input function, we get that with high probability a random function on n-inputs has at most

$$2^{O(\frac{N}{\log\log N})}$$

many near-optimal formulas where $N = 2^n$.

Thus, we have the following corollary.

▶ **Corollary 5** (also Corollary 34). *There is an algorithm A for solving* Search-MFSP *on all but a $o(1)$ fraction of instances that runs in time $2^{O(\frac{N}{\log\log N})}$ using an oracle to* MFSP.

Corollary 5 has a nice interpretation with respect to the perebor conjecture. The queries algorithm A (run on a truth table input of length N) makes to its MFSP-oracle can be answered using a deterministic brute-force algorithm in time $2^{(1+o(1))N}$. In particular, the queries A makes are of length at most $2N$ and have complexity at most $(1 + o(1))\frac{N}{\log\log N}$. On the other hand, the naive brute-force algorithm for Search-MFSP on an input of length N runs in time $2^{(1+o(1))N}$. Thus, we have the following further corollary.

▶ **Corollary 6** (Informal). *If the brute-force algorithm for* Search-MFSP *is essentially optimal on average, then the brute-force algorithm for* MFSP *is essentially optimal in the worst-case on a large subset of instances (in particular queries of length $2N$ with complexity at most $(1 + o(1))\frac{N}{\log\log N}$).*

It would be nice to improve the running-time of the algorithm in Corollary 5. The bound that $t \leq 2^{O(\frac{N}{\log\log N})}$ for a random function hardly seems tight. In fact, in the setting of Kolmogorov complexity, one can prove that a random string of length N has only $\mathsf{poly}(N)$

many near-optimal descriptions with high probability (this is because the worst-case upper bound for Kolmogorov complexity is much tighter than the one for formulas). If we could prove an analogous result for formulas, then Corollary 5 would give a polynomial-time search to decision reduction for a random function!

Solving Search-MFSP in the worst-case. We also give a reduction that shows that even in the worst-case, one can get exponential savings over the brute-force algorithm for Search-MFSP by using a MFSP-oracle. In light of Theorem 4, a natural approach is split into two cases:

- If there are a lot of near-optimal formulas for f, then just guess random formulas and see if they compute f.
- If there are not a lot of near-optimal formulas for f, then run the algorithm in Theorem 4.

However, this approach will only be able to output a near-optimal formula for computing f, and we desire to solve Search-MFSP exactly.

We manage to overcome this issue and prove the following theorem.

▶ **Theorem 7** (also Theorem 41). *There is a randomized algorithm for solving* Search-MFSP *using an oracle to* MFSP *that runs in $O(2^{.67N})$ time on instances of length N.*

By examining the queries that this algorithm makes to MFSP, we get the following consequence regarding the perebor conjecture.

▶ **Corollary 8** (Informal). *If brute-force is essentially optimal for solving* Search-MFSP*, then any algorithm solving* MFSP *can give at most an ϵ power speed up over the brute-force algorithm where $\epsilon = \frac{1}{7}$.*

A bottom-up approach for DeMorgan formulas. All of the results mentioned so far are proved by building an optimal formula for a function in a "top-down" way (i.e. starting from the output gate and working its way down to the tree leafs). It is natural to wonder if a "bottom-up" approach could also work.[4]

Indeed, we give such a bottom-up reduction for solving Search-MFSP using an oracle to MFSP that is efficient on average. Unfortunately, the guarantees we prove on the running time on this bottom-up algorithm are weaker than the guarantees provided in Theorem 4. Moreover, the proof of correctness for the algorithm requires our formulas to be DeMorgan formulas and not, say, $\mathbb{B}_2$ formulas. Still, we include this result because we think the algorithm is interesting and because it makes use of the following lemma (which is the part where DeMorgan formulas are crucial) that may be of independent interest. Roughly speaking, the lemma shows that optimal DeMorgan formulas must not have too large depth.

▶ **Lemma 9** (also Lemma 53). *Suppose φ is an optimal DeMorgan formula for a function on n-inputs. Then the depth of φ is at most $O(\frac{2^n}{n \log n})$.*

1.3 Techniques and Proof Overviews

The top-down approach. As mentioned earlier, our reduction works in a top-down manner. We formalize this as follows. For any Boolean function f on n-inputs, we define the set OptSubcomps(f) to consist of elements of the form $\{g, \nabla, h\}$ – where $g, h : \{0,1\}^n \to \{0,1\}$ and $\nabla \in \{\wedge, \vee\}$ – satisfying the property that there exists an optimal formula φ for computing f such that $\varphi = \varphi_g \nabla \varphi_h$ where φ_g and φ_h are subformulas computing g and h respectively.

[4] The idea that a bottom-up approach could also be an efficient way to solve Search-MFSP was given to me by Ryan Williams.

We can naturally define the Decomposition Problem, denoted DecompProblem as follows:

- **Given:** a non-trivial[5] function f,
- **Output:** some element of $\mathsf{OptSubcomps}(f)$.

Our two main reductions work by solving the DecompProblem. It is easy to show that one can solve Search-MFSP efficiently by recursively calling an DecompProblem oracle to build an optimal formula gate-by-gate from top to bottom. (See Theorem 20 for details.)

Thus, we now focus on trying to solve DecompProblem.

A high level approach to solving DecompProblem. Our two top-down reductions will use a similar approach to solving DecompProblem. (Actually, our worst-case reduction will use three different approaches, but this will be one of them.)

1. Find an efficient "test" that functions in[6] an optimal subcomputation of f pass, but not too many other functions pass.
2. Efficiently build the (not too long) list *Candidates* of functions that pass the "test."
3. Iterate through all pairs of functions in *Candidates* and each possible gate, and check if this constitutes an element of $\mathsf{OptSubcomps}(f)$.

We first describe how we do Item 3 since it is simpler and then describe our "test" for Item 1. Our method for Item 2 will be different in both reductions.

Item 3: checking membership in $\mathsf{OptSubcomps}(f)$. Given access to a MFSP oracle it is actually very easy to check whether some $\{g, \nabla, h\}$ is an element of $\mathsf{OptSubcomps}(f)$ or not. In Lemma 21 we observe that $\{g, \nabla, h\} \in \mathsf{OptSubcomps}(f)$ if and only if $f(x) = g(x) \nabla h(x)$ for all x and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$.

Item 1: the $\mathsf{Select}[f, g]$ test. The idea for our "test" is based on the gate elimination technique and the implications gate elimination has on the $\mathsf{Select}[\cdot, \cdot]$ function defined as follows. Given functions $f, g : \{0,1\}^n \to \{0,1\}$, we define $\mathsf{Select}[f, g] : \{0,1\}^n \times \{0,1\} \to \{0,1\}$ by

$$\mathsf{Select}[f, g](x, z) = \begin{cases} f(x) & , \text{ if } z = 0 \\ g(x) & , \text{ if } z = 1. \end{cases}$$

Our test for whether g might be part of an optimal subcomputation for f will be whether the quantity

$$\mathsf{L}(\mathsf{Select}[f, g]) - \mathsf{L}(f)$$

is small – in particular, no more than a parameter C. The exact value of C will depend on the reduction (we use this test in all three of our reductions with a different value for C), but to give a reader some idea, C will be an element of $\{1, n + 2, 10 \cdot \frac{2^n}{n}\}$ where n is the number of input bits f takes.

---

[5] here by non-trivial we mean a function that cannot be computed by a formula of size one

[6] In case it is not clear, we say a function g is *in an optimal subcomputation* for f if there exists a gate ∇ and function h such that $\{g, \nabla, h\}$ is an element of $\mathsf{OptSubcomps}(f)$.

Now, we needed our test to have two properties:

- Property 1: any function that is in an optimal subcomputation for f must pass this test, and
- Property 2: this test does not accept too many other functions.

With regards to Property 1, we show in Lemma 23 that if $\{g, \nabla, h\} \in \mathsf{OptSubcomps}(f)$, then $\mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + 1$ and $\mathsf{L}(\mathsf{Select}[f, h]) \leq \mathsf{L}(f) + 1$.

We can give the relatively straightforward proof that $\mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + 1$ here. Suppose that $\{g, \nabla, h\} \in \mathsf{OptSubcomps}(f)$. To avoid some case analysis, assume that $\nabla = \wedge$. Then there exists an optimal formula $\varphi = \varphi_g \wedge \varphi_h$ such that φ_g computes g and φ_h computes h. Then the formula $\varphi_g(x) \wedge (\varphi_h(x) \vee z)$ computes $\mathsf{Select}[f, g](x, z)$ and has size $\mathsf{L}(f) + 1$.

For Property 2, our test must be such that the set of all functions q satisfying

$$\mathsf{L}(\mathsf{Select}[f, q]) - \mathsf{L}(f) \leq C$$

is not too large. In Lemma 24, we show that the number of such q is bounded by

$$O(t \cdot 2^{C-1} N \log N)$$

where N is the length the truth table of f and t is the number of distinct formulas (modulo an isomorphism between formulas defined in Section 2.2) computing f of size $\mathsf{L}(f) + C - 1$. (In the case that $C = n + 2$, t is the number of "near-optimal" formulas discussed earlier in Section 1.2.)

The intuition behind this proof is to use gate elimination. In more detail, if φ is a formula of size $\mathsf{L}(f) + C$ computing $\mathsf{Select}[f, g]$, then we can set $z = 0$ in φ and eliminate between one and C gates from φ to obtain a new formula φ' of size at most $\mathsf{L}(f) + C - 1$ computing f. Hence, we can describe φ (and hence g) by first describing φ' (a small-ish formula for f) and the gates that need to be added back to φ' in order to obtain φ.

While this intuition is relatively straightforward, the proof itself is surprisingly tedious. In particular, the intuition, as stated, only gives a bound with a N^C factor dependence on C. To achieve the stated bound with a 2^C factor dependence on C requires some details. Moreover, this dependence on C is important since a N^C dependence would make Theorem 4 have a quasipolynomial dependence on t instead of a polynomial dependence.

Our top-down deterministic reduction. We now outline how the deterministic algorithm in Theorem 4 works to solve $\mathsf{DecompProblem}$ on an input f.

We have already introduced the some of the ideas for the algorithm in Theorem 4. In detail, let $BestFunctions$ be the set of functions that are in an optimal subcomputation of f. Let $GoodFunctions$ denote the set of functions g that pass the test $\mathsf{L}(\mathsf{Select}[f, g]) - \mathsf{L}(f) \leq n + 2$ (for this algorithm we set $C = n + 2$). From our previous discussions, we know that the size of $GoodFunctions$ can be bounded by a quantity related to the number of near-optimal formulas for f, and we know that $GoodFunctions$ contains all the functions in $BestFunctions$.

Later we explain how to construct the list $GoodFunctions$. Note though that once the list $GoodFunctions$ is constructed, we can then iterate through all pairs of functions in $GoodFunctions$ and efficiently check if they yield an optimal subcomputation, as we discussed previously.

Hence, the missing piece is to efficiently enumerate the elements of $GoodFunctions$. In fact, we do not quite need to enumerate *all* the elements of $GoodFunctions$. It suffices to enumerate a subset, that we call $Candidates$, of $GoodFunctions$ that contains all the elements of $BestFunctions$. Informally, one can think of the $Candidates$ subset as a set of "good enough functions."

The key observation is as follows. If q is a function on n-inputs and one defines the truth table $T_{q,i}$ of length 2^n that is equal to q on its first i bits and equals one on the remaining bits, then

$$\mathsf{L}(T_{q,i}) \leq \mathsf{L}(q) + n + 1$$

since one can compute $T_{q,i}$ by computing q, computing whether the input is greater than i, and ORing these two values. The $\mathsf{Select}[\cdot,\cdot]$ function actually respects this observation in a nice way. In particular, since functions g in $BestFunctions$ satisfy the stronger property that $\mathsf{L}(\mathsf{Select}[f,g]) - \mathsf{L}(f) \leq 1$, one can show that if $g \in BestFunctions$, then

$$\mathsf{L}(\mathsf{Select}[f,T_{g,i}]) \leq \mathsf{L}(f) + n + 2$$

for all i. In other words, if $g \in BestFunctions$, then $T_{g,i}$ is in $GoodFunctions$ for all i.

Using this fact, we can construct a subset $Candidates$ of $GoodFunctions$ that contains all the elements of $BestFunctions$ by bit-by-bit extending a set of prefixes $PartialCandidates$ that pass our test (and prefixes of functions in $BestFunctions$ do pass our test) until these prefixes become full functions.

In more detail, we start with a set $PartialCandidates$ that initially only contains the empty prefix. While $PartialCandidates$ is non-empty, we remove a prefix γ from it and try to extend it by one bit. That is, for each bit $b \in \{0,1\}$, we consider γ_b obtained by appending b to γ. We then see if the prefix γ_b "passes our test" by seeing if the truth table T_{γ_b}, obtained by padding γ_b with ones until it has length 2^n, has the property

$$\mathsf{L}(\mathsf{Select}[f,T_{\gamma_b}]) \leq \mathsf{L}(f) + n + 2.$$

If so, we either add γ_b to $Candidates$ or back to $PartialCandidates$ depending on whether the string γ_b is of length 2^n or not. We continue until $PartialCandidates$ is empty. The full details can be found in Algorithm 2.

Our top-down randomized worst-case reduction. The algorithm in Theorem 7 uses three different strategies for finding an optimal subcomputation in the worst-case using an oracle to MFSP. We give a a rough overview of each of these three parts.

Suppose the input to the algorithm is a function f on n-inputs. First, the algorithm picks $2^{2N/3}$ random formulas of size $\mathsf{L}(f)$ and checks if any of these formulas compute f. If so, we are done. Otherwise, we know that the number of optimal formulas for f cannot be too large (in particular, is upper bounded by roughly $2^{N/3}$ with high probability).

In the second part, we construct a set of candidate functions that pass a test. The guarantee on the number of optimal formulas from the previous step ensures that the size of the set

$$\{g : \mathsf{L}(\mathsf{Select}[f,g]) \leq \mathsf{L}(f) + 1\}$$

is bounded by $O(2^{N/3})$, and we know that all functions that are in an optimal subcomputation for f are in this set. Hence, what we would like to do is enumerate the functions in this set, however, the author does not know how to do this efficiently. Instead, we examine the subset of functions in this set that have not too large complexity. That is, we iterate through all functions with complexity at most $\frac{2}{3} \cdot \frac{2^n}{\log n}$ and build

$$Candidates = \{g : \mathsf{L}(\mathsf{Select}[f,g]) - \mathsf{L}(f) \leq 1 \text{ and } \mathsf{L}(g) \leq \frac{2}{3} \cdot \frac{2^n}{\log n}\}.$$

This takes time $O(2^{2N/3})$. We then try to find a pair of functions in $Candidates$ that form an optimal subcomputation.

If we succeed, we are done. Otherwise, we know that there exists an optimal subcomputation $\{g, \nabla, h\}$ of f where h has complexity greater than $\frac{2}{3} \cdot \frac{2^n}{\log n}$. This also implies that g has complexity at most $(1 + o(1))\frac{2^n}{3 \log n}$ since $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ and $\mathsf{L}(f) \leq (1 + o(1))\frac{2^n}{\log n}$.

In the third part, we look for such an "unbalanced" subcomputation as follows. We iterate through each g in *Candidates* with complexity at most $(1 + o(1))\frac{2^n}{3 \log n}$ and each $\nabla \in \{\wedge, \vee\}$ and try to find a matching h by considering each h satisfying $f = g \nabla h$. We argue that this is efficient because the the set of h satisfying the constraint that $f = g \nabla h$ is not too large (in particular, of size at most $2^{N/3}$). The reason for why this set must be small is that the constraint that $f = g \nabla h$ actually forces many of the values of h to a fixed zero/one value. Indeed, we argue that a large number of values must be "forced," since if only a small number of values of h were "forced," then a theorem of Pippenger [16] ensures that there would be a function h of too small complexity (smaller than $\frac{2}{3} \cdot \frac{2^n}{\log n}$) that satisfied $f = g \nabla h$, which would contradict that fact that the second part of the algorithm failed.

A bottom-up approach. Our final algorithm takes a different approach than our previous reductions, working bottom-up instead of top-down. The basic idea of the bottom-up approach is as follows. Begin with the set *Candidates* of all functions computed by formulas of size one. For each pair of functions g, h in *Candidates* and each $\nabla \in \{\wedge, \vee\}$, compute the function $q = g \nabla h$. Next, see if $g \nabla h$ is an optimal formula for q using the MFSP oracle. If so, use some one-sided heuristic (that never gives an incorrect NO answer) to test if q is computed by some gate in an optimal formula for f, and add q to *Candidates* if it passes this heuristic. Repeat this process until f is added to *Candidates*, in which case one can construct an optimal formula for f by tracing back through the functions that led to it.

The difficulty in this approach is in finding an appropriate heuristic that significantly prunes the search space of possible candidates. A natural contender for such a heuristic, in light of our previous algorithms, is testing if $\mathsf{L}(\mathsf{Select}[f, q]) - \mathsf{L}(f)$ is small. However, if our only guarantee is that q is computed by some gate in some optimal formula φ for f, the best upper bound we manage to prove for the quantity $\mathsf{L}(\mathsf{Select}[f, q]) - \mathsf{L}(f)$ is linear in the depth of φ.

Luckily, Lemma 53 shows that the depth of φ cannot be too large. In particular, if φ is an optimal DeMorgan formula, then the depth of φ is bounded by $O(\frac{2^n}{n})$ where n is the number of inputs f takes. At a high-level, the proof of this lemma works by saying that if a formula has very large depth, then there are many small subformulas that lie along a path in the binary tree of φ. Because there are so many of these small subformulas, there must be a pair that compute the same function, and this can be used to produce a slightly smaller formula.

Using this lemma, we show that the above bottom-up approach runs in time quadratic in the number of formulas for computing f that are within $O(\frac{2^n}{n})$ of being optimal, additively.

1.4 Open Questions

There are several intriguing questions raised by this work. Looking at our main theorem, the most obvious question is whether one can improve the bound we give on the number of near-optimal formulas for a random function. Our bound hardly seems correct, although its hard to imagine how one could do better with current techniques.

Perhaps an indirect approach could work. Is there any operation one can apply to a function in order to reduce the number of optimal formulas it has? It seems plausible that multiple applications of the $\mathsf{Select}[\cdot, \cdot]$ function might cut down the number of optimal formulas.

Another idea would be to try to modify the heuristic "tests" in our reduction. At their heart, all our "tests" are powered by the gate elimination technique. It seems reasonable that more powerful lower bound techniques (which we indeed do have for formulas) might lead to better heuristics and thus more efficient search-to-decision reductions.

There is also the question this paper began with: can one prove a non-trivial exact search to decision reduction for MCSP? The difficulty in adopting our approach to MCSP is that there are just too many ways to add a single gate to a circuit, which ruins the bounds we get on the number of functions passing our $\mathsf{Select}[f, g]$ test. Is there any way to get around this?

Taking a step back, one can also ask what role relativization plays in the search versus decision question. Can one show that there is an oracle relative to which MCSP or MFSP can be solved in linear time, but the corresponding search problem requires exponential time?

Finally, can one extend Lemma 9 to the case of formulas over $\mathbb{B}_2$ or even just prove a better bound for DeMorgan formulas?

1.5 Organization

In Section 2, we fix our notation and definitions, including our notion of formula isomorphism. In Section 3, we introduce the top-down approach and outline our basic strategy for solving Search-MFSP. Section 4 introduces the $\mathsf{Select}[\cdot,\cdot]$ function and proves bounds on number of functions that pass "tests" related to the $\mathsf{Select}[\cdot,\cdot]$ function. Section 5 gives a deterministic search to decision reduction for MFSP and shows it is efficient on average. Section 6 then gives a reduction that works in the worst case. Finally, Section 7 demonstrates a bottom-up approach for trying to solve Search-MFSP.

2 Preliminaries

For a positive integer n, we let $[n]$ denote the subset of integers $\{1, \ldots, n\}$.

2.1 DeMorgan Formulas and Formula Size

Our notion of formulas will be DeMorgan formulas. A DeMorgan formula φ on n-inputs of size s is given by:

- a directed rooted binary tree on the vertex set $[2s-1]$, specified by a subset $E_\varphi \subseteq [2s-1] \times [2s-1]$ of edges, and
- a gate labeling function $\tau_\varphi : [2s-1] \to \{\wedge, \vee\} \cup \{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$

where τ takes values in $\{\wedge, \vee\}$ on the internal nodes in φ and τ takes values in

$$\{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$$

on the leaf nodes in φ. The edges in E_φ point from inputs towards outputs. We note that our definition implicitly uses the fact that a binary tree with s leaf nodes has $s-1$ internal nodes. We also note that in our definition we do not need to specify the "left" and "right" child of an internal node since our gate set $\{\wedge, \vee\}$ is made up of symmetric functions. We will define a notion of formula isomorphism in Section 2.2.

We will use the notation $|\varphi|$ to denote the *size* of a formula φ (i.e. the number of leaves in the binary tree underlying φ). Given a Boolean function f, we denote the *minimum formula size* of f by

$$\mathsf{L}(f) = \min\{|\varphi| : \ \varphi \text{ is a formula computing } f\}.$$

We say a formula φ is an *optimal* formula for a Boolean function f, if φ computes f and $|\varphi| = \mathsf{L}(f)$.

We note, however, that all of our results except the ones presented in Section 7 apply equally well to formulas with arbitrary fan-in-two gates (i.e. the formulas over the $\mathbb{B}_2$ basis). Moreover, all our results are hold for other size notions such as gates and wires.

2.2 Optimal Formulas and Formula Isomorphism

Since our results will depend on the number of formulas satisfying certain properties, we will be clear about when exactly we are saying formulas are distinct in our count.

In particular, as we have defined formulas, one can obtain many optimal formulas from a single optimal formula by relabeling the nodes in underlying binary tree.

Thus, it will be useful to define an isomorphism on formulas and only count formulas modulo this isomorphism. In particular, we will define two formulas to be *isomorphic* if they are isomorphic as labelled binary trees.

In order to properly define this, we introduce some notation. If φ is a formula of size s with an underlying edge set E_φ and a labelling function τ_φ and $\sigma : [2s-1] \to [2s-1]$ is a permutation, then we let $\psi = \sigma(\varphi)$ be the formula of size s whose edge set E_ψ is given by

$$E_\psi = \{(\sigma(i), \sigma(j)) : (i,j) \in E_\varphi\}$$

and whose labelling function τ_ψ is given by

$$\tau_\psi(\sigma(i)) = \tau_\varphi(i).$$

We say two formulas φ and φ' are *isomorphic* if $|\varphi| = |\varphi'|$ and there is a permutation σ such that $\varphi' = \sigma(\varphi)$.

From each equivalence class of isomorphic formulas, we pick a single representative that we call the *canonical formula* for that equivalence class. Note that for our purposes we do not need that this canonical formula to be computable, as we will just be using them in our analysis. Then we define $\mathsf{CanonOpt}_k\mathsf{Formulas}(f)$ to be the set of canonical formulas that are optimal for computing f up to an additive k-term. In other words

$$\mathsf{CanonOpt}_k\mathsf{Formulas}(f) = \{\varphi : \varphi \text{ is a canonical formula and } |\varphi| \leq \mathsf{L}(f) + k\}.$$

2.3 MFSP, Search-MFSP and Conventions on n and N

We now define the *Minimum Formula Size Problem* denoted MFSP.

▶ **Definition 10** (MFSP). *We define the problem* MFSP *as follows*

- ***Given:*** *a truth table of a Boolean function f and an integer size parameter $s \geq 1$*
- ***Determine:*** *if $\mathsf{L}(f) \leq s$.*

We define the search version of MFSP analogously.

▶ **Definition 11** (Search-MFSP). Search-MFSP *is the problem defined as follows:*

- ***Given:*** *a truth table of a Boolean function f*
- ***Output:*** *a formula φ of size $\mathsf{L}(f)$ computing f.*

We note that $\mathsf{MFSP} \in \mathsf{NP}$ since given a minimum-sized formula as a witness, one can check that this indeed computes f efficiently since the truth table of f is provided and every function has a formula of size at most the length of its truth table (see Theorem 13).

When describing a function f that is an input to MFSP, one naturally wants to denote by n two different quantities: the number of variable inputs to a function f and the length of the truth table of f (which is the true input length for MFSP). We maintain the convention throughout this paper that n denotes the input arity of f and $N = 2^n$ denotes the length of the truth table of f.

2.4 Useful Facts About Formulas

We will make use of some basic facts about formulas in our work. First, one can easily bound the number of formulas of size at most s.

▶ **Proposition 12.** *The number of formulas on n-inputs of size at most s is at most* $2^{s \log n(1+o(1))}$

We also know tight upper bounds on the maximum formula complexity of a n-input function.

▶ **Theorem 13** (Lozhkin [11] improving on Lupanov [12])**.** *Let $f : \{0,1\}^n \to \{0,1\}$. Then*

$$\mathsf{L}(f) \leq \frac{2^n}{\log n}(1 + O(\frac{1}{\log n}))$$

Combining the size upper bound in Theorem 13 with the bound on the number of formulas of size s, we get the following proposition.

▶ **Proposition 14** (Random functions have not too many near optimal formulas)**.** *Let n and k be positive integers. Let $N = 2^n$. Assume $k = O(\frac{2^n}{\log^2 n})$. Then all but a $o(1)$-fraction of n-input Boolean functions f satisfy*

$$|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)| = 2^{O(\frac{N}{\log\log N})}.$$

Proof. Theorem 13 say that every n-input function has a formula of size at most

$$\frac{2^n}{\log n}(1 + O(\frac{1}{\log n})).$$

Thus, any formula for computing n-input function that is within an additive k of being optimal has size at most s where

$$s \leq k + \frac{2^n}{\log n}(1 + O(\frac{1}{\log n})) = \frac{2^n}{\log n}(1 + O(\frac{1}{\log n})).$$

Proposition 12 implies that the number of formulas of size at most s is upper bounded by

$$2^{s \log n(1+o(1))} = 2^{N(1+O(\frac{1}{\log\log N}))}.$$

Hence, since there are 2^N Boolean functions on n-inputs, it follows that in expectation a random function has at most

$$2^{O(\frac{N}{\log\log N})}$$

formulas within k of being optimal. The desired claim then follows by an application of Markov's inequality. ◀

We note that the bound given by Proposition 14 is actually counting formulas that are isomorphic to each other as distinct. Unfortunately removing this redundancy does not improve on the bound in Proposition 14. However, the fact that our results rely on the number of distinct formulas up to isomorphism means that there is no obvious obstruction to better bounds being proved and hence to our algorithms being more efficient.

We will also make use of the fact that integer comparison can be implemented by linear-sized formulas.

▶ **Proposition 15** (Small formulas for integer comparison). *Let $y \in \{0,1\}^n$. Let $GrtrThan_y : \{0,1\}^n \to \{0,1\}$ be the function given by $GrtrThan_y(x) = 1$ if and only if $x > y$ in the usual lexicographic order on $\{0,1\}^n$. Then $\mathsf{L}(GrtrThan_y(x)) \leq n$.*

Proof. We work by induction on n. If $n = 1$, then clearly $\mathsf{L}(GrtrThan_y) = 1$ (either it is 0 if $y = 1$ or it equals x if $y = 0$).

Now suppose $n > 1$. Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ denote the bits of x and y respectively where x_1 and y_1 denotes the highest order bit. Let $x', y' \in \{0,1\}^{n-1}$ be given by $x' = x_2 \ldots x_n$ and $y' = y_2 \ldots y_n$ respectively.

Now, $x > y$ if and only if one of the following two statements is true:

- $x_1 > y_1$, or
- $x_1 = y_1$ and $x' > y'$.

Since $x_1, y_1 \in \{0,1\}$, this is equivalent to

$$x > y \iff (x_1 > y_1) \vee (x' > y').$$

By induction this means $\mathsf{L}(GrtrThan_y) \leq 1 + n - 1 = n$. ◀

2.5 Partial Functions and their Formula Size

Partial functions will be a crucial building block in our reductions. A *partial Boolean function* is a function $\gamma : \{0,1\}^n \to \{0,1,?\}$ for some integer $n \geq 1$. We denote partial functions using Greek letters such as γ and μ, although sometimes we resort to the Roman alphabet with a ? subscript such as $h_?$.

In contrast, we say a Boolean function $f : \{0,1\} \to \{0,1\}$ is *total* Boolean function (though we allow for a partial Boolean function to indeed be total).

We say a total Boolean function g *agrees* with a partial Boolean function γ if

$$\gamma(x) \in \{0,1\} \implies \gamma(x) = g(x).$$

One can naturally define the minimum formula size of a partial Boolean function γ as follows

$$\mathsf{L}(\gamma) = \min\{\mathsf{L}(g) : g \text{ is a total function that agrees with } \gamma\}.$$

The following theorem regarding the formula complexity of partial functions will be useful in our randomized worst-case reduction.

▶ **Theorem 16** (Pippenger [16]). *Let $\gamma : \{0,1\}^n \to \{0,1,?\}$ be a partial function. Let $p_? = \frac{|\gamma^{-1}(?)|}{2^n}$. Then,*

$$\mathsf{L}(\gamma) \leq (1 + o(1)) \cdot (1 - p_?) \frac{2^n}{\log n}.$$

3 The Top-Down Approach

Our two main reductions both take a "top-down" approach to finding an optimal formula. That is, given a function f, they try to find functions g and h such that g and h are the two functions fed into the final output gate in an optimal formula for f and then recursing.

This is formalized as follows.

▶ **Definition 17** (Optimal Subcomputations Set). *Let $f : \{0,1\}^n \to \{0,1\}$. We define the set of* optimal subcomputations *for f, denoted* $\mathsf{OptSubcomps}(f)$*, as follows.*

Let $g, h : \{0,1\}^n \to \{0,1\}$ be Boolean functions of the same arity as f and $\nabla \in \{\wedge, \vee\}$. Then $\{g, \nabla, h\} \in \mathsf{OptSubcomps}(f)$ if and only if there exists an optimal formula $\varphi = \varphi_g \nabla \varphi_h$ for computing f such that φ_g computes g and φ_h computes h.

We note that in this definition we are implicitly using that the gate set $\{\wedge, \vee\}$ is symmetric with respect to its inputs.

We say a function g is *in an optimal subcomputation* for f if g is contained in some element of $\mathsf{OptSubcomps}(f)$. In other words, g is in an optimal subcomputation for f if there exists an h and ∇ such that $\{g, \nabla, h\} \in \mathsf{OptSubcomps}(f)$.

It is easy to see that $\mathsf{OptSubcomps}(f)$ is almost always non-empty.

▶ **Proposition 18.** *Let $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{L}(f) \geq 2$. Then $\mathsf{OptSubcomps}(f)$ is non-empty.*

Next, we can define the problem of finding an optimal subcomputation.

▶ **Definition 19** (Decomposition Problem). *The Decomposition Problem,* DecompProblem *is as follows:*

- ***Given:*** *the truth table of a Boolean function f satisfying $\mathsf{L}(f) \geq 2$*
- ***Output:*** *some element of $\mathsf{OptSubcomps}(f)$.*

It is easy to see that DecompProblem is equivalent to Search-MFSP. DecompProblem can be easily solved with an oracle to Search-MFSP. The following recursive procedure shows the reverse direction.

▶ **Theorem 20** (Search-MFSP reduces to DecompProblem). *There is a deterministic $O(N^2)$-time algorithm for solving* Search-MFSP *on inputs of length N given access to an oracle that solve* DecompProblem *on instances of length N.*

Proof. The pseudocode for this reduction is written in Algorithm 1.

Algorithm 1 Reduction from Search-MFSP to DecompProblem.

procedure FindOptFormula(f)
▷ Given the length-N truth table of a function f that takes n-inputs and oracle access to DecompProblem return an optimal formula for f.
  if there exists a size one formula φ computing f **then**
    return φ.
  end if
  Let $\{g, \nabla, h\}$ be the output returned by the oracle $\mathsf{DecompProblem}(f)$.
  Recursively compute the formula $\varphi_g \leftarrow FindOptFormula(g)$.
  Recursively compute the formula $\varphi_h \leftarrow FindOptFormula(h)$.
  return the formula given by $\varphi_g \nabla \varphi_h$.
end procedure

The correctness of this algorithm is easy to see as long as one is able to bound the number of recursive calls the algorithm makes. To see that the number of recursive calls is bounded by $O(N)$, notice that each iteration of the algorithm reveals one more gate in the optimal formula for f. Thus, since $\mathsf{L}(f) = O(N)$, we have that there are at most $O(N)$ recursive calls. ◀

Our goal is now to try to solve DecompProblem (i.e. find an element of OptSubcomps(f)) given an oracle to MFSP. Recall from the introduction that our high-level approach is as follows

1. Find an efficient "test" that functions that in an optimal subcomputation of f pass but not too many other functions pass.
2. Efficiently build the (not too long) list *Candidates* of things that pass the test.
3. Iterate through all pairs of elements in *Candidates* and all possible gates, and efficiently check if this yields an element of OptSubcomps(f).

Item 1 will be the subject of Section 4, Item 2 will be different in our two main reductions, and Item 3 is provided by the next lemma.

▶ **Lemma 21** (Test membership in OptSubcomps(f) efficiently with MFSP)**.** *Let* $f, g, h : \{0,1\}^n \to \{0,1\}$. *Then*

$$\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f) \iff f = g \triangledown h \text{ and } \mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h).$$

Proof. We prove the forward direction first. Suppose that $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$. Then there exists an optimal formula $\varphi = \varphi_g \triangledown \varphi_h$ for computing f such that φ_g computes g and φ_h computes h. Clearly this implies that $f = g \triangledown h$.

Moreover, $|\varphi| = |\varphi_g| + |\varphi_h|$. On the other hand, since φ is optimal, we have that $|\varphi| = \mathsf{L}(f)$, $|\varphi_g| = \mathsf{L}(g)$, and $|\varphi_h| = \mathsf{L}(h)$. (Otherwise, one could build a smaller formula for f by replacing φ_g or φ_h with a smaller formula computing the same function.) Hence $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$.

For the reverse direction, suppose that $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ and $f = g \triangledown h$. Let φ_g and φ_h be optimal formulas for g and h. Then $\varphi = \varphi_g \triangledown \varphi_h$ clearly computes f and has size $\mathsf{L}(f)$. Hence $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$. ◀

4 Using gate elimination to find functions in an optimal subcomputation

Our approach to solving DecompProblem involves finding a "test" that functions in an optimal subcomputation pass but not too many other functions pass. The test will be based off the following function.

▶ **Definition 22** (Select$[\cdot,\cdot]$)**.** *Let* $f, g : \{0,1\}^n \to \{0,1\}$. *We define the function* $\mathsf{Select}[f,g] : \{0,1\}^n \times \{0,1\} \to \{0,1\}$ *by*

$$\mathsf{Select}[f,g](x,z) = \begin{cases} f(x) & , \text{ if } z = 0 \\ g(x) & , \text{ if } z = 1 \end{cases}$$

We emphasize that $\mathsf{Select}[f,g]$ function is only defined when f and g have the same arity.

Now, our "test" will be to see if the quantity

$$\mathsf{L}(\mathsf{Select}[f,g]) - \mathsf{L}(f)$$

is small (how small will depend on our reduction).

Indeed, for functions in an optimal subcomputation, this quantity is exactly one![7]

[7] We will only prove that it is as most one, but the reader can check that if $g \neq f$ that a gate elimination argument actually implies equality.

▶ **Lemma 23.** *Suppose g is in an optimal subcomputation for f. Then*

$$\mathsf{L}(\mathsf{Select}[f,g]) \leq \mathsf{L}(f) + 1.$$

Proof. Since g is in an optimal subcomputation for f, there exists an optimal formula $\varphi = \varphi_g \bigtriangledown \varphi_h$ such that φ_g computes g. If $\bigtriangledown = \wedge$, then

$$\varphi_g \wedge (\varphi_h \vee z)$$

is a formula for $\mathsf{Select}[f,g]$ of size $\mathsf{L}(f)+1$. Otherwise $\bigtriangledown = \vee$. Then

$$\varphi_g \vee (\varphi_h \wedge \neg z)$$

is a formula for $\mathsf{Select}[f,g]$ of size $\mathsf{L}(f)+1$. ◀

On the other hand, the number of functions that "pass this test" can be upper bounded in terms of $|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)|$.

▶ **Lemma 24.** *Let k be a positive integer. Let $f : \{0,1\}^n \to \{0,1\}$. Assume $\mathsf{L}(f) \geq 2$. Let $TestPassers = \{g : \mathsf{L}(\mathsf{Select}[f,g]) - \mathsf{L}(f) \leq k+1\}$. Then*

$$|TestPassers| \leq O(|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)| \cdot 2^k N \log N)$$

where $N = 2^n$.

Proof. At a high-level the idea is that, given a formula φ of size $\mathsf{L}(f)+k+1$ for computing $\mathsf{Select}[f,g]$, one can replace the z-leaves in φ with 0-leaves to obtain a formula φ' of size $\mathsf{L}(f)+k+1$ for computing f with at least one constant leaf. One can then use a careful gate elimination argument to remove precisely one constant leaf from φ' to obtain a formula φ'' that still computes f but has size $\mathsf{L}(f)+k$. On the other hand, one can reverse this process by adding some constant leaf and gate to φ'' and then replacing some subset of the constant leaves by z-leaves.

This gives us a way to describe any g that passes the test, and thus allows us to bound the number of such g. In our bound, the $O(|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)|)$ factor corresponds to the choices for φ'', the $O(N \log N)$ corresponds to the number of ways to add a new constant leaf and gate to φ'' in order to obtain φ', and the $O(2^k)$ factor comes from the number of ways to chose a subset of the (at most k) 0-leaves in φ' into z-leaves.

In detail, we prove this statement by giving a series of injections. At a high-level First, let P denote the set of canonical formulas computing f with size exactly $\mathsf{L}(f)+k+1$ and with at least one constant-labelled leaf node in the formula.

We will give an injection from $TestPassers$ to $P \times [2^{k+1}]$. Before defining our injection, we will need the following claims and definitions.

▷ Claim 25. Suppose φ computes $\mathsf{Select}[f,g]$ and $f \neq g$, then φ has at least one leaf node labelled by z or $\neg z$.

Proof. If φ does not have any $\{z, \neg z\}$ labelled leaves, then the output of φ does not depend on z. But this contradicts that φ computes $\mathsf{Select}[f,g]$ since $\mathsf{Select}[f,g]$ does depend on the z input because $f \neq g$. ◁

▷ Claim 26. Suppose $\varphi \in P$. Then φ has at most $(k+1)$-many leaf nodes labelled by constants $\{0,1\}$.

Proof. Since $\varphi \in P$, we know that $|\varphi| = \mathsf{L}(f) + k + 1$ and φ computes f. Since $\mathsf{L}(f) \geq 2$, we know that $|\varphi| \geq k + 3$.

If φ had more than $(k+1)$-many constant labelled leaves, it follows by a standard gate elimination argument (note here it is important that $|\varphi| \geq k + 3$) that there is a φ' that computes the same function as φ such that

$$|\varphi'| < |\varphi| - (k+1) < \mathsf{L}(f).$$

But then φ would be a formula of size less than $\mathsf{L}(f)$ computing f which is a contradiction. ◁

We will also need the following definitions. Given a formula φ that can take z-variables as input, we define $\mathsf{Substitute}_{z=0}(\varphi)$ to be the formula φ' given by replacing the z-labeled leaves in φ with 0-labels and replacing the $(\neg z)$-labeled leaves in φ with 1-labels.

We note that the $\mathsf{Substitute}_{z=0}$ operation in some sense respects formula isomorphisms.

▷ Claim 27. Let φ be a formula of size s that takes a z-variable as input. Let $\sigma : [2s-1] \to [2s-1]$ be a permutation. Then

$$\sigma \circ \mathsf{Substitute}_{z=0}(\varphi) = \mathsf{Substitute}_{z=0} \circ \sigma(\varphi)$$

Proof. The proof is essentially just applying the definition to both sides and seeing that the resulting edge sets and labelling functions are equal. ◁

We can also define a reverse operation to $\mathsf{Substitute}_{z=0}$ as follows. Given a formula φ and a subset S of leaf nodes in φ that are labelled by constants $\{0,1\}$, define $\mathsf{Unsubstitute}_{z=0}(\varphi', S)$ to be the formula φ given by replacing 0-labeled leaves in S with z-labels leaves and by replacing 1-labeled leaves in S with $(\neg z)$-labels.

Indeed, the following claim whose proof we omit is easy to see.

▷ Claim 28. For all formulas φ there exists a set S such that

$$\varphi = \mathsf{Unsubstitute}_{z=0}(\mathsf{Substitute}_{z=0}(\varphi), S).$$

Being more precise, S is a subset of the leaf nodes in $\mathsf{Substitute}_{z=0}(\varphi)$ that are labelled by constants.

Now we are ready to describe our injection from $TestPassers \to P \times [2^{k+1}]$ on an input $g \in TestPassers$. Since $g \in TestPassers$, there is a φ of size $\mathsf{L}(f) + k + 1$ computing $\mathsf{Select}[f, g]$. Let $\varphi' = \mathsf{Substitute}_{z=0}(\varphi)$. Clearly φ' computes f since φ computes $\mathsf{Select}[f, g]$. Let $\overline{\varphi'}$ denote the canonical formula isomorphic to φ'. Then there exists a permutation σ such that

$$\begin{aligned}\overline{\varphi'} &= \sigma(\varphi') \\ &= \sigma \circ \mathsf{Substitute}_{z=0}(\varphi) \\ &= \mathsf{Substitute}_{z=0} \circ \sigma(\varphi)\end{aligned}$$

where the last equality comes from Claim 27. Thus, using Claim 28, we know that there exists a subset S of the leaf nodes of $\overline{\varphi'}$ labelled by constants such that

$$\mathsf{Unsubstitute}_{z=0}(\overline{\varphi'}, S) = \sigma(\varphi).$$

Moreover, the set S can be viewed as an element of $[2^{k+1}]$ because $\overline{\varphi'}$ has at most $k+1$ leaf nodes labelled by constants. In particular, by construction, we have that

$$|\overline{\varphi'}| = |\varphi|' = |\varphi| = \mathsf{L}(f) + k + 1,$$

and that $\overline{\varphi'}$ computes f, so Claim 26 ensures that φ' has at most $k+1$ many leaf nodes labelled by constants.

Hence, we define the output of our injection from $TestPassers$ to $P \times [2^{k+1}]$ on input $g \in TestPassers$ to be $(\overline{\varphi'}, S)$.

We must prove that this is indeed an injection. Towards this end, we claim that $\mathsf{Unsubstitute}_{z=0}(\overline{\varphi'}, S)$ is a formula computing $\mathsf{Select}[f, g]$. From this claim it is easy to see that this must be an injection.

▷ Claim 29. $\mathsf{Unsubstitute}_{z=0}(\overline{\varphi'}, S)$ is a formula computing $\mathsf{Select}[f, g]$.

Proof. S was chosen so that

$$\mathsf{Unsubstitute}_{z=0}(\overline{\varphi'}, S) = \sigma(\varphi).$$

Thus, $\mathsf{Unsubstitute}_{z=0}(\overline{\varphi'}, S)$ computes the same function as $\sigma(\varphi)$ which in turn computes the same function as φ, which computes $\mathsf{Select}[f, g]$ as desired. ◁

From this injection, we get that

$$|TestPassers| \leq 2^{k+1} \cdot |P|.$$

Next, we give an injection from P to the set

$$\mathsf{CanonOpt}_k\mathsf{Formulas}(f) \times [2\mathsf{L}(f)] \times \{\wedge, \vee\} \times \{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}.$$

To do this we will define an operation $\mathsf{DropLeaf}(\varphi, i)$ that takes as input a formula φ of size $s \geq 2$ and a leaf node $i \in [2s-1]$ from φ and outputs the formula φ' given as follows. We will first describe φ' informally and then give the formal description. φ' is obtained by deleting the leaf node i and making the output of the node i_p that i fed into simply the other node that was being fed into i_p.

Now, we formally describe φ'. Let $i_p \in [2s-1]$ be the internal node that i has an edge to in φ (we know this exists because $|\varphi| \geq 2$). If needed, apply a permutation to φ so $i = 2s-1$ and $i_p = 2s-2$. Let $u \in [2s-3]$ be the other node in φ that feeds into v_p. Let φ' be the formula given by the edge set

$$E' = (E \cap ([2s-3] \times [2s-3])) \cup \{\ (u, v_{pp}) : v_p \text{ feeds into } v_{pp} \text{ in } \varphi\}$$

and the labelling function

$$\tau' = \tau|_{[2s-3]}.$$

For example if $\varphi = (x_1 \vee x_2) \wedge x_3$ and 1 was then index of the x_1 leaf, then $\mathsf{DropLeaf}(\varphi, 1) = x_2 \wedge x_3$.

We show this operation in some sense commutes with formula isomorphisms.

▷ Claim 30. Let φ be formula of size s. Let i be the index a leaf node in φ, and let $\sigma : [2s-3] \to [2s-3]$ be a permutation. Then there exists an integer i' and a permutation $\sigma' : [2s-1] \to [2s-1]$ such that

$$\mathsf{DropLeaf}(\sigma'(\varphi), i') = \sigma \circ \mathsf{DropLeaf}(\varphi, i)$$

Proof. From our definition of $\mathsf{DropLeaf}(\cdot,\cdot)$, we can assume without loss of generality that $i = 2s-1$ and that the internal node that i feeds into in φ is $i_p = 2s-2$.

Then the claim follows from letting σ be equal to σ' on $[2s-3]$ and letting σ be the identity on $\{2s-2, 2s-1\}$ and applying the various definitions. ◁

We can also define a kind of inverse operation $\mathsf{AddLeaf}$ function that takes the following four inputs

- a formula φ' on n-inputs of size s,
- a node i in the tree given by φ',
- a gate $\nabla \in \{\wedge, \vee\}$, and
- a leaf label $\ell \in \{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$,

and outputs the formula φ of size $s+1$ given as follows. First, we give an informal description and then given a formal definition.

Intuitively, $\mathsf{AddLeaf}$ is adding a new ∇-gate into φ' between the i-node and wherever i was being output to (if i has an output), whose other input is a new ℓ-labeled leaf.

We define φ formally as follows. We will use $2s+1$ to add in our new leaf and $2s$ to add in our new gate. The edge set E_φ of φ is given by taking $E_{\varphi'}$ and adding in the edges $(2s+1, 2s)$ and $(i, 2s)$ and then, if there is a node i_p that i feeds into in φ', adding in an edge $(2s, i_p)$ and removing the (i, i_p) edge. The node labelling τ_φ of φ is given by

$$\tau_\varphi(i) = \begin{cases} \tau_{\varphi'} & , \text{ if } i \in [2s-1] \\ \nabla & , \text{ if } i = 2s \\ \ell & , \text{ if } i = 2s+1 \end{cases}$$

It is easy to see that $\mathsf{AddLeaf}$ can reverse a $\mathsf{DropLeaf}(\cdot,\cdot)$ operation.

▷ Claim 31. Let φ be a formula of size at least two. Let i be the index of a leaf node in φ. Then there exists an integer j, a gate $\nabla \in \{\wedge, \vee\}$, and a leaf label $\ell \in \{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$ such that

$$\mathsf{AddLeaf}(\, \mathsf{DropLeaf}(\varphi, i), j, \nabla, \ell) = \varphi$$

One of the main steps in our injection will be provided by the following claim.

▷ Claim 32. Let $\varphi \in P$. Then there is an i such that $\varphi' = \mathsf{DropLeaf}(\varphi, i)$ is a size $(\mathsf{L}(f)+k)$ formula for computing f.

Proof. Let $\varphi \in P$. Then there is some internal node j in φ that takes as input a leaf node indexed by i satisfying $\tau_\varphi(i) = b \in \{0,1\}$. We will assume $b = 0$ (the proof in the $b = 1$ case is similar).

Let φ_j be the the subformula computed at node j, and let $\nabla_j = \tau_\varphi(j) \in \{\wedge, \vee\}$ be the gate label of j. We already know that the ith leaf node feeds into φ_j. Let k be the other node feeding into j, and let φ_k be the subformula computed at node k. We split into cases depending on whether ∇_j is an $\wedge$-gate or a $\vee$-gate.

First, let us suppose ∇_j is a $\vee$-gate. Then the formula φ_j as a function is equivalent to the formula $0 \vee \varphi_k$ which is equivalent as a function to φ_k. Hence it follows that $\varphi' = \mathsf{DropLeaf}(\varphi, i)$ is an $(\mathsf{L}(f)+k)$-size formula (since we removed the ith leaf) computing f (since φ_j and φ_k compute the same function).

Now, suppose that it is a $\wedge$-gate. Then the output of φ_j is always zero. Since $|\varphi_j| \geq 2$, there exists some subformula φ_2 of φ_j of size 2 (i.e. there is some node in φ_v that has two leaves as children and φ_2 is the subformula computed at that node). Since φ_2 has two leaves, there exists one leaf index i' such that $i' \neq i$ (i.e. this leaf node is not the ith leaf node we were considering before).

Then we claim that $\varphi' = \mathsf{DropLeaf}(\varphi, i')$ is an $(\mathsf{L}(f)+k)$-size formula computing f. It is easy to see that $|\varphi'|$ is an $(\mathsf{L}(f)+k)$-size formula since we removed the i'th leaf node. To see that φ' still computes f, note that the 0-labeled ith leaf node in φ still exists in φ'. If the gate node j was removed by the $\mathsf{DropLeaf}(\cdot,\cdot)$ operation, then the output wire of φ_j that computed the 0 function in φ has been replaced by the 0-leaf i in φ' which still computes the 0 function, so φ' must still compute f. If the gate node v was not removed by the $\mathsf{DropLeaf}(\cdot,\cdot)$ operation, then the output corresponding gate to v in φ' is still computing 0 (since it is an $\wedge$-gate with a 0 input), so φ' still computes f. ◁

Now we can finally describe the injection from P to the set

$$\mathsf{CanonOpt}_k\mathsf{Formulas}(f) \times [2\mathsf{L}(f)+2k-1] \times \{\wedge,\vee\} \times \{0,1,x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n\}.$$

Given an input φ in P, we have by Claim 32 that there exists a least i-value such that $\varphi' = \mathsf{DropLeaf}(\varphi, i)$ is a formula computing f of size $\mathsf{L}(f)+k$. Let $\overline{\varphi'}$ be the canonical formula isomorphic to φ'. Then we have that $\overline{\varphi'} \in \mathsf{CanonOpt}_k\mathsf{Formulas}(f)$ and we have that there are permutations σ and σ' and an integer i' such that

$$\begin{aligned}\overline{\varphi'} &= \sigma(\varphi') \\ &= \sigma \circ \mathsf{DropLeaf}(\varphi\,, i) \\ &= \mathsf{DropLeaf}(\sigma'(\varphi), i')\end{aligned}$$

where the last equality comes from Claim 30.

Hence, by Claim 31, we have that there exists a gate index $j \in [2\mathsf{L}(f)+2k-1]$, a gate $\nabla \in \{\wedge,\vee\}$, and a leaf label $\ell \in \{0,1,x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n\}$ such that

$$\mathsf{AddLeaf}(\overline{\varphi'}, j, \nabla, \ell, D) = \sigma'(\varphi).$$

In other words, $\mathsf{AddLeaf}(\overline{\varphi'}, j, \nabla, \ell, D)$ outputs a formula isomorphic to φ.

Thus, we set the output of the injection on input φ to be $(\overline{\varphi'}, j, \nabla, \ell, D)$. The fact that this is an injection from P is ensured by the fact that $\mathsf{AddLeaf}(\overline{\varphi'}, j, \nabla, \ell, D)$ is isomorphic to φ and the fact that P contains only canonical formulas.

Hence, we get that

$$\begin{aligned}|P| &\leq |\mathsf{CanonOpt}_k\mathsf{Formulas}(f)| \cdot 2(\mathsf{L}(f)+k) \cdot 2 \cdot (2n+1) \\ &\leq O(|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)|(N+k)\log N).\end{aligned}$$

Combining this with upper bound on $TestPassers$ in terms of P, we get that

$$|TestPassers| \leq O(|\mathsf{CanonOpt}_k\mathsf{Formulas}(f)| \cdot 2^k N \log N)$$ ◀

5 A deterministic reduction that works on average

We will now use the tools developed in Section 3 and Section 4 to give a search to decision reduction that is efficient on functions with few near-optimal formulas.

▶ **Theorem 33.** *There is a deterministic algorithm solving* Search-MFSP *on inputs of length N given access to an oracle that solves* MFSP *on instances of length $2N$ that runs in time $O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)|^2 \cdot N^6 \log^2 N)$ where $n = \log N$.*

Before we prove Theorem 33, we state a corollary that follows from the bound on the size of $\mathsf{CanonOpt}_k\mathsf{Formulas}(f)$ for a random function given in Proposition 14.

▶ **Corollary 34.** *There is a deterministic algorithm solving* Search-MFSP *on inputs of length N given access to an oracle that solves* MFSP *on instances of length $2N$ that runs in time $2^{O(\frac{N}{\log\log N})}$ on all but a $o(1)$-fraction of instances.*

Proof of Corollary 34. This corollary follows by combining the algorithm in Theorem 33 with the recursive algorithm for Search-MFSP given in Theorem 20 and using Proposition 14 to bound $|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)|$ by $2^{O(\frac{N}{\log\log N})}$ for a random function.

There is actually a subtlety in appealing to Theorem 20 in that the running time of the algorithm in Theorem 33 has a dependence on the number near-optimal formulas of its input. Hence, we need to argue that when the algorithm in Theorem 20 makes recursive calls to the algorithm in Theorem 20 on functions g other than the original input f that g also has few near-optimal formulas. However, this is not a problem since it is easy to see that if a function g is computed by some gate in an optimal formula for f (as it must be if a recursive call is made to g), then the number of near-optimal formulas for g is at most the number of near-optimal formulas for f (since one can create a near optimal formula for f by taking the optimal formula for f that computes g at some gate and replacing the subformula at that gate with a near-optimal formula for g). ◀

Proof of Theorem 33. We provide the pseudocode of our DecompProblem algorithm in Algorithm 2, which we recommend the reader look at before proceeding.

5.1 Correctness of Algorithm 2

In this subsection we show that Algorithm 2 has the desired input/output behavior.

Fix some function f with n-inputs satisfying $\mathsf{L}(f) \geq 2$. Let $N = 2^n$.

Part 1: building *Candidates*. First, we will prove some loop invariants that will help us show that $Candidates$ and $PartialCandidates_{(i)}$ contain those functions we are interested in and do not contain many more things.

The following claim shows that the $x^\star$ described on Line 10 always exists and that the ?-values of partial functions in $PartialCandidates_{(i)}$ always have an easily computable structure.

▷ Claim 35. Before and after each iteration of the while loop, it is true that if $\gamma \in PartialCandidates_{(i)}$, then

- $\gamma(x) =? \iff x \geq i$ (interpreting i as a binary string in $\{0,1\}^n$ in the natural way),
- and consequently $|\gamma^{-1}(\{0,1\})| = i$.

Proof. Clearly the claim is satisfied before the first iteration of the while loop when $i = 0$ and $PartialCandidates_{(i)} = \{AllUnknown\}$.

Now, we must argue inductively. Suppose $1 \leq i \leq N$ and $\gamma' \in PartialCandidates_{(i)}$. Then, it follows that there is some $\gamma \in PartialCandidates_{(i-1)}$ and some $b \in \{0,1\}$ such that $\gamma' = \gamma_b$ where γ_b is as defined in the pseudocode. That is, γ_b is equal to γ except that the first ?-value (which occurs at $x^\star_{old} = i - 1$ by the inductive hypothesis) is replaced by a b. Thus, we have

$$\gamma'(x) =? \iff \gamma(x) =? \wedge (x \neq x^\star_{old}) \iff x > x^\star_{old} \iff x \geq i$$

where the first equivalence comes from the definition of $\gamma_b = \gamma'$ and the second equivalence comes from the fact that $x^\star_{old} = i - 1$. ◁

Algorithm 2 A deterministic search to decision reduction for MFSP whose run time depends on the number of "near-optimal formulas".

procedure OPTIMALSUBCOMPUTATION(f)
▷ Given the length-N truth table of a function f that takes n-inputs with $\mathsf{L}(f) \geq 2$, this procedure returns an element $\{g, \nabla, h\}$ of $\mathsf{OptSubcomps}(f)$.

Part 1: Building a *Candidates* list
  Let $allUnknown : \{0,1\}^n \to \{0,1,?\}$ be given by $allUnknown(x) =?$ for all x.
  Set $PartialCandidates_{(0)} = \{allUnknown\}$.
  Set $i = 0$.
  while $i < N$ **do**
    Set $PartialCandidates_{(i+1)} = \emptyset$.
    for all $\gamma \in PartialCandidates_{(i)}$ and for all $b \in \{0,1\}$ **do**
      Let $x^\star$ be the lexicographically first input satisfying $\gamma(x^\star) =?$.
      Let $\gamma_b : \{0,1\}^n \to \{0,1,?\}$ be given by $\gamma_b(x) = \begin{cases} b & , \text{ if } x = x^\star \\ \gamma(x) & , \text{ otherwise.} \end{cases}$
      Let g_{γ_b} be the (total) function given by $g_{\gamma_b}(x) = \begin{cases} 1 & , \text{ if } \gamma_b(x) =? \\ \gamma_b(x) & , \text{ otherwise.} \end{cases}$
      if $\mathsf{L}(\mathsf{Select}[f, g_{\gamma_b}]) \leq \mathsf{L}(f) + n + 2$ **then**
        Add γ_b to $PartialCandidates_{(i+1)}$.
      end if
    end for
    Set $i = i + 1$.
  end while
  Set $Candidates = PartialCandidates_{(N)}$.

Part 2: Finding an optimal pair within *Candidates*
  for all pairs $g, h \in Candidates$ and for all gates $\nabla \in \{\wedge, \vee\}$ **do**
    if $\mathsf{L}(g) + \mathsf{L}(h) = \mathsf{L}(f)$ **and** $f = g \nabla h$ **then**
      return $\{g, \nabla, h\}$.
    end if
  end for
end procedure

Next, we show that the $PartialCandidates_{(i)}$ never contains "redundant" partial functions.

▷ Claim 36. Before and after each iteration of the while loop, it is true that if γ' and γ'' are distinct elements of $PartialCandidates_{(i)}$, then no total function agrees with both γ' and γ''.

Proof. Before the first iteration of the while loop runs, $i = 0$ and $PartialCandidates_{(0)}$ only contains the single partial function $AllUnknown$, so the claim clearly holds.

Now we must show that the claim holds inductively. Assume $1 \leq i \leq N$. For contradiction, suppose there was some total function q that agrees with distinct elements μ and μ' from $PartialCandidates_{(i)}$. It follows that there exists some $b, b' \in \{0,1\}$ and some (possibly not distinct) $\gamma, \gamma' \in PartialCandidates_{(i-1)}$ such that $\mu = \gamma_b$ and $\mu' = \gamma'_{b'}$ (using the notation from the pseudocode where these functions γ_b and $\gamma'_{b'}$ are given by replacing the output of the first ?-valued input in γ or γ' respectively with a b-value or b'-value respectively). It follows that q must also agree with γ and γ'.

Either $\gamma \neq \gamma'$ or not. If $\gamma \neq \gamma'$, then q agrees with two distinct elements from $PartialCandidates_{(i-1)}$ which contradicts the inductive hypothesis.

Now suppose that $\gamma = \gamma'$. Then it must be that $b \neq b'$ (otherwise, $\mu = \mu'$ and we assumed they are distinct). But then, we have then γ and γ' have the same first ?-valued input $x^\star$, so

$$b = \mu(x^\star) = q(x^\star) = \mu'(x^\star) = b'$$

which contradicts that $b \neq b'$. ◁

Moreover, $PartialCandidates_{(i)}$ only contains partial functions that can be completed to total functions that pass a certain test.

▷ Claim 37. Before and after each iteration of the while loop, it is true that if $\gamma \in PartialCandidates_{(i)}$ then there exists a function g on n-inputs that agrees with γ such that

$$\mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + n + 2.$$

Proof. Before the first iteration of the while loop runs, $i = 0$ and $PartialCandidates_{(0)}$ only contains one partial function ($AllUnknown$). The function f clearly agrees with $AllUnknown$, and it is easy to see that $\mathsf{L}(\mathsf{Select}[f, f]) = \mathsf{L}(f) \leq \mathsf{L}(f) + n + 1$, as desired. Thus, the claim holds before the first iteration of the while loop.

Moreover, the claim clearly continues holding inductively because before any γ_b is added to $PartialCandidates_{(i)}$, we check to see if the function g_{γ_b} satisfies

$$\mathsf{L}(\mathsf{Select}[f, g_{\gamma_b}]) \leq \mathsf{L}(f) + n + 2$$

and g_{γ_b} agrees with γ_b by construction. ◁

Finally, we show that $PartialCandidates_{(i)}$ always contains the partial functions we want.

▷ Claim 38. Suppose some function q is in an optimal subcomputation for f. Then before and after each iteration of the while loop there is a $\gamma \in PartialCandidates_{(i)}$ such that q agrees with γ. Moreover, once part 1 is finished, $q \in Candidates$

Proof. Fix some q as in the statement of the claim.

Before the first iteration of the while loop runs, $i = 0$ and $PartialCandidates_{(0)}$ contains the all-? partial function $AllUnknown$, so q agrees with $AllUnknown$ and the claim holds.

Now, we must show the claim holds inductively. Assume $1 \leq i \leq N$. Then by induction there exists a $\gamma \in PartialCandidates_{(i-1)}$ such that q agrees with γ. Let $b = q(i-1)$. Then q agrees with γ_b as defined in the pseudocode (replacing the first ?-value in γ with a b-value) since Claim 35 implies that

$$\gamma_b(x) = \begin{cases} b & , \text{ if } x = i - 1 \\ \gamma(x) & , \text{ otherwise.} \end{cases}.$$

Thus, if we could show $\gamma_b \in PartialCandidates_{(i)}$, we would be done with showing the first part of the claim. From the pseudocode, it is clear $\gamma_b \in PartialCandidates_{(i)}$ if

$$\mathsf{L}(\mathsf{Select}[f, g_{\gamma_b}]) \leq \mathsf{L}(f) + n + 2,$$

where g_{γ_b} is as defined in the code (the function given by replacing the ?-values in γ_b with ones) which we now prove.

We already noted that

$$\gamma_b(x) = \begin{cases} b & , \text{ if } x = i-1 \\ \gamma(x) & , \text{ otherwise.} \end{cases}.$$

Thus, appealing to Claim 35, we know that $\gamma_b(x) =? \iff x > x^\star$ where $x^\star \in \{0,1\}^n$ is the binary string equivalent to $i-1$ (note that $0 \leq i-1 \leq N-1$ so this makes sense). Hence, since q agrees with γ_b, we have that $g_{\gamma_b}(x) = q(x) \vee GrtrThan_{x^\star}(x)$ where $GrtrThan_{x^\star}(x) = 1$ if and only if $x > x^\star$.

Thus, we have that

$$\begin{aligned} \mathsf{Select}[f, g_{\gamma_b}](x,z) &= \begin{cases} f(x) & \text{,if } z = 0 \\ g_{\gamma_b}(x) & \text{,if } z = 1 \end{cases} \\ &= \mathsf{Select}[f,q](x,z) \vee (z \wedge GrtrThan_{x^\star}(x)) \end{aligned}$$

Since $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$, we know that $\mathsf{L}(\mathsf{Select}[f,q]) = \mathsf{L}(f) + 1$ by Lemma 23, and Proposition 15 implies that $\mathsf{L}(GrtrThan_{x^\star}) \leq n$. Hence, we have that

$$\mathsf{L}(\mathsf{Select}[f, g_{\gamma_b}]) \leq \mathsf{L}(f) + n + 2.$$

Finally, we show that $q \in Candidates$ after part 1 finishes. Clearly, it suffices to show that $q \in PartialCandidates_{(N)}$ after part 1 finishes. We have already shown that there is a $\gamma \in PartialCandidates_{(N)}$ such that γ agrees with q. However, Claim 35 implies that γ is a total function and hence it equals q, so $q \in PartialCandidates_{(N)}$. ◁

Part 2: Finding a g, h pair within $Candidates$. First, we note that any output by Algorithm 2 must be correct.

▷ Claim 39. Any value Algorithm 2 outputs must be an element of $\mathsf{OptSubcomps}(f)$.

Proof. Any output $\{g, \triangledown, h\}$ of Algorithm 2 must satisfy $f = g \triangledown h$ and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ which implies $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$ by Lemma 21. ◁

Finally, we show that Algorithm 2 must output a value.

▷ Claim 40. Algorithm 2 must output a value (on input f).

Proof. Since $\mathsf{L}(f) \geq 2$, we have that $\mathsf{OptSubcomps}(f)$ is non-empty. Let $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$.

Claim 38 implies that $\{g, h\} \subseteq Candidates$. On the other hand, Lemma 21 implies that $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ and $f = g \triangledown h$. Thus, it is clear that part 2 will either output $\{g, \triangledown, h\}$ or output a value before that. ◁

5.2 Running Time of Algorithm 2

Fix some function f with n-inputs satisfying $\mathsf{L}(f) \geq 2$. Let $N = 2^n$. We break the running time analysis into the two pieces of the algorithm.

Part 1. It is easy to see that the run time of part 1 can be bounded by

$$O(N + \sum_{i \in [N]} N \cdot |PartialCandidates_{(i)}|)$$

where $|PartialCandidates_{(i)}|$ indicates the size of $PartialCandidates_{(i)}$ after Algorithm 2 is finished adding elements to it.

Moreover, we can bound the quantity $|PartialCandidates_{(i)}|$ as follows. Claim 37 implies that every partial function in $PartialCandidates_{(i)}$ must be consistent with some total function g on n-inputs satisfying

$$\mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + n + 2.$$

On the other hand, Claim 36 implies that any single (total) function can agree with at most partial function in $PartialCandidates_{(i)}$. Hence, we have that

$$|PartialCandidates_{(i)}| \leq |\{g : \mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + n + 2\}|$$

and Lemma 24 implies that

$$|\{g : \mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + n + 2\}| \leq O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)| \cdot N^2 \log N).$$

Thus, we have that part 1 runs in time at most $O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)| \cdot N^4 \log N)$. Moreover, part 1 only makes oracle calls of length at most $2N$ (to calculate $\mathsf{L}(\mathsf{Select}[f, g_{\gamma_b}])$).

Part 2. It is easy to see that this part runs in time $O(N \cdot |Candidates|^2)$. Hence, since $Candidates = PartialCandidates_{(N)}$, the analysis in part 1 above implies that

$$|Candidates| \leq O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)| \cdot N^2 \log N).$$

Thus, part 2 runs in time at most

$$O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)|^2 \cdot N^5 \log^2 N).$$

Moreover, part 2 only makes oracle calls of length N.

In total. Putting it all together, we have that Algorithm 2 runs in time at most

$$O(|\mathsf{CanonOpt}_{n+1}\mathsf{Formulas}(f)|^2 \cdot N^5 \log^2 N)$$

and only makes oracle queries of length $2N$. ◀

6 A worst-case randomized reduction

We now present a worst-case search to decision reduction for MFSP.

▶ **Theorem 41.** *There is a randomized algorithm solving* Search-MFSP *on inputs of length N in time $O(2^{.67N})$ given access to an oracle that solves* MFSP *on instances of length $2N$.*

Proof. We prove this theorem by giving an oracle algorithm solving DecompProblem and appealing to Theorem 20. We provide the pseudocode of our algorithm in Algorithm 3, which we recommend the reader look at before proceeding.

Algorithm 3 A randomized worst-case search to decision reduction for MFSP.

procedure WorstCaseOptimalSubcomputation(f)
▷ Given the length-N truth table of a function f that takes n-inputs with $\mathsf{L}(f) \geq 2$, this procedure returns an element $\{g, \triangledown, h\}$ of $\mathsf{OptSubcomps}(f)$.

Set $s = \frac{2}{3} \cdot \frac{2^n}{\log n}$
Set $t = 2^{2N/3}$

Part 1: Try random formulas
for $i = 1, \ldots, t$ **do**
Let G_i be a uniformly random binary tree with $\mathsf{L}(f)$-leaves. (Section 6.2 discusses how to sample G_i.)
Turn G_i into a uniformly random formula φ_i by picking uniformly random gates from $\{\wedge, \vee\}$ and uniformly random input leaves from $\{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$.
if φ_i computes f **then**
Write $\varphi_i = \varphi_{i,1} \triangledown \varphi_{i,2}$.
Let g and h be the function computed by $\varphi_{i,1}$ and $\varphi_{i,2}$ respectively.
if $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ **then**
return $\{g, \triangledown, h\}$.
end if
end if
end for

Part 2: Generate a small list of candidates for g
Set $SmallFuncs = \{g : g \text{ is a Boolean function with } n\text{-inputs and } \mathsf{L}(g) \leq s\}$.
Set $Candidates = \{g \in SmallFuncs : \mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + 1\}$.

Part 3: Try to find a g, h pair within Candidates
for each pair of functions $(g, h) \in Candidates$ and for each gate $\triangledown \in \{\wedge, \vee\}$ **do**
if $f = g \triangledown h$ and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ **then**
return $\{g, \triangledown, h\}$.
end if
end for

Part 4: Try to find a g, h pair by looking at functions h satisfying $f = g \triangledown h$
Set $SmallCandidates = \{g \in Candidates : \mathsf{L}(g) \leq \mathsf{L}(f) - s\}$.
for each function $g \in SmallCandidates$ and for each $\triangledown \in \{\wedge, \vee\}$ **do**
if $\forall x \in \{0,1\}^n$ $\exists b \in \{0,1\}$ such that $g(x) \triangledown b = f(x)$ **then**
Let $h_{?,g} : \{0,1\}^n \to \{0, 1, ?\}$ be the unique partial function on n-inputs such that $\forall\ h$, $f = g \triangledown h \iff h$ agrees with $h_{?,g}$.
for each total function h that agrees with $h_{?,g}$ **do**
if $f = g \triangledown h$ and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ **then**
return $\{g, \triangledown, h\}$.
end if
end for
end if
end for
end procedure

6.1 Correctness of Algorithm 3

In this section, we prove that Algorithm 3 has the desired input/output behavior. In our analysis, we will use s and t as parameters which we will set to the optimal values (which are written in the pseudocode) in Section 6.2 where we do the running time analysis for Algorithm 3.

Fix some function f on n-inputs with $\mathsf{L}(f) \geq 2$. We analyze the algorithm in parts.

Part 1. Since φ_i is chosen to have $\mathsf{L}(f)$ leaves and the algorithm in part 1 checks if φ_i computes f before returning any value, the following claim is clear.

▷ Claim 42. Any output by Algorithm 3 returned in part 1 must be an element of $\mathsf{OptSubcomps}(f)$.

Moreover, we can lower bound the probability that Algorithm 3 returns a value in part 1 as follows. Recall that $\mathsf{CanonOpt}_0\mathsf{Formulas}(f)$ is the set of optimal canonical formulas for f. We will show that part 1 succeeds if this set is large.

▷ Claim 43. If $t \geq 5 \cdot \frac{2^{N(1+o(1))}}{|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|}$, then part 1 of Algorithm 3 will return a value at least 99% of time.

Proof. Since we are picking each s-leaf formula φ_i uniformly at random, the probability that any fixed formula computes f is at least

$$\frac{|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|}{\text{the total number of formulas with } \mathsf{L}(f) \text{ leaves}}$$

Combining Theorem 13 with Proposition 12 upper bounds the denominator by $2^{N(1+o(1))}$, so

$$\Pr[\varphi_i \text{ computes } f] \geq \frac{|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|}{2^{N(1+o(1))}}.$$

Since each of these φ_i are chosen independently, we have that

$$\begin{aligned}\Pr[\exists i \in [t] \text{ such that } \varphi_i \text{ computes } f] &\geq 1 - (1 - \frac{|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|}{2^{N(1+o(1))}})^t \\ &\geq 1 - e^{-t \cdot \frac{|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|}{2^{N(1+o(1))}}} \\ &\geq 1 - e^{-5} \\ &\geq .99\end{aligned}$$

Hence, with probability at least 99%, part 1 will find a φ_i computing f at which point it will clearly return a value. ◁

Part 2. In part 2, Algorithm 3 constructs the *Candidates* set. We prove two claims about this set. First, that it contains the functions we care about, and second that its size can be bounded using the size of the $\mathsf{CanonOpt}_0\mathsf{Formulas}(f)$ set.

▷ Claim 44. Suppose g is in an optimal subcomputation for f. Then $\mathsf{L}(g) \leq s \implies g \in$ *Candidates*.

Proof. Since $\mathsf{L}(g) \leq s$, we know that $g \in \mathit{SmallFuncs}$. Next, since g is in an optimal subcomputation for f, we have by Lemma 23 that

$$\mathsf{L}(\mathsf{Select}[f, g]) \leq \mathsf{L}(f) + 1,$$

so g is an element of *Candidates*. ◁

▷ Claim 45.

$$|Candidates| = O(|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)| \cdot N \log N)$$

Proof. By construction, we have that

$$Candidates \subseteq \{g : \mathsf{L}(\mathsf{Select}[f, g]) = \mathsf{L}(f) + 1\}.$$

On the other hand, Lemma 24, we have that

$$|\{g : \mathsf{L}(\mathsf{Select}[f, g]) = \mathsf{L}(f) + 1\}| \leq O(|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)| \cdot N \log N)$$ ◁

Part 3. In Part 3, Algorithm 3 tries to find a g, h pair by looking within the $Candidates$ set. We show this works as long as there is a $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$ where $\mathsf{L}(g)$ and $\mathsf{L}(h)$ are small.

First, we note that part 3 can only return correct answers.

▷ Claim 46. Any output returned by Algorithm 3 in part 3 will be an element of $\mathsf{OptSubcomps}(f)$.

Proof. In order for a $\{g, \triangledown, h\}$ value to be returned in part 3 it must satisfy $f = g \triangledown h$ and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$. Thus, by Lemma 21, we know $\{g, \triangledown, h\} \in \mathsf{OptSubcomps}(f)$. ◁

Next, we give sufficient conditions on which part 3 will return an answer.

▷ Claim 47. If there exists an element $\{g', \triangledown, h'\}$ of $\mathsf{OptSubcomps}(f)$ such that

$$\max\{\mathsf{L}(g'), \mathsf{L}(h')\} \leq s,$$

then Algorithm 3 will return a value in part 3 or before.

Proof. Suppose there exists an element $\{g', \triangledown, h'\}$ of $\mathsf{OptSubcomps}(f)$ such that

$$\max\{\mathsf{L}(g'), \mathsf{L}(h')\} \leq s,$$

and assume that this procedure has not returned a value before it reaches part 3. Then by Claim 44, we have that both g' and h' are in $Candidates$. Moreover, since $\{g', \triangledown, h'\} \in \mathsf{OptSubcomps}(f)$, we know that $f = g' \triangledown' h'$ and $\mathsf{L}(f) = \mathsf{L}(g) + \mathsf{L}(h)$ by Lemma 21. Hence it is clear there are value the for loop will return an output if it reaches $g = g', h = h'$, and $\triangledown = \triangledown'$ (although it could return a value before that). ◁

Part 4. In the final part of Algorithm 3, we look for matching h functions for g candidates with small complexity.

First, we note that any output returned by part 4 must be correct by essentially the same proof as Claim 46 in part 3.

▷ Claim 48. Any output returned by Algorithm 3 in part 4 will be an element of $\mathsf{OptSubcomps}(f)$.

Next, we show sufficient conditions for part 4 returning an answer.

▷ Claim 49. If $s \geq \mathsf{L}(f)/2$ and there exists a $\{g', \triangledown, h'\} \in \mathsf{OptSubcomps}(f)$ such that $\mathsf{L}(h') \geq s$, then Algorithm 3 will return a value in part 4 or earlier.

Proof. Using Lemma 21, we have $\mathsf{L}(f) = \mathsf{L}(g') + \mathsf{L}(h')$. Thus, since $s \geq \mathsf{L}(f)/2$ and $\mathsf{L}(h') \geq s$, we have that

$$\mathsf{L}(g') = \mathsf{L}(f) - \mathsf{L}(h') \leq \mathsf{L}(f) - s \leq 2s - s = s.$$

Hence, by Claim 44, we have that $g' \in \mathit{Candidates}$. Moreover, since $\mathsf{L}(g') \leq \mathsf{L}(f) - s$, it follows that $g' \in \mathit{SmallCandidates}$. Thus, it is clear that part 4 will return a value if its for loop ever reaches $g = g', \nabla = \nabla'$, and $h = h'$ (though it could return a value before that). ◁

In total. Finally, we can prove the correctness of the input/output behavior of Algorithm 3.

▷ Claim 50. If $s \geq \mathsf{L}(f)/2$, then Algorithm 3 (run on input f) returns an element of $\mathsf{OptSubcomps}(f)$.

Proof. Put together, Claim 42, Claim 46, and Claim 48 ensures that any output returned by Algorithm 3 must be an element of $\mathsf{OptSubcomps}(f)$.

Hence, it suffices to show that Algorithm 3 will always output a value. We divide into two cases. Either there exists an element $\{g', \nabla, h'\} \in \mathsf{OptSubcomps}(f)$ such that $\max\{\mathsf{L}(g'), \mathsf{L}(h')\} \leq s$ or not.

If there exists such an element, then Claim 47 ensures that Algorithm 3 will output a value.

Now suppose that for all $\{g', \nabla, h'\} \in \mathsf{OptSubcomps}(f)$ we have $\max\{\mathsf{L}(g'), \mathsf{L}(h')\} \geq s$. Since $\mathsf{L}(f) \geq 2$, we know $\mathsf{OptSubcomps}(f)$ is non-empty by Proposition 18. Hence we can fix some $\{g', \nabla, h'\} \in \mathsf{OptSubcomps}(f)$ satisfying $\max\{\mathsf{L}(g'), \mathsf{L}(h')\} \geq s$. Without loss of generality we can assume that $\mathsf{L}(g') \leq \mathsf{L}(h')$. Thus, we have that $\mathsf{L}(h') \geq s$, and by hypothesis $s \geq \mathsf{L}(f)/2$, so Claim 49 ensures Algorithm 3 outputs a value. ◁

Thus Algorithm 3 is correct as long as $s \geq \mathsf{L}(f)/2$. Indeed, in the next section we will set s so that $s \geq \max\{\mathsf{L}(f)/2 : \ f \text{ takes } n\text{-inputs}\}$.

6.2 Runtime of Algorithm 3

In this subsection, we bound the runtime of Algorithm 3 and set s and t to the optimal values. We analyze Algorithm 3 in its parts.

Part 1. The for loop in part 1 clearly runs t times, so we just need to bound the running time of each iteration. Generating a uniformly random binary with $\mathsf{L}(f)$-leaves can be done in linear time (see [15] for a survey of various approaches). The other operations in the for loop can clearly be done in time $O(N + \mathsf{L}(f))$. Hence, all of part 1 runs in time $O(t \cdot (N + \mathsf{L}(f))$ which is $O(t \cdot N)$ using the worst-case formula upper bound from Theorem 13.

Moreover, part 1 only makes oracle calls of length N (to calculate $\mathsf{L}(f)$).

Part 2. Building the $\mathit{SmallFuncs}$ set requires iterating through all formulas of size s (which is bounded by $2^{(1+o(1))\cdot s \log n}$ using Proposition 12) and then computing the truth table of each of these size s formulas (which can be done in time $O(Ns)$). Hence, computing $\mathit{SmallFuncs}$ can be done in $O(N \cdot 2^{(1+o(1))\cdot s \log n})$ time. Moreover, $|\mathit{SmallFuncs}|$ is clearly upper bounded by the upper bound on the number of formulas of size s: $2^{(1+o(1))\cdot s \log n}$.

Next, building the $\mathit{Candidates}$ set can be done in time $O(|\mathit{SmallFuncs}| + N) = O(N \cdot 2^{(1+o(1))\cdot s \log n})$, and we use oracle calls of length $2N$ in this step (for $\mathsf{Select}[f, g]$).

Hence, part 2 runs in time $O(N \cdot 2^{(1+o(1))\cdot s \log n})$.

We will make use of the following claim later, which bounds the size of the $Candidates$ set if part 1 did not return a value.

▷ Claim 51. Fix some function f. Then with 99% probability (over the algorithm's choice of random formulas) either Algorithm 3 on input f returns before reaching part two or

$$|Candidates| \leq \frac{2^{N(1+o(1))}}{t}.$$

Proof. Suppose that *Algorithm* 3 reaches part two on input f and

$$|Candidates| > \frac{2^{N(1+o(1))}}{t}.$$

Then Claim 45 implies that

$$\frac{2^{N(1+o(1))}}{t} = O(|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)|N \log n)$$

which implies that

$$|\mathsf{CanonOpt}_0\mathsf{Formulas}(f)| \geq \frac{2^{N(1+o(1))}}{t}.$$

Hence, Claim 43 implies that Algorithm 3 will return in part 1 with 99% probability. ◁

Part 3. It is easy to see that part 3 runs in time $O(|Candidates|^2 + N)$ and makes oracle calls of length N.

Thus, using Claim 51, we have that with 99% probability part 3 runs in time $\frac{2^{2N(1+o(1))}}{t^2}$.

Part 4. Computing $SmallCandidates$ can be done in $O(|Candidates| + N)$ time, and the outer for loop runs at most $O(|Candidates|)$ many times.

It remains to bound the running time of each iteration of the outer for loop. The if condition can be checked in $O(N)$ time. Constructing $h_{?,g}$ also takes $O(N)$ time (similar to the if condition, just iterate through each input $x \in \{0,1\}^n$ and see which values of $b \in \{0,1\}$ satisfy $f(x) = g(x) \nabla b$). Each iteration of the inner for loop takes $O(N)$ time. Finally, the inner for loop runs $2^{|h_{?,g}^{-1}(?)|}$ many times.

Thus, the total running time for part 4 is

$$O(|Candidates| \cdot (N + N \cdot 2^{\max_g\{|h_{?,g}^{-1}(?)|\}}))$$

time.

Moreover, we can bound the quantity $\max_g\{|h_{?,g}^{-1}(?)|\}$ as follows.

▷ Claim 52. $\max_g\{|h_{?,g}^{-1}(?)|\} \leq (1 + o(1))(N - s \log n)$.

Proof. Since any function h that agrees with $h_{?,g}$ satisfies $g \nabla h = f$ and, we have that

$$\mathsf{L}(f) \leq \mathsf{L}(g) + \mathsf{L}(h_{?,g}).$$

Since $\mathsf{L}(g) \leq \mathsf{L}(f) - s$ (since $g \in SmallCandidates$), we have that

$$\mathsf{L}(h_{?,g}) \geq s.$$

On the other hand, the upper bound on the formula complexity of partial functions from Theorem 16 implies that

$$\mathsf{L}(h_{?,g}) \leq (1+o(1))(1-\frac{|h_{?,g}^{-1}(?)|}{N})\frac{N}{\log n}.$$

Hence

$$s \leq (1+o(1))(1-\frac{|h_{?,g}^{-1}(?)|}{N})\frac{N}{\log n}$$

so

$$|h_{?,g}^{-1}(?)| \leq N - \frac{s\log n}{(1+o(1))} \leq (1+o(1))(N - s\log n). \qquad \lhd$$

Thus, using Claim 51, we have that with 99% probability part 4 runs in $\frac{2^{2N(1+o(1))}}{t^2}$.

$$O(\frac{2^{N(1+o(1))}}{t}\cdot N\cdot 2^{(1+o(1))(N-s\log n)}) = \frac{2^{(1+o(1))(2N-s\log n)}}{t}$$

time.

In total. Thus, we get that with 99% probability Algorithm 3 runs in time

$$O(t\cdot N) + O(N\cdot 2^{(1+o(1))\cdot s\log n}) + \frac{2^{2N(1+o(1))}}{t^2} + \frac{2^{(1+o(1))(2N-s\log n)}}{t}.$$

Letting $s = \frac{2}{3}\frac{2^n}{\log n}$ and $t = 2^{\frac{2}{3}N}$, we get that the running time is bounded by

$$2^{(1+o(1))\frac{2}{3}N}.$$

Moreover, s will satisfy $s \geq \mathsf{L}(f)/2$ (as required for the correctness of the algorithm) for all f with n-inputs when n is sufficiently large by Theorem 13. ◀

7 A "bottom-up" reduction for DeMorgan Formulas

In this section, we provide another algorithm for solving Search-MFSP that is also efficient on average, though with worse guarantees than the one given by Theorem 33. Despite its worse guarantees, we present the algorithm because it uses a different "bottom-up" approach that we think is interesting.

We begin by proving a lemma that bounds the depth of optimal DeMorgan formulas.

▶ **Lemma 53** (Large optimal DeMorgan formulas have not too large depth). *Let $f : \{0,1\}^n \to \{0,1\}$. Let φ be an optimal DeMorgan formula for computing f. Then the depth of f is at most $\frac{10}{n}\cdot 2^n$ for sufficiently large n.*

Proof. Let d be a parameter we set later. For contradiction, suppose that φ is an optimal formula for computing f with depth greater than d. Clearly then $\mathsf{L}(f) > d$ as well. For φ to have depth greater than d, there must be gates $\nabla_1, \ldots, \nabla_{d-1} \in \{\wedge, \vee\}$ and subformulas $\varphi_1, \ldots, \varphi_d$ such that

$$\varphi = \varphi_1 \ \nabla_1 \ \varphi_2 \ \nabla_2 \ \ldots \ \nabla_{d-1} \ \varphi_d$$

where we evaluate gates from left to right, so this formula with parentheses would be

$$(\dots((\varphi_1 \bigtriangledown_1 \varphi_2) \bigtriangledown_2 \varphi_3)\dots) \bigtriangledown_{d-1} \varphi_d,$$

and $|\varphi_i| \geq 1$ for all $i \in [d]$. In other words, consider a d-length path from some subformula φ_1 to the output gate g_{d-1} in the formula and let $\varphi_2, \dots, \varphi_d$ be all the subformulas in φ from bottom to top (viewing the output gate as the top) intersecting this path. Similarly, let $\bigtriangledown_1, \dots, \bigtriangledown_{d-1}$ be the gates in order from bottom to top along this path.

Then, we have that

$$\mathsf{L}(f) = |\varphi| \geq \sum_{i \in [d]} |\varphi_i|.$$

Thus, we have that

$$\mathbb{E}_{i \in [d] \setminus \{1\}}[|\varphi_i|] \leq \frac{\mathsf{L}(f) - 1}{d - 1}.$$

Hence by Markov's inequality, we have that there exists a subset $S \subseteq [d] \setminus \{1\}$ of size at least $\frac{d-1}{2}$ such that for all $i \in S$ we have $|\varphi_i| \leq 2 \cdot \frac{\mathsf{L}(f)-1}{d-1}$.

On the other hand the number of distinct formulas on n-inputs with size at most $2 \cdot \frac{\mathsf{L}(f)-1}{d-1}$ is bounded by

$$2^{2 \cdot \frac{\mathsf{L}(f)-1}{d-1} \log n (1+o(1))}$$

according to Proposition 12. Assume that we have chosen d so that

$$|S| \geq \frac{d-1}{2} > 2^{\frac{\mathsf{L}(f)-1}{d-1} \log n (1+o(1))}.$$

Then, by the pigeonhole principle, there exists $i \leq j \in S$ such that φ_i and φ_j compute the same function. We can use this to get a contradiction to optimality as follows. Assume that $\bigtriangledown_{i-1} = \bigtriangledown_{j-1} = \wedge$ (the other cases are similar). Then, substituting in $\bigtriangledown_{i-1} = \bigtriangledown_{j-1} = \wedge$ we would have that the subformula of φ computed at $\bigtriangledown_{i-1}$, that is

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \dots \bigtriangledown_{j-2} \varphi_{j-1} \bigtriangledown_{j-1} \varphi_j \bigtriangledown_j \dots \bigtriangledown_{i-1} \varphi_i,$$

equals the function computed by

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \dots \bigtriangledown_{j-2} \varphi_{j-1} \wedge \varphi_j \bigtriangledown_j \dots \wedge \varphi_i$$

However if $\varphi_i(x) = 0$, then intuitively this formula outputs 0 no matter what happens on the to the "left" of φ_i in the formula. Thus, we might as well assume on the "left" that $\varphi_i(x) = \varphi_j(x) = 1$. Thus we get that the function computed at gate $\bigtriangledown_{i-1}$ is also computed by the following simplified formula:

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \dots \bigtriangledown_{j-2} \varphi_{j-1} \wedge 1 \bigtriangledown_j \dots \wedge \varphi_i.$$

which equals

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \dots \bigtriangledown_{j-2} \varphi_{j-1} \bigtriangledown_j \dots \wedge \varphi_i.$$

Thus, while the original subformula of φ computed at gate $\bigtriangledown_{i-1}$ given by

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \dots \varphi_{j-1} \bigtriangledown_{j-1} \varphi_j \dots \bigtriangledown_{i-1} \varphi_i$$

had size $\sum_{k\in[i]} |\varphi_k|$, the new equivalent formula given by

$$\varphi_1 \bigtriangledown_1 \varphi_2 \bigtriangledown_2 \cdots \varphi_{j-1} \bigtriangledown_j \varphi_{j+1} \cdots \wedge \varphi_i$$

has the smaller size $\sum_{k\in[i]} |\varphi_k| - |\varphi_j| < \sum_{k\in[i]} |\varphi_k|$ which contradicts the optimality of φ for f.

It remains to chose a value for d. We need to satisfy that

$$\frac{d-1}{2} > 2^{2\cdot\frac{\mathsf{L}(f)-1}{d-1}\log n(1+o(1))}.$$

By Theorem 13, we have that $\mathsf{L}(f) \leq (1+o(1))\frac{2^n}{\log n}$. So setting $d = \frac{10}{n}\cdot 2^n$, we get that

$$2^{2\cdot\frac{\mathsf{L}(f)-1}{d-1}\log n(1+o(1))} \leq 2^{2n(1/10+o(1))} \leq d = \frac{10}{n}\cdot 2^n$$ ◀

Using this lemma, we prove a "bottom-up" search to decision reduction for Search-MFSP.

▶ **Theorem 54.** *There is a deterministic "bottom-up" algorithm solving* Search-MFSP *on inputs of length N given access to an oracle that solves* MFSP *on instances of length $2N$ that runs in time $O(N^3 \cdot |\mathsf{CanonOpt}_{(\frac{10}{n}\cdot 2^n)}\mathsf{Formulas}(f)|^2)$ where f is the input truth table of length N.*

Algorithm 4 A bottom up search to decision reduction.

```
1:  procedure OptimalFormula(f)
    ▷ Given the length-N truth table of a function f that takes n-inputs, this procedure
    finds an optimal formula computing f
2:      Set Candidates_(1) = ∅.
3:      Let OptForm be a empty lookup table.
4:      for each size one formula φ on n-inputs do
5:          Let q be the function computed by φ.
6:          Add q to Candidates_(1).
7:          Let OptForm(q) = φ.
8:      end for
9:      Set s = 1.
10:     while s < L(f) do
11:         Set Candidates_(s+1) ← ∅.
12:         for every pair g, h in Candidates and every gate ▽ ∈ {∧, ∨} do
13:             Let q be the function computed by g▽h.
14:             if L(q) = L(g) + L(h) and L(Select[f, q]) ≤ L(f) + (10/n)·2^n then
15:                 Add q to Candidates_(s+1).
16:                 Set OptForm(q) to the formula given by OptForm(g)▽OptForm(h).
17:             end if
18:         end for
19:         Set s = s + 1.
20:     end while
21:     return OptForm(f).
22: end procedure
```

Proof. The pseudocode for our reduction is presented in Algorithm 4.

Since this algorithm is weaker than the one presented in Theorem 33, we only sketch the main observation needed to see that the "test" implicit in Algorithm 4 that

$$\mathsf{L}(\mathsf{Select}[f,q]) - \mathsf{L}(f) \leq d \leq \frac{10}{n} \cdot 2^n$$

is passed by any function q that is computed by some gate in an optimal formula. The bound on the total number of functions that pass this test is given by Lemma 24.

Fix a function f on n-inputs and set $N = 2^n$. The correctness of this algorithm follows from showing that if φ is an optimal formula for f and q is an n-input function computed by the ith gate node in φ, then

$$\mathsf{L}(\mathsf{Select}[f,q]) - \mathsf{L}(f) \leq d$$

where d is the depth of φ. If there were the case, then

$$\mathsf{L}(\mathsf{Select}[f,q]) - \mathsf{L}(f) \leq \frac{10}{n} \cdot 2^n$$

using the depth bound on optimal DeMorgan formulas from Lemma 53.

We now show that

$$\mathsf{L}(\mathsf{Select}[f,q]) - \mathsf{L}(f) \leq d$$

by producing a formula φ' for $\mathsf{L}(\mathsf{Select}[f,q])$ of size at most $\mathsf{L}(f) + d$.

Before, we give our formula construction of φ', we give an example of what our construction does that will hopefully be enough to convince the reader. To give an example,if $\varphi = x_1 \vee x_2 \wedge x_3$ and x_1 computes q, then $\varphi' = x_1 \vee (x_2 \wedge \neg z) \wedge (x_3 \vee z)$.

We formally construct φ' as follows. Recall we assumed that φ has depth d that q is the function computed by the ith gate in φ. Then, we can write

$$\varphi = \varphi_i \nabla_{i+1} \varphi_{i+1} \nabla_{i+2} \dots \nabla_k \varphi_k$$

(associating from left to right) where $k \leq d$ and $\varphi_i, \dots, \varphi_{k+1}$ are subformulas of φ and $\nabla_i, \dots, \nabla_k$ are the gates connecting those subformulas in φ and φ_i computes q.

We can then construct φ' by replacing each φ_j in φ for $i+1 \leq j \leq k$ with a new formula φ'_j given by

$$\varphi'_j = \begin{cases} \varphi_j \wedge \neg z, \text{ if } \nabla_j = \vee \\ \varphi_j \vee z, \text{ if } \nabla_j = \wedge \end{cases}.$$

Then

$$\varphi' = \varphi_i \nabla_{i+1} \varphi'_{i+1} \nabla_{i+2} \dots \nabla_k \varphi'_k$$

computes $\mathsf{Select}[f,q]$ because these φ'_j are chosen so that ∇_j will always just output its other input when $z = 1$.

Hence,

$$\mathsf{L}(\mathsf{Select}[f,q]) - \mathsf{L}(f) \leq d$$ ◀

References

1. Eric Allender. The new complexity landscape around circuit minimization. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications - 14th International Conference, LATA 2020, Milan, Italy, March 4-6, 2020, Proceedings*, volume 12038 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2020. doi:10.1007/978-3-030-40608-0_1.
2. Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.
3. Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017.
4. Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011.
5. David Buchfuhrer and Christopher Umans. The complexity of boolean formula minimization. *J. Comput. Syst. Sci.*, 77(1):142–153, 2011.
6. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Proceedings of the 31st Conference on Computational Complexity*, 2016.
7. Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. $\mathrm{Ac}^0[\mathrm{p}]$ lower bounds against MCSP via the coin problem. In *ICALP*, volume 132 of *LIPICS*, pages 66:1–66:15, 2019.
8. Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 247–258, 2018.
9. Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference, CCC*, volume 102, pages 5:1–5:31, 2018.
10. Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 73–79, 2000.
11. S. A. Lozhkin. Tighter bounds on the complexity of control systems from some classes. *Mat. Voprosy Kibernetiki 6*, pages 189–214 (in Russian), 1996.
12. Oleg B. Lupanov. Complexity of formula realization of functions of logical algebra. *Problemy Kibernetiki*, 3:61–80, 1960.
13. William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.
14. Cody D. Murray and R. Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017.
15. Erkki Mäkinen. Generating random binary trees — a survey. *Information Sciences*, 115(1):123–136, 1999.
16. Nicholas Pippenger. Information theory and the complexity of boolean functions. *Mathematical Systems Theory*, 10:129–167, January 1977.
17. Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.
18. Michael Rudow. Discrete logarithm and minimum circuit size. *Inf. Process. Lett.*, 128:1–4, 2017.
19. Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS*, volume 151 of *LIPIcs*, pages 68:1–68:26, 2020.
20. B. A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, October 1984.

Quantum Query-To-Communication Simulation Needs a Logarithmic Overhead

Sourav Chakraborty
Indian Statistical Institute, Kolkata, India
sourav@isical.ac.in

Arkadev Chattopadhyay
Tata Institute of Fundamental Research, Mumbai, India
arkadev.c@tifr.res.in

Nikhil S. Mande
Georgetown University, Washington, D.C., USA
nikhil.mande@georgetown.edu

Manaswi Paraashar
Indian Statistical Institute, Kolkata, India
manaswi.isi@gmail.com

Abstract

Buhrman, Cleve and Wigderson (STOC'98) observed that for every Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and $\bullet : \{-1,1\}^2 \to \{-1,1\}$ the two-party bounded-error quantum communication complexity of $(f \circ \bullet)$ is $O(Q(f) \log n)$, where $Q(f)$ is the bounded-error quantum query complexity of f. Note that the bounded-error randomized communication complexity of $(f \circ \bullet)$ is bounded by $O(R(f))$, where $R(f)$ denotes the bounded-error randomized query complexity of f. Thus, the BCW simulation has an extra $O(\log n)$ factor appearing that is absent in classical simulation. A natural question is if this factor can be avoided. Razborov (IZV MATH'03) showed that the bounded-error quantum communication complexity of Set-Disjointness is $\Omega(\sqrt{n})$. The BCW simulation yields an upper bound of $O(\sqrt{n} \log n)$. Høyer and de Wolf (STACS'02) showed that this can be reduced to $c^{\log^* n}$ for some constant c, and subsequently Aaronson and Ambainis (FOCS'03) showed that this factor can be made a constant. That is, the quantum communication complexity of the Set-Disjointness function (which is $\mathsf{NOR}_n \circ \wedge$) is $O(Q(\mathsf{NOR}_n))$.

Perhaps somewhat surprisingly, we show that when $\bullet = \oplus$, then the extra $\log n$ factor in the BCW simulation is unavoidable. In other words, we exhibit a total function $F : \{-1,1\}^n \to \{-1,1\}$ such that $Q^{cc}(F \circ \oplus) = \Theta(Q(F) \log n)$.

To the best of our knowledge, it was not even known prior to this work whether there existed a total function F and 2-bit function $\bullet$, such that $Q^{cc}(F \circ \bullet) = \omega(Q(F))$.

2012 ACM Subject Classification Theory of computation → Quantum query complexity; Theory of computation → Quantum communication complexity

Keywords and phrases Quantum query complexity, quantum communication complexity, approximate degree, approximate spectral norm

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.32

Related Version A full version of the paper is available at `https://arxiv.org/pdf/1909.10428.pdf`.

Acknowledgements We thank Ronald de Wolf and Srinivasan Arunachalam for various discussions on this problem. They provided us with a number of important pointers that were essential for our understanding of the problem and its importance. We thank Justin Thaler for referring us to Claim 31. We thank the anonymous CCC 2020 reviewers for invaluable comments. In particular, we thank one of the reviewers for pointing out the contrasts between the classical query model and the quantum query model that our results imply (Remark 5).

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 32; pp. 32:1–32:15

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

1 Introduction

Classical communication complexity, introduced by Yao [24], is aptly called the "swiss-army-knife" for understanding, especially the limitations of, classical computing. Quantum communication complexity holds the same promise with regards to quantum computing. Yet, there are many problems that remain open. One broad theme is to understand the fundamental differences between classical randomized and quantum protocols, especially for computing total functions.

Recall a standard way to derive a communication problem from a function $f : \{-1,1\}^n \to \{-1,1\}$. Each input bit of f is *encoded* between the two players Alice and Bob, using an instance of a binary primitive, denoted by $\bullet : \{-1,1\} \times \{-1,1\} \to \{-1,1\}$, giving rise to the communication problem of evaluating $f \circ \bullet$. Each input bit to f is obtained by evaluating $\bullet$ on the relevant bit of Alice and that of Bob, i.e. $\big(f \circ \bullet\big)\big(x,y\big) = f\big(\bullet(x_1,y_1),\ldots,\bullet(x_n,y_n)\big)$ and x, y are each n-bit strings given to Alice and Bob respectively. Many well known functions in communication complexity are derived in this way: Set-Disjointness is $\mathsf{NOR} \circ \wedge$, Inner-Product being $\mathsf{PARITY} \circ \wedge$, Equality being $\mathsf{NOR} \circ \oplus$.[1] Set-Disjointness is also a standard total function where quantum protocols provably yield a significant cost saving over their classical counterpart.

A natural and well studied question in this regard is what is the relationship between the query complexity of f and the communication problem of $f \circ \bullet$, when the $\bullet$ is $\wedge$ or $\oplus$. This question has been studied for particular interesting functions or special classes of functions. Classically, it is folklore that

$$R^{cc}\big(f \circ \bullet\big) = O(R(f)),$$

where $R(f)$ denotes the bounded-error randomized query complexity of f and $R^{cc}(f \circ \bullet)$ denotes the bounded-error randomized communication complexity for computing $f \circ \bullet$. In an influential work, Buhrman, Cleve and Wigderson [9] observed that a general and natural recipe exists for constructing a quantum communication protocol for $f \circ \bullet$, using a quantum query algorithm for f as a black-box.

▶ **Theorem 1** ([9]). *For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, we have*

$$Q^{cc}\big(f \circ \bullet\big) = O\big(Q(f) \cdot \log n\big),$$

where $\bullet$ is either $\wedge$ or $\oplus$.

Here $Q(f)$ denotes the bounded-error quantum query complexity of f, and $Q^{cc}(f \circ \bullet)$ denotes the bounded-error quantum communication complexity for computing $f \circ \bullet$. We remark here that the BCW simulation works for any constant-sized primitive $\bullet : \{-1,1\}^k \times \{-1,1\}^k \to \{-1,1\}$, but the focus of this paper is on the case where $k = 1$. Thus in the quantum world one incurs a logarithmic factor in the natural BCW simulation while no such factor is needed in the randomized setting. The basic question that arises naturally and which we completely answer in this work, is the following: analogous to the classical model, can this multiplicative $\log n$ blow-up in the communication cost be always avoided by designing quantum communication protocols that more cleverly simulate quantum query algorithms?

[1] Here $\wedge$ and $\oplus$ are the AND function and the XOR functions on 2 bits, respectively.

A priori, it is not clear what the answer to this question ought to be. For certain special functions and some classes of functions, quantum protocols exist where the $\log n$ factor can be saved. Theorem 1 implies a communication upper bound of $O(\sqrt{n}\log n)$ for the Set-Disjointness function. Høyer and de Wolf [15] designed a quantum protocol for Set-Disjointness of cost $O(\sqrt{n}c^{\log^* n})$, speeding up the BCW simulation significantly. Later, Aaronson and Ambainis [1] gave a more clever protocol that only incurred a constant factor overhead from Grover's search using more involved ideas, matching an $\Omega(\sqrt{n})$ lower bound due to Razborov [20].

For partial functions, tightness of the BCW simulation is known in *some* settings. For example, consider the Deutsch-Jozsa (DJ) problem, where the input is an n-bit string with the promise that its Hamming weight is either 0 or $n/2$, and DJ outputs -1 if the Hamming weight is $n/2$, and 1 otherwise. DJ has quantum query complexity 1 whereas the *exact* quantum communication complexity of $(\mathsf{DJ} \circ \oplus)$ is $\log n$. Note that it is unclear whether the $\log n$ factor loss here is additive or multiplicative.[2] Montanaro, Nishimura and Raymond [19] exhibited a partial function for which the BCW simulation is tight (up to constants) in the *exact* and *non-deterministic* quantum settings. They also observed the existence of a total function for which the BCW simulation is tight (up to constants) in the *unbounded-error* setting.

As far as we know, there was no (partial or total) Boolean-valued function f known prior to our work for which the *bounded-error* quantum communication complexity of $f \circ \bullet$ (i.e. $Q^{cc}(f \circ \bullet)$) is even $\omega(Q(f))$, where $\bullet$ is either $\wedge$ or $\oplus$.

In this paper, we exhibit the first *total function* witnessing the tightness of the BCW simulation in arguably the most well-known quantum model, which is the bounded-error model.

▶ **Theorem 2.** *There exists a total function $F : \{-1,1\}^n \to \{-1,1\}$ for which,*

$$Q^{cc}(F \circ \oplus) = \Theta(Q(F)\log n). \tag{1}$$

The statement above does not necessarily guarantee that a function exists that both satisfies Equation 1 and has bounded-error quantum query complexity (as a function of n) arbitrarily close to n. We answer this question by proving a more general result, from which Theorem 2 follows.

▶ **Theorem 3** (Main Theorem)**.** *For any constant $0 < \delta < 1$, there exists a total function $F : \{-1,1\}^n \to \{-1,1\}$ for which $Q(F) = \Theta(n^\delta)$ and*

$$Q^{cc}(F \circ \oplus) = \Theta(Q(F)\log n).$$

1.1 Overview of our approach and techniques

To demonstrate the tightness of the BCW simulation for a total function in the quantum bounded-error setting we have to find a function F such that $Q^{cc}(F \circ \bullet) = \Theta(Q(F)\log n)$ for some choice of $\bullet$ (that is, either $\bullet$ is $\wedge$ or $\oplus$). This requires us to prove an upper bound of $Q(F)$ and a lower bound on $Q^{cc}(F \circ \bullet)$. We consider the case when $\bullet$ is the $\oplus$ function.

[2] Indeed, there are well-known situations where complexity of 1 vs. $\log n$ can be deceptive. The classical private-coin randomized communication complexity of Equality is $\Theta(\log n)$, whereas the public-coin cost is well known to be $O(1)$. Newman's Theorem shows that this difference in costs, in general, is *not multiplicative* but merely *additive*.

For the inner function the $\oplus$ function is preferred over the $\wedge$ function for one crucial reason: we have an analytical technique for proving lower bounds on $Q^{cc}(F \circ \oplus)$, due to Lee and Shraibman [18]. They reduced the problem of lower bounding the bounded-error quantum communication complexity of $(F \circ \oplus)$ to proving lower bounds on an analytic property of F, called its *approximate spectral norm*. The ϵ-approximate spectral norm of F, denoted by $\|\hat{F}\|_{1,\epsilon}$, is defined to be the minimum ℓ_1-norm of the coefficients of a polynomial that approximates F uniformly to error ϵ (see Definition 18). Lee and Shraibman [18] showed that $Q^{cc}(F \circ \oplus) = \Omega(\log \|\widehat{F}\|_{1,1/3})$. Thus, the lower bound on Theorem 3 follows immediately from our result below.

▶ **Theorem 4.** *For any constant $0 < \delta < 1$, there exists a total function $F : \{-1,1\}^n \to \{-1,1\}$ for which $Q(F) = \Theta(n^\delta)$ and*

$$\log\left(\|\widehat{F}\|_{1,1/3}\right) = \Theta(Q(F)\log n).$$

▶ Remark 5. It is interesting to note that, in contrast, it is well known that the extra $\log n$ factor does not appear for classical randomized query complexity. That is, if $R(F)$ denotes the bounded-error randomized query complexity of F, then $\log(\|\widehat{F}\|_{1,1/3}) = O(R(F))$. This indicates a significant structural difference between the classical and quantum query models.

There are not many techniques known to bound the approximate spectral norm of a function. This sentiment was expressed both in [18] and in the work of Ada, Fawzi and Hatami [2]. On the other hand, classical approximation theory offers tools to prove bounds on a simpler and better known concept called *approximate degree* which has been invaluable, particularly for quantum query complexity. The ϵ-approximate degree of f, denoted by $\widetilde{\deg}_\epsilon(f)$, is the minimum degree required by a real polynomial to uniformly approximate f to error ϵ (see Definition 17). Recently, two of the authors [11] devised a way of lifting approximate degree bounds to approximate spectral norm bounds. We first show here that technique works a bit more generally, to yield the following: let $\mathsf{ADDR}_{m,t} : \{-1,1\}^m \to [t]$ be a (possibly partial) addressing function (see Definition 13). For any function $f : \{-1,1\}^n \to \{-1,1\}$, define the (partial) function $f^{\mathsf{ADDR}_{m,t}} : \{-1,1\}^{n\times t} \times \{-1,1\}^{n\times m} \to \{-1,1\}$ as follows (formally defined in Definition 15):

$$f^{\mathsf{ADDR}_{m,t}}(x,y) = f\left(x_{1,\mathsf{ADDR}_{m,t}(y_1)}, x_{2,\mathsf{ADDR}_{m,t}(y_2)}, \ldots, x_{n,\mathsf{ADDR}_{m,t}(y_n)}\right).$$

Our main result on lower bounding the spectral norm is stated below.

▶ **Lemma 6** (extending [11])**.** *Let $t > 1$ be any integer, $\mathsf{ADDR}_{m,t}$ be any (partial) addressing function and $f : \{-1,1\}^n \to \{-1,1\}$ be any function. Then,*

$$\log\left(\|\widehat{f^{\mathsf{ADDR}_{m,t}}}\|_{1,1/3}\right) = \Omega\left(\widetilde{\deg}(f)\log t\right).$$

The functions F constructed for the proof of Theorem 4 are completions of instances of $\mathsf{PARITY}^{\mathsf{ADDR}_{\ell,\ell}}$, and hence Lemma 6 yields lower bounds on the approximate spectral norm of F in terms of the approximate degree of PARITY (which is known to be maximal).

For the upper bound on $Q(F)$ we use two famous query algorithms - Grover's search [14] and the Bernstein-Vazirani algorithm [7]. The use of these algorithms for upper bounding $Q(F)$ is in the same taste as in the work of Ambainis and de Wolf [3] although their motivation was quite different than ours. Interestingly, Ambainis and de Wolf used their function to pin down the minimal *approximate degree* of a total Boolean function, all of whose input variables are influential.

1.2 Intuition behind the function construction

- From Theorem 1 it is known that for all Boolean functions f, $Q^{cc}(f \circ \oplus) \leq O(Q(f) \log n)$.
- In order to prove a matching lower bound, we construct a Boolean function F on n variables such that $\|\widehat{F}\|_{1,1/3} = 2^{\Omega(Q(F)\log n)}$ (Theorem 4). From Theorem 22, this shows that $Q^{cc}(F \circ \oplus) = \Omega(\log \|\widehat{F}\|_{1,1/3}) = \Omega(Q(F) \log n)$. We want to additionally ensure that $Q(F) = \Theta(n^\delta)$ for a given constant $0 < \delta < 1$. A formal definition of F is given in Figure 1, we attempt to provide an overview on how we arrived at this function below.
- Assume δ is a constant that is least $1/2$, else the argument follows along similar lines by ignoring suitably many input variables when defining the function. A natural first attempt is to try to construct a composed function of the form $F = f^{\mathsf{ADDR}}$, for some addressing function ADDR (see Definition 13) with $\Omega(n^{1-\delta})$ many target bits, for which $Q(f^{\mathsf{ADDR}}) = \Theta(\widetilde{\deg}(f))$. For the lower bound we use Lemma 6 to show that $\log \|\widehat{f^{\mathsf{ADDR}}}\|_{1,1/3} = \Omega(\widetilde{\deg}(f) \log(n^{1-\delta})) = \Omega(\widetilde{\deg}(f) \log n)$.
- Given the upper bound target, we are led to a natural choice of addressing function. We refer the reader to Definition 13 for the definition of an addressing function and the selector function of an addressing function. Let $\mathsf{HADD}_{n^{1-\delta}}$ be the $(n^{1-\delta}, n^{1-\delta})$-addressing function defined as follows. Fix an arbitrary order on the $n^{1-\delta}$-bit Hadamard codewords (see Definition 12) , say $w_1, \ldots, w_{n^{1-\delta}}$. Define the selector function of $\mathsf{HADD}_{n^{1-\delta}}$, which we denote g, by $g(w_i) = i$ for all $i \in [n^{1-\delta}]$, and $g(x) = \star$ for $x \neq w_i$ for any $i \in [n^{1-\delta}]$.
- For any function f on $n^\delta/2$ bits, the *partial* function $f^{\mathsf{HADD}_{n^{1-\delta}}}$ on n inputs has quantum query complexity $O(Q(f)+n^\delta/2)$, as we sketch in the next step. We select f appropriately such that this is $\Theta(Q(f))$. Finally, we define the *total* function $F = f^{\mathsf{HADD}_{n^{1-\delta}}}$ to be the completion of $f^{\mathsf{HADD}_{n^{1-\delta}}}$ that evaluates to -1 on the non-promise inputs of $f^{\mathsf{HADD}_{n^{1-\delta}}}$.
- We choose the outer function to be $f = \mathsf{PARITY}_{n^\delta/2}$ to ensure $Q(F) = \Theta(n^\delta)$. To prove the upper bound on $Q(F)$, we crucially use the Bernstein-Vazirani and Grover's search algorithms.
  - Run $n^\delta/2$ instances of the Bernstein-Vazirani algorithm [7], one on each block. This algorithm guarantees that if the address variables were all Hadamard codewords, then we would receive the correct indices of the target variables with probability 1, and just $n^\delta/2$ queries.
  - In the next step, we run Grover's search [14, 8] on two $n/2$-bit strings to test whether the output of the first step was correct. If it was correct, we succeed with probability 1, and proceed to query the $n^\delta/2$ selected target variables and output the parity of them. If it was not correct, Grover's search catches a discrepancy with probability at least $2/3$ and we output -1, succeeding with probability at least $2/3$ in this case.
  - The $n^\delta/2$ invocations of the Bernstein-Vazirani algorithm use a total of $n^\delta/2$ queries, Grover's search uses another $O(\sqrt{n})$ queries, and the final parity (if Grover's search outputs that the strings are equal) uses another $n^\delta/2$ queries, for a cumulative total of $O(n^\delta + \sqrt{n}) = O(n^\delta)$ queries (recall that we assume $\delta \geq 1/2$).

1.3 Other implications of our result

Zhang [26] showed that for all Boolean functions f, there must exist gadgets $\bullet_i$, each either $\wedge$ or $\vee$, such that $Q^{cc}(f(\bullet_1, \ldots, \bullet_n)) = \Omega(\text{poly}(Q(f)))$. For monotone f, they showed that either $Q^{cc}(f \circ \wedge) = \Omega(\text{poly}(Q(f)))$ or $Q^{cc}(f \circ \vee) = \Omega(\text{poly}(Q(f)))$. They also state that it is unclear how tight the BCW simulation is. Our result implies that there exists a function for which it is tight up to constants (on composition with $\oplus$).

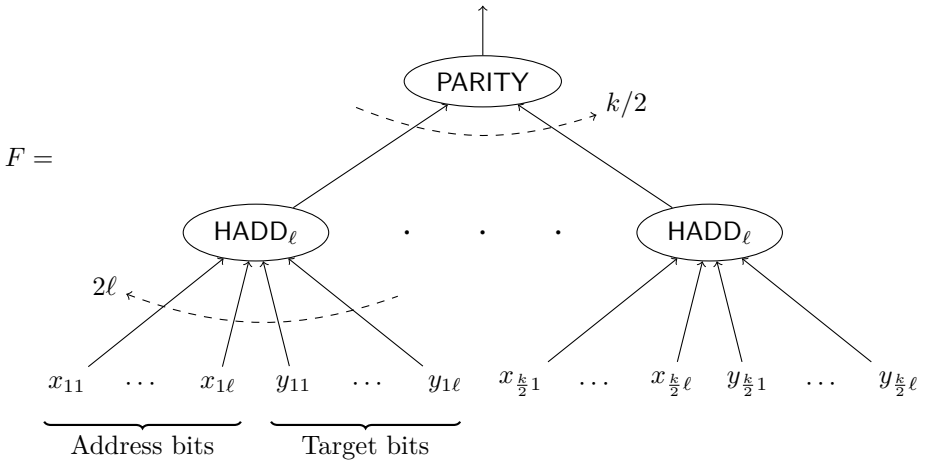


Figure 1 $k = n^\delta, \ell = n^{1-\delta}$. If the address bits of an input to the r'th HADD$_\ell$ is the j'th Hadamard codeword, then y_{rj} is selected. If on an input, there exists at least one HADD$_\ell$ for which the address bits do not correspond to a Hadamard codeword, F outputs -1. Else it outputs the parity of the $k/2$ selected target bits.

Another implication of our result is related to the Entropy Influence Conjecture, which is an interesting question in the field of analysis of Boolean functions, posed by Friedgut and Kalai [13]. This conjecture is open for general functions. A much weaker version of this conjecture is called the Min-Entropy Influence Conjecture. For the statement of the conjecture we need to consider the Fourier expansion of Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$ as

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S)\chi_S(x),$$

where $\{\chi_S : S \subseteq [n]\}$ are the *parity* functions ($\chi_S(x) = \Pi_{i \in S} x_i$, when $x = (x_1, \ldots, x_n) \in \{-1,1\}^n$) and $\{\widehat{f}(S) : S \subseteq [n]\}$ are the corresponding Fourier coefficients.

▶ **Conjecture 7** (Min-Entropy Influence Conjecture). *For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ there exists a non-zero Fourier coefficient $\widehat{f}(S)$ such that*

$$\log\left(1/|\widehat{f}(S)|\right) = O(I(f)),$$

where $I(f)$ denotes the influence (or average sensitivity) of f ($I(f) = \sum_{S \subseteq [n]} |S|\widehat{f}(S)^2$).

While this conjecture is also open, some attempts have been made to prove it and various implications of it [5, 16]. One interesting implication of the Min-Entropy Influence Conjecture that is still open is that the min-entropy of the Fourier spectrum (that is, $\log\left(1/\max_{S \subseteq [n]} |\widehat{f}(S)|\right)$) is less than $O(Q(f))$. In [5] using a primal-dual technique it was shown that the min-entropy of the Fourier spectrum is less than a constant times $\log(\|\hat{f}\|_{1,\epsilon})$, where the constant depends on ϵ. Thus if it were the case that $\log(\|\hat{f}\|_{1,\epsilon}) = O(Q(f))$, we would have upper bounded the min-entropy of Fourier spectrum by $O(Q(f))$. As $\widetilde{\deg}(f) \leq 2Q(f)$ [6], the following was stated in [5] as a possible approach and was left as an open problem.

▶ **Question 8** ([5, Section 4]). *Is it true that for all Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$,*

$$\log(\|\hat{f}\|_{1,\epsilon}) = O(\widetilde{\deg}(f))?$$

While their conjecture holds true for certain special classes of functions like the symmetric functions (proof given in Appendix A), our result in this paper nullifies this approach for general Boolean functions, since Theorem 4 yields the following along with the fact that $\widetilde{\deg}(f) \leq 2Q(f)$ [6].

▶ **Theorem 9.** *For any constant $0 < \delta < 1$, there exists a total function $F : \{-1,1\}^n \to \{-1,1\}$ for which $\widetilde{\deg}(F) = O(n^\delta)$ and*

$$\log \|\widehat{F}\|_{1,1/3} = \Omega(\widetilde{\deg}(F) \log n).$$

2 Preliminaries

For any positive integer n, we denote the set $\{1,\ldots,n\}$ by $[n]$. For $d \leq n$ we use the notation $\binom{n}{\leq d} := \binom{n}{0} + \cdots + \binom{n}{d}$. Note that $\binom{n}{\leq d} < (n+1)^d$.

In this section we review the necessary preliminaries. We first review some basic notions of Fourier analysis on the Boolean cube. Consider the vector space of functions from $\{-1,1\}^n$ to $\mathbb{R}$, equipped with an inner product defined by

$$\langle f, g\rangle := \mathbb{E}_{x\in\{-1,1\}^n}[f(x)g(x)] = \frac{1}{2^n}\sum_{x\in\{-1,1\}^n} f(x)g(x)$$

for every $f, g : \{-1,1\}^n \to \mathbb{R}$. For any set $S \subseteq [n]$, define the associated *parity* function χ_S by $\chi_S(x) = \prod_{i\in S} x_i$. The set of parity functions $\{\chi_S : S \subseteq [n]\}$, forms an orthonormal basis for this vector space. Thus, every function $f : \{-1,1\}^n \to \mathbb{R}$ has a unique multilinear expression as $f = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S$. The coefficients $\{\widehat{f}(S) : S \subseteq [n]\}$ are called the *Fourier coefficients* of f. We also use the notation PARITY_n to denote the function $\chi_{[n]} : \{-1,1\}^n \to \{-1,1\}$.

▶ **Fact 10** (Parseval's Identity). *For any function $f : \{-1,1\}^n \to \mathbb{R}$, we have $\sum_{S\subseteq[n]} \widehat{f}(S)^2 = \frac{\sum_{x\in\{-1,1\}^n} f(x)^2}{2^n}$.*

▶ **Definition 11** (Spectral Norm). *For any function $f : \{-1,1\}^n \to \mathbb{R}$, define its* spectral norm*, which we denote $\|\widehat{f}\|_1$, to be the sum of absolute values of the Fourier coefficients of f. That is, $\|\widehat{f}\|_1 := \sum_{S\subseteq[n]} |\widehat{f}(S)|$.*

▶ **Definition 12** (Hadamard Codeword). *If an ℓ-bit string $(x_1,\ldots,x_\ell) \in \{-1,1\}^\ell$ (alternatively, view the indices of x as subsets of $[\log \ell]$) is of the form $x_S = \prod_{i\in S} z_i$ for all $S \subseteq [\log \ell]$ for some $z \in \{-1,1\}^{\log \ell}$, then define such an $x = x_1 \ldots x_\ell$ to be the* ℓ-bit Hadamard codeword *$h(z)$ of the $(\log \ell)$-bit string z.*

2.1 Addressing functions

▶ **Definition 13** ((m,k)-addressing function). *We define a (partial) function $f : \{-1,1\}^{m+k} \to \{-1,1,\star\}$ to be an* (m,k)-addressing function *if there exists $g : \{-1,1\}^m \to \{[k] \cup \star\}$ such that*

- *$f(x_1,\ldots,x_m,y_1,\ldots,y_k) = y_{g(x_1,\ldots,x_m)}$ if $g(x_1,\ldots,x_m) \in [k]$, $f(x_1,\ldots,x_m,y_1,\ldots,y_k) = \star$ otherwise.*
- *For all $j \in [k]$, there exists $(x_1,\ldots,x_m) \in \{-1,1\}^m$ such that $g(x_1,\ldots,x_m) = j$.*

We call the variables $\{x_1,\ldots,x_m\}$ the address variables *and the variables $\{y_1,\ldots,y_k\}$ the* target variables*. The function g is called the* selector *function of f.*

▶ **Definition 14** (Indexing Function). *The Indexing function, which we denote by* IND_k*, is a* $(k, 2^k)$*-addressing function defined by* $\mathsf{IND}(x_1, \ldots, x_k, y_1, \ldots, y_{2^k}) = y_{\mathrm{bin}(x)}$*, where* $\mathrm{bin}(x)$ *denotes the integer represented by the binary string* $x_1, \ldots, x_k$*.*

▶ **Definition 15** (Composition with addressing functions). *For any function* $f : \{-1,1\}^n \to \{-1,1\}$ *and an* (m,k)*-addressing function* ADDR*, define the (partial) function* $f^{\mathsf{ADDR}} : \{-1,1\}^{n(m+k)} \to \{-1,1,\star\}$ *by* $f^{\mathsf{ADDR}}(x_1, y_1, \ldots, x_n, y_n) =$

$$\begin{cases} f(\mathsf{ADDR}(x_1,y_1), \ldots, \mathsf{ADDR}(x_n,y_n)) & \text{if } \forall i \in [n], \mathsf{ADDR}(x_i,y_i) \in \{-1,1\} \\ \star & \text{otherwise.} \end{cases}$$

where $x_i \in \{-1,1\}^m$ *and* $y_i \in \{-1,1\}^k$ *for all* $i \in [n]$*.*

▶ **Definition 16** (Hadamard Addressing Function). *We define the* Hadamard addressing function*, which we denote* $\mathsf{HADD}_\ell : \{-1,1\}^{2\ell} \to \{-1,1,\star\}$*, as follows. Fix an arbitrary order on the* ℓ*-many Hadamard codewords of* $(\log \ell)$*-bit strings, say* $w_1, \ldots, w_\ell$*. Define the selector function of* HADD_ℓ *by*

$$g(x) = \begin{cases} i & \text{if } x = w_i \text{ for some } i \in [\ell] \\ \star & \text{otherwise.} \end{cases}$$

Note that HADD_ℓ *is an* (ℓ, ℓ)*-addressing function.*

2.2 Polynomial approximation

▶ **Definition 17** (Approximate Degree). *The* ϵ-approximate degree *of* $f : \{-1,1\}^n \to \{-1,1,\star\}$*, denoted by* $\widetilde{\deg}_\epsilon(f)$ *is defined to be the minimum degree of a real polynomial* $p : \{-1,1\}^n \to \mathbb{R}$ *that satisfies* $\left|p(x) - f(x)\right| \le \epsilon$ *for all* $x \in \{-1,1\}^n$ *for which* $f(x) \in \{-1,1\}$*.*[3] *That is,*

$$\widetilde{\deg}_\epsilon(f) := \min\{d : \deg(p) \le d, \left|p(x) - f(x)\right| \le \epsilon \ \forall x \in \{-1,1\}^n \text{ for which } f(x) \in \{-1,1\}\}.$$

Henceforth, we will use the notation $\widetilde{\deg}(f)$ to denote $\widetilde{\deg}_{1/3}(f)$.

▶ **Definition 18** (Approximate Spectral Norm). *The* approximate spectral norm *of a function* $f : \{-1,1\}^n \to \{-1,1,\star\}$*, denoted by* $\|\widehat{f}\|_{1,\epsilon}$ *is defined to be the minimum spectral norm of a real polynomial* $p : \{-1,1\}^n \to \mathbb{R}$ *that satisfies* $\left|p(x) - f(x)\right| \le \epsilon$ *for all* $x \in \{-1,1\}^n$ *for which* $f(x) \in \{-1,1\}$*.*

$$\|\widehat{f}\|_{1,\epsilon} := \min\{\|\widehat{p}\|_1 : \left|p(x) - f(x)\right| \le \epsilon \text{ for all } x \in \{-1,1\}^n \text{ for which } f(x) \in \{-1,1\}\}.$$

▶ **Lemma 19** ([10]). *Let* $f : \{-1,1\}^n \to \{-1,1\}$ *be a total function. Then for all constants* $0 < \delta, \epsilon < 1$ *we have* $\widetilde{\deg}_\epsilon(f) = \Theta(\widetilde{\deg}_\delta(f))$*. The constant hidden in the* Θ *notation depends on* δ *and* ϵ*.*

The following is a standard upper bound on the approximate spectral norm of a Boolean function in terms of its approximate degree.

[3] When dealing with partial functions, another notion of approximation is sometimes considered, where the approximating polynomial p is required to have bounded values even on the non-promise inputs of f. For the purpose of this paper, we do not require this constraint.

▷ Claim 20. For all total functions $f : \{-1,1\}^n \to \{-1,1\}$, we have $\log \|\widehat{f}\|_{1,1/3} = O(\widetilde{\deg}(f) \log n)$.

Proof. Let d denote the approximate degree of f. Take any 1/3-approximating polynomial of degree d, say p, to f. Then, $\sum_{S\subseteq[n]} |\widehat{p}(S)| \leq \sqrt{\binom{n}{\leq d}} \cdot \sqrt{\sum_{S:|S|\leq d} \widehat{p}(S)^2} \leq 4/3 \cdot (n+1)^{d/2} = 2^{O(d \log n)}$, where the first inequality follows by the Cauchy-Schwarz inequality, the second inequality follows by Parseval's identity (Fact 10) and the fact that the absolute value of p is at most 4/3 for any input $x \in \{-1,1\}^n$. ◁

It is easy to exhibit functions $f : \{-1,1\}^n \to \{-1,1\}$ such that $\log \|\widehat{f}\|_{1,1/3} = \Omega(\widetilde{\deg}(f))$. Bent functions satisfy this bound, for example.

Building upon ideas in [17], the approximate spectral norm of $f \circ \mathsf{IND}_1$ was shown to be bounded below by $2^{\Omega(\widetilde{\deg}(f))}$ in [11].

▶ **Theorem 21** ([11]). *Let $f : \{-1,1\}^n \to \{-1,1\}$ be any function. Then $\|\widehat{f \circ \mathsf{IND}_1}\|_{1,1/3} \geq 2^{c \cdot \widetilde{\deg}_{2/3}(f)}$ for any constant $c < 1 - 3/\widetilde{\deg}_{2/3}(f)$.*

2.3 Communication complexity

The classical model of communication complexity was introduced by Yao in [24]. In this model two parties, say Alice and Bob, wish to compute a function whose output depends on both their inputs. Alice is given an input $x \in \mathcal{X}$, Bob is given $y \in \mathcal{Y}$, and they want to jointly compute the value of a given function $F(x,y)$ by communicating with each other. Alice and Bob individually have unbounded computational power and the number of bits communicated is the resource we wish to minimize. Alice and Bob communicate using a *protocol* that is agreed upon in advance. In the randomized model, Alice and Bob have access to unlimited public random bits and the goal is to compute the correct value of $F(x,y)$ with probability at least 2/3 for all inputs $(x,y) \in \mathcal{X} \times \mathcal{Y}$. The *bounded-error randomized communication complexity* of a function F, denoted $R^{cc}(F)$, is the number of bits that must be communicated in the worst case by any randomized protocol to compute the correct value of the function $F(x,y)$, with probability at least 2/3, for every $(x,y) \in \mathcal{X} \times \mathcal{Y}$.

The quantum model of communication complexity was introduced by Yao in [25]. We refer the reader to the survey [23] for details. The *bounded-error quantum communication complexity* of a function F, denoted $Q^{cc}(F)$ is the number of bits that must be communicated by any quantum communication protocol in the worst case to compute the correct value of the function $F(x,y)$, with probability at least 2/3, for every (x,y) in domain of F. Buhrman, Cleve and Wigderson [9] observed a quantum simulation theorem, which gives an upper bound on the bounded-error quantum communication complexity of a composed function of the form $f \circ \wedge$ or $f \circ \oplus$ in terms of the bounded-error quantum query complexity of f (see Theorem 1).

Lee and Shraibman [18] showed that the bounded-error quantum communication complexity of $f \circ \oplus$ is bounded below by the logarithm of the approximate spectral norm of f. Also see [11] for an alternate proof.

▶ **Theorem 22** ([18]). *For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$,*

$$Q^{cc}(f \circ \oplus) = \Omega(\log \|\widehat{f}\|_{1,1/3}).$$

We remark that to the best of our knowledge, it is not known whether the same lower bound holds on the bounded-error quantum communication complexity of $f \circ \wedge$.

3 Proof of Theorem 3

In this section, we prove Theorem 3. We first formally define the function we use.

3.1 Definition of the function

If $\delta < 1/2$, then ignore the last $n - 2n^{2\delta}$ bits of the input, and define the following function on the first $2n^{\delta}$ bits of the input. The same argument as in Sections 3.2 and 3.3 give the required bounds for Theorem 3 and Theorem 4. Hence, we may assume without loss of generality that $\delta \geq 1/2$.

Define the partial function $f : \{-1,1\}^n \to \{-1,1,\star\}$ by $f = \mathsf{PARITY}_{n^{\delta}/2}^{\mathsf{HADD}_{n^{1-\delta}}}$. Define F to be the completion of f that evaluates to -1 on the non-promise domain of f (see Figure 1).

3.2 Upper bound

In this section, we prove the following.

▷ Claim 23. For $F : \{-1,1\}^n \to \{-1,1\}$ defined as in Section 3.1, we have $Q(F) = \Theta(n^{\delta})$.

The upper bound follows along the lines of a proof in [3], and the lower bound just uses the fact that F is at least as hard as $\mathsf{PARITY}_{n^{\delta}/2}$.

Proof. For convenience, set $\ell = n^{1-\delta}$ and $k = n^{\delta}$. Recall that an input to F is viewed as $(x_{11}, \ldots, x_{1,\ell}, y_{11}, \ldots, y_{1,\ell}, \ldots, x_{\frac{k}{2}1}, \ldots, x_{\frac{k}{2}\ell}, y_{\frac{k}{2}1}, \ldots, y_{\frac{k}{2}\ell})$. The following is an $O(n^{\delta})$-query quantum algorithm computing F. Note that since $\delta \geq 1/2$, we have $k = \Omega(\ell)$.

1. Run $k/2$ instances of the Bernstein-Vazirani algorithm on the $(k/2)$-many input strings $(x_{11}, \ldots, x_{1\ell}), \ldots, (x_{\frac{k}{2}1}, \ldots, x_{\frac{k}{2}\ell})$ to obtain $k/2$ strings $z_1, \ldots, z_{k/2}$.
2. Run Grover's search [14, 8] to check equality of the two strings: $h(z_1), \ldots, h(z_\ell)$ and $x_{11}, \ldots, x_{1\ell}, \ldots, x_{\frac{k}{2}1}, \ldots, x_{\frac{k}{2}\ell}$, i.e. to check whether the addressing bits of the input are indeed all Hadamard codewords which are output by the first step.
3. If the step above outputs that the strings are equal, then query the $k/2$ selected variables and output their parity. Else, output -1.
   - If the input was indeed of the form as claimed in the first step, then Bernstein-Vazirani outputs the correct $z_1, \ldots, z_\ell$ with probability 1, and Grover's search verifies that the strings are equal with probability 1. Hence the algorithm is correct with probability 1 in this case.
   - If the input was not of the claimed form, then the two strings for which equality is to be checked in the second step are not equal. Grover's search catches a discrepancy with probability at least $2/3$. Hence, the algorithm is correct with probability at least $2/3$ in this case.

The correctness of the algorithm is argued above, and the cost is $k/2$ queries for the first step, $O(\sqrt{k\ell})$ queries for the second step, and at most $k/2$ for the third step. Thus, we have $Q(F) = O(k + \sqrt{k\ell}) = O(k)$, since $k = \Omega(\ell)$. The upper bound in the lemma follows.

For the lower bound, we argue that F is at least as hard as $\mathsf{PARITY}_{k/2}$. To see this formally, set all the address variables such that the selected target variables are the first target variable in each block. Under this restriction, F equals $\mathsf{PARITY}(y_{11}, \ldots, y_{\frac{k}{2}1})$. Thus any quantum query algorithm computing F must be able to compute $\mathsf{PARITY}_{k/2}$, and thus $Q(F) = \Omega(k)$. ◁

▶ Remark 24. The same argument as above works when the function f is defined to be g^{HADD_ℓ} for any $g : \{-1,1\}^{n^\delta} \to \{-1,1\}$ satisfying $\widetilde{\deg}(g) = \Omega(n^\delta)$, and F is the completion of f that evaluates to -1 on all non-promise inputs. The same proof of Theorem 3 also goes through, but we fix $g = \mathsf{PARITY}_{n^\delta/2}$ for convenience.

3.3 Lower bound

In this section, we first prove Lemma 6. We require the following observation.

▶ **Observation 25.** *For any $S \subseteq [n]$ and any $j \in S$, we have $\mathbb{E}_{x_j \sim \{-1,1\}}[\chi_S(x)] = 0$, where x_j is distributed uniformly over $\{-1,1\}$.*

Proof of Lemma 6. Let $F = f^{\mathsf{ADDR}_{m,t}}$. Recall that our goal is to show that $\log \|\widehat{F}\|_{1,1/3} = \Omega(\widetilde{\deg}(f) \log t)$. We may assume $\widetilde{\deg}(f) \geq 1$, because the lemma is trivially true otherwise.

Towards a contradiction, suppose there exists a polynomial P of spectral norm strictly less than $2^{\frac{1}{10}\widetilde{\deg}_{0.99}(f)\log t}$ uniformly approximating F to error $1/3$ on the promise inputs (recall that from Lemma 19, we have $\widetilde{\deg}(f) = \Theta(\widetilde{\deg}_{0.99}(f))$).

Let ν be a distribution on the address bits of $\mathsf{ADDR}_{m,t}$ such that ν is supported only on assignments to the address variables that do not select $\star$, and is the uniform distribution over these assignments. Let $\mu = \nu^n$ be the product distribution over the address bits of the addressing functions in F.

- For any assignment z of the address variables from the support of μ, define a relevant (target) variable, with respect to z, to be one that is selected by z. Analogously, define a target variable to be irrelevant if it is not selected by z. Define a monomial to be relevant if it does not contain irrelevant variables, and irrelevant otherwise.
- Note that for any target variable, the probability with which it is selected is exactly $1/t$.
- In the analysis that follows in this proof, we are interested in the *set of target variables* that are present in a monomial χ_S, which we denote by S_{target}. Also define the *target-degree* of S to be $|S_{\text{target}}|$, i.e. the degree of a monomial χ_S when restricted to the target variables.
- Thus under any assignment z drawn from μ, for any monomial of the function P of target-degree $t \geq \widetilde{\deg}_{0.99}(f)$, the probability that it is relevant is at most $1/t^{\widetilde{\deg}_{0.99}(f)}$. Hence

$$\begin{aligned}
&\mathop{\mathbb{E}}_{z\sim\mu}[\ell_1\text{-norm of relevant monomials w.r.t. } z \text{ in } P \text{ of target-degree } \geq \widetilde{\deg}_{0.99}(f)] \\
&= \sum_{|S_{\text{target}}| \geq \widetilde{\deg}_{0.99}(f)} |\widehat{P}(S)| \Pr_{z\sim\mu}[\chi_S \text{ is relevant w.r.t. } z] \\
&\leq \max_{|S_{\text{target}}| \geq \widetilde{\deg}_{0.99}(f)} \{\Pr_{z\sim\mu}[\chi_S \text{ is relevant w.r.t. } z]\} \cdot \|\widehat{P}\|_1 \\
&< \frac{1}{t^{\widetilde{\deg}_{0.99}(f)}} \cdot 2^{\frac{1}{10}\widetilde{\deg}_{0.99}(f)\log t} = 2^{(-\frac{9}{10})\widetilde{\deg}_{0.99}(f)\log t} < \frac{3}{5},
\end{aligned}$$

  where the second last inequality holds because of Claim 20 and the last inequality holds because $t \geq 2$ and $\widetilde{\deg}_{0.99}(f) \geq 1$.
- Fix an assignment to the address variables from the support of μ such that under this assignment, the ℓ_1-norm of relevant monomials in P of degree $\geq \widetilde{\deg}_{0.99}(f)$ is less than $3/5$.
- Note that under this assignment (in fact under any assignment in the support of μ), the restricted F is just the function f on the n variables selected by the addressing functions. Denote by P_1 the polynomial on the target variables obtained from P by fixing address variables as per this assignment.

- Drop the relevant monomials of degree $\geq \widetilde{\deg}_{0.99}(f)$ from P_1 to get a polynomial P_2, which uniformly approximates the restricted F (which is f on n variables) to error $1/3 + 3/5 < 0.99$.
- Take expectation over irrelevant variables (from the distribution where each irrelevant variable independently takes values uniformly from $\{-1, 1\}$). Under this expectation, the value of F does not change (since irrelevant variables do not affect F's output by definition), and all irrelevant monomials of P_2 become 0 (using Observation 25 and linearity of expectation). Hence, under this expectation we have $\mathbb{E}[P_2] = P_3$, where P_3 is a polynomial of degree strictly less than $\widetilde{\deg}_{0.99}(f)$. Furthermore, P_3 uniformly approximates f to error less than 0.99 which is a contradiction. ◀

As a corollary of Lemma 6, we obtain a lower bound on the approximate spectral norm of F, where F is defined as in Section 3.1. This yields a proof of Theorem 4.

Proof of Theorem 4. Construct F as in Section 3.1. Claim 23 implies $Q(F) = \Theta(n^\delta)$.

Let $f = \mathsf{PARITY}_{n^\delta/2}^{\mathsf{HADD}_{n^{1-\delta}}}$. Lemma 6 implies that $\|\widehat{f}\|_{1,1/3} = \Omega(n^\delta \log n)$.

Since F is a completion of f, we have $\|\widehat{F}\|_{1,1/3} = \Omega(n^\delta \log n)$, which proves the lower bound in Theorem 4. The upper bound follows from Theorem 1. ◀

We are now ready to prove our main theorem.

Proof of Theorem 3. It immediately follows from Theorem 4 and Theorem 22. ◀

4 Conclusions

We conclude with the following points: first, we find our main result somewhat surprising that simulating a query algorithm by a communication protocol in the quantum context has a larger overhead than in the classical context. Second, it is remarkable that this relatively fine overhead of $\log n$ can be detected using analytic techniques that are an adaptation of the generalized discrepancy method. Third, the function that we used in this work is an XOR function. Study of this class of functions is proving to be very insightful. A recent example is the refutation of the log-approximate-rank conjecture [12] and even its quantum version [4, 22]. Our work further advocates the study of XOR functions.

An open question that remains is whether there exists a Boolean function $F : \{-1, 1\}^n \to \{-1, 1\}$ such that $Q^{cc}(F \circ \wedge) = \Omega(Q(F) \log n)$. Or does there exist a better quantum communication protocol for $(F \circ \wedge)$ that does not incur the logarithmic factor loss?

It is easy to verify that the constructions of F that yield Theorem 4 for any fixed constant $0 < \delta < 1$, also satisfy $\widetilde{\deg}(F) = \Theta(n^\delta)$. Recall that Theorem 9 states that such functions F satisfy $\log \|\widehat{F}\|_{1,1/3} = \Omega(\widetilde{\deg}(F) \log n)$. This gives a negative answer to Question 8 ([5, Section 4]), where it was asked if any degree-d approximating polynomial to a Boolean function of approximate degree d has spectral norm at most $2^{O(d)}$ (it is interesting to note that their conjecture holds true for symmetric functions, which we prove in Appendix A). Thus to prove min-entropy of the Fourier spectrum of a Boolean function is upper bounded by approximate degree, it cannot follow from their observation that min-entropy is upper bounded by the logarithm of the approximate spectral norm. The following remains an interesting and important open problem: (how) can one prove that the min-entropy of the Fourier spectrum of a Boolean function is upper bounded by a constant multiple of its approximate degree? Such an inequality is implied by the Fourier Entropy Influence (FEI) Conjecture.

References

1 Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005.

2 Anil Ada, Omar Fawzi, and Hamed Hatami. Spectral norm of symmetric functions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 338–349, 2012.

3 Andris Ambainis and Ronald de Wolf. How low can approximate degree and quantum query complexity be for total boolean functions? *Computational Complexity*, 23(2):305–322, 2014.

4 Anurag Anshu, Naresh Goud Boddu, and Dave Touchette. Quantum log-approximate-rank conjecture is also false. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 982–994, 2019.

5 Srinivasan Arunachalam, Sourav Chakraborty, Michal Koucký, Nitin Saurabh, and Ronald de Wolf. Improved bounds on fourier entropy and min-entropy. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, pages 45:1–45:19, 2020. doi:10.4230/LIPIcs.STACS.2020.45.

6 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

7 Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

8 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

9 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 63–68, 1998.

10 Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007.

11 Arkadev Chattopadhyay and Nikhil S. Mande. A lifting theorem with applications to symmetric functions. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, pages 23:1–23:14, 2017.

12 Arkadev Chattopadhyay, Nikhil S. Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 42–53, 2019.

13 Ehud Friedgut and Gil Kalai. Every monotone graph property has a sharp threshold. *Proceedings of the American Mathematical Society*, 124(10):2993–3002, 1996.

14 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219, 1996.

15 Peter Høyer and Ronald de Wolf. Improved quantum communication complexity bounds for disjointness and equality. In *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, pages 299–310, 2002.

16 Esty Kelman, Guy Kindler, Noam Lifshitz, Dor Minzer, and Muli Safra. Revisiting bourgain-kalai and fourier entropies. *CoRR*, abs/1911.10579, 2019. arXiv:1911.10579.

17 Matthias Krause and Pavel Pudlák. On the computational power of depth-2 circuits with threshold and modulo gates. *Theor. Comput. Sci.*, 174(1-2):137–156, 1997.

18 Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–398, 2009.

19 Ashley Montanaro, Harumichi Nishimura, and Rudy Raymond. Unbounded-error quantum query complexity. *Theor. Comput. Sci.*, 412(35):4619–4628, 2011.

20 Alexander A Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics*, 67(1):145, 2003.

21 Alexander A. Sherstov. Algorithmic polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 311–324, 2018.

22 Makrand Sinha and Ronald de Wolf. Exponential separation between quantum communication and logarithm of approximate rank. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 966–981, 2019.

23 Ronald de Wolf. Quantum communication and complexity. *Theor. Comput. Sci.*, 287(1):337–353, 2002. doi:10.1016/S0304-3975(02)00377-8.

24 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979.

25 Andrew Chi-Chih Yao. Quantum circuit complexity. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 352–361, 1993.

26 Shengyu Zhang. On the tightness of the Buhrman-Cleve-Wigderson simulation. In *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, pages 434–440, 2009.

A Upper bound on the approximate spectral norm of symmetric functions

Recall from Section 1.3 that Theorem 9 gives a negative answer to Question 8, where it was asked if for all Boolean functions of approximate degree d, there exists an approximating polynomial with spectral norm $2^{O(d)}$. We show in this section that the upper bound in Question 8 does hold true for symmetric functions.

▶ **Definition 26** (Multilinear Polynomial). *A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a multilinear polynomial if ϕ is of the form:*

$$\phi(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i$$

where $a_S \in \mathbb{R}$.

▶ **Definition 27** (Spectral Norm of a Multilinear Polynomial). *Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be a multilinear polynomial of the form $\phi(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i$. The spectral norm of ϕ, denoted by $\|\phi\|_1$, is defined as*

$$\|\phi\|_1 = \sum_{S \subseteq [n]} |a_S|.$$

▶ **Fact 28** (Properties of Spectral Norm of Multilinear Polynomials). *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ be any symmetric polynomials and let $\alpha \in \mathbb{R}$ be any real number. Then,*

1. $\|\alpha f\|_1 = |\alpha| \|f\|_1$,
2. $\|f + g\|_1 \leq \|f\|_1 + \|g\|_1$,
3. $\|fg\|_1 \leq \|f\|_1 \|g\|_1$.

▶ **Lemma 29.** *Let $S \subseteq [n]$ and $\chi_S : \{-1, 1\}^n \to \mathbb{R}$ be the symmetric multilinear polynomial defined as*

$$\chi_S(x_1, \ldots, x_n) = \prod_{i \in S} \frac{(1 - x_i)}{2}.$$

Then $\|\chi_S\|_1 = 1$.

Proof. Since for all $i \in [n]$, the spectral norm of $\frac{(1-x_i)}{2} = 1$, the proof follows from 28 (3). ◀

▶ **Definition 30** (Symmetric Multilinear Polynomial). *A multilinear polynomial $\phi : \mathbb{R}^n \to \mathbb{R}$ is said to be symmetric if $\phi(x_1, \ldots, x_n) = \phi(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ for all $(x_1, \ldots, x_n) \in X$ and $\sigma \in S_n$.*

Sherstov [21] showed the following upper bound on the spectral norm of symmetric multilinear polynomials.

▷ Claim 31 ([21]). Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be a symmetric multilinear polynomial. Then

$$\|\phi\|_1 \leq 8^{\deg(\phi)} \max_{x \in \{0,1\}^n} |\phi(x)|.$$

▶ **Lemma 32.** *Let $f : \{-1,1\}^n \to \{-1,1\}$ be a symmetric Boolean function. Then*

$$\log(\|f\|_{1,1/3}) = O(\widetilde{\deg}(f)).$$

Proof. Let $f' : \{0,1\}^n \to \{-1,1\}$ be defined as $f'(x_1, \ldots, x_n) = f\left(\frac{1-x_1}{2}, \ldots, \frac{1-x_n}{2}\right)$. It is not hard to show, since we have done a linear transformation on the input domain, that $\widetilde{\deg}(f') = \widetilde{\deg}(f)$. Let p' be a polynomial that 1/3-approximates the symmetric function f'. By symmetrization we can assume that p' is symmetric, and is of the form $p'(x) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i$.

Define the polynomial $p : \{-1,1\}^n \to \mathbb{R}$ as follows:

$$p(x_1, \ldots, x_n) = p'\left(\frac{1-x_1}{2}, \ldots, \frac{1-x_n}{2}\right).$$

Clearly p is a 1/3-approximation to f since p' is a 1/3-approximation to f' and we can write

$$p(x) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} \frac{(1-x_i)}{2}.$$

Thus we can upper bound the ℓ_1-norm of p as follows:

$$\|p\|_1 = \left\| \sum_{S \subseteq [n]} a_S \prod_{i \in S} \frac{(1-x_i)}{2} \right\|_1 \leq \sum_{S \subseteq [n]} \left\| a_S \prod_{i \in S} \frac{(1-x_i)}{2} \right\|_1 \tag{2}$$

$$\leq \sum_{S \subseteq [n]} |a_S| \left\| \prod_{i \in S} \frac{(1-x_i)}{2} \right\|_1 \tag{3}$$

$$= \sum_{S \subseteq [n]} |a_S| = \|p'\|_1, \tag{4}$$

where Equation (2) follows from Fact 28 (2), Equation (3) follows from Fact 28 (1) and Equation (4) follows from 29.

Hence, $\log(\|f\|_{1,1/3}) \leq \log(\|p\|_1) \leq \log(\|p'\|_1) = O(\deg(p'))$, where the last equality follows by Claim 31 since p' is symmetric. Since p' was assumed to have degree $\deg(p') = \widetilde{\deg}(f)$, the lemma follows. ◀

Factorization of Polynomials Given By Arithmetic Branching Programs

Amit Sinhababu
Aalen University, Germany
amitks@cse.iitk.ac.in

Thomas Thierauf
Aalen University, Germany
thomas.thierauf@uni-ulm.de

Abstract

Given a multivariate polynomial computed by an arithmetic branching program (ABP) of size s, we show that all its factors can be computed by arithmetic branching programs of size $\text{poly}(s)$. Kaltofen gave a similar result for polynomials computed by arithmetic circuits. The previously known best upper bound for ABP-factors was $\text{poly}(s^{\log s})$.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Arithmetic Branching Program, Multivariate Polynomial Factorization, Hensel Lifting, Newton Iteration, Hardness vs Randomness

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.33

Funding Research supported by DFG grant TH 472/5-1.

Acknowledgements We thank Nitin Saxena, Pranjal Dutta, Arpita Korwar, Sumanta Ghosh, Zeyu Guo and Mrinal Kumar for helpful discussions. A.S would like to thank the Institute of Theoretical Computer Science at Ulm University for the hospitality.

1 Introduction

Polynomial factoring is a classical question in algebra. For factoring multivariate polynomials, we have to specify a model for representing polynomials. A standard model in algebraic complexity to represent polynomials are *arithmetic circuits* (aka *straight-line programs*). Other well known models are *arithmetic branching programs* (ABP), *arithmetic formulas*, *dense representations*, where the coefficients of *all* n-variate monomials of degree $\leq d$ are listed, or *sparse representations*, where only the non-zero coefficients are listed. Given a polynomial in some model, one can ask for efficient algorithms for computing its factors represented in the same model. That leads to the following question.

▶ **Question** (Factor size upper bound). *Given a polynomial of degree d and size s in a representation, do all of its factors have size* $\text{poly}(s, d)$ *in the same representation?*

For example in the dense representation the size of the input polynomial and the output factors is the same, namely $\binom{n+d}{d}$, for n-variate polynomials of degree d. But for other representations, the factor of a polynomial may take *larger* size than the polynomial itself. For example, in the sparse representation the polynomial $x^d - 1$ has size 2, but its factor $1 + x + \cdots + x^{d-1}$ has size d.

Arithmetic circuits. The algebraic complexity class VP contains all families of polynomials $\{f_n\}_n$ that have degree $\text{poly}(n)$ and arithmetic circuits of size $\text{poly}(n)$. Kaltofen [11] showed that VP is closed under factoring: Given a polynomial $f \in \text{VP}$ of degree d computed by an arithmetic circuit of size s, all its factors can be computed by an arithmetic circuit of size $\text{poly}(s, d)$.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 33; pp. 33:1–33:19
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Arithmetic branching programs. Kaltofen's [11] proof technique for circuit factoring does not directly extend to formulas or ABPs. The construction there results in a circuit, even if the input polynomial is given as a formula or an ABP. Converting a circuit to an arithmetic formula or an ABP *may* cause super-polynomial blow-up of size.

Analogous to VP, classes VF and VBP contain families of polynomials that can be computed by polynomial-size arithmetic formulas and branching programs, respectively. Note that the size also bounds the degree of the polynomials in these models. Arithmetic branching programs are an intermediate model in terms of computational power, between arithmetic formulas and arithmetic circuits,

$$\mathrm{VF} \subseteq \mathrm{VBP} \subseteq \mathrm{VP}\,.$$

ABPs are interesting in algebraic complexity as they essentially capture the power of linear algebra, for example they can efficiently compute determinants. ABPs have several equivalent characterizations. They can be captured via iterated matrix multiplication, weakly-skew circuits, skew circuits, and determinants of a symbolic matrices. See [14] for an overview of these connections.

Proof technique. A standard technique to factor multivariate polynomials has typically two main steps. The first step uses Hensel lifting to lift a factorization to high enough precision, starting from two coprime univariate factors. The second step, sometimes called the *jump step* or *reconstruction step*, consists of reconstructing a factor from a corresponding lifted factor by solving a system of linear equations.

The earlier works for polynomial factorization use a version of Hensel lifting, where in each iteration the lifted factors remain *monic*. It seems as this version is not efficient for ABPs. We observe that monicness of the lifted factor is not necessary for the jump step. This allows us to use a simple version of Hensel lifting that is efficient for ABPs.

Another point in some earlier works is that, in a pre-processing step, the input polynomial is transformed into a square-free polynomial. It is not clear how to achieve this transformation with small ABPs. We get around this problem by observing that square-freeness is not necessary. It suffices to have one irreducible factor of multiplicity one. This weaker transformation can be computed by small ABPs.

Finally, we use the fact that the determinant can be computed efficiently by ABPs.

▶ Remark 1. Whatever ABP we construct, the same can be done for circuits. Hence, as a by-product, we also literally provide another proof for the classical circuit factoring result of Kaltofen.

Comparison with prior works. There are several proofs of the closure of VP under factors [9, 10, 11, 1, 2, 12, 16, 6, 3]. None of the previous proofs directly extends to the closure of VBP, i.e. branching programs, under factors.

Recently, Dutta, Saxena, and Sinhababu [6] and also Oliveira [16] considered factoring in restricted models like formulas, ABPs and small depth circuits. They reduce polynomial factoring to approximating power series roots of the polynomial to be factored. Then they use versions of Newton iteration for approximating the roots. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$. If $p(\boldsymbol{x}, y)$ is the given polynomial and $q(\boldsymbol{x})$ is a root w.r.t. y, i.e. $p(\boldsymbol{x}, q(\boldsymbol{x})) = 0$, Newton iteration repeatedly uses the following recursive formula to approximate q:

$$y_{t+1} = y_t - \frac{p(\boldsymbol{x}, y_t)}{p'(\boldsymbol{x}, y_t)}.$$

If p is given as a circuit, the circuit for y_{t+1} is constructed from the circuit of y_t. For the circuit model, we can assume that $p(\boldsymbol{x}, y)$ has a single leaf node y where we feed y_t. But for formula and branching programs, we may have d many leaves labeled by y, where d is the degree of p in terms of y. As we cannot reuse computations in formula or branching programs, we have to make d copies of y_t in each round. This leads to $d^{\log d}$ blow-up in size.

Oliveira [16] used the idea of approximating roots via an approximator polynomial function of the coefficients of a polynomial. This gives good upper bound on the size of factors of ABPs, formulas, and bounded depth circuits under the assumption that the individual degrees of the variables in the input polynomial are bounded by a constant.

Recently, Chou, Kumar, and Solomon [3] proved closure of VP under factoring using Newton iteration for several variables for a system of polynomial equations. This approach also faces the same problem for the restricted models.

Instead of lifting roots, another classical technique for multivariate factoring is *Hensel lifting*, where factors modulo an ideal are lifted. Hensel lifting has a slow version, where the power of the ideal increases by one in each round. The other version due to Zassenhaus [21] is fast, the power of the ideal gets doubled in each round.

Kaltofen's [11, 10] proofs uses slow versions of Hensel lifting iteratively for d rounds, where d is the degree of the given polynomial. That leads to an exponential blow-up of size in models where the previous computations cannot be reused, as using previous lifts twice would need two copies each time. .

Kopparty, Saraf, and Shpilka [12] use the standard way of doing fast Hensel lifting for $\log d$ rounds, where in each round the lifted factors are kept monic. To achieve this, one has to compute a polynomial division with remainder. Implementing this version of Hensel lifting for ABPs or formulas seems to require to make d copies of previous computations in each round. Thus, that way would lead to a $d^{\log d}$ size blow-up. Also, they compute the gcd of polynomials, for which a priori no size upper bound was known for ABP or formulas.

Here, we use a classic version of fast Hensel lifting, that needs $\log d$ rounds and additionally in each round we have to make copies of previous computations only constantly many times. As we mentioned earlier, we avoid to maintain the monicness, and also gcd-computations.

Though various versions of Hensel lifting (factorization lifting) and Newton iteration techniques (root lifting) are equivalent in a mathematical sense [19], it is interesting that the former gives a better factor size upper bound for the model of ABP.

Application in hardness vs. randomness. Closure under factoring is used in the hardness vs. randomness trade-off results in algebraic complexity. See for example the excellent survey of Kumar and Saptharishi [13] for details on this topic. The celebrated result of Kabanets and Impagliazzo [8, Theorem 7.7] showed that a sufficiently strong lower bound for arithmetic circuits would *derandomize* polynomial identity testing (PIT). The proof of derandomization uses a hard polynomial as well as the upper bound on the size of factors of a polynomial computed by the circuit [11]. As a corollary of our result, we get a similar statement in terms of ABPs: An exponential (or super-polynomial) lower bound for ABPs for an explicit multilinear polynomial yields quasi-poly (or sub-exponential) black-box derandomization of PIT for polynomials in VBP.

Closure under factoring is relevant in the connection between algebraic complexity and proof complexity [7]. If a class $\mathcal{C}$ is closed under factoring, then the following holds. If a polynomial is *hard* for the class $\mathcal{C}$, then all its nonzero multiples are hard for $\mathcal{C}$. Lower bounds for all the nonzero multiples of an explicit hard polynomial may lead to lower bounds for ideal proof systems [7].

2 Preliminaries

We consider multivariate polynomials over a field $\mathbb{F}$ of characteristic 0. A polynomial p is called *square-free*, if for any non-constant irreducible factor q, the polynomial q^2 is not a factor of p.

By $\deg(p)$ we denote the total degree of p. Let x and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be variables and $p(x, \boldsymbol{z})$ be a $(n+1)$-variate polynomial. Then we can view p as a univariate polynomial $p = \sum_i a_i(\boldsymbol{z})\, x^i$ over $\mathbb{K}[x]$, where $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$. The *$x$-degree of p* is denoted by $\deg_x(p)$. It is the highest degree of x in p. Polynomial p is called *monic in x*, if the coefficient $a_{d_x}(\boldsymbol{z})$ is a nonzero constant, where $d_x = \deg_x(p)$.

By $\mathrm{poly}(n)$ we denote the class of polynomials in $n \in \mathbb{N}$.

We denote by $\mathcal{I} = \langle x \rangle$ the ideal of polynomials generated by x over the ring $\mathbb{F}[x, \boldsymbol{z}]$. The *$k$-th power of the ideal $\mathcal{I}$* is the ideal $\mathcal{I}^k = \langle x^k \rangle$.

Computational models. An *arithmetic circuit* is a directed acyclic graph, whose leaf nodes are labeled by the variables $x_1, \ldots, x_n$ and various constants from the underlying field. The other nodes are labeled by sum gates or product gates. A node labeled by a variable or constant computes the same. A node labeled by sum or product compute the sum or product of the polynomials computed by nodes connected by incoming edges. The *size of an arithmetic circuit* is the total number of its edges.

An *arithmetic formula* is a special kind of arithmetic circuit. A formula has the structure of a directed acyclic tree. Every node in a formula has out-degree at most one. As we can not *reuse* computations in a formula, it is considered to be weaker than circuits.

An *arithmetic branching program* (ABP) is a layered directed acyclic graph with a single source node and a single sink node. An edge of an ABP is labeled by a variable or a constant from the field. The weight corresponding to a path from the source to the sink is the product of the polynomials labeling the edges on the path. The polynomial $f(x_1, \ldots, x_n)$ computed by the ABP is the sum of the weights of the all possible paths from source to sink.

The *size of an ABP* is the number of its edges. The size of the smallest ABP computing f is denoted by $\mathrm{size}_{\mathrm{ABP}}(f)$. The degree of a polynomial computed by an ABP of size s is at most $\mathrm{poly}(s)$.

Properties of ABPs. *Univariate* polynomials have small ABPs. Let $p(x)$ be a univariate polynomial of degree d. It can be computed by an ABP of size $2d+1$, actually even by a formula of that size.

For univariate polynomials $p(x), q(x)$, the *extended Euclidian algorithm* computes the gcd $h = \gcd(p, q)$ and also the *Bézout-coefficents*, polynomials a, b such that $ap + bq = h$, where $\deg(a) < \deg(q)$ and $\deg(b) < \deg(p)$. Let p have the larger degree, $d = \deg(p) \geq \deg(q)$. Then clearly also $\deg(h), \deg(a), \deg(b) \leq d$, and consequently, all these polynomials, p, q, h, a, b have ABP-size at most $2d+1$.

Let $p(\boldsymbol{x}), q(\boldsymbol{x})$ be multivariate polynomials in $\boldsymbol{x} = (x_1, \ldots, x_n)$. For the ABP-size with respect to *addition* and *multiplication*, we have

1. $\mathrm{size}_{\mathrm{ABP}}(p+q) \leq \mathrm{size}_{\mathrm{ABP}}(p) + \mathrm{size}_{\mathrm{ABP}}(q)$,
2. $\mathrm{size}_{\mathrm{ABP}}(pq) \leq \mathrm{size}_{\mathrm{ABP}}(p) + \mathrm{size}_{\mathrm{ABP}}(q)$.

For the sum of two ABPs B_p, B_q one can put B_p and B_q in parallel by merging the two source nodes of B_p and B_q into one new source node, and similar for the two sink nodes. For the product, one can put B_p and B_q in series by merging the sink of B_p with the source of B_q.

Another operation is *substitution*. Let $p(x_1, \ldots, x_n)$ and $q_1(\boldsymbol{x}), \ldots, q_n(\boldsymbol{x})$ be polynomials. Let $\text{size}_{\text{ABP}}(q_i) \leq s$, for $i = 1, \ldots, n$. Then we have

$$\text{size}_{\text{ABP}}(p(q_1, \ldots, q_n)) \leq s \cdot \text{size}_{\text{ABP}}(p).$$

To get an ABP for $p(q_1(\boldsymbol{x}), \ldots, q_n(\boldsymbol{x}))$, replace an edge labeled x_i in the ABP B_p for p by the ABP B_{q_i} for q_i.

It is known that the *determinant of a symbolic matrix* of dimension n can be computed by an ABP of size $\text{poly}(n)$ [15]. By substitution, the entries of the matrix can itself be polynomials computed by ABPs.

Resultant. Given two polynomials $p(x, \boldsymbol{y})$ and $q(x, \boldsymbol{y})$ in variables x and $\boldsymbol{y} = (y_1, \ldots, y_n)$, consider them as polynomials in x with coefficients in $\mathbb{F}[\boldsymbol{y}]$. The *resultant of p and q w.r.t. x*, denoted by $\text{Res}_x(p, q)$, is the determinant of the Sylvester matrix of p and q. For the definition of the Sylvester matrix, see [20]. Note that $\text{Res}_x(p, q)$ is a polynomial in $\mathbb{F}[\boldsymbol{y}]$.

Basic properties of the resultant are that it can be represented as a linear combination of p and q, and that it provides information about the gcd of p and q.

▶ **Lemma 2** (See [20]). *Let $p(x, \boldsymbol{y})$ and $q(x, \boldsymbol{y})$ be polynomials of degree $\leq d$ and $h = \gcd(p, q)$.*

1. $\deg(\text{Res}_x(p, q)) \leq 4d^2$,
2. $\exists u, v \in \mathbb{F}[x, \boldsymbol{y}] \quad up + vq = \text{Res}_x(p, q)$,
3. $\text{Res}_x(p, q) = 0 \iff \deg_x(h) > 0$.

Note that the problem whether $\text{Res}_x(p, q) = 0$ is a polynomial identity test (PIT), because $\text{Res}_x(p, q) \in \mathbb{F}[\boldsymbol{y}]$. It can be solved in a randomized way by the DeMillo-Lipton-Schwartz-Zippel Theorem (see [4] and the references therein for more details and history of this theorem).

▶ **Theorem 3** (Polynomial Identity Test). *Let $p(\boldsymbol{x})$ be an n-variate nonzero polynomial of total degree d. Let $S \subseteq \mathbb{F}$ be a finite set. For $\boldsymbol{\alpha} \in S^n$ picked independently and uniformly at random,*

$$\Pr[\, p(\boldsymbol{\alpha}) = 0 \,] \;\leq\; \frac{d}{|S|}\,.$$

3 Pre-processing Steps and Algebraic Tool Kit

Before we start the Hensel lifting process, a polynomial should fulfill certain properties that the input polynomial might not have. In this section, we describe transformations of a polynomial that achieve these properties such that ABPs can compute the transformation and its inverse, and factors of the polynomials are maintained.

We also explain how to compute homogeneous components and how to solve linear systems via ABPs. We show how handle the special case when the given polynomial is just a power of an irreducible polynomial.

3.1 Computing homogeneous components and coefficients of a polynomial

Let $p(x, \boldsymbol{z})$ be polynomial of degree d in variables x and $\boldsymbol{z} = (z_1, \ldots, z_n)$. Let B_p be an ABP of size s that computes a polynomial p. Write p as a polynomial in x, with coefficients from $\mathbb{F}[\boldsymbol{z}]$,

$$p(x, \boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})\, x^i\,.$$

We show that all the coefficients $p_i(\boldsymbol{z})$ have ABPs of size $\mathrm{poly}(s, d)$.

The argument is similar to Strassen's *homogenization* technique for arithmetic circuits, an efficient way to compute all the homogeneous components of a polynomial. The same technique can be used for ABPs (see [17, Lemma 5.2 and Remark]). Here we sketch the proof idea.

Each node v of B_p we split into $d+1$ nodes $v_0, \ldots, v_d$, such that node v_i computes the degree i part of the polynomial computed by node v, for $i = 0, 1, \ldots, d$. Consider an edge e between node u and v in B_p.

- If e is labeled with a constant $c \in \mathbb{F}$ or a variable z_i, then we put an edge between u_i and v_i with label c or z_i, respectively.
- If e is labeled with variable x, then we put an edge between u_i and v_{i+1} with label 1.

The resulting ABP has one source node and $d+1$ sink nodes. The i-th sink node computes $p_i(\boldsymbol{z})$.

For each edge of B_p we get either d or $d+1$ edges in the new ABP. Hence, its size is bounded by $s(d+1)$.

▶ **Lemma 4** (Coefficient extraction). *Let* $p(x, \boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})\, x^i$ *be a polynomial. Then* $\mathrm{size}_{\mathrm{ABP}}(p_i) \le (d+1)\,\mathrm{size}_{\mathrm{ABP}}(p)$*, for* $i = 0, 1, \ldots, d$.

The technique can easily be extended to constantly many variables. For two variables, consider $p(x, y, \boldsymbol{z}) = \sum_{i,j} p_{i,j}(\boldsymbol{z})\, x^i y^j$. Then, from an ABP of size s for p we get ABPs for the coefficients $p_{i,j}(\boldsymbol{z})$ of size $s(d+1)^2$ similarly as above.

In *homogenization*, we want to compute the homogeneous components of p. That is, write $p(\boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})$, where $\deg(p_i) = i$. From an ABP B_p for p we get ABPs for the p_i's similarly as above: In the definition of the new edges, only for constant label, we put the edge from u_i to v_i. In case of any variable label z_j, we put the edge from u_i to v_{i+1} with label z_j. Then the i-th sink node computes $p_i(\boldsymbol{z})$. The size is bounded by $s(d+1)$.

▶ **Lemma 5** (Homogenization). *Let* $p(\boldsymbol{z}) = \sum_{i=0}^{d} p_i(\boldsymbol{z})$ *be a polynomial with* $\deg(p_i) = i$*, for* $i = 0, 1, \ldots, d$*. Then* $\mathrm{size}_{\mathrm{ABP}}(p_i) \le (d+1)\,\mathrm{size}_{\mathrm{ABP}}(p)$*, for* $i = 0, 1, \ldots, d$.

3.2 Computing q from $p = q^e$

A special case is when the given polynomial $p(\boldsymbol{x})$ is a power of one irreducible polynomial $q(\boldsymbol{x})$, i.e., $p = q^e$, for some $e > 1$. This case is handled separately. Kaltofen [10] showed how to compute q for circuits, ABPs, and arithmetic formulas. Here, we give a short proof from Dutta [5].

▶ **Lemma 6.** *Let* $p = q^e$*, for polynomials* $p(\boldsymbol{x}), q(\boldsymbol{x})$*. Then* $\mathrm{size}_{\mathrm{ABP}}(q) \le \mathrm{poly}(\mathrm{size}_{\mathrm{ABP}}(p))$.

Proof. We may assume that p is nonzero; otherwise the claim is trivial. We want to apply Newton's binomial theorem to compute $q = p^{1/e}$. For this we need that $p(0, \ldots, 0) = 1$. If this is not the case, we first transform p as follows.

1. If $p(0, \ldots, 0) = 0$, let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a point where $p(\boldsymbol{\alpha}) \neq 0$. By the PIT-Theorem, a random point $\boldsymbol{\alpha}$ will work, with high probability. Now we shift the variables and work with the shifted polynomial $\tilde{p}(\boldsymbol{x}) = p(\boldsymbol{x} + \boldsymbol{\alpha})$.
   Still, $\tilde{p}(0, \ldots, 0)$ might be different from 1. In this case, we also apply the next item to $\tilde{f}$.

2. If $p(0,\ldots,0) = a_0 \neq 0$, then we work with $\tilde{p}(\boldsymbol{x}) = p(\boldsymbol{x})/a_0$. Then $\tilde{p}(0,\ldots,0) = 1$.
Note that both transformations are easily reversible. Hence, in the following we simply assume that $p(0,\ldots,0) = 1$.

By Newton's binomial theorem, we have

$$q = p^{1/e} = (1+(p-1))^{1/e} = \sum_{i=0}^{\infty} \binom{1/e}{i} (p-1)^i \,. \tag{1}$$

Note that $p^{1/e}$ is a polynomial of degree $d = \deg(q)$. Since $p-1$ is constant free, the terms $(p-1)^j$ in the RHS of (1) have degree $> d$, for $j > d$. Thus (1) turns into a finite sum modulo the ideal $\langle \boldsymbol{x} \rangle^{d+1}$,

$$q = \sum_{i=0}^{d} \binom{1/e}{i} (p-1)^i \bmod \langle \boldsymbol{x} \rangle^{d+1} \,. \tag{2}$$

Let $\mathrm{size}_{\mathrm{ABP}}(p) = s$. For the polynomial $Q = \sum_{i=0}^{d} \binom{1/e}{i}(p-1)^i$ from (2), we clearly have $\mathrm{size}_{\mathrm{ABP}}(Q) \leq \mathrm{poly}(s)$. Finally, to get $q = Q \bmod \langle \boldsymbol{x} \rangle^{d+1}$, we have to truncate the terms in Q with degree $> d$. This can be done by computing the homogeneous components of Q as described in Lemma 5. We conclude that $\mathrm{size}_{\mathrm{ABP}}(q) \leq \mathrm{poly}(s)$. ◀

3.3 Reducing the multiplicity of a factor

In the earlier works on bivariate and multivariate polynomial factoring, typically the problem is reduced to factoring a *square-free* polynomial. This is convenient at various places in the Hensel lifting process. The technique to reduce to the square-free case is via taking the gcd of the input polynomial and its derivative. However, for getting upper bounds on the ABP-size of the factors, we want to avoid gcd-computations, because no polynomial size upper bound for the gcd of two ABPs is known.

We avoid this problem by observing that we do not need the polynomial to be square-free. As we will see, it suffices to have one irreducible factor with multiplicity one, and another coprime factor.

Let $p(\boldsymbol{x})$ be the given polynomial, for $\boldsymbol{x} = (x_1,\ldots,x_n)$. The special case that p is a power of one irreducible polynomial we just handled in Section 3.2. Hence, we may assume that p has at least two irreducible factors. So let $p = q^e\, p_0$, where q is irreducible and coprime to p_0.

Consider the derivative of p w.r.t. some variable, say x_1.

$$\frac{\partial p}{\partial x_1} = q^{e-1}\left((e-1)\frac{\partial q}{\partial x_1}p_0 + q\frac{\partial p_0}{\partial x_1}\right). \tag{3}$$

Note that q does not divide the factor $\left((e-1)\frac{\partial q}{\partial x_1}p_0 + q\frac{\partial p_0}{\partial x_1}\right)$ in (3). Hence, the multiplicity of factor q in $\frac{\partial p}{\partial x_1}$ is reduced by one compared to p.

For the ABP-size, we write p as a polynomial in x_1, i.e. $p(\boldsymbol{x}) = \sum_{i=0}^{d} a_i x_1^i$, where the coefficients a_i are polynomials in $x_2,\ldots,x_n$. By Lemma 4, when p has an ABP of size s, then the coefficients a_i can be computed by ABPs of size $s' = s(d+1)$. We observe that then the coefficients of the derivative polynomial $\frac{\partial p}{\partial x_1} = \sum_{i=1}^{d} i a_i x_1^{i-1}$ have ABPs of size $s'+1$.

We repeat taking derivatives $k = e-1$ times and get $\frac{\partial^k p}{\partial x_1^k}$, which has the irreducible factor q with multiplicity one, as desired.

The coefficients of $\frac{\partial^k p}{\partial x_1^k}$ can be computed by ABPs of size $s'+1$. This yields an ABP of size $\mathrm{poly}(s)$ that computes $\frac{\partial^k p}{\partial x_1^k}$.

3.4 Transforming to a monic polynomial

Given any polynomial $p(\boldsymbol{z})$ in variables $\boldsymbol{z} = (z_1, \ldots, z_n)$, there is a standard trick to make it monic in a new variable x by applying a linear transformation on the variables: for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$, let

$$\tau_{\boldsymbol{\alpha}} : \quad z_i \mapsto \alpha_i x + z_i,$$

for $i = 1, \ldots, n$. Let $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ be the resulting polynomial. Note that p and $p_{\boldsymbol{\alpha}}$ have the same degree. We show that $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ is monic in x, for a random transformation $\tau_{\boldsymbol{\alpha}}$.

▶ **Lemma 7** (Transformation to monic). *Let $p(\boldsymbol{z})$ be polynomial of total degree d. Let $S \subseteq \mathbb{F}$ be a finite set. For $\boldsymbol{\alpha} \in S^n$ picked independently and uniformly at random,*

$$\Pr[\, p_{\boldsymbol{\alpha}}(x, \boldsymbol{z}) \text{ is monic in } x \,] \geq 1 - \frac{d}{|S|}.$$

Proof. Consider the terms of degree d in p. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ such that $|\boldsymbol{\beta}| = \sum_{i=1}^n \beta_i = d$. We denote the term $\boldsymbol{z}^{\boldsymbol{\beta}} = z_1^{\beta_1} \cdots z_n^{\beta_n}$. Then the homogeneous component of degree d in p can be written as $a_d(\boldsymbol{z}) = \sum_{|\boldsymbol{\beta}|=d} c_{\boldsymbol{\beta}} \boldsymbol{z}^{\boldsymbol{\beta}}$. Note that a_d is a nonzero polynomial.

Now consider the transformed polynomial $p_{\boldsymbol{\alpha}}$. We have $\deg_x(p_{\boldsymbol{\alpha}}) = d$ and the coefficient of x^d in $p_{\boldsymbol{\alpha}}$ is $a_d(\alpha) = \sum_{|\boldsymbol{\beta}|=d} c_{\boldsymbol{\beta}} \boldsymbol{\alpha}^{\boldsymbol{\beta}}$. When we pick $\boldsymbol{\alpha}$ at random, $a_d(\alpha)$ will be a nonzero constant with probability $\geq 1 - \frac{d}{|S|}$ by the PIT-Theorem, and in this case $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$ is monic in x. ◀

Given an ABP of size s that computes $p(\boldsymbol{z})$, we can construct another ABP of size $3s$ that computes $p_{\boldsymbol{\alpha}}(x, \boldsymbol{z})$. For the new ABP replace edge labeled by z_i by the ABP computing $\alpha_i x + z_i$. For each old edge, this requires adding two new edges with labels α_i and x.

3.5 Handling the starting point of Hensel lifting

After doing the above pre-processing steps on the given polynomial $p(\boldsymbol{z})$, we call the transformed polynomial $f(x, \boldsymbol{z})$. We can assume that f of degree d can be factorized as $f = gh$, where g and h are coprime and g is irreducible. In the first step of Hensel lifting, we factorize the univariate polynomial $f(x, 0, \ldots, 0) \equiv f(x, \boldsymbol{z}) \pmod{\boldsymbol{z}}$. Now, clearly we have the factorization $f(x, 0, \ldots, 0) = g(x, 0, \ldots, 0)\, h(x, 0, \ldots, 0)$, but these two factors might not be coprime. In this case we do another transformation.

▶ Remark. Although it would suffice for our purpose to start with two coprime factors, the transformation below produces one irreducible factor.

Let g_0 be an irreducible factor of $g(x, 0, \ldots, 0)$. Then we have for some univariate polynomial $h_0'(x)$ and for $h_0(x) = h_0'(x)\, h(x, 0, \ldots, 0)$,

$$g \equiv g_0\, h_0' \pmod{\boldsymbol{z}},$$
$$f \equiv g_0\, h_0 \pmod{\boldsymbol{z}}.$$

We want that g_0 is coprime to h_0' and h_0. Directly, this might *not* be the case because all factors of $f(x, 0, \ldots, 0)$ might have multiplicity > 1. However, we argue how to ensure this after a random shift $\boldsymbol{\alpha}$ of f. That is, we consider the function $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$

1. First, we show how to achieve that g_0 is coprime to h_0'.
   Since g is irreducible, it is also square-free, and hence, $\gcd(g, \frac{\partial g}{\partial x}) = 1$. By Lemma 2, the resultant $r(\boldsymbol{z}) = \mathrm{Res}_x(g, \frac{\partial g}{\partial x})$ is a polynomial of degree $\leq 4d^2$ and $r(\boldsymbol{z}) \neq 0$. Hence, at a

random point $\boldsymbol{\alpha} \in [8d^2]^n$, we have $r(\boldsymbol{\alpha}) \neq 0$ with high probability. At such a point $\boldsymbol{\alpha}$, we have that $g(x, \boldsymbol{\alpha})$ is square-free. Therefore, $g(x, \boldsymbol{z})$ is square-free modulo $(\boldsymbol{z} - \boldsymbol{\alpha})$, or, equivalently, $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ is square-free modulo $\boldsymbol{z}$. Hence, when we define g_0 and h'_0 from $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ instead of $g(x, \boldsymbol{z})$, they will be coprime.

2. Similarly, we can achieve that g_0 is coprime to h_0. By the first item, it now suffices to get g_0 coprime to $h(x, 0, \ldots, 0)$.
   For showing this, we use that g_0 is coprime to h'_0 and prove that $g(x, 0, \ldots, 0)$ is coprime to $h(x, 0, \ldots, 0)$. Consider the resultant of g and h w.r.t. x, the polynomial $r'(\boldsymbol{z}) = \mathrm{Res}_x(g, h)$ has degree $\leq 4d^2$. Since g and h are coprime, $r'(\boldsymbol{z}) \neq 0$. Hence, at a random point $\boldsymbol{\alpha} \in [8d^2]^n$, we have $r'(\boldsymbol{\alpha}) \neq 0$ with high probability, and hence $g(x, \boldsymbol{\alpha})$ and $h(x, \boldsymbol{\alpha})$ are coprime univariate polynomials. Therefore, $g(x, \boldsymbol{z})$ and $h(x, \boldsymbol{z})$ are coprime modulo $(\boldsymbol{z} - \boldsymbol{\alpha})$, or, equivalently, $g(x, \boldsymbol{z} + \boldsymbol{\alpha})$ and $h(x, \boldsymbol{z} + \boldsymbol{\alpha})$ are coprime modulo $\boldsymbol{z}$.

Combining the two items, a random point $\boldsymbol{\alpha} \in [8d^2]^n$ will fulfill both properties with high probability. So instead of factoring $f(x, \boldsymbol{z})$, we do a coordinate transformation $\boldsymbol{z} \mapsto \boldsymbol{z} + \boldsymbol{\alpha}$ and factor $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$ instead. From these factors, we easily get the factors of $f(x, \boldsymbol{z})$ by inverting the transformation. Note also that when $f(x, \boldsymbol{z})$ is monic in x, the same holds for $f(x, \boldsymbol{z} + \boldsymbol{\alpha})$.

In the next section, we do another transformation on the input polynomial. We apply a map on the variables that maps x to x and z_i is mapped to yz_i, for a new variable y and $i = 1, \ldots, n$. Then we factorize the transformed polynomial modulo y. Note that in this case, going modulo y has the same effect of going modulo $\boldsymbol{z}$. So we can use the above argument to ensure the starting condition for Hensel lifting is satisfied.

3.6 Reducing multivariate factoring to the bivariate case

Factoring multivariate polynomials can be reduced to the case of *bivariate* polynomials (see [12]). Let x, y and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be variables and let $f(x, \boldsymbol{z})$ be the given polynomial. With $f \in \mathbb{F}[x, \boldsymbol{z}]$, we associate the polynomial $\widehat{f} \in \mathbb{F}[x, y, \boldsymbol{z}]$ defined by

$$\widehat{f}(x, y, \boldsymbol{z}) = f(x, yz_1, \ldots, yz_n)\,.$$

The point now is to consider $\widehat{f}$ as a polynomial in $\mathbb{F}[\boldsymbol{z}][x, y]$, that is, as a bivariate polynomial in x and y with coefficients in $\mathbb{F}[\boldsymbol{z}]$. We list some properties.

1. $f(x, \boldsymbol{z}) = \widehat{f}(x, 1, \boldsymbol{z})$,
2. $\deg(\widehat{f}) \leq 2\deg(f)$,
3. f monic in $x \implies \widehat{f}$ monic in x,
4. $f = gh \implies \widehat{f} = \widehat{g}\,\widehat{h}$,
5. $\widehat{f} = g'\,h' \implies f = g'(x, 1, \boldsymbol{z})\,h'(x, 1, \boldsymbol{z})$.

By property 4, factors of f yield factors of $\widehat{f}$. The following lemma shows that also the irreducibility of the factors is maintained.

▶ **Lemma 8.** *Let f be monic in x and g be a non-trivial irreducible factor of f. Then $\widehat{g}$ is a non-trivial irreducible factor of $\widehat{f}$.*

Proof. By property 4 above, $\widehat{g}$ is a factor of $\widehat{f}$. We argue that $\widehat{g}$ is irreducible.

Let $\widehat{g} = uv$ be a factorization of $\widehat{g}$. By item 5 above, this yields a factorization of g as $g = u(x, 1, \boldsymbol{z})\,v(x, 1, \boldsymbol{z})$. Since g is monic in x, the same holds for $\widehat{g}$, and therefore also for factors u, v. Hence, either u or v must be constant, because otherwise they would provide a non-trivial factorization of g. ◀

Thus, to get an ABP for an irreducible factor g of f, first we show that there is an ABP for the irreducible factor $\widehat{g}$. This yields an ABP for g by substituting $g = \widehat{g}(x, 1, \boldsymbol{z})$.

Given an ABP B_f of size s for f, we get an ABP $B_{\widehat{f}}$ for $\widehat{f}$ by putting an edge labeled y in series with every edge labeled z_i in B_f, so that $B_{\widehat{f}}$ computes yz_i at every place where B_f uses z_i. Hence, the size of $B_{\widehat{f}}$ is at most $2s$.

3.7 Solving a linear system with polynomials as matrix entries

We show how to solve a linear system $M\boldsymbol{v} = 0$ for a polynomial matrix M with entries from $\mathbb{F}[\boldsymbol{z}]$ given as ABPs. We are seeking for a nonzero vector $\boldsymbol{v}$. Note that such a $\boldsymbol{v}$ exists over the ring $\mathbb{F}[\boldsymbol{z}]$ iff it exists over the field $\mathbb{F}(\boldsymbol{z})$.

Except for minor modifications, this follows from classical linear algebra. Kopparty, Saraf, and Shpilka [12, Lemma 2.6] have shown the same result for circuits. The proof works as well for ABPs.

▶ **Lemma 9** (Solving linear systems [12]). *Let $M = (m_{i,j}(\boldsymbol{z}))_{i,j}$ be a polynomial matrix of dimension $k \times m$ and variables $\boldsymbol{z} = (z_1, \ldots, z_n)$, where the entries are polynomials $m_{i,j} \in \mathbb{F}[\boldsymbol{z}]$ that can be computed by ABPs of size s.*

Then there is an ABP of size $\mathrm{poly}(k, m, s)$ *computing a nonzero vector $\boldsymbol{v} \in \mathbb{F}[\boldsymbol{z}]^m$ such that $M\boldsymbol{v} = 0$ (if it exists).*

Proof. After swapping rows of M, we ensure that the $j \times j$ submatrix M_j that consists of the first j rows and the first j columns has full rank, iteratively for $j = 1, 2, \ldots$.

For $j = 1$ this means to find a nonzero entry in the first column and swap that row with the first row. If the first column is a zero-column, then $\boldsymbol{v} = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^T$ is a solution and we are done. To extend from j to $j+1$, suppose we have ensured that M_j has full rank. Now we search for a row from row $j+1$ on, such that after a swap with row $j+1$, the submatrix M_{j+1} has full rank. This can be tested by Lemma 3. If no such row exists, then the process stops at j. If $j = m$ then M has full rank and the zero vector is the only solution. Otherwise, assume the above process stops with $j < m$.

Now Cramer's rule can be used to find the unique solution $\boldsymbol{u} = \begin{pmatrix} u_1 & u_2 & \cdots & u_j \end{pmatrix}^T$ of the system

$$M_j \boldsymbol{u} = \begin{pmatrix} m_{1,j+1} & m_{2,j+1} & \cdots & m_{j,j+1} \end{pmatrix}^T .$$

We have $u_i = \frac{\det M_j^i}{\det M_j}$, where M_j^i is the matrix obtained by replacing the i-th column of M_j by the vector $\begin{pmatrix} m_{1,j+1} & \cdots & m_{j,j+1} \end{pmatrix}^T$. Now, define

$$\boldsymbol{v} = \begin{pmatrix} \det M_j^1 & \det M_j^2 & \cdots & \det M_j^j & -\det M_j & 0 & \cdots & 0 \end{pmatrix}^T .$$

Then $\boldsymbol{v}$ is a solution to the original system. Its entries are determinants of matrices with entries computed by ABPs of size s. Hence, all the entries of $\boldsymbol{v}$ have ABPs of size $\mathrm{poly}(k, m, s)$. ◀

4 Factors of Arithmetic Branching Programs

In this section, we prove that ABPs are closed under factoring.

▶ **Theorem 10.** *Let p be a polynomial over a field $\mathbb{F}$ with characteristic 0. For all factors q of p, we have*

$$\mathrm{size}_{\mathrm{ABP}}(q) \leq \mathrm{poly}(\mathrm{size}_{\mathrm{ABP}}(p)) .$$

We prove Theorem 10 in the rest of this section. First observe that it suffices to prove the poly(s) size upper bound for the irreducible factors of p. This yields the same bound for all the factors. The case when $p = q^e$ is proved in Section 3.2. So it remains to consider the general case when $p = p_1^{e_1} \cdots p_m^{e_m}$, for $m \geq 2$, where $p_1, \ldots, p_m$ are the different irreducible factors of p. We want to prove an ABP size upper bound for an irreducible factor, say p_1.

We start by several transformations on the input polynomial $p(\boldsymbol{z})$, where $\boldsymbol{z} = (z_1, \ldots, z_n)$.

1. As described in Section 3.3, taking $k = e_1 - 1$ times the derivative w.r.t. some variable, say z_1, we get the polynomial $p'(\boldsymbol{z}) = \frac{\partial^k p(\boldsymbol{z})}{\partial z_1^k}$, where the factor p_1 has multiplicity 1.
2. Next, by Lemma 7, we transform $p'(\boldsymbol{z})$ to a polynomial $p''(x, \boldsymbol{z})$ that is monic in x, for a new variable x. Thereby also the factors of $p'(\boldsymbol{z})$ are transformed, maintaining their irreducibilty and multiplicity. The degree of p'' is twice the degree of p'.
3. At this point, we may have to shift the variables $\boldsymbol{z}$ as described in Section 3.5 to ensure the properties needed for starting the Hensel lifting. This shift preserves the monicness and the irreducibility of the factors.
4. Finally, the transformation to a bivariate polynomial is explained in Section 3.6. This yields polynomial $p'''(x, y, \boldsymbol{z})$, with new variable y and monic in x. We rewrite p''' as a polynomial in x and y with coefficients in the ring $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$ and call the representation f. That is, $f(x, y) \in \mathbb{K}[x, y]$. By Lemma 8, the tranformation maintains irreducible factors. Note also that by the definition of p''' we have $f(x, 0) = p'''(x, 0, 0, \ldots, 0) = f(x, y) \bmod y$, so that $f(x, y) \bmod y$ is univariate.

The main part now is to factor $f(x, y) \in \mathbb{K}[x, y]$, say $f = gh$, where $g \in \mathbb{K}[x, y]$ is irreducible and coprime to $h \in \mathbb{K}[x, y]$, and f, g, h are monic in x and have x-degree ≥ 1. Let d be the total degree of f in x, y.

From the factor g of f, we will recover the factor f_1 of p by reversing the above transformations. We show that g can be computed by an ABP of size poly(s). It follows that the irreducible factor f_1 has an ABP of size poly(s).

The basic strategy is to first factor the univariate polynomial $f \bmod y$, and then apply Hensel lifting to get a factorization of $f \bmod y^t$, for large enough t. Finally, from the lifted factors modulo y^t, we compute the absolute factors of f.

4.1 Hensel lifting

There are various versions of Hensel lifting in the literature (see for example [18]). In our case, an ABP should be able to perform several iterations of the lifting. Therefore we use the lifting in a way suitable for ABPs. In particular, in contrast to other presentations, we will *not* maintain the monicness of the lifted factors.

Hensel lifting works over rings $\mathcal{R}$ modulo an ideal $\mathcal{I} \subseteq \mathcal{R}$. In our case, $\mathcal{R} = \mathbb{K}[x, y]$, where $\mathbb{K} = \mathbb{F}[\boldsymbol{z}]$, and $\mathcal{I} = \langle y \rangle^t$, for some $t \geq 1$.

▶ **Definition 11** (Lifting). *Let $\mathcal{R}$ be a ring and $\mathcal{I} \subseteq \mathcal{R}$ be an ideal. Let $f, g, h, a, b \in R$ such that $f \equiv gh \pmod{\mathcal{I}}$ and $ag + bh \equiv 1 \pmod{\mathcal{I}}$. Then we call $g', h' \in \mathcal{R}$* a lift of g, h with respect to f, *if*

- **(i)** $f \equiv g'h' \pmod{\mathcal{I}^2}$,
- **(ii)** $g' \equiv g \pmod{\mathcal{I}}$ *and* $h' \equiv h \pmod{\mathcal{I}}$, *and*
- **(iii)** $\exists a', b' \in \mathcal{R} \quad a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$.

▶ Remark. The three conditions in Definition 11 are the *invariants* when iterating the lifting.

Note that the last condition is actually redundant. It follows from the assumptions together with the second condition. This can be seen in the proof of Lemma 12 below, where a lift g', h' from g, h is constructed, together with a', b'. When we show that condition (iii) holds, we do *not* use the specific form of g', h' constructed there, it suffices to have condition (ii).

▶ **Lemma 12** (Hensel Lifting). *Let $\mathcal{R}$ be a ring and $\mathcal{I} \subseteq \mathcal{R}$ be an ideal. Let $f, g, h, a, b \in R$ such that $f \equiv gh \pmod{\mathcal{I}}$ and $ag + bh \equiv 1 \pmod{\mathcal{I}}$. Then we have:*

1. (Existence). *There exists a lift g', h' of g, h w.r.t. f.*
2. (Uniqueness). *For any other lift g^*, h^* of g, h w.r.t. f, there exists a $u \in \mathcal{I}$ such that*

$$g^* \equiv g'(1+u) \pmod{\mathcal{I}^2} \quad \text{and} \quad h^* \equiv h'(1-u) \pmod{\mathcal{I}^2}.$$

Proof. We first show the existence part. Let

1. $e = f - gh$,
2. $g' = g + be$ and $h' = h + ae$,
3. $c = ag' + bh' - 1$,
4. $a' = a(1-c)$ and $b' = b(1-c)$.

We verify that g', h' is a lift of g, h. Because $f \equiv gh \pmod{\mathcal{I}}$, we have $e = f - gh \equiv 0 \pmod{\mathcal{I}}$. In other words, $e \in \mathcal{I}$. It follows that $g' \equiv g \pmod{\mathcal{I}}$ and $h' \equiv h \pmod{\mathcal{I}}$.

Next we show that $f \equiv g'h' \pmod{\mathcal{I}^2}$.

$$\begin{aligned} f - g'h' &= f - (g+be)(h+ae) \\ &= f - gh - e(ag+bh) - abe^2 \\ &\equiv e - e(ag+bh) \pmod{\mathcal{I}^2} \\ &\equiv e(1-(ag+bh)) \pmod{\mathcal{I}^2} \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

In the second line, note that $e^2 \in \mathcal{I}^2$. The last equality holds because $e \in \mathcal{I}$ and $1-(ag+bh) \in \mathcal{I}$.

Now, we verify that $a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$. First, observe that

$$\begin{aligned} c &= ag' + bh' - 1 \\ &\equiv ag + bh - 1 \pmod{\mathcal{I}} \\ &\equiv 0 \pmod{\mathcal{I}} \end{aligned}$$

Hence, $c \in \mathcal{I}$ and we conclude that $a' \equiv a \pmod{\mathcal{I}}$ and $b' \equiv b \pmod{\mathcal{I}}$. Now,

$$\begin{aligned} a'g' + b'h' - 1 &= a(1-c)g' + b(1-c)h' - 1 \\ &= ag' + bh' - 1 - c(ag' + bh') \\ &= c - c(ag' + bh') \\ &= c(1-(ag'+bh')) \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

The last equality holds because $c \in \mathcal{I}$ and $1 - (ag' + bh') \equiv -c \equiv 0 \pmod{\mathcal{I}}$.

For the uniqueness part, let g^*, h^* be another lift of g, h. Let $\alpha = g^* - g'$ and $\beta = h^* - h'$. By Definition 11 (*ii*), we have $g' \equiv g \equiv g^* \pmod{\mathcal{I}}$ and $h' \equiv h \equiv h^* \pmod{\mathcal{I}}$, and therefore $\alpha, \beta \in \mathcal{I}$.

We first show

$$\beta g' + \alpha h' \equiv 0 \pmod{\mathcal{I}^2}. \tag{4}$$

$$\begin{aligned} \beta g' + \alpha h' &= \beta g' + (g^* - g')h' \\ &= \beta g' + g^* h' - g'h' \\ &\equiv \beta g' + g^* h' - g^* h^* \pmod{\mathcal{I}^2} \\ &\equiv \beta g' - \beta g^* \pmod{\mathcal{I}^2} \\ &\equiv -\alpha\beta \pmod{\mathcal{I}^2} \\ &\equiv 0 \pmod{\mathcal{I}^2} \end{aligned}$$

Define $u = a'\alpha - b'\beta$. Because $\alpha, \beta \in \mathcal{I}$, also $u \in \mathcal{I}$. Then, by (4) and because $a'g' + b'h' \equiv 1 \pmod{\mathcal{I}^2}$, we have

$$\begin{aligned} g'(1+u) &= g'(1 + (a'\alpha - b'\beta)) \\ &= g' + a'g'\alpha - b'g'\beta \\ &\equiv g' + a'g'\alpha + b'h'\alpha \pmod{\mathcal{I}^2} \\ &\equiv g' + \alpha \pmod{\mathcal{I}^2} \\ &\equiv g^* \pmod{\mathcal{I}^2}. \end{aligned}$$

Similarly, we get $h^* \equiv h'(1-u) \pmod{\mathcal{I}^2}$. ◀

For the ABP-size, recall that the size just adds up when doing additions or multiplications. Hence, when f, g, h, a, b have ABPs of size $\leq s$ and we construct ABPs for g', h', a', b' according to steps 1 - 4 in the above proof, then we get ABPs of size $O(s)$.

▶ Remark. In the *monic version* of Hensel Lifting there is a division in addition to the 4 steps from above. When we assume that g is monic, we can compute polynomials q and r such that $g' - g = qg + r$, where $\deg_x(r) < \deg_x(g)$. Then one can show that $\hat{g} = g + r$ and $\hat{h} = h'(1+q)$ are a lift of g, h w.r.t. f. Moreover, when g, h are monic, so are $\hat{g}, \hat{h}$. Also the Bézout-coefficients $\hat{a}, \hat{b}$ can be computed. For $\hat{c} = a\hat{g} + b\hat{h} - 1$, let $\hat{a} = a(1-\hat{c})$ and $\hat{b} = b(1-\hat{c})$.

The advantage of the monic version is that the result is really unique. There is no $1+u$ factor between monic lifts. The disadvantage is the extra division which would blow up the ABP-size too much.

4.2 Iterating Hensel lifting

Let $f = gh$, where g is irreducible and coprime to h, and f, g, h are monic in x with x-degree ≥ 1.

To start the Hensel lifting procedure, we factor the univariate polynomial $f(x,0) = f \bmod y$ as $f(x,0) = g_0(x)\, h_0(x)$, where g_0 is a divisor of $g \bmod y$, and coprime to h_0, and $\deg_x(g_0) \geq 1$. Recall that by the pre-processing in Section 3.5, we may assume that there is such a decomposition of $f(x,0)$.

By the Euclidian algorithm, there are polynomials $a_0(x), b_0(x)$ such that $a_0 g_0 + b_0 h_0 = 1$. Hence, for $\mathcal{I}_0 = \langle y \rangle$, we have $a_0 g_0 + b_0 h_0 \equiv 1 \pmod{I_0}$ and initiate Hensel lifting with

$$f \equiv g_0 h_0 \pmod{\mathcal{I}_0}.$$

We iteratively apply Hensel lifting to g_0, h_0 as described in the proof of Lemma 12. Each time, the ideal gets squared. Let $\mathcal{I}_k = \mathcal{I}_0^{2^k}$. That is, we get polynomials g_k, h_k such that

$$f \equiv g_k h_k \pmod{\mathcal{I}_k},$$

and g_k, h_k is a lift of g_{k-1}, h_{k-1} w.r.t. f. The following lemma states that g_k divides g modulo $\mathcal{I}_k$, for all $k \geq 0$. In a sense, the g_k's approximate g modulo increasing powers of y.

▶ **Lemma 13.** *With the notation from above, for all $k \geq 0$ and some polynomial h'_k,*

$$g \equiv g_k h'_k \pmod{\mathcal{I}_k} \quad \textit{and} \quad h_k \equiv h\, h'_k \pmod{\mathcal{I}_k}.$$

Moreover, g_k, h'_k is a lift of g_{k-1}, h'_{k-1} w.r.t. g and $\deg_x(h'_k) \leq \deg_x(h_k) = 2^{O(k)}$, for $k \geq 1$.

Proof. The proof is by induction on $k \geq 0$. For the base case, we have that g_0 divides g modulo I_0, as explained above. Thus, for some polynomial h'_0 that is coprime to g_0, we have

$$g \equiv g_0 h'_0 \pmod{\mathcal{I}_0},$$

Hence, we have $h_0 \equiv h'_0 h \pmod{\mathcal{I}_0}$. Note that h'_0 can be just 1.

For the inductive step, assume that

$$g \equiv g_{k-1} h'_{k-1} \pmod{\mathcal{I}_{k-1}} \quad \text{and} \quad h_{k-1} \equiv h\, h'_{k-1} \pmod{\mathcal{I}_{k-1}}. \tag{5}$$

Let g'_k, h''_k be a lift of g_{k-1}, h'_{k-1} w.r.t. g, so that in particular

$$g'_k h''_k \equiv g \pmod{\mathcal{I}_k}. \tag{6}$$

We claim that then $g'_k,\ h\, h''_k$ is a lift of $g_{k-1},\ h\, h'_{k-1}$, i.e., of g_{k-1}, h_{k-1} by (5), w.r.t. f.

▷ Claim 14. $g'_k,\ h\, h''_k$ is a lift of g_{k-1}, h_{k-1} w.r.t. f.

Proof. We check the three conditions for a lift in Definition 11. For the product condition (i), we have by (6)

$$g'_k\, h\, h''_k = (g'_k\, h''_k)\, h \equiv gh \pmod{\mathcal{I}_k}.$$

For condition (ii), we have $g'_k \equiv g_{k-1} \pmod{\mathcal{I}_{k-1}}$ by assumption and similarly

$$h\, h''_k \equiv h\, h'_{k-1} \equiv h_{k-1} \pmod{\mathcal{I}_{k-1}}.$$

By the remark after Definition 11, the condition (iii) already follows now. This proves Claim 14. ◁

Recall that also g_k, h_k is a lift of g_{k-1}, h_{k-1}. Hence, by the uniqueness property of Hensel lifting, there is a $u \in \mathcal{I}_{k-1}$ such that

$$g'_k \equiv g_k\,(1+u) \pmod{\mathcal{I}_k} \quad \text{and} \quad h\, h''_k \equiv h_k\,(1-u) \pmod{\mathcal{I}_k} \tag{7}$$

Now observe that we can move the factor $1+u$: we have that $g_k\,(1+u),\ h\, h''_k$ is a lift of g_{k-1}, h_{k-1}, then also $g_k,\ h\, h''_k\,(1+u)$ is a lift of g_{k-1}, h_{k-1}.

▷ Claim 15. $g_k,\ h\, h''_k\,(1+u)$ is a lift of g_{k-1}, h_{k-1} w.r.t. f.

Proof. We check the conditions for a lift in Definition 11. The first two of them are trivial: moving the factor $1+u$ clearly does not change the product. Because $u \in \mathcal{I}_{k-1}$ we still have the equality with the factors g_{k-1} and h_{k-1} modulo $\mathcal{I}_{k-1}$, respectively.

By the remark after Definition 11, the third condition already follows, but it also easy to check now:

Let $a, b \in \mathcal{R}$ such that $ag_k + bh_k \equiv 1 \pmod{\mathcal{I}_k}$. It follows by (7) that

$$ag_k + bh\, h''_k(1+u) \equiv ag_k + bh_k(1-u)(1+u) \equiv ag_k + bh_k(1-u^2) \equiv 1 \pmod{\mathcal{I}_k}.$$

This proves Claim 15. ◁

Now, define $h'_k = h''_k(1+u)$. Note that

$$h'_k \equiv h''_k \equiv h'_{k-1} \pmod{\mathcal{I}_{k-1}}. \tag{8}$$

By (7), we have

$$h\,h'_k \equiv h\,h''_k\,(1+u) \equiv h_k(1-u)(1+u) \equiv h_k \pmod{\mathcal{I}_k}. \tag{9}$$

By (9) we have

$$f = gh \equiv g_k h_k \equiv g_k h\,h'_k \pmod{\mathcal{I}_k}. \tag{10}$$

It follows from (10) that $gh \equiv g_k h'_k h \pmod{\mathcal{I}_k}$. Now we want to cancel h in the last equation and conclude that $g \equiv g_k h'_k \pmod{\mathcal{I}_k}$. This we can do because h is monic in x, it does not contain a factor y, i.e. $h \notin \mathcal{I}_0$. Hence, together with (8), we conclude that g_k, h'_k is a lift of g_{k-1}, h'_{k-1} w.r.t. g.

For the x-degree of h'_k, consider the equation $h_k \equiv h\,h'_k \pmod{\mathcal{I}_k}$. Since h is monic in x the highest x-degree term in the product $h\,h'_k$ will survive the modulo operation. Therefore $\deg_x(h'_k) \leq \deg_x(h) + \deg_x(h'_k) = \deg_x(h_k)$.

To bound the degree of h_k observe that in each round of the Hensel lifting, the maximum possible degree is bounded by a constant multiple (≤ 5) of the maximum degree in the previous round. After k iterations, the degree of the lifted factors is therefore bounded by $2^{O(k)}$. ◀

For the ABP-size, we observed at the end of Section 4.1 that the size increases by a constant factor in one iteration. Hence, after k iterations, the size increases by a factor $2^{O(k)}$.

4.3 Factor reconstruction for ABP

We show how to get the absolute factor g of f from the lifted factor. This is called the *jump step* in Sudan's lecture notes [18]. The difference to the earlier presentations is that our lifted factor might not be monic.

Let $f = gh$, where f has degree d, factor g is irreducible and coprime to h, and f, g, h are monic in x. In the previous section, we started with a factorization $f \equiv g_0 h_0 \pmod{\mathcal{I}_0}$, where g_0 is irreducible and coprime to h_0. Moreover, $g \equiv g_0 h'_0 \pmod{\mathcal{I}_0}$, for some h'_0 such that $h_0 = h\,h'_0 \pmod{\mathcal{I}_0}$.

Then we apply Hensel lifting, say t-times, for some t to be determined below. By Lemma 13, we get a factorization $f \equiv g_t h_t \pmod{\mathcal{I}_t}$ such that

$$g \equiv g_t h'_t \pmod{\mathcal{I}_t}, \tag{11}$$

for some h'_t such that $h_t \equiv h\,h'_t \pmod{\mathcal{I}_t}$.

Equation (11) gives us a relation between the known g_t and the unknown g, via the unknown h'_t. We set up a linear system of equations to find a polynomial $\tilde{g} \in \mathbb{K}[x, y]$ that is monic in x and has minimal degree in x such that

$$\tilde{g} \equiv g_t \tilde{h} \pmod{\mathcal{I}_t}, \tag{12}$$

for some polynomial $\tilde{h}$. We give some more details to the linear system below, after the next lemma. Before, we show that $\tilde{g}$ is indeed the factor we were looking for, for large enough t.

▶ **Lemma 16.** *$\tilde{g} = g$, for $t \geq \log(4d^2 + 1)$.*

Proof. Consider the resultant $r(y) = \mathrm{Res}_x(g, \tilde{g})$. We show that $r(y) = 0$. Then it follows from Lemma 2 that g and $\tilde{g}$ share a common factor with positive x-degree. Since g is irreducible it must be a divisor of $\tilde{g}$. Since both of them are monic and have the same x-degree, we get equality $\tilde{g} = g$, up to constant multiples.

To argue that $r(y) = 0$, recall from Lemma 2 that the resultant can be written as $r(y) = ug + v\tilde{g}$, for some polynomials u and v. By (11) and (12), we have

$$ug + v\tilde{g} \equiv g_t(uh'_t + v\tilde{h}) \pmod{\mathcal{I}_t}$$

Consider g_t and $w = uh'_t + v\tilde{h}$ as polynomials in y with coefficients in x. Suppose $g_t = c_0(x) + c_1(x)y + \ldots + c_{d'}(x)y^{d'}$. By the properties of Hensel lifting, we have $g_t \equiv g_0 \pmod{\mathcal{I}_0}$, and therefore $c_0(x) = g_0(x)$. Recall that g_0 is non-constant, $\deg(g_0) \geq 1$.

Let $j \geq 0$ be the least power of y that appears in w and let its coefficient be $w_j(x)$. Suppose for the sake of contradiction that $j < 2^t$. Then the least power of y in $g_t w$ is also j, and its coefficient is $g_0(x)w_j(x)$, which is a nonzero polynomial in x.

The monomials present in $g_0(x)w_j(x)y^j$ cannot be canceled by other monomials in $g_t w$ because they have larger y-degree. It follows that $g_t w \bmod \mathcal{I}_t$ is not free of x. On the other hand, $r(y) \equiv g_t w \pmod{\mathcal{I}_t}$ and $r(y) \in \mathbb{K}[y]$ is a polynomial with no variable x. This is a contradiction.

We conclude that $j \geq 2^t$, which means that $w \equiv 0 \pmod{\mathcal{I}_t}$. Hence, we get $r(y) \equiv 0 \pmod{\mathcal{I}_t}$. Recall that $\deg(r) \leq 4d^2$. Now we choose $t \geq \log(4d^2 + 1)$. Then we can conclude that indeed $r(y) = 0$. ◀

Details for setting up the linear system. Equation (12) can be used to set up a homogeneous system of linear equations. For the degree bounds of the polynomials, let $d_x = \deg_x(g)$ and $d_y = \deg_y(g)$. We may assume that we know d_x and d_y. Let $D_x = \deg_x(g_t)$ and $D_y = 4d^2$, where $d = \deg(f)$. Let $D'_x = \deg_x(h_t)$. Recall from Lemma 13 that $\deg_x(\tilde{h}) \leq D'_x$. Let

$$\begin{aligned} g_t &= \sum_{i \leq D_x, j \leq D_y} c_{i,j}\, x^i y^j, \\ \tilde{g} &= x^{d_x} + \sum_{i < d_x, j \leq d_y} r_{i,j}\, x^i y^j, \\ \tilde{h} &= \sum_{i \leq D'_x, j \leq D_y} s_{i,j}\, x^i y^j, \end{aligned}$$

where the coefficients $c_{i,j}, r_{i,j}, s_{i,j}$ are polynomials in the variables $z_1, \ldots, z_n$. To ensure that $\tilde{g}$ is monic, we set the coefficient of x^{d_x} in $\tilde{g}$ to be 1.

Note that we have an ABP that computes g_t. Hence, there are ABPs for computing the coefficients $c_{i,j}$ of g_t by Lemma 4. The coefficients $r_{i,j}, s_{i,j}$ of $\tilde{g}$ and $\tilde{h}$ we treat as unknowns. Equation (12) now becomes

$$\sum_{i \leq d_x, j \leq d_y} r_{i,j} x^i y^j \equiv \sum_{i \leq D_x, j \leq D_y} c_{i,j} x^i y^j \sum_{i \leq D'_x, j \leq D_y} s_{i,j} x^i y^j \pmod{y^{2d^2+1}} \tag{13}$$

Now we equate the coefficients of the monomials $x^k y^l$ on both sides in (13). Then we get $(D_x + D'_x + 1)(D_y + 1)$ homogeneous linear equations in $d_x(d_y + 1) + (D'_x + 1)(D_y + 1)$ many unknowns $r_{i,j}$ and $s_{i,j}$. This system can be expressed in the form $M\boldsymbol{v} = 0$, for a matrix M and unknown vector $\boldsymbol{v}$.

By Lemma 9, an ABP can efficiently compute a solution vector $\boldsymbol{v}$ of polynomials from $\mathbb{F}[\boldsymbol{z}]$. Note that by (11), a nontrivial solution is guaranteed to exist. From $\boldsymbol{v}$, we get the coefficients $r_{i,j}$ of $\tilde{g}$, and hence of the factor g.

4.4 Size Analysis

We summarize the bound on the ABP-size of the factor computed. Given polynomial p of degree d_p and $\text{size}_{\text{ABP}}(p) = s$. We have seen that the pre-processing transformations yield a polynomial f of degree $d_f \leq 2d_p$ and $\text{size}_{\text{ABP}}(f) = \text{poly}(s)$. Then we do $t = \log(2d_f^2 + 1)$ iterations of Hensel lifting. The initial polynomials f_0, g_0, h_0 have ABP-size bound by $2d_f$. Hence, the polynomials after the last iteration have ABP-size bounded by $2^t \text{poly}(s) = \text{poly}(s, d_p) = \text{poly}(s)$.

From the lifted factor we construct the actual factor of f. This step involves solving a linear system. We argued that the resulting polynomial g has ABP-size $\text{poly}(s)$.

Finally, we reverse the transformations from the beginning and get a factor of p that has an ABP of size $\text{poly}(s)$. This finishes the proof of Theorem 10.

5 Applications

5.1 Root Finding

Given a polynomial $p \in \mathbb{F}[x, \boldsymbol{y}]$, the *root finding* problem asks for a polynomial $r \in \mathbb{F}[\boldsymbol{y}]$ such that $p(r(\boldsymbol{y}), \boldsymbol{y}) = 0$. By a lemma of Gauß, r is a root of p iff $x - r(\boldsymbol{y})$ is an irreducible factor of p. By Theorem 10, when p is given by an ABP, we get an ABP for $x - r(\boldsymbol{y})$. Setting $x = 0$ and inverting the sign gives an ABP for $r(\boldsymbol{y})$.

▶ **Corollary 17.** *The solutions of the root finding problem for a polynomial p given by an ABP can be computed by ABPs of size* $\text{poly}(\text{size}_{\text{ABP}}(p))$.

5.2 Hardness vs. Randomness

As an application of Theorem 10, we get that lower bounds for ABPs imply a black-box derandomization of polynomial identity tests (PIT) for ABPs, similar to the result of Kabanets and Impagliazzo [8, Theorem 7.7] for arithmetic circuits.

▶ **Theorem 18** (Hitting-set from hard polynomial)**.** *Let $\{q_m\}_{m \geq 1}$ be a multilinear polynomial family such that q_m is computable in time $2^{O(m)}$, but has no ABP of size $2^{o(m)}$. Then one can compute a hitting set for ABPs of size s in time $s^{O(\log s)}$.*

The proof is similar to the proof given by Kabanets and Impagliazzo [8, Theorem 7.7] for circuits. At one point, they invoke Kaltofen's factor result for circuits. This can be replaced now by Theorem 10 for ABPs. Finally, from a given ABP of size s with n variables, we can get another ABP of size $\text{poly}(s)$ and $\log n$ variables by replacing the original variables by a hitting set generator, n polynomials computed by small size ABPs. This final composition step also goes through for ABPs. We will give more details in the final version of the paper.

6 Conclusion and Open Problems

We prove that the class of polynomials computed by ABPs is closed under factors. As a direct corollary, we get that the gcd of two polynomials computed by small-sized ABPs has small ABP size.

Our proof seems not to extend to the model of arithmetic formulas. The bottleneck is the last step, as the determinant of a symbolic matrix $(x_{i,j})_{n \times n}$ may not have $\text{poly}(n)$ size formulas. One way to avoid computing the determinant is by making the lifted factor monic

in each round of Hensel lifting. But the direct implementation of monic Hensel lifting leads to a quasi-poly blow-up of formula size because it involves polynomial division in each step. So the closure of formulas under factors remains an open problem.

If one could show that arithmetic formulas are *not* closed under factors, i.e. if some polynomial $f(x_1, \ldots, x_n)$ exists that requires formula of size $\geq n^{\log n}$, but has a nonzero multiple of formula-size $\text{poly}(n)$, then, by our result, VF would be separated from VBP and by Kaltofen's result, VF would be separated from VP.

Besides arithmetic formulas, there are other models for which $\text{poly}(s, d)$ upper bound on the size of factors are not known. For example, read-once oblivious arithmetic branching programs (ROABP) and constant depth arithmetic circuits.

References

1 Peter Bürgisser. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004.

2 Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7 of *Algorithms and Computation in Mathematic*. Springer, 2013.

3 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure of VP under taking factors: a short and simple proof. Technical Report arXiv:1903.02366, arXiv, 2019. `arXiv:1903.02366`.

4 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing*, 15(13):1–34, 2019. `doi:10.4086/toc.2019.v015a013`.

5 Pranjal Dutta. Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity. Master's thesis, Chennai Mathematical Institute, 2018.

6 Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the roots: Uniform closure results for algebraic classes under factoring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1152–1165. ACM, 2018.

7 Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof complexity lower bounds from algebraic circuit complexity. In *Proceedings of the 31st Conference on Computational Complexity (CCC)*, page 32. LIPIcs, 2016.

8 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pages 355–364. ACM, 2003.

9 Erich Kaltofen. Uniform closure properties of p-computable functions. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 330–337, 1986.

10 Erich Kaltofen. Single-factor Hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the 19th annual ACM Symposium on Theory of Computing (STOC)*, pages 443–452. ACM, 1987.

11 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989.

12 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015.

13 Mrinal Kumar and Ramprasad Saptharishi. Hardness-randomness tradeoffs for algebraic computation. *Bulletin of EATCS*, 3(129), 2019.

14 Meena Mahajan. Algebraic complexity classes. In *Perspectives in Computational Complexity*, pages 51–75. Springer, 2014.

15 Meena Mahajan and V Vinay. Determinant: Old algorithms, new insights. *SIAM Journal on Discrete Mathematics*, 12(4):474–490, 1999.

16 Rafael Oliveira. Factors of low individual degree polynomials. *computational complexity*, 2(25):507–561, 2016.

17 Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. `https://github.com/dasarpmar/lowerbounds-survey/releases`, 2016.

18 Madhu Sudan. Algebra and computation. `http://people.csail.mit.edu/madhu/FT98/course.html`, 1998. Lecture Notes.

19 Joachim von zur Gathen. Hensel and Newton methods in valuation rings. *Mathematics of Computation*, 42(166):637–661, 1984.

20 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

21 Hans Zassenhaus. On Hensel factorization, I. *Journal of Number Theory*, 1(3):291–311, 1969.

On the Complexity of Branching Proofs

Daniel Dadush
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
dadush@cwi.nl

Samarth Tiwari
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
samarth.tiwari@cwi.nl

Abstract

We consider the task of proving integer infeasibility of a bounded convex K in $\mathbb{R}^n$ using a general branching proof system. In a general branching proof, one constructs a branching tree by adding an integer disjunction $\mathbf{ax} \leq b$ or $\mathbf{ax} \geq b+1$, $\mathbf{a} \in \mathbb{Z}^n$, $b \in \mathbb{Z}$, at each node, such that the leaves of the tree correspond to empty sets (i.e., K together with the inequalities picked up from the root to leaf is empty).

Recently, Beame et al (ITCS 2018), asked whether the bit size of the coefficients in a branching proof, which they named stabbing planes (SP) refutations, for the case of polytopes derived from SAT formulas, can be assumed to be polynomial in n. We resolve this question in the affirmative, by showing that any branching proof can be recompiled so that the normals of the disjunctions have coefficients of size at most $(nR)^{O(n^2)}$, where $R \in \mathbb{N}$ is the radius of an ℓ_1 ball containing K, while increasing the number of nodes in the branching tree by at most a factor $O(n)$. Our recompilation techniques works by first replacing each disjunction using an iterated Diophantine approximation, introduced by Frank and Tardos (Combinatorica 1986), and proceeds by "fixing up" the leaves of the tree using judiciously added Chvátal-Gomory (CG) cuts.

As our second contribution, we show that Tseitin formulas, an important class of infeasible SAT instances, have quasi-polynomial sized cutting plane (CP) refutations. This disproves a conjecture that Tseitin formulas are (exponentially) hard for CP. Our upper bound follows by recompiling the quasi-polynomial sized SP refutations for Tseitin formulas due to Beame et al, which have a special enumerative form, into a CP proof of the same length using a serialization technique of Cook et al (Discrete Appl. Math. 1987).

As our final contribution, we give a simple family of polytopes in $[0,1]^n$ requiring exponential sized branching proofs.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Branching Proofs, Cutting Planes, Diophantine Approximation, Integer Programming, Stabbing Planes, Tseitin Formulas

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.34

Funding Supported by ERC Starting Grant QIP–805241.

Acknowledgements The first author would like to deeply thank Noah Fleming, Denis Pankratov, Toni Pitassi and Robert Robere for posing the bit-size vs length question for SP and for very stimulating conversations while the author was visiting the University of Toronto. The authors are also very grateful for the comments from the anonymous reviewers, which have greatly helped us improve the quality of the presentation.

1 Introduction

A principal challenge in SAT solving is finding short proofs of unsatisfiability of SAT formulas. This task is particularly important in the automatic verification of computer programs, where incorrect runs of the program or bugs (e.g., divide by zero) can be encoded as satisfying

assignments to SAT formulas derived from the program specification. In this case, the corresponding formula is an UNSAT instance if the corresponding program is correct, or at least devoid of certain types of bugs.

The study of how long or short such UNSAT proofs can be is the main focus of the field of proof complexity. Indeed, popular SAT algorithms, such as DPLL search, i.e. branching on variables combined with unit propagation, or Conflict Driven Clause Learning (CDCL), implicitly generate infeasibility proofs in standard proof systems such as Resolution or Cutting Planes. From the negative perspective, lower bounds on the length of UNSAT proofs in these systems automatically imply lower bounds on the running time of the corresponding SAT algorithms. On the positive side, understanding which UNSAT instances have short proofs can inspire the design of good heuristics and algorithms for trying to find such proofs automatically.

The analoguous problem in the context of Integer Programming (IP) is that of showing that a linear system of inequalities has no integer solutions. This problem also encapsulates SAT: for a formula $\Phi(\mathbf{x}) := \wedge_{j\in[m]} C_j(\mathbf{x})$, where $C_j(\mathbf{x}) = \vee_{i\in L_j} x_i \vee_{i\in \bar{L}_j} \bar{x}_i$, $j \in [m]$, Φ is unsatisfiable if and only if the linear system

$$\sum_{i\in L_j} x_i + \sum_{i\in \bar{L}_j} (1 - x_i) \geq 1, j \in [m] \qquad \text{(SAT-LP)}$$
$$0 \leq x_i \leq 1, i \in [n]$$

has no integer solutions (in this case $\{0,1\}$). IP solvers such as CPLEX or Gurobi routinely produce such infeasibility proofs in the so-called proof of optimality phase of the solution process. More precisely, once a solver has found a candidate optimal solution $\mathbf{x}^*$ to an integer linear program

$$\min \mathbf{cx} \quad \text{subject to } \mathsf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n \qquad \text{(IP)}$$

optimality is proved by showing that the linear system

$$\mathbf{cx} < \mathbf{cx}^* \qquad \text{(IP-LP)}$$
$$\mathsf{A}\mathbf{x} \leq \mathbf{b}$$

has no integer solutions. In practice, this is most often achieved by a mixture of Branch & Bound and Cutting Planes. We note that most applications are modelled using *mixed* integer linear programs (MIP), where a decision variable x_i can be continuous ($x_i \in \mathbb{R}$), binary ($x_i \in \{0,1\}$) or general integer ($x_i \in \mathbb{Z}$), with binary and continuous variables being the most common.

1.1 Branching Proofs

For proving infeasibility of a SAT formula or an integer linear program, where we denote the continuous relaxation of the feasible region by $K \subseteq \mathbb{R}^n$ (e.g., (SAT-LP) or (IP-LP)), the most basic strategy is to build a search tree based on so-called variable branching. That is, we build a rooted binary tree $\mathcal{T}$, where at each internal node v we choose a "promising" candidate integer variable x_i and create two children v_l, v_r corresponding either side of the disjunction $x_i \leq b$ (left child v_l) and $x_i \geq b+1$ (right child v_r), for some $b \in \mathbb{Z}$. The edge from the parent to its child is labelled with the corresponding inequality. If x_i is binary, one always sets $b = 0$, corresponding to branching on $x_i = 0$ or $x_i = 1$. To each node is associated its continuous relaxation K_v, corresponding to K together with the inequalities on the edges

of the unique path from the root to v in $\mathcal{T}$. To be a valid proof of integer infeasibility, we require that the continuous relaxation K_v be empty at every leaf node $v \in \mathcal{T}$. We then call the proof tree $\mathcal{T}$ as above a *variable* branching proof of integer infeasibility for K. We will consider the length of branching proof, interpreted as the "number of lines" of the proof, to be equal to the number of nodes in $\mathcal{T}$, which we denote $|\mathcal{T}|$.

When applied to a SAT formula as in (SAT-LP), a variable branching tree $\mathcal{T}$ as above is in correspondance with a run of DPLL search, noting that LP infeasibility of a node is equivalent to unit propagation (i.e., iteratively propagating the values of variables appearing in single literal clauses) yielding a conflict[1]. Similarly when applied to an integer program as in (IP-LP) for which the optimal value is known, the above is equivalent to standard Branch and Bound.

Branching on General Integer Disjunctions. To obtain a more general proof strategy one may examine a richer class of disjunctions. Instead of branching only on variables as above, one may also branch on a general integer disjunction $\mathbf{a}\mathbf{x} \leq b$ or $\mathbf{a}\mathbf{x} \geq b+1$, where $\mathbf{a} \in \mathbb{Z}^n$ and $b \in \mathbb{Z}$, noting that any integer point $\mathbf{x} \in \mathbb{Z}^n$ must satisfy exactly one of these inequalities. One may then define branching proofs of infeasibility for K using general integer disjunctions exactly as above, which we call *general* branching proofs. We note that in principle, the continuous relaxation K can be arbitrary, i.e. it need not be a polytope. In this work, we will in fact consider the case where K is a compact convex set in $\mathbb{R}^n$. Furthermore, it is easy to extend branching proofs to the case of *mixed* integer infeasibility, where we want to certify that $K \cap \mathbb{Z}^k \times \mathbb{R}^{n-k} = \emptyset$, that is, where only the first k variables are restricted to be integer. In this setting, one need only restrict the disjunctions $\mathbf{a}\mathbf{x} \leq b$ or $\geq b$ to have support on the integer variables; precisely, we enforce $\mathbf{a} \in \mathbb{Z}^k \times \{0\}^{n-k}, b \in \mathbb{Z}$.

As formalized above, the attentive reader may have noticed that there is no mechanism to "certify" the emptiness of the leaf nodes of the tree. In many cases, such certificates can be appended to the leaves yielding a *certified* branching proof, however their exact form will differ depending on the representation of K (e.g., LP, SOCP or SDP). In the important case where the continuous relaxation is a polytope $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}$, emptiness of a leaf node can indeed be certified efficiently using a so-called Farkas certificate of infeasibility. Let $\mathcal{T}$ be branching proof for K and let $v \in \mathcal{T}$ be a leaf node with $K_v = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}, \mathsf{A}_v\mathbf{x} \leq \mathbf{b}_v\}$, where $\mathsf{A}_v\mathbf{x} \leq \mathbf{b}_v$ represents all the inequalities induced by the branching decisions on the path from the root to v. Then, by Farkas's lemma $K_v = \emptyset$ iff there exists multipliers $\boldsymbol{\lambda}_v := (\boldsymbol{\lambda}_{v,1}, \boldsymbol{\lambda}_{v,2}) \geq 0$, known as a *Farkas certificate*, such that $\boldsymbol{\lambda}_{v,1}\mathsf{C} + \boldsymbol{\lambda}_{v,2}\mathsf{A}_v = 0$ and $\boldsymbol{\lambda}_{v,1}\mathbf{d} + \boldsymbol{\lambda}_{v,2}\mathbf{b} < 0$. Therefore, for a polyhedral feasible region, we may certify the branching proof by labeling each leaf node $v \in \mathcal{T}$ with its Farkas certificate $\boldsymbol{\lambda}_v$.

For a variable branching proof $\mathcal{T}$, especially for $\{0,1\}$ IPs, the tree size $|\mathcal{T}|$ is arguably the most important measure of the complexity of the proof. However, for a general branching proof $\mathcal{T}$, the tree size $|\mathcal{T}|$ ignores the "complexity" of the individual disjunctions. Note that we have not a priori set any restrictions on the size of the coefficients for the disjunctions $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$ used in the nodes of the tree. To accurately capture this complexity, we will also measure the number of bits needed to write down the description of $\mathcal{T}$, which we denote by $\langle \mathcal{T} \rangle$. Here, $\langle \mathcal{T} \rangle$ includes the bit-size of all the disjunctions $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$ used

[1] Note that if unit propagation finds a conflict at a node of the tree, the corresponding node LP (i.e. (SAT-LP) with some variables fixed to 0 or 1) is also infeasible. If unit propagation terminates without a conflict, then setting all non-propagated variables to 1/2 yields a feasible LP solution since every surviving clause has at least 2 literals.

in $\mathcal{T}$. For a certified branching proof, as introduced above, we also include the bit-length of the infeasibility certificates at the leaves to $\langle\mathcal{T}\rangle$. Understanding how large the coefficients need to be to ensure near-optimal tree size will be one of the principal interests of this work.

Applications of General Branching. While variable branching is the most prevalent in practice, due to its simplicity and ease of implementation, it is well-known that branching on general integer disjunctions can lead to much smaller search trees. In practice, general branching is used when certain simple constraints such as $\sum_{i=1}^{n} x_i = 1$, x_i binary, are present in the model, which is part of the family of specially ordered set constraints [3]. In this context, one may branch on $\sum_{i=1}^{n/2} x_i = 0$ or $\sum_{i=1}^{n/2} x_i = 1$ to a get a more balanced search tree. A more recent idea of Fischetti and Lodi [11], known as local branching, is to branch on disjunctions which control the Hamming distance to the best incumbent solution $\mathbf{x}^*$, e.g. $\sum_{i:x_i^*=0} x_i + \sum_{i:x_i^*=1}(1-x_i) \leq k$ or $\geq k+1$. This provides a very effective way of controlling the search neighborhood, and allows one to find improving solutions more quickly.

From the theoretical side, a seminal result is that of Lenstra [17], who gave a fixed dimension polynomial time algorithm for Integer Programming based on basis reduction and general branching. Relating to branching proofs, his result directly implies that every integer free compact convex set admits a general branching proof of length $O(f(n)^n)$, where $f(n)$, the so-called flatness constant, is the supremum of the lattice width over integer free compact convex set in dimension n. It is known that $f(n) = \tilde{O}(n^{4/3})$ [2, 22] and $f(n) = \Omega(n)$. We note that already in $\mathbb{R}^2$, there are simple integer free polytopes, e.g., $\{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 = 1/2, 0 \leq x_1 \leq k\}$, for $k \in \mathbb{N}$, with arbitrarily long variable branching proofs. Inspired by Lenstra's result, there has been a line of work on the use of basis reduction techniques to reformulate IPs so that they become "easy" for variable branching. This approach has been successfully theoretically analyzed for certain classes of knapsack problems as well as random IPs (see [20] for a survey) and experimentally analyzed on various classes of instances [1, 16]. There has also been experimental work on how to come up with good general branching directions in practice using heuristic methods [19, 18, 14].

1.2 Cutting Planes

Another fundamental proof system, studied extensively within both the IP and SAT contexts are cutting planes (CP) proofs. The most fundamental class of cutting planes are so-called Chvátal-Gomory (CG) cuts, which are the principal class studied within SAT and one of the most important classes of cuts in IP [13].

CG cuts for a set $K \subseteq \mathbb{R}^n$ are derived geometrically as follows. Assume that the inequality $\mathbf{a}\mathbf{x} \leq r$, $\mathbf{a} \in \mathbb{Z}^n$, $r \in \mathbb{R}$, is valid for K, that is, $\mathbf{x} \in K \Rightarrow \mathbf{a}\mathbf{x} \leq r$. Then, the inequality $\mathbf{a}\mathbf{x} \leq \lfloor r \rfloor$ is valid for $K \cap \mathbb{Z}^n$, since $\mathbf{x} \in \mathbb{Z}^n$ implies that $\mathbf{a}\mathbf{x} \in \mathbb{Z}$. Given $\mathbf{a} \in \mathbb{Z}^n$, the strongest cut of this form one can derive for K is clearly $\mathbf{a}\mathbf{x} \leq \lfloor \sup_{\mathbf{z} \in K} \mathbf{a}\mathbf{z} \rfloor$. We therefore denote this cut to be the CG cut of K induced by $\mathbf{a}$, and we use the notation $\mathrm{CG}(K, \mathbf{a}) := \{\mathbf{x} \in K : \mathbf{a}\mathbf{x} \leq \lfloor \sup_{\mathbf{z} \in K} \mathbf{a}\mathbf{z} \rfloor\}$ to denote applying the CG cut induced by $\mathbf{a}$ to K. We may extend this to an ordered list $\mathcal{L} = (\mathbf{a}_1, \ldots, \mathbf{a}_k)$, letting $\mathrm{CG}(K, \mathcal{L})$ be the result of applying the CG cuts induced by $\mathbf{a}_1, \ldots, \mathbf{a}_k$ to K one by one in this order (from left to right).

In terms of certifying such cuts, if $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}$, $\mathsf{C} \in \mathbb{Q}^{m \times n}, \mathbf{d} \in \mathbb{Q}^m$, is a polyhedron, then by Farkas's lemma, every CG cut can be obtained as a conic combination of the constraints after rounding down the right hand side. That is, for each $\boldsymbol{\lambda} \geq 0$ such that $\boldsymbol{\lambda}\mathsf{C} \in \mathbb{Z}^n$, we have the corresponding CG cut $\boldsymbol{\lambda}\mathsf{C}\mathbf{x} \leq \lfloor \boldsymbol{\lambda}\mathbf{d} \rfloor$, and every CG cut for K can be derived in this way.

A cutting plane proof (CP) of integer infeasibility for $K \subseteq \mathbb{R}^n$ can now be described as a list $\mathcal{L} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$, $\mathbf{a}_i \in \mathbb{Z}^n$, such that $\mathrm{CG}(K, \mathcal{L}) = \emptyset$. In this context, the number of CG cuts N denotes the length of the CP proof. When $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}$ is a polyhedron as above, to get a certified proof, we can augment $\mathcal{L}$ with multipliers $\boldsymbol{\lambda}_1 \in \mathbb{R}^m_+, \boldsymbol{\lambda}_2 \in \mathbb{R}^{m+1}_+, \ldots, \boldsymbol{\lambda}_{N+1} \in \mathbb{R}^{m+N}_+$ (we still refer to the length of $\mathcal{L}$ as N in this case). Letting $\mathcal{L}_i := (\mathbf{a}_1, \ldots, \mathbf{a}_i)$, $i \in [N]$, the multipliers $\boldsymbol{\lambda}_i \in \mathbb{R}^{m+i-1}_+$, $0 \leq i \leq N$, certify the cut $\mathbf{a}_i\mathbf{x} \leq \lfloor \sup\{\mathbf{a}_i\mathbf{z} : \mathbf{z} \in \mathrm{CG}(K, \mathcal{L}_{i-1})\}\rfloor$, in the manner described in the previous paragraph, using the the original inequalities $\mathsf{C}\mathbf{x} \leq \mathbf{d}$ (the first m components of $\boldsymbol{\lambda}_i$) and the previous cuts $\mathbf{a}_j\mathbf{x} \leq \lfloor \sup\{\mathbf{a}_j\mathbf{z} : \mathbf{z} \in \mathrm{CG}(K, \mathcal{L}_{j-1})\}\rfloor$, $j \in [i-1]$. Finally, $\boldsymbol{\lambda}_{N+1} \in \mathbb{R}^{m+N}_+$ provides the Farkas certificate of infeasibility for $\mathrm{CG}(K, \mathcal{L})$, using the original system together with all the cuts.

As with branching proofs, it is important to be able to control the bit-size $\langle \mathcal{L} \rangle$ of a CP proof and not just its length (i.e., the number of cuts in the list $\mathcal{L}$). Here $\langle \mathcal{L} \rangle$ corresponds to the number of bits needed to describe $\langle \mathbf{a}_1, \ldots, \mathbf{a}_N \rangle$, as well as $\langle \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{N+1} \rangle$ for a certified proof. When $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}$ is a polyhedron as above, a fundamental theorem of Cook, Coullard and Turán [7] is that any CP proof $\mathcal{L}$ of integer infeasibility for K can be recompiled into a *certified* CP proof $\mathcal{L}'$, such that $N := |\mathcal{L}| = |\mathcal{L}'|$ and $\langle \mathcal{L}' \rangle = \mathrm{poly}(N, L)$, where $L := \langle \mathsf{C}, \mathbf{d} \rangle$ is the number of bits needed to describe the linear system defining K. Thus, for CP proofs on polyhedra, one can, without loss of generality, assume that the bit-size of a CP proof is polynomially related to its length and the bit-size of the defining linear system.

In terms of general complexity upper bounds, another important theorem of [7] is that every integer free rational polytope $K \subseteq \mathbb{R}^n$ admits a CP proof of infeasibility of length $O(f(n)^n)$, where $f(n)$ is the flatness constant. This bound was achieved by showing that a run of Lenstra's algorithm can effectively be converted into a CP proof.

To relate CP and branching proofs, there is a simple disjunctive characterization of CG cuts. Namely, $\mathbf{a}\mathbf{x} \leq b$, $\mathbf{a} \in \mathbb{Z}^n$, $b \in \mathbb{Z}$ is a CG cut for K iff $\{\mathbf{x} \in K : \mathbf{a}\mathbf{x} \geq b+1\} = \emptyset$. That is, if and only if the *right side* of the disjunction $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$ is empty for K. From this observation, one can easily show that any CP proof of infeasibility can be converted into a branching proof of infeasibility with only an $O(1)$ factor blowup in length (see [5] for a formal proof).

1.3 Complexity of Branching Proofs

Despite its long history of study within IP, general branching has only recently been studied from the SAT perspective. In [5], Beame et al rediscovered the concept of general branching proofs in the context SAT, naming them stabbing planes (SP) refutations, and analyzed them from the proof complexity perspective. To keep with this nomenclature, we use the term stabbing planes (SP) refutations to refer specifically to a certified branching proofs of infeasibility for SAT formulas. In terms of results, they showed that SP refutations can size or depth simulate CP proofs and showed that they are equivalent to Krajíček's [15] tree-like R(CP) refutations. They further gave lower bounds and impossibility results, showing an $\Omega(n/\log n)$ lower bound on the depth of SP refutations and showed that SP refutations cannot be balanced.

Lastly, they provided upper bounds on the length of SP refutations, showing that any Tseitin formula has a quasi-polynomial sized SP refutation. We recall that a Tseitin formula is indexed by a constant degree graph $G = (V, E)$ and a set of parities $l_v \in \{0, 1\}$, $v \in V$, satisfying $\sum_{v \in V} l_v \equiv 1 \mod 2$. The variables $\mathbf{x} \in \{0, 1\}^E$ index the corresponding subset of edges where the assignment $\mathbf{x}$ is a satisfying assignment iff $\sum_{e \in E : v \in e} x_e \equiv l_v \mod 2$,

$\forall v \in V$. Note that such a formula is clearly unsatisfiable, since the sum of degrees of any (sub)graph is even whereas $\sum_{v \in V} l_v$ is odd by assumption. For such formulas, Beame et al gave a $2^{\Delta}(n\Delta)^{O(\log n)}$ length SP refutations, where Δ is the maximum degree of G. A long standing conjecture [4, 5] is that Tseitin formulas are hard for cutting planes, and thus the above was seen as evidence that SP refutations are strictly stronger than CP. We note that exponential lower bounds for CP were first proven by Pudlák [21], who showed how to derive CP lower bounds from monotone circuit lower bounds. The corresponding monotone circuit problem for Tseitin formulas is easy however, and hence cannot be used for proving strong lower bounds.

Beame et al [5] left open some very natural proof complexity theoretic questions about branching proofs, which highlighted fundamental gaps in our understanding of the proof system. Their first question relates to the relationship between bit-size $\langle \mathcal{T} \rangle$ and length $|\mathcal{T}|$ of an SP proof. Precisely, they asked whether one can always assume that the bit size of an SP refutation is bounded by a polynomial in the dimension and the length of the proof, that is, can an SP refutation be "recompiled" so that it satisfies this requirement without increasing its length by much. As mentioned previously, the corresponding result for CP refutations was already shown by Cook et al [7], though the techniques there do not seem to apply to SP. Their second question was whether one could show a separation between CP and SP, which would follow if Tseitin formulas are (say exponentially) hard for CP. Lastly, they asked whether one can prove super-polynomial lower bounds for SP.

1.4 Our Contributions

In this work, we give answers to many of the questions above. Firstly, we resolve Beame et al's bit-size vs length question affirmatively. Secondly, we show that Tseitin formulas have quasi-polynomial size CP proofs, showing that they do not provide an exponential separation between CP and SP. Lastly, we give a very simple family of n-dimensional (mixed-)integer free polytopes for which any branching proof has size exponential in n. We describe these contributions in detail below.

Bit-size of Branching Proofs. As our first main contribution, we resolve Beame et al's bit-size vs length question, by proving the following more general result:

▶ **Theorem 1.** *Let $K \subseteq \mathbb{R}^n$ be an integer free compact convex set satisfying $K \subseteq R\mathbb{B}_1^n$, where $\mathbb{B}_1^n$ is the ℓ_1 ball and $R \in \mathbb{N}$. Let $\mathcal{T}$ be a branching proof of integer infeasibility for K. Then, there exists a branching proof $\mathcal{T}'$ for K, such that $|\mathcal{T}'| \leq O(n|\mathcal{T}|)$, and where every edge e of $\mathcal{T}'$ is labeled by an inequality $\mathbf{a}'_e\mathbf{x} \leq b'_e$, $\mathbf{a}'_e \in \mathbb{Z}^n$, $b'_e \in \mathbb{Z}$, where $\max\{\|\mathbf{a}'_e\|_\infty, |b'_e|\} \leq (10nR)^{(n+2)^2}$. Moreover, $\langle \mathcal{T}' \rangle = O(n^3 \log_2(nR)|\mathcal{T}|)$.*

The above theorem says that at the cost of increasing the number of nodes in the branching tree by a factor $O(n)$, one can reduce the coefficients in the normals of the disjunctions to $(10nR)^{(n+2)^2}$. In particular, since $\mathbf{a}'_e, b'_e$ are integral, they can be described with $O(n^3 \log_2(nR))$ bits. We note that the final bound on $\langle \mathcal{T}' \rangle$ ends up being better than $O(n^3 \log_2(nR)|\mathcal{T}'|) = O(n^4 \log_2(nR)|\mathcal{T}|)$, due to the fact that "extra" nodes we need in $\mathcal{T}'$ use smaller disjunctions needing only $O(n^2 \log_2(nR))$ bits. In the context of SAT, the desired bound on the coefficients of SP proofs follows directly from the fact that any SAT polytope, as in (SAT-LP), is contained inside $[0,1]^n \subseteq n\mathbb{B}_1^n$.

As mentioned in the previous subsection, one would generally want a branching proof to come with certificates of infeasibility for the leaf nodes. For a rational polytope K, the following corollary bounds the cost of extending the branching proof produced by Theorem 1

to a certified branching proof. Precisely, the bit-size of the final certified proof can be made proportional to the size of the original tree, the bit-encoding length of the defining system for K and a polynomial in the dimension.

▶ **Corollary 2.** *Let $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}$ be rational polytope with $\mathsf{C} \in \mathbb{Q}^{m\times n}, \mathbf{d} \in \mathbb{Q}^m$ having bit-size $L := \langle \mathsf{C}, \mathbf{d}\rangle$. Let $\mathcal{T}$ be a branching proof for K. Then there exists a certified branching proof $\mathcal{T}'$ for K such that $|\mathcal{T}'| \leq O(n)|\mathcal{T}|$ and $\langle \mathcal{T}'\rangle = O(n^6 L)|\mathcal{T}|$.*

The bit-size $L := \langle \mathsf{C}, \mathbf{d}\rangle$ of K in Corollary 2 shows up for two related reasons. Firstly, we need L to upper bound the ℓ_1 circumradius R of K, which is in turn used to bound the bit-size of the disjunctions in Theorem 1. For a rational polytope K, R is in fact always upper bounded by $2^{O(L)}$. We stress that $2^{O(L)}$ more directly upper bounds the ℓ_1 norm of the vertices of K, which in turns upper bounds the ℓ_1 circumradius of K only under the assumption that K is indeed bounded (i.e., that K is polytope and not just a polyhedron). However, it is well known that for a rational polyhedron K, $K \cap \mathbb{Z}^n = \emptyset$ iff $K \cap 2^{O(L)}\mathbb{B}_1^n \cap \mathbb{Z}^n = \emptyset$ (see Schrijver [24] Chapter 17). Therefore, the boundedness assumption above is essentially without loss of generality. More precisely, one can simply add box constraints $-2^{O(L)} \leq x_i \leq 2^{O(L)}$, $i \in [n]$, to the description of K, which increases the description length by an $O(n)$ factor. The second reason for needing L is to bound the bit-complexity of the Farkas infeasibility certificates at the leaves of the modified branching tree. By standard bounds, such a certificate has bit-size bounded by $O(n)$ times the bit description length of a minimal infeasible subsystem (over the reals) at the corresponding leaf. By Helly's theorem, a minimal infeasible subsystem has at most $n + 1$ inequalities consisting of a subset of the inequalities defining K and the inequalities from branching, where each of these inequalities has bit-size at most $O(n^3 L)$ by Theorem 1.

Sketch of Theorem 1. We now give some intuition about the difficulties in proving Theorem 1, which is technically challenging, and sketch the high level proof ideas.

We first note that any disjunction $\mathbf{a}x \leq b$ or $\geq b + 1$, where $\mathbf{a}$ has very large coefficients, only cuts off a very thin slice of K. In particular, the width of the band $b \leq \mathbf{a}\mathbf{x} \leq b + 1$ is exactly $1/\|\mathbf{a}\|_2$. Thus, it is perhaps intuitive that any "optimal" proof should use wide disjunctions instead of thin ones, and hence should have reasonably small coefficients. This intuition turns out to be false however, as the angle of a disjunction can in fact be more important than its width for a proof of optimal length.

The following simple 2 dimensional example shows that if one wishes to exactly preserve the length of a branching proof, then large coefficients are unavoidable even for sets of constant radius. Examine the line segment

$$K = \{(x_1, x_2) : Mx_1 + x_2 = 1/2, 0 \leq x_2 \leq 2\}$$

for $M \geq 1$. Clearly, branching on $Mx_1 + x_2 \leq 0$ or ≥ 1 certifies integer infeasibility in one step. Now let $\mathbf{a} \in \mathbb{Z}^2$ be any branching direction that also certifies infeasibility in one step. Then, the width of K with respect to $\mathbf{a}$ must be less than one:

$$\max_{\mathbf{x}\in K} \mathbf{a}\mathbf{x} - \min_{\mathbf{x}\in K} \mathbf{a}\mathbf{x} = |2(a_2 - a_1/M)| < 1.$$

Now if $a_2 \neq 0$, then $|a_1| \geq M/2$, so $\|\mathbf{a}\|_\infty \geq M/2$. If $a_2 = 0$, then we should let $\mathbf{a} = (1, 0)$, since this choice yields the widest possible disjunctions under this restriction. Branching on $\mathbf{a} = (1, 0)$ cannot certify infeasibility in one step however, since $\mathbf{x} = (0, 1/2) \in K$ and $\mathbf{a}\mathbf{x} = 0$.

To recompile a proof $\mathcal{T}$ using only small coefficients, we are thus forced to make do with a discrete set of disjunction angles that may force us to increase the length of the proof. Given an arbitrary branching direction $\mathbf{a}$, the standard tool for approximating the direction of $\mathbf{a}$ using small coefficients is so-called *Diophantine* approximation (see Lemma 13). Thus, the natural first attempt would be to take every disjunction $\mathbf{ax} \leq b$ or $\geq b+1$ in $\mathcal{T}$ and replace it by its small coefficient Diophantine approximation $\mathbf{a}'\mathbf{x} \leq b'$ or $\geq b'+1$ to get $\mathcal{T}'$. As shown above, there are examples where any such small coefficient $\mathcal{T}'$ will no longer be valid, namely some of the leaf nodes may become feasible.

Let $v \in \mathcal{T}$ be a leaf node with relaxation $K_v = \{\mathbf{x} \in K : \mathsf{A}\mathbf{x} \leq \mathbf{b}\} = \emptyset$ and corresponding approximation $v' \in \mathcal{T}'$ with $K_{v'} = \{\mathbf{x} \in K : \mathsf{A}'_v\mathbf{x} \leq \mathbf{b}'_v\} \neq \emptyset$. To transform $\mathcal{T}'$ to a valid proof, we must therefore add branching decisions to $\mathcal{T}'$ below v' to certify integer-freeness of $K_{v'}$. From here, the main intuitive observation is that since $P_v := \mathsf{A}_v\mathbf{x} \leq \mathbf{b}_v$ and $P_{v'} := \mathsf{A}'_v\mathbf{x} \leq \mathbf{b}'_v$ have almost the same inequalities, $P_{v'} \cap K$ should be very close to infeasible.

By inspecting a Farkas-type certificate of infeasibility fo $K \cap P_v$ (see section 2.4), for a good enough Diophantine approximation $P_{v'}$ to P_v, one can in fact pinpoint an inequality of $P_{v'}$, say $\mathbf{a}'_{v,1}\mathbf{x} \leq b'_{v,1}$, such that replacing $b'_{v,1}$ by $b'_{v,1} - 1$ makes $K \cap P_{v'}$ empty. This uses the boundedness of K, i.e., $K \subseteq R\mathbb{B}_1^n$, and that the disjunctions induced by the rows of A' are much wider than those induced by A. Note that the emptiness of $\mathbf{a}'_{v,1}\mathbf{x} \leq b'_{1,v} - 1$ corresponds to saying that $\mathbf{a}'_{v,1}\mathbf{x} \geq b'_{v,1}$ is a valid CG cut for $K \cap P_{v'}$. Furthermore, this CG cut has the effect of reducing dimension by one since now $\mathbf{a}'_{v,1}\mathbf{x} = b'_{v,1}$.

Given the above, it is natural to hope than one can simply repeat the above strategy recursively. Namely, at each step, we try to find a new CG cut induced by a row of A' which reduces dimension of $K \cap P_{v'}$ by one. Unfortunately, as stated, the strategy breaks down after one step. The main problem is that, after the first step, we have no "information" about $\mathbf{a}_{v,1}\mathbf{x} \leq b_{v,1}$ restricted to $\mathbf{a}'_{v,1}\mathbf{x} = b'_{v,1}$. Slightly more precisely, we no longer have a proxy for $\mathbf{a}_{v,1}\mathbf{x} \leq b_{v,1}$ in $P_{v'}$ that allows us to push this constraint "backwards" on the subspace $\mathbf{a}'_{v,1}\mathbf{x} = b'_{v,1}$. Since we must somehow compare $P_{v'}$ to P_v to deduce infeasibility, this flexibility turns out to be crucial for being able to show the existence of a dimension reducing CG cut.

To fix this problem, we rely on a more sophisticated iterated form of Diophantine approximation due to Frank and Tardos [12]. At a high level (with some simplification), for a disjunction $\mathbf{ax} \leq b$ or $\geq b+1$, $\mathbf{a} \in \mathbb{Z}^n$, $b \in \mathbb{Z}$, we first construct of sequence of Diophantine approximations $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{Z}^n$, containing $\mathbf{a}$ in their span, which intuitively represent the highest to lower order bits of the direction of $\mathbf{a}$. From here, we carefully choose a sequence $b_1, \ldots, b_k \in \mathbb{Z}$ indexing inequalities $\mathbf{a}_i\mathbf{x} \leq b_i$, $i \in [k]$, which allow us to get better and better approximations of $\mathbf{ax} \leq b$. Since we are in reality replacing the disjunction $\mathbf{ax} \leq b$ or $\geq b+1$, we will in fact need a sequence that somehow approximates both sides of the disjunction at the same time. This will correspond to requiring that a "flipped" version of the sequence, namely $\mathbf{a}_i\mathbf{x} \geq b_i$, $i \in [k-1]$, and $\mathbf{a}_k\mathbf{x} \geq b_k + 1$, gives improving approximations of $\mathbf{ax} \geq b+1$. Restricting attention to just the $\mathbf{ax} \leq b$ side, we will show the existence of improving "error levels" $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_k = 0$, such that $\|\mathbf{x}\|_1 \leq R, \mathbf{a}_l\mathbf{x} \leq b_l, \mathbf{a}_i\mathbf{x} = b_i, i \in [l-1] \Rightarrow \mathbf{ax} \leq b + \gamma_l$. Furthermore, we will ensure that branching on $\mathbf{a}_l\mathbf{x} \leq b_l - 1$, not only reduces the error bound α_l, but in fact implies a far stronger inequality than $\mathbf{ax} \leq b$. Precisely, we will require $\|\mathbf{x}\|_1 \leq R, \mathbf{a}_l\mathbf{x} \leq b_l - 1, \mathbf{a}_i\mathbf{x} = b_i, i \in [l-1] \Rightarrow \mathbf{ax} \leq b - n\gamma_l$. Hence, once we have learned the equalities $\mathbf{a}_i\mathbf{x} = b_i$, $i \in [l-1]$, $\mathbf{a}_l$ becomes a suitable proxy for $\mathbf{a}$ which we can use to push the constraint $\mathbf{ax} \leq b$ "backwards". Note that if $l = k$, we have in fact fully learned $\mathbf{ax} \leq b$ since $\alpha_k = 0$. If $l < k$ and $\mathbf{ax} \leq b$ is the "closest inequality to infeasibility" in the current relaxation, corresponding to the inequalities in $P_{v'}$ for some leaf v' together with the additional equalities

as above, we will be able to guarantee that the CG cuts induced by $\mathbf{a}_l$ and $-\mathbf{a}_l$ induce the new equality $\mathbf{a}_l\mathbf{x} = b_l$. Note that if we always manage to reduction dimension by at least 1, we will terminate with an infeasible node after adding at most $n+1$ pairs of CG cuts. So far, we have discussed replacing a disjunction $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$ by sequence instead of a single disjunction, and the latter is what is actually needed. For this purpose, the new disjunction will have the form $\mathbf{a}'\mathbf{x} \leq b'$ or $\geq b'+1$ where $\mathbf{a}' = \sum_{i=1}^{k} M^{k-i}\mathbf{a}_i$ and $b' = \sum_{i=1}^{k} M^{k-i}b_i$ for M chosen large enough. This is chosen to ensure that $\|\mathbf{x}\|_1 \leq R, \mathbf{a}'\mathbf{x} \leq b', \mathbf{a}_i\mathbf{x} = b_i, i \in [l-1]$ "almost implies" $\mathbf{a}_l\mathbf{x} \leq b_l$, with a symmetric guarantee for the flipped sequence. The full list of $(a', b', k, a_1, b_1, \gamma_1, \ldots, a_k, b_k, \gamma_k)$ satisfying the requisite properties is what we call a *valid substitution sequence* of $\mathbf{a}\mathbf{x} \leq b$ (see definition (24)). The main difficulty in constructing and analyzing the disjunction $\mathbf{a}'\mathbf{x} \leq b'$ or $\geq b'+1$, is that each side of the disjunction should induce a valid substitution sequence for the corresponding side of $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$. That is, we need to work for "both sides" at once. As the remaining details technical, we defer further discussion of the proof to Section 3 of the paper.

As an interesting point of comparison, we note that in constrast to Theorem 1 the recompilation result of [7] does not give a *length independent bound* on the size of normals of the CG cuts it produces (e.g., depending only on the ℓ_1 radius of K). An interesting question is whether one can give length independent bounds for CP proofs based only on the bit-complexity L of the starting system. Perhaps one avenue for such a reduction, would be to first convert the CP proof to a branching proof and try to apply the techniques above. The main issue here is that first reduction phase above, which approximates each disjunction in the tree with a small coefficient one, need not preserve the CP structure. Namely, after the replacement, it is not clear how to guarantee that every disjunction in the replacement tree has at least one "empty" side (note that this problem is compounded by the approximation errors going up the tree).

Upper Bounds for Tseitin formulas. As our second contribution, we show that Tseitin formulas have quasi-polynomial CP proofs, refuting the conjecture that these formulas are (exponentially) hard for CP.

▶ **Theorem 3.** *Let $G = (V, E)$ be an n-vertex graph, $l_v \in \{0, 1\}$, for $v \in V$, be parities and Φ be the corresponding Tseitin formula. Then Φ has a CP refutation of length $2^{\Delta}(n\Delta)^{O(\log n)}$, where Δ is the maximum degree of G.*

To prove the theorem our main observation is that the quasi-polynomial SP proof of Beame et al [5] is of a special type, which we dub an *enumerative branching proof*, that can be automatically converted to a CP proof of the same size.

We define an *enumerative* branching proof for a compact convex set K to correspond, as before, to a tree $\mathcal{T}$ with root r and root relaxation $K_r := K$. At every node $v \in \mathcal{T}$ with $K_v \neq \emptyset$, we choose a branching direction $\mathbf{a}_v \in \mathbb{Z}^n \setminus \{0\}$ and immediately branch on all possible choices $b \in \mathbb{Z}$ that intersect the current relaxation K_v. Note that tree $\mathcal{T}$ need no longer be binary. Formally, we first label v with the bounds $l_v, u_v \in \mathbb{R}$ satisfying

$$\{\mathbf{a}_v\mathbf{x} : \mathbf{x} \in K_v\} \subseteq [l_v, u_v].$$

From here, we create a child node v_b, for every $b \in \mathbb{Z}$ such that $l_v \leq b \leq u_v$. The edge $e = \{v, v_b\}$ is now labeled with the *equality* $\mathbf{a}_v\mathbf{x} = b$ and the updated relaxation becomes $K_{v_b} = \{\mathbf{x} \in K_v : \mathbf{a}_v\mathbf{x} = b\}$. From here, each leaf node $v \in \mathcal{T}$ can be of two different types. Either $K_v = \emptyset$, or if $K_v \neq \emptyset$, the interval $[l_v, u_v]$ is defined and does not contain integer points, i.e., $\lfloor u_v \rfloor < l_v$. A tree $\mathcal{T}$ satisfying the above properties is a valid enumerative branching proof of integer infeasibility for K.

It is an easy exercise to check that any enumerative branching proof can be converted to a standard branching proof incurring only a constant factor blowup in the number of nodes. Theorem 3 follows directly from the observation that the Beame et al SP proof is enumerative together with the following simulation result.

▶ **Theorem 4.** *Let $K \subseteq \mathbb{R}^n$ be a compact convex set. Let $\mathcal{T}$ be an enumerative branching proof of K. Then there exists $\mathcal{L} = (\mathbf{a}_1, \ldots, \mathbf{a}_N) \in \mathbb{Z}^n$ such that $\mathrm{CG}(K, \mathcal{L}) = \emptyset$ and $N \leq 2|\mathcal{T}| - 1$.*

While in the above generality the result is new, the main ideas (at least for rational polytopes) are implicit in Cook et al [7]. In particular, their proof that any integer free rational polytope admits a CP proof of length at most $O(f(n)^n)$ in effect treats Lenstra's algorithm as an enumerative branching proof which they serialize to get a CP proof. Theorem 4 shows that their serialization technique is fully general and in fact can be applied to any enumerative branching proof. To get a certified CP proof of small bit-size from Theorem 4 for a rational polyhedron K, we note that it suffices to apply the recompilation technique of Cook et al [7] to the output of Theorem 4. While there is some technical novelty in the generalization to arbitrary compact convex sets, we feel the main contribution of Theorem 4 is conceptual. As evidenced by Theorem 3, the formalization of enumerative branching proofs and their relationship to CP can be a useful tool for constructing CP proofs.

We now sketch the main ideas for serializing an enumerative branching proof $\mathcal{T}$ for K. We start from the root $r \in \mathcal{T}$, with branching direction $\mathbf{a}_r \in \mathbb{Z}^n$ and $\{\mathbf{a}_r\mathbf{x} : \mathbf{x} \in K\} \subseteq [l_r, u_r]$. The idea is to iteratively "push" the hyperplane $H_b = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_r\mathbf{x} = b\}$, with b initialized to u_r, backwards through K, until K is empty (i.e., iteratively decreasing b until it goes below l_r). The first push is given by the CG cut induced by $\mathbf{a}_r$ which pushes H_{u_r} to $H_{\lfloor u_r \rfloor}$. That is, $b \leftarrow \lfloor b \rfloor$. Since b is now integral, we can no longer decrease b just using CG cuts induced by $\mathbf{a}_r$. At this point, we note that the subtree $\mathcal{T}_{r_b}$ of $\mathcal{T}$ rooted at the child r_b is a valid branching proof for $K \cap H_b$. We can thus apply the procedure recursively on $K \cap H_b$ and $\mathcal{T}_{r_b}$ to "chop off" $K \cap H_b$. For this purpose, one crucially needs to be able to lift CG cuts applied to the face $K \cap H_b$ to CG cuts one can apply to K that have the same effect on $K \cap H_b$. Such a *lifting lemma* is classical for rational polyhedra [6] and was established more recently for compact convex sets in [10], a variant of which we use here. Applying the lifted CG cuts to K, we can thus guarantee that $K \cap H_b = \emptyset$. This allows us to push once more with the cut induced by $\mathbf{a}_r$, pushing H_b to H_{b-1}. The process now continues in a similar fashion until K is empty. We note that the enumerative structure is crucial here, as it allows one to keep the "action" on the boundary K throughout the entire proof.

Lower Bounds for Branching Proofs. As our final contribution, we give a simple family of n-dimensional (mixed-)integer free polytopes which require branching proofs of length exponential in n.

▶ **Theorem 5.** *The integer-free SAT polytope*

$$P_n := \{\mathbf{x} \in [0,1] : \sum_{i \in S} x_i + \sum_{i \notin S} (1 - x_i) \geq 1, \forall S \subseteq [n]\}$$

requires branching proofs of length $2^n/n$.

The above example is due to Cook et al [7], which they used to give a $2^n/n$ lower bound for CP. In the above theorem, we show that their lower bound technique extends to branching proofs. As it is very simple and short, we give the full proof below.

Proof. The first observation is that P_n is "integer critical", namely, removing any constraint from P_n makes the polytope integer feasible. In particular, removing $\sum_{i\in S} x_i + \sum_{i\notin S}(1-x_i) \geq 1$, for any $S \subseteq [n]$, makes the vector $\mathbf{1}_{\bar{S}}$, the indicator of the complement of S, feasible.

Let $\mathcal{T}$ denote any branching proof for P_n. For any leaf node v of $\mathcal{T}$, by Farkas's lemma, the infeasibility of the continuous relaxation $(P_n)_v$ is certified by at most $n+1$ constraints. Since P_n is non-empty, at most n of these constraints can come from the description of P_n. Letting N denote the number of leaves of $\mathcal{T}$, one can therefore certify the infeasibility of each leaf of $\mathcal{T}$ using at most nN original constraints from P_n. If $nN < 2^n$, then $\mathcal{T}$ would certify the integer infeasibility of P_n with at least one constraint removed. By integer criticality of P_n, this is impossible. Therefore $|\mathcal{T}| \geq N \geq 2^n/n$, as needed. ◀

One notable criticism of the above example is that it already has 2^n constraints. Thus, the length of the proof is simply proportional to the initial representation. Interestingly, P_n has a very simple extended formulation in $\mathbb{R}^{2n}$ requiring only $O(n)$ constraints. In particular, a direct computation reveals that

$$\begin{aligned} P_n &= \{\mathbf{x} \in [0,1]^n : \|(x_1 - 1/2, \ldots, x_n - 1/2)\|_1 \leq n/2 - 1\} \\ &= \{\mathbf{x} \in [0,1]^n : \exists \mathbf{y} \in [0,1]^n, \sum_{i=1}^n y_i \leq n/2 - 1, \pm(x_i - 1/2) \leq y_i, i \in [n]\}. \end{aligned}$$

Combining the above with Theorem 5, we immediately get an exponential lower bound for proving the mixed-integer infeasibility of a compactly represented polytope. We note that in this setting, the lower bound is indeed exponential in the description length of P.

▶ **Corollary 6.** *Let* $Q_n = \{(\mathbf{x}, \mathbf{y}) \in [0,1]^{2n} : \sum_{i=1}^n y_i \leq n/2 - 1, \pm(x_i - 1/2) \leq y_i, i \in [n]\}$. *Then any branching proof of mixed-integer infeasibility for* Q_n, *proving* $Q_n \cap \mathbb{Z}^n \times \mathbb{R}^n = \emptyset$, *has length at least* $2^n/n$.

To see the above, recall that a mixed-integer branching proof for Q_n only branches on integer disjunctions supported on the first n variables. Thus, it is entirely equivalent to a branching proof for the projection of Q_n onto these variables, namely, to a branching proof for P_n.

As a final remark, we note that in the extended space, Q_n does in fact have a very short proof of infeasibility using only n split cuts, which are perhaps the most important class of cutting planes in practice (in fact, the most generically effective cuts are the Gomory mixed-integer cuts (GMI), which are equivalent to split cuts for rational polyhedra [8]). Roughly speaking, a split cut here is any linear inequality that is *valid for both sides* $\mathbf{ax} \leq b$ or $\geq b+1$, $\mathbf{a} \in \mathbb{Z}^n$, $b \in \mathbb{Z}$, of an integer disjunction. In particular, $y_i \geq 1/2$ is a valid split cut for Q_n, for $i \in [n]$, since it is valid for $x_i \leq 0$ and $x_i \geq 1$. These n splits together imply that $\sum_{i=1}^n y_i \geq n/2$, and thus adding them to Q_n makes the system infeasible.

1.5 Conclusions

In this work, we have continued the proof complexity theoretic study of branching proofs started in [5], establishing analogues of the CP results in [7] for branching proofs. In the process, we have clarified basic properties of the branching proof system, including how to control the size of coefficients, how to simulate important classes of branching proofs using CP, and how to construct elementary lower bound examples for them. We hope that these results will help motivate a further study of this important proof system.

In terms of open questions, there are many. On the lower bound side, in the context of SAT, the example we use has exponentially many clauses. It would be much more interesting to find polynomial sized formulas with exponential sized branching proofs. In the context of integer programming, as mentioned previously, the best known algorithms for general integer programming require $n^{O(n)}$ time. A very interesting question is whether one can find an example of an integer free compact convex set $K \subseteq \mathbb{R}^n$, requiring branching proofs of size $n^{\Omega(n)}$. Such a lower bound would show that Lenstra-type algorithms for IP, which in fact yield enumerative branching proofs, cannot be substantially improved. We note that this still leaves open the possibility that so-called Kannan-type algorithms can do much better (see [9] Chapter 7 for a reference). In terms of upper bounds, a natural question is whether one can leverage the simulation of enumerative branching proofs by CP to give new upper bounds beyond Tseitin formulas. It was shown by Cook et al [7] that for SAT, CP can be simulated by extended resolution. A natural question is whether stabbing planes can also be simulated by extended resolution. Lastly, as mentioned previously, it would be interesting to establish length independent bounds for the coefficients of the normals in CP proofs.

1.6 Organization

In Section 2, we collect basic notation, formalize the definition of branching proofs and cover the necessary tools from Diophantine approximation. In Section 3, we present our branching proof recompilation theorem, which ensures that the bit-size of branching proofs can be polynomially bounded. In Section 4, we show how to simulate enumerative branching proofs via CP, and apply this simulation to get a quasi-polynomial CP bound for Tseitin formulas.

2 Preliminaries

Basic Notation. The natural numbers are denoted by $\mathbb{N}$, the reals and non-negative reals by $\mathbb{R}, \mathbb{R}_+$ respectively. For $m \in \mathbb{N}$, we denote the set $\{1, \dots, m\}$ by $[m]$. Vectors $\mathbf{x} \in \mathbb{R}^n$ are denoted in bold and scalars by $x \in \mathbb{R}$. The standard basis vectors of $\mathbb{R}^n$ are denoted by $\mathbf{e}_i, i \in [n]$. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we write $\mathbf{xy} := \sum_{i=1}^n x_i y_i$ for their inner product. The ℓ_1 and ℓ_∞ norm of $\mathbf{x}$ are $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$ and $\|\mathbf{x}\|_\infty = \max_{i \in [n]} |x_i|$ respectively. We denote the ℓ_1 ball in $\mathbb{R}^n$ by $\mathbb{B}_1^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \leq 1\}$. For a vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$, we let $\lfloor \mathbf{x} \rceil := (\lfloor x_1 \rceil, \dots, \lfloor x_n \rceil)$ denote the vector whose coordinates are those of $\mathbf{x}$ rounded to the nearest integer.

Since we shall study convex bodies lying in the l_1 ball of some radius $R \in \mathbb{N}$, it is helpful to define the following shorthand notation: for a set of linear inequalities $\mathsf{A}\mathbf{x} \leq \mathbf{b}$ and a vector $\mathbf{c}$, the expression $\mathsf{A}\mathbf{x} \leq \mathbf{b} \Rightarrow_R \mathbf{c}\mathbf{x} \leq d$ stands for

$$\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \leq R, \mathsf{A}\mathbf{x} \leq \mathbf{b}\} \subseteq \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \leq R, \mathbf{c}\mathbf{x} \leq d\}.$$

▶ **Definition 7** (Halfspace, Hyperplane). *For $\mathbf{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$, we define the halfspace $H_{\mathbf{a},b} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}\mathbf{x} \leq b\}$ and the hyperplane $H^=_{\mathbf{a},b} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}\mathbf{x} = b\}$.*

▶ **Definition 8** (Support Function). *Let $K \subseteq \mathbb{R}^n$. The* support function $h_K : \mathbb{R}^n \to \mathbb{R}$ *is defined as $h_K(\mathbf{a}) := \sup_{\mathbf{x} \in K} \mathbf{a}\mathbf{x}$. The support function is always convex and is continuous if K is non-empty and bounded. If K is non-empty and compact, the supremum in $h_K(\mathbf{a})$ is always attained. By convention, if $K = \emptyset$ we define $h_K(\mathbf{a}) = -\infty$, $\forall \mathbf{a} \in \mathbb{R}^n$.*

For $K \subseteq \mathbb{R}^n$ non-empty and compact and $\mathbf{a} \in \mathbb{R}^n$, we define the supporting hyperplane of K induced by $\mathbf{a}$ to be $H^=_K(\mathbf{a}) := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}\mathbf{x} = h_K(\mathbf{a})\}$. We define the set of maximizers of $\mathbf{a}$ in K to be $F_K(\mathbf{a}) := K \cap H^=_K(\mathbf{a})$.

2.1 Bit-Sizes

▶ **Definition 9** (Bit-size). *The notation $\langle x \rangle$ is reserved for the number of bits required to express the object x, or the bit-size of x. We build up the precise definitions as follows:*

For $r \in \mathbb{Q}, r = p/q, p \in \mathbb{Z}, q \in \mathbb{Z}, q > 0, \langle r \rangle := 1 + \lceil \log(|p|+1) \rceil + \lceil \log(q+1) \rceil$. Next, for $\mathbf{c} \in \mathbb{Q}^n$ with $\mathbf{c} = (c_1, c_2 \dots c_n), \langle c \rangle := n + \sum_{i=1}^n \langle c_1 \rangle$. Similarly for matrices $\mathsf{A} \in \mathbb{Q}^{m \times n}$, $\langle \mathsf{A} \rangle := mn + \sum_{i=1}^m \sum_{j=1}^n \langle \mathsf{A}_{ij} \rangle$. $\langle A, B \rangle$ is simply $\langle A \rangle + \langle B \rangle$ when these terms are well-defined.

For a labeled rooted tree $\mathcal{T}$ with n nodes and m edges $E[\mathcal{T}]$, and where edges $e \in E[\mathcal{T}]$ have labels L_e and nodes v have labels L_v, and if the labels belong to a class for which the bit-size has already been defined, then $\langle \mathcal{T} \rangle := n + m + \sum_{e \in E[\mathcal{T}]} \langle L_e \rangle + \sum_{v \in \mathcal{T}} \langle L_v \rangle$.

2.2 Branching Proofs

▶ **Definition 10** (Branching Proof). *A branching proof of integer infeasibility for a convex set $K \subseteq \mathbb{R}^n$ is represented by a rooted binary tree $\mathcal{T}$ with root $r := r_{\mathcal{T}}$. Each node $v \in \mathcal{T}$ is labeled with $(\mathbf{a}_v, b_v), \mathbf{a}_v \in \mathbb{Z}^n, b_v \in \mathbb{Z}$ and has two children nodes: the left child v_l and right child v_r. Since the inner product of two integer vectors is an integer, the integer lattice $\mathbb{Z}^n$ can be partitioned into $\{\mathbf{x} \in \mathbb{Z}^n : \mathbf{a}_v \mathbf{x} \leq b_v\}, \{\mathbf{x} \in \mathbb{Z}^n : \mathbf{a}_v \mathbf{x} \geq b_v + 1\}$. This partition is referred to as the integer disjunction given by $(\mathbf{a}_v, b_v)$.*

Every edge $e \in E[\mathcal{T}]$ is labeled with an inequality $\mathbf{a}_e \mathbf{x} \leq b_e$. A left edge $e_l = \{v, v_l\}$ is labeled with $\mathbf{a}_v \mathbf{x} \leq b_v$. Thus we have $\mathbf{a}_e = \mathbf{a}_v, b_e = b_l$. However, a right edge $e_r = \{v, v_r\}$ is labeled with $\mathbf{a}_v \mathbf{x} \geq b_v + 1$, so that $\mathbf{a}_e = -\mathbf{a}_v, b_e = -b_1 - 1$.

For each node $v \in \mathcal{T}$, we define $P_{\mathcal{T}}(v)$ to be the the unique path from the root r of $\mathcal{T}$ to v. Also define for each v a polyhedron $P_v = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}_v \mathbf{x} \leq \mathbf{b}_v\}$ where the rows of A_v are given by $\mathbf{a}_{v,e}, e \in E[P_{\mathcal{T}}(v)]$, and the coordinates of $\mathbf{b}_v$ are $b_{v,e}, e \in E[P_{\mathcal{T}}(v)]$. Let $K_v := K \cap P_v$. Note that $K_r = K$.

For $\mathcal{T}$ to be a proof of integer infeasibility for K, we require that every leaf $v \in \mathcal{T}$ (v is a leaf if its has no children) satisfies $K_v = \emptyset$.

We denote the length of the branching proof by $|\mathcal{T}|$, which is defined to be the number of nodes of $\mathcal{T}$. The size of a branching proof $\langle \mathcal{T} \rangle$ is simply its bit-size as a labeled rooted tree as given above in definition 9.

▶ **Definition 11** (Certified Branching Proof). *Suppose $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}\}, \mathsf{C} \in \mathbb{Q}^{r \times n}, \mathbf{d} = \mathbb{Q}^r$ belongs to the class of rational polyhedra. A certified branching proof of integer infeasibility for K is a standard branching proof $\mathcal{T}$ of infeasibility of K, but where every leaf node v of $\mathcal{T}$ is also labeled with a Farkas certificate $\boldsymbol{\lambda}_v \in \mathbb{Q}^{r+m_v}, \lambda_i \geq 0, \forall i \in [r + m_v]$, where now $K_v = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{C}\mathbf{x} \leq \mathbf{d}, \mathsf{A}_v \leq \mathbf{b}_v\}$, for $\mathsf{A}_v \in \mathbb{R}^{m_v \times n}, \mathbf{b}_v \in \mathbb{R}^{m_v}$, $m_v = |P_{\mathcal{T}}(v)|$. Let $\boldsymbol{\lambda}_v = (\boldsymbol{\lambda}_{v,1}, \boldsymbol{\lambda}_{v,2}), \boldsymbol{\lambda}_{v,1} \in \mathbb{Q}^r, \boldsymbol{\lambda}_{v,2} \in \mathbb{Q}^{m_v}$. The requirement that every $K_v = \emptyset$ for a leaf nodes v is certified by requiring $\boldsymbol{\lambda}_{v,1}\mathsf{C} + \boldsymbol{\lambda}_{v,2}\mathsf{A}_v = 0, \boldsymbol{\lambda}_{v,1}\mathbf{d} + \boldsymbol{\lambda}_{v,2}\mathbf{b}_v < 0$.*

The bit-size of a certified branching proof is its bit-size when viewed as a labeled rooted tree.

▶ **Definition 12** (Enumerative Branching Proof). *For a compact convex set K, an* enumerative branching proof *consists of a tree $\mathcal{T}$ with root r and root relaxation $K_r := K$. Every node $v \in \mathcal{T}$ is labeled with $(\mathbf{a}_v, l_v, u_v)$, where $\mathbf{a}_v \in \mathbb{Z}^n, l_v, u_v \in \mathbb{Q}$ satisfying*

$$\{\mathbf{a}_v \mathbf{x} : \mathbf{x} \in K_v\} \subseteq [l_v, u_v].$$

There is a child of v denoted v_b for every $b \in \mathbb{Z}, l_v \leq b \leq u_v$, and the edge $e = \{v, v_b\}$ is labeled with the equality $\mathbf{a}_v \mathbf{x} = b$. The relaxation at K_{v_b} becomes $\{\mathbf{x} \in K_v : \mathbf{a}_v \mathbf{x} = b\}$.

$\mathcal{T}$ is a valid enumerative branching proof of infeasibility if every leaf node $v \in \mathcal{T}$ satisfies $K_v = \emptyset$ or $K_v \neq \emptyset$ but $[l_v, u_v]$ contains no integer points, i.e., $\lfloor u_v \rfloor < l_v$.

$\langle \mathcal{T} \rangle$ is again simply the bit-size of $\mathcal{T}$ as a labeled rooted tree.

2.3 Simultaneous Diophantine Approximation

The existence of a rational vector of small bit-size that well approximates an arbitrary real vector is of prime importance in this paper. For this purpose, we shall require standard tools from Diophantine approximation (see [23] for a reference). The following is a slightly adapted version of the Dirichlet's simultaneous approximation theorem, which will be convenient for our purposes. We provide a proof for completeness.

▶ **Lemma 13.** *Let $\mathbf{a} \in \mathbb{R}^n$ satisfy $\|\mathbf{a}\|_\infty = 1$ and let $N \geq 1$. Then, there exists a positive integer $l \leq N^n$ such that $\mathbf{a}' := \lfloor l\mathbf{a} \rceil$ satisfies*

$$\|l\mathbf{a} - \mathbf{a}'\|_\infty < 1/N \quad \text{and} \quad \|\mathbf{a}'\|_\infty = l \geq 1.$$

Proof. Let $\mathcal{C} = \{I_{\mathbf{z}} : \mathbf{z} \in [N]^n\}$ denote the collection of N^n half-open cubes forming a partition of $[0,1)^n$, where $I_{\mathbf{z}} = \times_{i=1}^n [(z_i - 1)/N, z_i/N)$ for $\mathbf{z} \in [N]^n$. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, let $[\mathbf{x}] = \mathbf{x} - \lfloor \mathbf{x} \rfloor \in [0,1)^n$ denote the fractional part of $\mathbf{x}$. Examine the sequence $[0\mathbf{a}], [1\mathbf{a}], \ldots, [N^n\mathbf{a}]$. Since the sequence has length $N^n + 1$ and each element of the sequence lands in one of the cubes in $\mathcal{C}$, by the pigeonhole principle there must be distinct indices l_1, l_2, $0 \leq l_1 < l_2 \leq N^n$ and $\mathbf{z} \in [N]^n$ such that $[l_1\mathbf{a}], [l_2\mathbf{a}] \in I_{\mathbf{z}}$. Since $I_{\mathbf{z}} - I_{\mathbf{z}} = (-1/N, 1/N)^n$, we note that $\|[l_1\mathbf{a}] - [l_2\mathbf{a}]\|_\infty < 1/N$. Let $l = l_2 - l_1$ and $\mathbf{a}' = \lfloor l\mathbf{a} \rceil$, where we note that $1 \leq l \leq N^n$. For any $i \in [n]$, we have that

$$|la_i - \lfloor la_i \rceil| = \min_{k \in \mathbb{Z}} |la_i - k| \leq |(l_1 - l_2)a_i - (\lfloor l_1 a_i \rfloor - \lfloor l_2 a_i \rfloor)| = |[l_1 a_i] - [l_2 a_i]| < 1/N.$$

In particular, $\|l\mathbf{a} - \mathbf{a}'\|_\infty = \|l\mathbf{a} - \lfloor l\mathbf{a} \rceil\|_\infty < 1/N$, as needed. We now show that $\|\mathbf{a}'\|_\infty = l$. By assumption on $\mathbf{a}$, there is a coordinate $i \in [n]$ such that $a_i = 1 = \|\mathbf{a}\|_\infty$. Thus, $a'_i = \lfloor la_i \rceil = l$ and $\|\mathbf{a}'\|_\infty \geq l$. For any $j \in [n]$, also clearly have $la_j \in [-l, l] \Rightarrow a'_j = \lfloor la_j \rceil \in [-l, l]$ since $l \in \mathbb{N}$. Thus, $\|\mathbf{a}'\|_\infty = l$ as needed. ◀

▶ Remark 14. For $\mathbf{a} \in \mathbb{R}^n, \mathbf{a}' \in \mathbb{Z}^n, 1 \leq l \leq N^n$ as above, observe that $a_i = 0 \Rightarrow a'_i = \lfloor la_i \rceil = 0$. Furthermore, $\|\mathbf{a}'\|_\infty = l \leq N^n$.

▶ **Definition 15** (Diophantine Approximation of Precision N)**.** *For a vector $\mathbf{a} \in \mathbb{R}^n \setminus \{0\}$ and $N \geq 1$, we say that $\mathbf{a}'$ is a precision N Diophantine approximation of $\mathbf{a}$ if $\mathbf{a}'$ satisfies the conditions of Lemma 13 on inputs $\mathbf{a}/\|\mathbf{a}\|_\infty$ and N.*

We note that in our application, we will set $N = 10nR$, where R is an integer upper bound on the ℓ_1 radius of the convex set $K \subseteq \mathbb{R}^n$ whose branching proof we are modifying.

2.4 Farkas Certificates for General Convex Sets

A Farkas certificate $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ certifies the infeasibility of the system $\mathsf{A}\mathbf{x} \leq \mathbf{b}, \mathsf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$ if $\boldsymbol{\lambda}^\mathsf{T}\mathsf{A} = 0, \boldsymbol{\lambda}^\mathsf{T}\mathbf{b} = -1$. It is possible to extend this definition to show a linear system is infeasible whenever $\mathbf{x} \in K$ for a compact convex set K.

▶ **Definition 16** (Generalized Farkas Certificate)**.** *Let $K \subseteq \mathbb{R}^n$ be a compact convex set, and $P := \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}\mathbf{x} \leq \mathbf{b}\}, \mathsf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m$. $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ is a generalized Farkas certificate of infeasibility for $K \cap P$ if*

$$\min_{\mathbf{x} \in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b}) > 0$$

▶ **Lemma 17.** *With the notation of definition 16, $K \cap P = \emptyset$ if and only if there exists a generalized Farkas certificate $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ of its infeasibility. Furthermore, if one generalized Farkas certificate exists, then so does one with at most $n + 1$ non-zero coordinates.*

Proof. That a generalized Farkas certificate implies infeasibility is trivial.

Now let us suppose $K \cap P = \emptyset$. K is compact and convex by assumption, and P is clearly closed and convex. Therefore, there exists a strictly separating hyperplane $\mathbf{c}\mathbf{x} = d$ so that K and P lie on "opposite sides" of this hyperplane. More precisely, $\mathbf{c}\mathbf{x} - d > 0$ for $\mathbf{x} \in K$, and $\mathbf{c}\mathbf{x} - d < 0$ for $\mathbf{x} \in P$.

$\mathbf{c}\mathbf{x} < d$ for every $\mathbf{x} \in P$ means the system $\mathsf{A}\mathbf{x} \leq \mathbf{b}, -\mathbf{c}\mathbf{x} \leq -d$ is infeasible. Let $(\boldsymbol{\lambda}, \gamma) \geq 0$ be a (conventional) Farkas certificate of the infeasibility of this system: $\boldsymbol{\lambda}^\mathsf{T}\mathsf{A} = \gamma\mathbf{c}, \boldsymbol{\lambda}^\mathsf{T}\mathbf{b} < \gamma d$. We now claim that $\boldsymbol{\lambda} \geq 0$ is a generalized Farkas certificate of infeasibility for $K \cap P$. Firstly, if $\gamma = 0$, we have that $\min_{\mathbf{x}\in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b}) = -\boldsymbol{\lambda}^\mathsf{T} b > 0$. If $\gamma > 0$, then

$$\mathbf{x} \in K \Rightarrow \gamma(\mathbf{c}\mathbf{x} - d) > 0 \Rightarrow \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b}) > 0.$$

In particular, $\min_{\mathbf{x}\in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b}) > 0$, noting that the minimum is indeed achieved since K is compact.

By Caratheodory's theorem, there exists a generalized Farkas certificate of at most $n + 1$ non-zero coordinates whenever a generalized Farkas certificate exists. ◀

Although the correctness of a conventional Farkas certificate can be verified with simple matrix multiplication, this is not the case for a generalized Farkas certificate. In particular, one must exactly solve the (convex) minimization problem in definition 16 to verify the certificate. This is why the notion of a certified branching proof is sensible only for specific classes of compact convex sets, such as polyhedra.

The following lemma will be crucial for enabling us to deduce infeasibility information for "nearby" polyhedra. The proof relies upon the existence of generalized Farkas certificates as defined above.

▶ **Lemma 18.** *Let $K \subseteq \mathbb{R}^n$ be a compact convex set and let $P = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}\mathbf{x} \leq \mathbf{b}\}, \mathsf{A} \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$, be a polyhedron satisfying $P \cap K = \emptyset$. For $\boldsymbol{\varepsilon} \in \mathbb{R}^m$, define $P_{\boldsymbol{\varepsilon}} := \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}\mathbf{x} \leq \mathbf{b} + \boldsymbol{\varepsilon}\}$. Then, for any $\boldsymbol{\varepsilon} \in \mathbb{R}^m$, either $K \cap P_{\boldsymbol{\varepsilon}} = \emptyset$, or there exists $j \in [m]$ such that $\varepsilon_j > 0$ and $K \cap P_{\boldsymbol{\varepsilon}-(n+1)\varepsilon_j \mathbf{e}_j} = \emptyset$.*

Proof. We assume that $K \cap P_{\boldsymbol{\varepsilon}} \neq \emptyset$, since otherwise there is nothing to prove.

Let $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ be a generalized Farkas certificate of infeasibility for $K \cap P$ with at most $n + 1$ non-zero coordinates as guaranteed by Lemma 17. Let $j_* = \arg\max_{j\in[m]} \varepsilon_j\lambda_j$. We claim that $\varepsilon_{j_*}\lambda_{j_*} > 0$. Assume not, then $\varepsilon_j\lambda_j \leq 0$ for all $i \in [m]$. In particular,

$$\min_{\mathbf{x}\in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b} - \boldsymbol{\varepsilon}) = \min_{\mathbf{x}\in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b}) - \boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\varepsilon} > -\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\varepsilon} \geq 0. \tag{2.1}$$

Thus, $\boldsymbol{\lambda}$ is a generalized Farkas certificate of infeasibility for $K \cap P_\varepsilon$. But this contradicts our assumption that $K \cap P_\varepsilon \neq \emptyset$. Therefore, we must have that $\varepsilon_{j_*}\lambda_{j_*} > 0$. In particular, since $\boldsymbol{\lambda} \geq 0$, we have that $\varepsilon_{j_*} > 0$ and $\lambda_{j_*} > 0$.

We now show that $\boldsymbol{\lambda}$ is in fact a valid generalized Farkas certificate of infeasibility for $K \cap P_{\boldsymbol{\varepsilon}-(n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}}$. Let $S = \{j \in [m] : \lambda_j > 0\}$, and note that by assumption $|S| \leq n + 1$. Using a similar calculation to (2.1), we see that

$$\begin{aligned}\min_{\mathbf{x}\in K} \boldsymbol{\lambda}^\mathsf{T}(\mathsf{A}\mathbf{x} - \mathbf{b} - \boldsymbol{\varepsilon} + (n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}) &> -\boldsymbol{\lambda}^\mathsf{T}\boldsymbol{\varepsilon} + (n+1)\varepsilon_{j_*}\lambda_{j_*} \\ &= -\sum_{j\in S} \varepsilon_j\lambda_j + (n+1)\varepsilon_{j_*}\lambda_{j_*} \geq (n+1-|S|)\varepsilon_{j_*}\lambda_{j_*} \geq 0.\end{aligned}$$

Since $\boldsymbol{\lambda}$ is a valid certificate of infeasibility, we have that $K \cap P_{\boldsymbol{\varepsilon}-(n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}} = \emptyset$, as needed. ◀

2.5 Chvátal-Gomory Cuts

▶ **Definition 19** (Chvátal-Gomory Cut). *For* $\mathbf{a} \in \mathbb{Z}^n$*, the* CG cut of K induced by $\mathbf{a}$ *is the halfspace* $H_K^{\text{cg}}(\mathbf{a}) := H_{\mathbf{a},\lfloor h_K(\mathbf{a})\rfloor}$*. We define* $\text{CG}(K,\mathbf{a}) := K \cap H_K^{\text{cg}}(\mathbf{a})$ *to be the result of applying the CG cut induced by* $\mathbf{a}$ *to* K*.*

This definition is extended to an ordered list $\mathcal{L} = (\mathbf{a}_1,\ldots,\mathbf{a}_k)$ *of integer vectors as* $\text{CG}(K,\mathcal{L}) := \text{CG}(\text{CG}(K,\mathbf{a}_1),(\mathbf{a}_2,\ldots,\mathbf{a}_k))$*. That is, we first apply the CG cut induced by* $\mathbf{a}_1$ *to* K *yielding* $\text{CG}(K,\mathbf{a}_1)$*, then we apply the CG cut induced by* $\mathbf{a}_2$ *to* $\text{CG}(K,\mathbf{a}_1)$ *yielding* $\text{CG}(K,(\mathbf{a}_1,\mathbf{a}_2))$*, and so forth. By convention,* $\text{CG}(K,\emptyset) = K$*, that is, applying the empty list of CG cuts does nothing to* K*.*

The following *lifting lemma*, adapted from [10], shows that CG cuts on a "rational face" F of K can be lifted to a CG cut of K having the same effect on the face. We note that lifting is also possible from "irrational faces" [10], however this requires intersecting multiple CG cuts to achieve the desired effect. The corresponding lemma for rational polyhedra is classical [6].

We include its proof for clarity and completeness. The proof follows the standard approach of adding a large integer multiple of the normal vector to F to the cut.

▶ **Lemma 20** (Lifting CG cuts). *Let* $K \subseteq \mathbb{R}^n$ *be a non-empty compact set. Let* $\mathbf{c} \in \mathbb{Z}^n$*,* $F := F_K(\mathbf{c})$ *and assume that* $h_K(\mathbf{c}) \in \mathbb{Z}$*. Then for any* $\mathbf{a} \in \mathbb{Z}^n$*, there exists* $N \geq 0$ *such that*

$$H_K^{\text{cg}}(\mathbf{a}+i\mathbf{c}) \cap H_K^{=}(\mathbf{c}) = H_F^{\text{cg}}(\mathbf{a}) \cap H_K^{=}(\mathbf{c}), \forall i \geq N.$$

For the proof, we will need the following technical lemma, which shows converge properties of a sequence of maximizing faces.

▶ **Lemma 21.** *Let* $K \subseteq \mathbb{R}^n$ *be a non-empty compact set. Let* $(\mathbf{a}_i)_{i=1}^{\infty} \in \mathbb{R}^n$ *be a convergent sequence with* $\mathbf{a}_\infty := \lim_{i\to\infty} \mathbf{a}_i$ *and let* $F_i := F_K(\mathbf{a}_i)$*,* $i \in \mathbb{N} \cup \{\infty\}$*. Then,* $\forall \varepsilon > 0$ *there exists* $N_\varepsilon \geq 1$ *such that* $\forall i \geq N_\varepsilon$*,* $F_i \subseteq F_\infty + \varepsilon\mathbb{B}_1^n$*.*

Proof. For the sake of contradiction, let us assume that there exists a sequence $(\mathbf{x}_i)_{i=1}^{\infty}$ and an $\varepsilon > 0$ such that $\mathbf{x}_i \in F_i$ and $\mathbf{x}_i \notin F_\infty + \varepsilon\mathbb{B}_1^n$. Letting $K' = \text{closure}(K \setminus (F_\infty + \varepsilon\mathbb{B}_1^n))$, we see that $K' \subseteq K$ is compact and that $K' \cap F_\infty = \emptyset$. Furthermore, $\mathbf{x}_i \in F_i \subseteq K'$, $\forall i \in \mathbb{N}$. Therefore, by compactness of K' there exists a convergent subsequence $(\mathbf{x}_{s_i})_{i=1}^{\infty}$ with limit point $\mathbf{y} := \lim_{i\to\infty} \mathbf{x}_{s_i} \in K'$. Note that by construction $\mathbf{y} \in K$ and $\mathbf{y} \notin F_\infty$. Since K is compact, its support function h_K is continuous. By continuity of h_K and the standard inner product, we conclude that

$$\begin{aligned}\mathbf{v}_\infty\mathbf{y} &= \lim_{i\to\infty} \mathbf{v}_{s_i}\mathbf{x}_{s_i} = \lim_{i\to\infty} h_K(\mathbf{v}_{s_i}) \quad (\text{ since } \mathbf{x}_{s_i} \in F_{s_i}) \\ &= h_K(\mathbf{v}_\infty).\end{aligned}$$

But then $\mathbf{y} \in F_\infty$, a clear contradiction. The lemma thus follows. ◀

We now give the proof of the lifting lemma.

Proof of Lemma 20. Firstly, if $\mathbf{c} = 0$ then $F = K$ and the statement trivially holds for $N = 0$. Thus, we may assume that $\mathbf{c} \neq 0$.

Let $b = h_F(\mathbf{a})$ and recall that $H_F^{\text{cg}}(\mathbf{a}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}\mathbf{x} \leq \lfloor b \rfloor\}$. For $i \geq 0$, let $b_i := h_K(\mathbf{a} + N\mathbf{c}) - ih_K(\mathbf{c})$. From here, we see that

$$\begin{aligned}
\mathbf{x} \in H_K^{\text{cg}}(\mathbf{a} + i\mathbf{c}) \cap H_K^{=}(\mathbf{c}) &\Leftrightarrow (\mathbf{a} + i\mathbf{c})\mathbf{x} = \lfloor h_K(\mathbf{a} + i\mathbf{c})\mathbf{x} \rfloor, \mathbf{c}\mathbf{x} = h_K(\mathbf{c}) \\
&\Leftrightarrow (\mathbf{a} + i\mathbf{c})\mathbf{x} \leq \lfloor b_i + ih_K(\mathbf{c}) \rfloor, \mathbf{c}\mathbf{x} = h_K(\mathbf{c}) \\
&\Leftrightarrow (\mathbf{a} + i\mathbf{c})\mathbf{x} \leq \lfloor b_i \rfloor + ih_K(\mathbf{c}), \mathbf{c}\mathbf{x} = h_K(\mathbf{c}) \\
&\qquad \text{(since } ih_K(\mathbf{c}) \in \mathbb{Z}\text{)} \\
&\Leftrightarrow \mathbf{a}\mathbf{x} \leq \lfloor b_i \rfloor, \mathbf{c}\mathbf{x} = h_K(\mathbf{c}).
\end{aligned}$$

Given the above, it suffices to show that there exists $N \geq 0$ such that $\lfloor b_i \rfloor = \lfloor b \rfloor$, $\forall i \geq N$. Since F is the set of maximizers of $\mathbf{c}$ in K, note that

$$b_i = h_K(\mathbf{a} + i\mathbf{c}) - ih_K(\mathbf{c}) \geq h_F(\mathbf{a} + i\mathbf{c}) - ih_K(\mathbf{c}) = h_F(\mathbf{a}) = b, \forall i \geq 0.$$

Letting $\varepsilon_1 = \lfloor b + 1 \rfloor - b > 0$, note that $\lfloor b' \rfloor = \lfloor b \rfloor$ for $b' \in [b, b + \varepsilon_1)$. Given this, it now suffices to show the existence of $N \geq 0$ such that $b_i < b + \varepsilon_1$, for $i \geq N$. Let $F_i := F_K(\mathbf{a} + i\mathbf{c})$, for $i \in N$. Since $\mathbf{a}/i + \mathbf{c} \to \mathbf{c}$ as $i \to \infty$ and K is compact, by Lemma 21 for $\varepsilon_2 > 0$ there exists $N_{\varepsilon_2} \geq 0$ such that $F_i \subseteq F + \varepsilon_2 \mathbb{B}_1^n$, for $i \geq N_{\varepsilon_2}$. For $i \geq N_{\varepsilon_2}$, we may thus choose $\mathbf{x}_i \in F_i$ and $\mathbf{y}_i \in F$ satisfying $\|\mathbf{x}_i - \mathbf{y}_i\|_1 \leq \varepsilon_2$. From here, for $i \geq N_{\varepsilon_2}$ we have that

$$\begin{aligned}
b_i &= h_K(\mathbf{a} + i\mathbf{c}) - ih_K(\mathbf{c}) = (\mathbf{a} + i\mathbf{c})\mathbf{x}_i - ih_K(\mathbf{c}) \quad (\text{ since } \mathbf{x}_i \in F_i) \\
&\leq \mathbf{a}\mathbf{x}_i + ih_K(\mathbf{c}) - ih_K(\mathbf{c}) = \mathbf{a}(\mathbf{x}_i - \mathbf{y}_i) + \mathbf{a}\mathbf{y}_i \quad (\text{ since } \mathbf{x}_i \in K) \\
&\leq \|\mathbf{x} - \mathbf{y}\|_1 \|\mathbf{a}\|_\infty + h_F(\mathbf{a}) \leq \varepsilon_2 \|\mathbf{a}\|_\infty + b \quad (\text{ since } \mathbf{y}_i \in F)\,.
\end{aligned}$$

Setting $\varepsilon_2 := \varepsilon_1/(2\|\mathbf{a}\|_\infty)$ and $N := N_{\varepsilon_2}$ yields the desired bound. The lemma thus follows. ◀

3 Bounding the coefficients of Branching Proofs

In this section, we show how to transform any branching proof $\mathcal{T}$ for a compact convex set $K \subseteq R\mathbb{B}_1^n$ into a branching proof $\mathcal{T}'$ having small coefficients with length $|\mathcal{T}'| = O(n|\mathcal{T}|)$.

The construction of $\mathcal{T}'$ is a two step process. In the first step, we substitute each integer disjunction given by $(\mathbf{a}, b)$ by an approximation $(\mathbf{a}', b')$ with coefficients of size $(nR)^{O(n^2)}$. This bounds $\langle \mathcal{T}' \rangle$ while keeping $|\mathcal{T}'| = |\mathcal{T}|$. We shall use the "iterated Diophantine approximation" technique introduced by Frank and Tardos [12] to construct $\mathbf{a}', b'$ from $\mathbf{a}, b$.

It is possible that the new inequalities are "stronger"; e.g., it is possible that for $\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{a}\mathbf{x} \leq b$ and $\mathbf{a}'\mathbf{x} \geq b' + 1 \Rightarrow_R \mathbf{a}\mathbf{x} \geq b + 1$. However, one cannot always ensure this, and in general we will only be able to guarantee that $\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{a}\mathbf{x} \leq b + \varepsilon$ and $\mathbf{a}'\mathbf{x} \geq b' + 1 \Rightarrow_R \mathbf{a}\mathbf{x} \geq b + 1 - \varepsilon$ for some "small" $\varepsilon > 0$. As explained in the introduction, the combined error from all the substitutions may render the continuous relaxations at the leaves nonempty. In a second step, we "fix-up" these newly feasible leaf nodes by adding $O(n)$ judiciously chosen CG cuts to arrive at infeasible sets, causing the $O(n)$ factor increase in $|\mathcal{T}'|$. These cuts will be derived from so-called valid substitution sequences (see Definition 24) of the original disjunctions in $\mathcal{T}$, which we construct together with the replacement disjunctions $\mathbf{a}'\mathbf{x} \leq b'$ or $\geq b' + 1$ described above.

From here until the end of subsection 3.1, we explain the first step, showing how to construct appropriate replacement disjunctions together with substitution sequences and how to compute the initial (partial) replacement tree $\mathcal{T}'$ from $\mathcal{T}$. In subsection 3.2, we explain

the second step, showing how to construct the requisite $O(n)$-size CP proof of infeasibility for each leaf node of $\mathcal{T}'$. Finally, in subsection 3.3, we give the proof of Theorem 1 which combines both steps.

We begin with the following lemma, which collects the properties of Diophantine approximations we will need to construct the replacement disjunctions and substitution sequences. In particular, part (ii) of the lemma will be used to choose the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k$ and right hand sides $b_1, \ldots, b_k$ inducing the pair of inequality sequences (we think of one as the "flipped" version of the other) $\mathbf{a}_1\mathbf{x} \leq b_1, \ldots, \mathbf{a}_k\mathbf{x} \leq b_k$ and $\mathbf{a}_1\mathbf{x} \geq b_1, \ldots, \mathbf{a}_{k-1}\mathbf{x} \geq b_{k-1}, \mathbf{a}_k\mathbf{x} \geq b_k + 1$ respectively used to approximate the left side $\mathbf{ax} \leq b$ and right side $\mathbf{ax} \geq b + 1$ of an initial disjunction. In this context, the disjunction $(\mathbf{a}, b)$ in the lemma will represent an initial disjunction $(\mathbf{a}, b)$ after "projecting out" $\mathbf{a}_1\mathbf{x} = b_1, \ldots, \mathbf{a}_i\mathbf{x} = b_i$, for some intermediate $i \in [k-1]$, and $\mathbf{a}', b'$ will correspond to the next $(\mathbf{a}_{i+1}, b_{i+1})$ in the sequence.

▶ **Lemma 22.** *For any vector $\mathbf{a} \in \mathbb{R}^n \setminus \{0\}, b \in \mathbb{R}, R, N \in \mathbb{N}$, let $\mathbf{a}'$ be a Diophantine approximation of $\mathbf{a}$ of precision N, and let $\alpha = \frac{\|\mathbf{a}\|_\infty}{\|\mathbf{a}'\|_\infty}$. Then the following statements hold:*

(i) *$\forall b' \in \mathbb{R}, \mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{ax} \leq \alpha\left(b' + \frac{R}{N}\right)$ and symmetrically, $\mathbf{a}'\mathbf{x} \geq b' \Rightarrow_R \mathbf{ax} \geq \alpha\left(b' - \frac{R}{N}\right)$.*

(ii) *When $\frac{R}{N} < \frac{1}{4}, \alpha \geq 2$, we can uniquely set $b' \in \mathbb{Z}$ according to exactly one of following cases:*

- *(non-R-dominating case): $-R\|\mathbf{a}\|_\infty - 1 < b < R\|\mathbf{a}\|_\infty$ and $\exists$ unique $b' \in \mathbb{Z}, |b'| \leq R\|\mathbf{a}'\|_\infty$,*

$$\text{such that } (b, b+1) \cap \left[\alpha\left(b' - \frac{R}{N}\right), \alpha\left(b' + \frac{R}{N}\right)\right] \neq \emptyset.$$

- *(R-dominating case): $\exists$ unique $b' \in \mathbb{Z}, -R\|\mathbf{a}'\|_\infty \leq b' \leq R\|\mathbf{a}'\|_\infty - 1$,*

$$\text{such that } (b, b+1) \subseteq \left(\alpha\left(b' + \frac{R}{N}\right), \alpha\left(b' + 1 - \frac{R}{N}\right)\right),$$

or

$$b \geq R\|\mathbf{a}\|_\infty, b' = R\|\mathbf{a}'\|_\infty,$$

or

$$b + 1 \leq -R\|\mathbf{a}\|_\infty, b' = -R\|\mathbf{a}'\|_\infty - 1.$$

Furthermore, in the R-dominating case we have that

$$\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{ax} \leq b \text{ and } \mathbf{a}'\mathbf{x} \geq b' + 1 \Rightarrow_R \mathbf{ax} \geq b + 1.$$

Proof.

(i) By definition of $\mathbf{a}'$, $\left\|\frac{\mathbf{a}}{\alpha} - \mathbf{a}'\right\|_\infty < 1/N$. We have for any $b' \in \mathbb{Z}$:

$$\|\mathbf{x}\|_1 \leq R, \mathbf{a}'\mathbf{x} \leq b' \Rightarrow \frac{\mathbf{a}}{\alpha}\mathbf{x} \leq b' + (\frac{\mathbf{a}}{\alpha} - \mathbf{a}')\mathbf{x} \leq b' + \left\|\frac{\mathbf{a}}{\alpha} - \mathbf{a}'\right\|_\infty \|\mathbf{x}\|_1 \leq b' + \frac{R}{N}.$$

Summarizing, we have that

$$\|\mathbf{x}\|_1 \leq R, \mathbf{a}'\mathbf{x} \leq b' \Rightarrow \mathbf{ax} \leq \alpha\left(b' + \frac{R}{N}\right).$$

By a symmetric argument, we also have

$$\|\mathbf{x}\|_1 \leq R, \mathbf{a}'\mathbf{x} \geq b' \Rightarrow \mathbf{ax} \geq \alpha\left(b' - \frac{R}{N}\right).$$

(ii) When $\frac{R}{N} < \frac{1}{4}$, the intervals of the form $I(b') := [\alpha\left(b' - \frac{R}{N}\right), \alpha\left(b' + \frac{R}{N}\right)]$, $b' \in \mathbb{Z}$, are pairwise disjoint. In fact, when $\alpha \geq 2$, they are more than unit distance apart. This implies that the interval $(b, b+1)$ cannot intersect more than one of the intervals $I(b')$, $b' \in \mathbb{Z}$.
Let us now suppose $-R\|\mathbf{a}\|_\infty - 1 < b < R\|\mathbf{a}\|_\infty$. We now show that only $b' \in [-R\|\mathbf{a}\|_\infty, R\|\mathbf{a}\|_\infty] \cap \mathbb{Z}$ need be considered in this case.
For $b' = -R\|\mathbf{a}'\|_\infty$, we have $I(b') = [R\alpha\left(\|\mathbf{a}'\|_\infty - \frac{1}{N}\right), R\alpha\left(\|\mathbf{a}'\|_\infty + \frac{1}{N}\right)]$. The left end point $-R\|\mathbf{a}\|_\infty - \frac{R\alpha}{N}$ of $I(-R\|\mathbf{a}'\|_\infty)$ lies to the left of $b+1$ on the real line because

$$-R\|\mathbf{a}\|_\infty - \frac{R\alpha}{N} < -R\|\mathbf{a}\|_\infty < b+1.$$

Similarly the right end point $R\|\mathbf{a}\|_\infty + \frac{R\alpha}{N}$ of $I(R\|\mathbf{a}'\|_\infty)$ lies to the right of b as

$$R\|\mathbf{a}\|_\infty + \frac{R\alpha}{N} > R\|\mathbf{a}\|_\infty > b.$$

Thus, either $(b, b+1)$ intersects some $I_{b'}$ for $b' \in [-R\|\mathbf{a}'\|_\infty, R\|\mathbf{a}'\|_\infty] \cap \mathbb{Z}$ or it lies in between two such consecutive intervals $I_{b'}, I_{b'+1}$: these are the non-dominating and dominating cases respectively.
In the dominating case for $b \in (-R\|\mathbf{a}\|_\infty - 1, R\|\mathbf{a}\|_\infty)$, the fact that

$$(b, b+1) \subseteq \left(\alpha\left(b' + \frac{R}{N}\right), \alpha\left(b' + 1 - \frac{R}{N}\right)\right)$$

implies $b \geq \alpha\left(b' + \frac{R}{N}\right)$. Applying part (i) we see that

$$\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{a}\mathbf{x} \leq \alpha\left(b' + \frac{R}{N}\right) \Rightarrow \mathbf{a}\mathbf{x} \leq b.$$

On the other side, $b+1 \leq \alpha\left(b' + 1 - \frac{R}{N}\right)$ gives

$$\mathbf{a}'\mathbf{x} \geq b'+1 \Rightarrow_R \mathbf{a}\mathbf{x} \geq \alpha\left(b' + 1 - \frac{R}{N}\right) \Rightarrow \mathbf{a}\mathbf{x} \geq b+1.$$

Now let us consider the situation where $b \geq R\|\mathbf{a}\|_\infty$. Then $\forall \mathbf{x} \in R\mathbb{B}_1^n$ we have $\mathbf{a}\mathbf{x} \leq \|\mathbf{a}\|_\infty \|\mathbf{x}\|_1 \leq \|\mathbf{a}\|_\infty R \leq b$. Since this inequality holds for every vector in $R\mathbb{B}_1^n$, we have that $\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{a}\mathbf{x} \leq b$ $\forall b' \in \mathbb{R}$ and in particular for $b' = R\|\mathbf{a}'\|_\infty$. Furthermore, for $b' = R\|\mathbf{a}'\|_\infty$, $\mathbf{a}'\mathbf{x} \geq b'+1$ does not hold for any $\mathbf{x} \in R\mathbb{B}_1^n$ and thus $\mathbf{a}'\mathbf{x} \geq b'+1 \Rightarrow_R \mathbf{a}\mathbf{x} \geq b+1$.
The symmetric reasoning applies to case $b+1 \leq -R\|\mathbf{a}\|_\infty$. Setting $b' = -R\|\mathbf{a}\|_\infty - 1$, we firstly have that $\mathbf{a}\mathbf{x} \geq b+1$ is a valid inequality for $R\mathbb{B}_1^n$ and hence $\mathbf{a}'\mathbf{x} \geq b'+1 \Rightarrow_R \mathbf{a}\mathbf{x} \geq b+1$ trivially. Secondly, the system $\mathbf{a}'\mathbf{x} \leq b', \|\mathbf{x}\|_1 \leq R$ is empty and hence $\mathbf{a}'\mathbf{x} \leq b' \Rightarrow_R \mathbf{a}\mathbf{x} \leq b$ trivially as well. ◀

▶ Remark 23. In the sequel, we will say that $(\mathbf{a}', b')$ R-dominates or R-non-dominates $(\mathbf{a}, b)$ when the corresponding case holds in Lemma 22. We will also drop label R- when R is clear from context.

When applying the above lemma to an initial disjunction $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1$, the R-dominating case above is a scenario in which the naive replacement of the disjunction $\mathbf{a}\mathbf{x} \leq b$ or $\geq b+1 \to \mathbf{a}'\mathbf{x} \leq b'$ or $\geq b'+1$ by the "first pass" Diophantine approximation does the job. Indeed, if every branching decision in $\mathcal{T}$ was dominated by its first pass Diophantine approximation, then the naive replacements would be sufficient to obtain a branching proof with bounded coefficients.

Since domination does not always occur at the first level of approximation for an original disjunction $\mathbf{ax} \leq b$ or $\geq b+1$, we will require the use of an substitution sequence $\mathbf{a}_1\mathbf{x} \leq b_1, \ldots, \mathbf{a}_1\mathbf{x} \leq b_k$ as described previously. The exact properties needed from this sequence as well as the algorithm to compute it are provided in the next subsection. At a high level, we continue creating additional levels of approximation until the dominating case occurs. More precisely, for every level $l \in [k]$, we will reduce the inequality $\mathbf{ax} \leq b$ by subtracting non-negative combinations of the equalities $\mathbf{a}_i\mathbf{x} = b_i$, $i \in [l-1]$, to get a "remainder inequality" $\hat{\mathbf{a}}_l\mathbf{x} \leq \hat{b}_l$. The remainder is then given as input to Lemma 22 to get next level approximator $\mathbf{a}_l\mathbf{x} \leq b_l$. The final iteration k will correspond to the first time where the dominating case occurs (i.e., all previous iterations are non-dominating).

Since we are interested in approximating not just an inequality but a disjunction, it will be crucial that (non-)domination is well-behaved with respect to both sides of the disjunction. For this purpose, we will heavily make use of the following "flip-symmetry" in the definition of the R-domination and R-non-domination. Namely, if $(\mathbf{a}', b')$ dominates $(\mathbf{a}, b)$, then $(-\mathbf{a}', -b'-1)$ dominates $(-\mathbf{a}, -b-1)$, and if $(\mathbf{a}', b')$ non-dominates $(\mathbf{a}, b)$ then $(-\mathbf{a}', -b')$ non-dominates $(-\mathbf{a}, -b-1)$. One can easily check that these symmetries follow directly from simple manipulations of the definitions. These symmetries are what will allow us to conclude that the substitution sequence $\mathbf{a}_1\mathbf{x} \leq b_1, \ldots, \mathbf{a}_k\mathbf{x} \leq b_k$ and its flipped version $\mathbf{a}_1\mathbf{x} \geq b_1, \ldots, \mathbf{a}_k\mathbf{x} \geq b_k + 1$ will yield good approximations to $\mathbf{ax} \leq b$ and $\mathbf{ax} \geq b+1$ respectively.

3.1 Step 1: Replacing Large Coefficient Branches by Small Coefficient Approximations

Given a branching proof $\mathcal{T}$ of infeasibility for $K \subseteq RB_1^n$, where $R \in \mathbb{N}$, we begin the construction of the replacement proof $\mathcal{T}'$ as follows:

We let $\mathcal{T}'$ be a tree with vertex set V' containing a vertex v' for each $v \in V[\mathcal{T}]$, and an edge $e' = (v', w')$ for each $e = (v, w) \in E[\mathcal{T}]$. For each internal node $v \in V[T]$ with children v_l, v_r, we compute through Algorithm 1 (see below) an approximation $(\mathbf{a}_{v'}, b_{v'})$ of the disjunction $(\mathbf{a}_v, b_v)$ at v of precision $R, N := 10nR, M := (10nR)^{n+2}$. We label the left edge $e_l = (v', v'_l)$ in $\mathcal{T}'$ by $\mathbf{a}_{v'}\mathbf{x} \leq b_{v'}$ and the right edge (v', v'_r) by $\mathbf{a}_{v'}\mathbf{x} \leq -b_{v'} - 1$ (equivalently, $\mathbf{a}_{v'}\mathbf{x} \geq b' + 1$).

In this first phase of construction, note that $\mathcal{T}'$ retains the same tree structure as $\mathcal{T}$. Furthermore, note that the output of Algorithm 1 must serve equally well to approximate $\mathbf{a}_v\mathbf{x} \leq b_v$ and $\mathbf{a}_v\mathbf{x} \geq b_v + 1$ for $v \in \mathcal{T}$.

The required properties of the replacements of the form $\mathbf{ax} \leq b \to \mathbf{a}'\mathbf{x} \leq b'$ are collected in the definition of a valid substitution sequence defined below. A valid substitution sequence of $(\mathbf{a}, b)$ of precision R, N, M consists of, along with the approximations $\mathbf{a}', b'$, auxiliary information in the form of integers $k, b_1, b_2 \ldots b_k$, integer vectors $\mathbf{a}_1, \ldots \mathbf{a}_k$ and nonnegative reals $\gamma_1, \ldots \gamma_k$. While this auxiliary information is not included in the labels of $\mathcal{T}'$, it is computed by Algorithm 1 (and hence its existence is guaranteed). Furthermore, the existence of the valid substitution sequence is crucial for second part of the tree construction (see subsection 3.2), where the inequalities from the substitution sequences are used to construct CP proofs of infeasibility for the (possibly non-empty) leaves of $\mathcal{T}'$ above.

▶ **Definition 24** (Valid Substitution Sequence). *We define a* valid substitution sequence *of an integer inequality* $\mathbf{ax} \leq b$, $\mathbf{a} \in \mathbb{Z}^n \setminus \{0\}$, $b \in \mathbb{Z}$ *of precision* $R, N, M \in \mathbb{N}$ *to be a list*

$$(\mathbf{a}', b', k, \mathbf{a}_1, b_1, \gamma_1, \ldots, \mathbf{a}_k, b_k, \gamma_k := 0),$$

where $k \in [n+1]$ *and* $\mathbf{a}', \mathbf{a}_i \in \mathbb{Z}^n, b', b_i \in \mathbb{Z}$, $\gamma_i \in \mathbb{R}_+$, *for* $i \in [k]$, *satisfying:*

1. $\|\mathbf{a}'\|_\infty \le N^n M^{n+1}, |b'| \le RN^n M^{n+1}$ *and* $\|\mathbf{a}_i\|_\infty \le 11nN^n, |b_i| \le R\|\mathbf{a}_i\|_\infty + 1, i \in [k]$.
2. *For* $l \in [k-1]$, *we have*

$$\mathbf{a}'\mathbf{x} \le b', \mathbf{a}_i\mathbf{x} = b_i, \forall\ i \in [l-1] \Rightarrow_R \mathbf{a}_l\mathbf{x} < b_l + 1.$$

3. *For* $l \in [k]$, *we have*

$$\mathbf{a}'\mathbf{x} \le b', \mathbf{a}_i\mathbf{x} = b_i, \forall\ i \in [l-1] \Rightarrow_R \mathbf{a}\mathbf{x} \le b + \gamma_l.$$

4. *For* $l \in [k-1]$, *we have*

$$\mathbf{a}_l\mathbf{x} \le b_l - 1, \mathbf{a}_i\mathbf{x} = b_i, \forall\ i \in [l-1] \Rightarrow_R \mathbf{a}\mathbf{x} \le b - n\gamma_l.$$

▶ Remark 25. In light of property 3, the terms γ_i are measures of precision for our approximation of $\mathbf{a}\mathbf{x} \le b$. If $l_1 < l_2$, property 3 when applied to l_2 assumes more statements than when applied l_1. Intuitively, this suggests that the implications should also be stronger (or at least not weaker). That is, one would expect $\gamma_{l_2} \ge \gamma_{l_1}$. Indeed, this assumption can be made without loss of generality. More precisely, if $(a', b', k, \mathbf{a}_1, b_1, \gamma_1, \ldots, \mathbf{a}_k, b_k, \gamma_k := 0)$ is a valid substitution, then so is $(a', b', k, \mathbf{a}_1, b_1, \bar{\gamma}_1, \ldots, \mathbf{a}_k, b_k, \bar{\gamma}_k)$, where $\bar{\gamma}_i := \min_{j\in[i]} \gamma_i$.

Algorithm 1 LongToShort$(\mathbf{a}, b, R, N, M)$.

Input: $\mathbf{a} \in \mathbb{Z}^n, b \in \mathbb{Z}, R, N, M \in \mathbb{N}$ such that $\frac{R}{N} < \frac{1}{4}$.
Output: $\mathbf{a}' \in \mathbb{Z}^n, b' \in \mathbb{Z}, k \in \mathbb{N}$, and $\mathbf{a}_i \in \mathbb{Z}^n, b_i \in \mathbb{Z}, \gamma_i \in \mathbb{R}_+$, $i \in [k]$, satisfying:
1. $(\mathbf{a}', b', k, \mathbf{a}_1, b_1, \gamma_1, \ldots, \mathbf{a}_k, b_k, \gamma_k)$ is a valid substitution sequence of $\mathbf{a}\mathbf{x} \le b$ of precision R, M, N.
2. $(-\mathbf{a}', -b' - 1, k, -\mathbf{a}_1, -b_1, \gamma_1, \ldots, -\mathbf{a}_{k-1}, -b_{k-1}, \gamma_{k-1}, -\mathbf{a}_k, -b_k - 1, \gamma_k)$ is a valid substitution sequence of $-\mathbf{a}\mathbf{x} \le -b - 1$ of precision R, M, N.

initialize $\hat{\mathbf{a}}_1 = \mathbf{a}, \hat{b}_1 = b, j = 1$;
while $\|\hat{\mathbf{a}}_j\|_\infty > 10nN^n$ **do**
  Set $\mathbf{a}_j$ as a Diophantine approximation of $\hat{\mathbf{a}}_j$ of precision N;
  Apply Lemma 22 part (ii) to $\hat{\mathbf{a}}_j, \hat{b}_j, \mathbf{a}_j, R, N$ to obtain b_j;
  if $(\mathbf{a}_j, b_j)$ *dominates* $(\hat{\mathbf{a}}_j, \hat{b}_j)$ **then**
    Set $k = j, \gamma_k = 0, \mathbf{a}' = \sum_{i=1}^k M^{k-i}\mathbf{a}_i$ and $b' = \sum_{i=1}^k M^{k-i}b_i$;
    return $\mathbf{a}', b', k, \mathbf{a}_i, b_i, \gamma_i, i \in [k]$;
  Set $\alpha_j = \frac{\|\hat{\mathbf{a}}_j\|_\infty}{\|\mathbf{a}_j\|_\infty}, \gamma_j = \frac{2\alpha_j}{5n}$;
  Set $\hat{\mathbf{a}}_{j+1} = \hat{\mathbf{a}}_j - \alpha_j\mathbf{a}_j, \hat{b}_{j+1} = \hat{b}_j - \alpha_j b_j$;
  Increment j;
Set $k = j, \gamma_k = 0, \mathbf{a}_k = \mathbf{a} - \sum_{i=1}^{k-1} \lfloor \alpha_i \rceil \mathbf{a}_i, \tilde{b}_k = b - \sum_{i=1}^{k-1} \lfloor \alpha_i \rceil b_i$;
Set $b_k = \begin{cases} \tilde{b}_k & : \quad -R\|\mathbf{a}_k\|_\infty - 1 < \tilde{b}_k < R\|\mathbf{a}_k\|_\infty \\ -R\|\mathbf{a}_k\|_\infty - 1 & : \quad \tilde{b}_k \le -R\|\mathbf{a}_k\|_\infty - 1 \\ R\|\mathbf{a}_k\|_\infty & : \quad R\|\mathbf{a}_k\|_\infty \le \tilde{b}_k \end{cases}$;
Set $\mathbf{a}' = \sum_{i=1}^k M^{k-i}\mathbf{a}_i$ and $b' = \sum_{i=1}^k M^{k-i}b_i$;
return $\mathbf{a}', b', k, \mathbf{a}_i, b_i, \gamma_i, i \in [k]$;

▶ **Lemma 26.** *Algorithm 1 with input* $\mathbf{a}, b, R, N, M$ *such that* $N = 10nR, M = (10nR)^{n+2}$ *terminates within* $k \leq n+1$ *iterations and outputs valid substitution sequences of* $\mathbf{a}, b$ *and* $-\mathbf{a}, -b-1$ *of precision* R, N, M*.*

Proof. We begin by showing that the number of coordinates of $\hat{\mathbf{a}}_{k+1}$ that are zero is strictly greater than that of $\hat{\mathbf{a}}_k$. At iteration j, let p be such that $|(\hat{\mathbf{a}}_j)_p| = \|\hat{\mathbf{a}}_j\|_\infty$. As $\mathbf{a}_j$ is the Diophantine approximation of $\hat{\mathbf{a}}_j$, we know that $|(\frac{\hat{\mathbf{a}}_j}{\alpha_j} - \mathbf{a}_j)_p| = |(\frac{\|\mathbf{a}_j\|_\infty}{\|\hat{\mathbf{a}}_j\|_\infty}\hat{\mathbf{a}}_j)_p - (\mathbf{a}_j)_p| < 1/10nR$. By assumption, $(\frac{\hat{\mathbf{a}}_j}{\|\hat{\mathbf{a}}_j\|_\infty})_p = \pm 1$. Thus $(\frac{\|\mathbf{a}_j\|_\infty}{\|\hat{\mathbf{a}}_j\|_\infty}\hat{\mathbf{a}}_j)_p \in \mathbb{Z}$, and so $(\frac{\|\mathbf{a}_j\|_\infty}{\|\hat{\mathbf{a}}_j\|_\infty}\hat{\mathbf{a}}_j)_p = (\mathbf{a}_j)_p$. As a result, $(\hat{\mathbf{a}}_{j+1})_p = 0$. As observed in remark 1.2, any zero entry of $\hat{\mathbf{a}}_j$ is also zero for $\hat{\mathbf{a}}_{j+1}$.

By this reasoning, either Algorithm 1 terminates with $k \leq n$, or $\hat{\mathbf{a}}_{n+1} = 0$. The while loop terminates as $\|\hat{\mathbf{a}}_{n+1}\|_\infty \leq 10nN^n$. This proves that Algorithm 1 terminates within $n+1$ iterations.

We now show that Algorithm 1 outputs valid substitution sequences. For this purpose, we first prove below that $(\mathbf{a}', b', k, \mathbf{a}_1, b_1, \gamma_1, \ldots, \mathbf{a}_k, b_k, \gamma_k)$ is a valid substitution sequence of $\mathbf{ax} \leq b$ of precision R, M, N. After this, we will argue that the flipped version of this sequence yields a valid substitution of $-\mathbf{ax} \leq -b-1$ using the symmetries the algorithm.

1. When Algorithm 1 returns from line (7), we have $\|\mathbf{a}_i\|_\infty \leq N^n$, $i \in [k]$, since every $\mathbf{a}_i$ is the result of Diophantine approximation of precision N. Furthermore, since every b_i, $i \in [k]$, is then the output of Lemma 22 part (ii), we also have $|b_i| \leq R\|\mathbf{a}_i\|_\infty + 1 \leq RN^n + 1, i \in [k]$. Therefore,

$$\|\mathbf{a}'\|_\infty \leq \sum_{i=1}^{k} M^{k-i}\|\mathbf{a}_i\|_\infty \leq N^n \sum_{i=1}^{k} M^{k-i} \leq N^n \sum_{i=0}^{n} M^i = N^n \frac{M^{n+1}-1}{M-1} \leq N^n M^{n+1}.$$

Similarly for b', using $R, N \geq 1$ and $M \geq 3$,

$$|b'| \leq \sum_{i=1}^{k} M^{k-i}|b_i| \leq (RN^n+1)\sum_{i=1}^{k} M^{k-i} = (RN^n+1)\frac{M^{n+1}-1}{M-1} \leq RN^n M^{n+1}.$$

When Algorithm 1 returns from line (14), $\|\mathbf{a}_i\|_\infty \leq N^n$ for every $i \in [k-1]$ and $\|\hat{\mathbf{a}}_k\| \leq 10N^n$. As in the previous case, we also have $|b_i| \leq R\|\mathbf{a}_i\|_\infty + 1$, $i \in [k]$. Note that for $i = k$, this is enforced on line (12) of the algorithm. Furthermore, b_k is indeed an integer since $R \in \mathbb{N}$ and $\tilde{b}_k, \|\mathbf{a}_k\|_\infty \in \mathbb{Z}$ by construction. To bound $\|\mathbf{a}_k\|_\infty$, we first note that

$$\|\mathbf{a}_k - \hat{\mathbf{a}}_k\|_\infty = \left\|\sum_{i=1}^{k-1}(\alpha_i - \lfloor\alpha_i\rceil)\mathbf{a}_i\right\|_\infty \leq \sum_{i=1}^{k-1}\|(\alpha_i - \lfloor\alpha_i\rceil)\mathbf{a}_i\|_\infty \leq (k-1)N^n \leq nN^n.$$

Since $\|\hat{\mathbf{a}}_k\|_\infty \leq 10nN^n$, we get that

$$\|\mathbf{a}_k\|_\infty \leq \|\hat{\mathbf{a}}_k\|_\infty + \|\mathbf{a}_k - \hat{\mathbf{a}}_k\|_\infty \leq 10nN^n + nN^n = 11nN^n.$$

To bound $\|\mathbf{a}'\|_\infty$, by the triangle inequality

$$\begin{aligned}\|\mathbf{a}'\|_\infty &\leq \sum_{i=1}^{k-1} M^{k-i}\|\mathbf{a}_i\|_\infty + \|\mathbf{a}\|_k \leq N^n(\sum_{i=1}^{n} M^i + 11n) \\ &= N^n(\frac{M^{n+1}-1}{M-1} + 11n - 1) \leq N^n(\frac{2M^{n+1}}{M-1} + 11n - 1) \leq N^n M^{n+1},\end{aligned} \tag{3.1}$$

where it can be easily be checked that the last inequality holds for $M = (10nR)^{n+2}$ and $R, n \geq 1$. The bound on $|b'|$ is computed in a manner similar to (3.1):

$$|b'| \leq \sum_{i=1}^{k} M^{k-i}(R\left\|\mathbf{a}_i\right\|_\infty + 1) \leq RN^n(\frac{2M^{n+1}}{M-1} + 11n - 1) \leq RN^nM^{n+1}.$$

2. Let $l \in [k-1]$. When $\mathbf{a}_i\mathbf{x} = b_i, \forall\ i \in [l-1]$, we have that

$$\mathbf{a}'\mathbf{x} \leq b' \Leftrightarrow \sum_{i=l}^{k} M^{k-i}\mathbf{a}_i\mathbf{x} \leq \sum_{i=l}^{k} M^{k-i}b_i \Leftrightarrow \mathbf{a}_l\mathbf{x} + \sum_{i=l+1}^{k} M^{l-i}\mathbf{a}_i\mathbf{x} \leq b_l + \sum_{i=l+1}^{k} M^{l-i}b_i.$$

By the proof of part 1, $\left\|\mathbf{a}_i\right\|_\infty \leq N^n, b_i \leq R\left\|\mathbf{a}_i\right\|_\infty + 1$, for $i \in [k-1]$. Using these bounds, we get that

$$\mathbf{a}_l\mathbf{x} + \sum_{i=l+1}^{k} M^{l-i}\mathbf{a}_i \leq b_l + \sum_{i=l+1}^{k} M^{l-i}b_i \Rightarrow_R \mathbf{a}_l\mathbf{x} \leq b_l + \sum_{i=l+1}^{k} M^{l-i}b_i + R\left\|\sum_{i=l+1}^{k} M^{l-i}\mathbf{a}_i\right\|_\infty. \tag{3.2}$$

The error in the last term is bounded by

$$\sum_{i=l+1}^{k} M^{l-i}b_i + R\left\|\sum_{i=l+1}^{k} M^{l-i}\mathbf{a}_i\right\|_\infty \leq (2R+1)N^n\left(\sum_{i=1}^{k-l} M^{-i}\right) \leq \frac{(2R+1)N^n}{M-1} \leq \frac{1}{10n}, \tag{3.3}$$

where the last inequality is easily checked for $M = (10nR)^{n+2} = N^{n+2}$ and $n, R \in \mathbb{N}$. Combining (3.2) and (3.3), we conclude that

$$\mathbf{a}'\mathbf{x} \leq b', \mathbf{a}_i\mathbf{x} = b_i, i \in [l-1] \Rightarrow_R \mathbf{a}_l\mathbf{x} \leq b_l + \frac{1}{10n} < b_l + 1, \tag{3.4}$$

as needed.

3. We first deal with the case $l = k$. We have $\mathbf{a}_k + \sum_{i=1}^{k-1} M^{k-i}\mathbf{a}_i = \mathbf{a}'$ and $b_k + \sum_{i=1}^{k-1} M^{k-i}b_i = b'$. So if $\mathbf{a}_i\mathbf{x} = b_i$, $\forall\ i \in [k-1]$, we have that $\mathbf{a}'\mathbf{x} \leq b' \Leftrightarrow \mathbf{a}_k\mathbf{x} \leq b_k$.
When Algorithm 1 returns from line (14), we have $\mathbf{a}_k + \sum_{i=1}^{k-1} \lfloor \alpha_i \rceil \mathbf{a}_i = \mathbf{a}$ and $\tilde{b}_k + \sum_{i=1}^{k-1} \lfloor \alpha_i \rceil b_i = b$ by construction. By the same argument as above, under the assumption $\mathbf{a}_i\mathbf{x} = b_i, \forall\ i \in [k-1]$, we have that $\mathbf{a}_k\mathbf{x} \leq \tilde{b}_k \Leftrightarrow \mathbf{a}\mathbf{x} \leq b$. It thus suffices to show that $\mathbf{a}_k\mathbf{x} \leq b_k \Rightarrow_R \mathbf{a}_k\mathbf{x} \leq \tilde{b}_k$, recalling that $\gamma_k = 0$. This proceeds in an analoguous fashion to the analysis of the dominating case in Lemma 22 part (ii). Firstly, by our choice of b_k on line (12), if $-R\left\|\mathbf{a}_k\right\|_\infty - 1 < \tilde{b}_k < R\left\|\mathbf{a}_k\right\|_\infty$ then $b_k = \tilde{b}_k$, so this case is trivial. If $\tilde{b}_k \geq R\left\|\mathbf{a}_k\right\|_\infty$, then $b_k = R\left\|\mathbf{a}_k\right\|$ and $\mathbf{a}_k\mathbf{x} \leq \tilde{b}_k$ is valid inequality for $R\mathbb{B}_1^n$. Thus, the implication $\mathbf{a}_k\mathbf{x} \leq b_k \Rightarrow_R \mathbf{a}_k\mathbf{x} \leq \tilde{b}_k$ is again trivial. Lastly, if $\tilde{b}_k \leq -R\left\|\mathbf{a}_k\right\|_\infty - 1$, then $b_k = -R\left\|\mathbf{a}_k\right\|_\infty - 1$ and the system $\mathbf{a}_k\mathbf{x} \leq b_k, \left\|\mathbf{x}\right\|_\infty \leq R$ is empty. In particular, $\mathbf{a}_k\mathbf{x} \leq b_k \Rightarrow_R \mathbf{a}_k\mathbf{x} \leq \tilde{b}_k$, as needed.
Next, when the Algorithm 1 returns from line (7), by the guarantees of the R-dominating case in Lemma 22 part (ii), we have that

$$\mathbf{a}_k\mathbf{x} \leq b_k \Rightarrow_R \hat{\mathbf{a}}_k\mathbf{x} \leq \hat{b}_k.$$

Similarly to the previous case, under the assumption $\mathbf{a}_i\mathbf{x} = b_i$, $\forall\ i \in [k-1]$, we have that $\mathbf{a}'\mathbf{x} \leq b' \Leftrightarrow \mathbf{a}_k\mathbf{x} \leq b_k$ and $\mathbf{a}\mathbf{x} \leq b \Leftrightarrow \hat{\mathbf{a}}_k\mathbf{x} \leq \hat{b}_k$. The desired implication, $\mathbf{a}'\mathbf{x} \leq b'$, $\mathbf{a}_i\mathbf{x} = b_i$, $\forall\ i \in [k-1] \Rightarrow_R \mathbf{a}\mathbf{x} \leq b$, thus follows.

Now suppose $l \in [k-1]$. From (3.4) in part 2, we have that

$$\mathbf{a}'\mathbf{x} \leq b', \mathbf{a}_i\mathbf{x} = b_i, i \in [l-1] \Rightarrow_R \mathbf{a}_l\mathbf{x} \leq b_l + \frac{1}{10n}.$$

By lemma 22 part (i) applied to $\hat{\mathbf{a}}_{l+1}$ and $\mathbf{a}_{l+1}$,

$$\mathbf{a}_l\mathbf{x} \leq b_l + \frac{1}{10n} \Rightarrow_R \hat{\mathbf{a}}_l\mathbf{x} \leq \alpha_l\left(b_l + \frac{1}{10n} + \frac{R}{N}\right). \tag{3.5}$$

Since $l \in [k-1]$, $\mathbf{a}_l, b_l$ non-dominates $\hat{\mathbf{a}}_l, \hat{b}_l$ and thus by Lemma 22 part (ii),

$$(\hat{b}_l, \hat{b}_l + 1) \cap \left[\alpha_l\left(b_l - \frac{R}{N}\right), \alpha_l\left(b_l + \frac{R}{N}\right)\right] \neq \emptyset.$$

In particular, we get that

$$\alpha_l\left(b_l + \frac{R}{N}\right) = \alpha_l\left(b_l - \frac{R}{N}\right) + \alpha_l(\frac{2R}{N}) \leq \hat{b}_l + 1 + \alpha_l(\frac{2}{10n}). \tag{3.6}$$

Using $\alpha_l = \frac{\|\hat{\mathbf{a}}_l\|_\infty}{\|\mathbf{a}_l\|_\infty} \geq \frac{10nN^n}{N^n} = 10n$ combined with (3.6), we get that

$$\begin{aligned}\alpha_l\left(b_l + \frac{1}{10n} + \frac{R}{n}\right) &\leq \hat{b}_l + 1 + \alpha_l\left(\frac{3}{10n}\right) \leq \hat{b}_l + \alpha_l\left(\frac{1}{10n} + \frac{3}{10n}\right) \\ &= \hat{b}_l + \alpha_l(\frac{2}{5n}) = \hat{b}_l + \gamma_l,\end{aligned} \tag{3.7}$$

where the last equality follows by definition of γ_l. Finally, under the assumption that $\mathbf{a}_i\mathbf{x} = b_i, \forall i \in [l-1]$, observe that for any $\delta \in \mathbb{R}$, we have that

$$\hat{\mathbf{a}}_l\mathbf{x} \leq \hat{b}_l + \delta \Leftrightarrow \mathbf{a}\mathbf{x} \leq b + \delta. \tag{3.8}$$

The desired implication $\mathbf{a}'\mathbf{x} \leq b', \mathbf{a}_i\mathbf{x} = b_i, \forall i \in [l-1] \Rightarrow_R \mathbf{a}\mathbf{x} \leq b + \gamma_l$ now follows directly from the above combined with (3.4),(3.5) and (3.7).

4. Repeating the argument from part 3 starting from (3.5) with $\mathbf{a}_l\mathbf{x} \leq b_l - 1$, we have that

$$\mathbf{a}_l\mathbf{x} \leq b_l - 1 \Rightarrow_R \hat{\mathbf{a}}_l\mathbf{x} \leq \alpha_l(b_l - 1 + \frac{R}{N}). \tag{3.9}$$

Using (3.7) in part 3, we see that

$$\alpha_l(b_l - 1 + \frac{R}{N}) \leq \hat{b}_l - \alpha_l + \alpha_l(\frac{2}{5n} - \frac{1}{10n}) = \hat{b}_l - \alpha_l(1 - \frac{3}{10n}). \tag{3.10}$$

Recalling $\gamma_l := \alpha_l(\frac{2}{5n})$, observe that $n\gamma_l = \alpha_l(\frac{2}{5}) \leq \alpha_l(1 - \frac{3}{10n})$ since $n \geq 1$. Combining together with (3.9), (3.10) and (3.8) from part 3, we conclude that

$$\mathbf{a}\mathbf{x} \leq b_l - 1, \mathbf{a}_i\mathbf{x} = b_i, \forall i \in [l-1] \Rightarrow_R \mathbf{a}\mathbf{x} \leq b - \alpha_l(1 - \frac{3}{10n}) \leq b - n\gamma_l,$$

as needed.

We conclude the proof by showing that

$$(-\mathbf{a}', -b' - 1, k, -\mathbf{a}_1, -b_1, \gamma_1, \ldots, -\mathbf{a}_{k-1}, -b_{k-1}, \gamma_{k-1}, -\mathbf{a}_k, -b_k - 1, \gamma_k)$$

is a valid substitution sequence of $-\mathbf{a}\mathbf{x} \leq -b - 1$ of precision R, M, N. We have already proved that Algorithm 1 correctly outputs a valid substitution sequence of $\mathbf{a}, b$, so we are done if we show that $(-\mathbf{a}', -b' - 1, k, -\mathbf{a}_1, -b_1, \gamma_1, \ldots, -\mathbf{a}_{k-1}, -b_{k-1}, \gamma_{k-1}, -\mathbf{a}_k, -b_k - 1, \gamma_k)$ could have been output by Algorithm 1 upon input $-\mathbf{a}, -b - 1$.

If $\mathbf{a}_i$ is a Diophantine approximation of $\hat{\mathbf{a}}_i$, then $-\mathbf{a}_i$ is a Diophantine approximation of $-\hat{\mathbf{a}}_i$. Referring to Remark 23 as our next step: when $j < k$, $(\mathbf{a}_j, b_j)$ non-dominates $(\hat{\mathbf{a}}_j, \hat{b}_j)$, $(-\mathbf{a}_j, -b_j)$ also non-dominates $(-\hat{\mathbf{a}}_j, -\hat{b}_j - 1)$. As a ratio of norms, the α_j values are identical for both executions of Algorithm 1.

If Algorithm 1 with input $\mathbf{a}, b$ returned from line (7), then the algorithm with input $-\mathbf{a}, -b-1$ must also return from line (7). This is also a consequence of Remark 23: if $(\mathbf{a}_k, b_k)$ dominates $(\hat{\mathbf{a}}_k, \hat{b}_k)$, then $(-\mathbf{a}_k, -b_k - 1)$ dominates $(-\hat{\mathbf{a}}_k, -\hat{b}_k - 1)$ and the algorithm returns with the expected valid substitution sequence of $-\mathbf{a}, -b - 1$.

If Algorithm 1 with input $\mathbf{a}, b$ returned from line (14), this means $\|\hat{\mathbf{a}}_k\|_\infty \leq 10nN^n$. Running Algorithm 1 with input $-\mathbf{a}, -b - 1$ would give $-\hat{\mathbf{a}}_k$, also obviously of small norm. Line 11 would consequently give $-\mathbf{a} - \sum_{i=1}^{k-1} \lfloor \alpha_i \rceil (-\mathbf{a}_i) = -\mathbf{a}_k$ and $-b-1-\sum_{i=1}^{k-1} \lfloor \alpha_i \rceil (-b_i) = -\tilde{b}_k - 1$. It now suffices to check that output of line (12) given $-\tilde{b}_k - 1$ is $-b_k - 1$, recalling that b_k is the output of line (12) on input $\tilde{b}_k$. This follows by direct inspection, noting that it is analoguous to the "flip-symmetry" of Lemma 22 part (ii). ◀

Replacing branches with large coefficients with their valid approximations reduces their bit-size, since valid approximations have coefficients of size $O(n^2)\log(nR)$. However, we are not yet done. We do not yet have a valid branching proof as the convex sets $K_{v'}$ associated to leaf nodes v' of $\mathcal{T}'$ are not necessarily empty. We deal with this in Step 2.

3.2 Step 2: Adding Chvátal-Gomory (CG) Cuts to Trim the Leaves

We now show how to add CG cuts at each leaf of the current replacement tree $\mathcal{T}'$ for $\mathcal{T}$, whose construction is described in the previous section, to ensure that all the leaf nodes in the final tree have empty continuous relaxations. The final tree will simply simulate the effect of the CG cuts applied to the leaves of $\mathcal{T}'$ using additional branching decisions (see the proof of Theorem 1 in the next section).

Recall from the last section, that every leaf node $v \in \mathcal{T}$ has an associated leaf $v' \in \mathcal{T}'$ in the current replacement tree. The continuous relaxation for v' is $K_{v'} = P_{v'} \cap K$ (recall that unlike $K_v := P_v \cap K$, $K_{v'}$ need not be empty), where the inequalities defining $P_{v'}$ are derived from *valid substitution sequences* (as in Definition 24) of the original defining inequalities for P_v. Given this setup, our task is to add "low-weight" CG cuts to $P_{v'} \cap K$ to derive the empty set.

The main result of this section is a general procedure for deriving such CG cuts for any polyhedron P' induced by valid substitution sequences of the defining inequalities of a polyhedron P, where P satisfies $K \cap P = \emptyset$. The procedure will return a list of at most $2(n+1)$ CG cuts, which is responsible for the $O(n)$ factor blowup in the final tree size. Second, the normals of these CG cuts will all come from the substitution lists for the inequalities defining P, which ensures that they have low weight. The formal statement of this result is given below:

▶ **Theorem 27.** *Let $K \subseteq R\mathbb{B}_1^n$, $R \in \mathbb{N}$, be a compact convex set, and let $P = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}\mathbf{x} \leq \mathbf{b}\}$, $\mathsf{A} \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$, be a polyhedron satisfying $P \cap K = \emptyset$. For each defining inequality $\mathbf{a}_i\mathbf{x} \leq b_i$ of P, for $i \in [m]$, let $(\mathbf{a}'_i, b'_i, k_i, \mathbf{a}_{i,1}, b_{i,1}, \gamma_{i,1}, \ldots, \mathbf{a}_{i,k_i}, b_{i,k_i}, \gamma_{i,k_i})$ be a valid substitution sequence of precision $R, N := 10nR, M := (10nR)^{n+2}$. Let $P' = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}'\mathbf{x} \leq \mathbf{b}'\}$ be the corresponding "substitution" polyhedron, where $\mathsf{A}' \in \mathbb{Z}^{m \times n}$ has rows $\mathbf{a}'_1, \ldots, \mathbf{a}'_m$ and $\mathbf{b}' \in \mathbb{Z}^m$ has rows $b'_1, \ldots, b'_m$.*

Then, there exists an ordered list $\mathcal{L} := (\mathbf{a}_{j_1,p_1}, -\mathbf{a}_{j_1,p_1}, \ldots, \mathbf{a}_{j_l,p_l}, -\mathbf{a}_{j_l,p_l}) \subseteq \mathbb{Z}^n$, where $j_r \in [m]$ and $p_r \in [k_{j_r} - 1]$, $r \in [l]$, satisfying $\mathrm{CG}(K \cap P', \mathcal{L}) = \emptyset$ and $|\mathcal{L}| = 2l \leq 2(n+1)$.

Proof. To prove theorem 27, we give a procedure to construct such a list $\mathcal{L}$ in Algorithm 2. To prove the theorem, it thus suffices to prove the correctness of Algorithm 2.

Algorithm 2 Generate CG Cuts.

Input: $K, P := \mathsf{A}\mathbf{x} \leq \mathbf{b}, P' := \mathsf{A}'\mathbf{x} \leq \mathbf{b}', R, M, N \in \mathbb{N}$, $(\mathbf{a}'_i, b'_i, k_i, \mathbf{a}_{i,1}, b_{i,1}, \gamma_{i,1}, \ldots, \mathbf{a}_{i,k_i}, b_{i,k_i}, \gamma_{i,k_i})$, for $i \in [m]$, a valid substitution sequence of $\mathbf{a}_i\mathbf{x} \leq b_i$ of precision R, M, N, as in theorem 27.

Output: An ordered list $\mathcal{L} := (\mathbf{a}_{j_1,p_1}, -\mathbf{a}_{j_1,p_1}, \ldots, \mathbf{a}_{j_l,p_l}, -\mathbf{a}_{j_l,p_l})$ satisfying $\mathrm{CG}(K \cap P', \mathcal{L}) = \emptyset$ and $0 \leq l \leq n+1$.

1 **initialize** $\mathcal{L} = \emptyset$, $V = \mathbb{R}^n$, $p(i) = 1$, for $i \in [m]$, and $\boldsymbol{\varepsilon} = (\gamma_{1,p(1)}, \ldots, \gamma_{m,p(m)})$;
2 Define $P_{\boldsymbol{\varepsilon}} := \{\mathbf{x} \in \mathbb{R}^n : \mathsf{A}\mathbf{x} \leq \mathbf{b} + \boldsymbol{\varepsilon}\}$;
3 **while** $K \cap P_{\boldsymbol{\varepsilon}} \neq \emptyset$ *and* $V \neq \emptyset$ **do**
4 Apply Lemma 18 to $K, P, \boldsymbol{\varepsilon}$ to obtain $j_* \in [m]$ satisfying $\varepsilon_{j_*} > 0$ and $K \cap P_{\boldsymbol{\varepsilon} - (n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}} = \emptyset$;
5 Append vectors $\mathbf{a}_{j_*,p(j_*)}, -\mathbf{a}_{j_*,p(j_*)}$ to the list $\mathcal{L}$;
6 Update $V \leftarrow V \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} = b_{j_*,p(j_*)}\}$;
7 **for** j *from* 1 *to* m **do**
8 Increment $p(j)$ to the largest integer $p \in [k_j]$ satisfying $V \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j,i}\mathbf{x} = b_{j,i}, 1 \leq i < p\}$;
9 $\varepsilon_j \leftarrow \gamma_{j,p(j)}$;
10 **return** $\mathcal{L}$;

To begin, we first give a high level description of the algorithm and explain the key invariants it maintains. The algorithm proceeds in iterations, associated with runs of the while loop on line 3. At each iteration, we append the pair of CG cuts induced by $\mathbf{a}_{j,p}, -\mathbf{a}_{j,p}$, $j \in [m]$, $p \in [k_j - 1]$, from one of our substitution lists to the end $\mathcal{L}$. These cuts are chosen so that after adding them to $\mathcal{L}$, we can guarantee that $\mathrm{CG}(K \cap P', \mathcal{L})$ satisfies the equality $\mathbf{a}_{j,p}\mathbf{x} = b_{j,p}$.

We keep track of these learned equalities using the affine subspace $V \subseteq \mathbb{R}^n$, which is initialized as $V = \mathbb{R}^n$ at the beginning of the algorithm. The principal invariant needed to prove correctness of the algorithm is as follows: at the beginning of an iteration $l \geq 1$, $\mathcal{L}, V$ satisfy

$$(i) \quad \mathrm{CG}(K \cap P', \mathcal{L}) \subseteq V \quad \text{and} \quad \dim(V) \leq n - l + 1.$$

The condition on the dimension of V above will be achieved by ensuring that the new equality we add is not already implied by V. Precisely, the dimension of V will decrease by at least one at every iteration where we pass the while loop check. Using (i), at the beginning of iteration $l = n + 2$ (i.e., after $n + 1$ iterations) we will have that $\dim(V) \leq -1$ and hence $\mathrm{CG}(K \cap P', \mathcal{L}) \subseteq V = \emptyset$. In particular, the while loop check $V \neq \emptyset$ will fail and we will correctly terminate. Thus, assuming (i) holds, the algorithm always terminates after at most $n + 1$ iterations. Since we add only 2 CG cuts per iteration, the total number of cuts in the list $\mathcal{L}$ will be at most $2(n + 1)$ by the end the algorithm. To prove that (i) holds, we first introduce two other important invariants.

To keep track of the learned equalities in each substitution list, we keep a counter $p(j) \in [k_j]$, for $j \in [m]$. For the second invariant, the algorithm maintains that at the beginning of each iteration we have that

$$(ii) \quad V = \cap_{j \in [m]} \cap_{1 \leq i \leq p(j)-1} \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j,i}\mathbf{x} = b_{j,i}\},$$

and that each $p(j) \in [k_j]$,$j \in [m]$, is maximal subject to the above equality. That is, for each $j \in [m]$, V satisfies all the equalities $\mathbf{a}_{j,i}\mathbf{x} = b_{j,i}$ for $i \in [p(j)-1]$, and, if $p(j) < k_j$, V does not satisfy $\mathbf{a}_{j,p(j)}\mathbf{x} = b_{p(j)}$. The counters are initialized to $p(1) = \cdots = p(m) = 1$ corresponding to $V = \mathbb{R}^n$ (i.e., we have not yet learned any equalities), which indeed yields a maximal choice. That the affine space V can expressed in the above form is a simple consequence of how we update it on line (6). Namely, we only update V when we add the equality $\mathbf{a}_{j_*,p(j_*)}\mathbf{x} = b_{j_*,p(j_*)}$ to V on line (6). Note that since $\varepsilon_{j_*} > 0$ on line (4), we must have $p(j_*) < k_{j_*}$, since otherwise $\varepsilon_{j_*} = \gamma_{j_*,k_{j_*}} = 0$ (by definition of a valid substitution sequence). Thus, we only add an inequality $\mathbf{a}_{j,p}\mathbf{x} = b_{j,p}$, $j \in [m]$, to V if $1 \le p < k_j$ and if V satisfies $\mathbf{a}_{j,i}\mathbf{x} = b_{j,i}$ for all $i \in [p-1]$, as needed. Lastly, the required maximality is directly ensured by line (8). This proves that (ii) is indeed maintained. Note that under maximality, for each $j \in [m]$ such that $p(j) < k_j$, adding the equality $\mathbf{a}_{j,p(j)}\mathbf{x} = b_{p(j)}$ to V must reduce the dimension of V by at least one (more precisely, adding this equality either makes V empty or reduces its dimension by exactly 1). We will use this in the proof of (i).

With this notation, we may state the final invariant, which will be a direct consequence of the first two and the definition of a valid substitution sequence. Letting $\boldsymbol{\varepsilon} := (\gamma_{1,p(i)}, \ldots, \gamma_{m,p(m)})$ denote the "error level" for each constraint of P (note that this equality is maintained on line (9)), at the beginning of each iteration we maintain

$$(iii) \quad \mathrm{CG}(K \cap P', \mathcal{L}) \subseteq P_{\boldsymbol{\varepsilon}},$$

where $P_{\boldsymbol{\varepsilon}} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \le \mathbf{b} + \boldsymbol{\varepsilon}\}$. Crucially, invariant (iii) justifies the first termination condition $K \cap P_{\boldsymbol{\varepsilon}} = \emptyset$, since if this occurs $\mathrm{CG}(K \cap P', \mathcal{L}) \subseteq K \cap P_{\boldsymbol{\varepsilon}} = \emptyset$. Note that for a constraint $j \in [m]$, with $p(j) = k_j$, the effective error level $\varepsilon_j = \gamma_{j,k_j} = 0$ (by definition of valid substitution). That is, we have effectively "learned" the defining constraint $\mathbf{a}_j\mathbf{x} \le b_j$ for P for any $j \in [m]$ with $p(j) = k_j$. Clearly, once all the constraints of P have been learned, we will have $\mathrm{CG}(K \cap P', \mathcal{L}) \subseteq K \cap P = \emptyset$, where the last equality is by assumption.

We now show that (iii) is a consequence of (i) and (ii). Let $\mathcal{L}$,V,p and $\boldsymbol{\varepsilon}$ be the state at the beginning of some iteration $l \ge 1$, and assume that (i) and (ii) hold. Then, for each $j \in [m]$, we have that

$$\begin{aligned}
\mathrm{CG}(K \cap P', \mathcal{L}) &\subseteq K \cap P' \cap V \quad (\text{ by (i) and} \mathrm{CG}(K \cap P', \mathcal{L}) \subseteq K \cap P') && (3.11)\\
&\subseteq R\mathbb{B}_1^n \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}'_j\mathbf{x} \le b'_j, \mathbf{a}_{j,i}\mathbf{x} = b_{j,i}, \forall\, i \in [p(j)-1]\} && (3.12)\\
&\qquad (\text{ by (ii) and } K \subseteq R\mathbb{B}_1^n)\\
&\subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_j\mathbf{x} \le b_j + \gamma_{j,p(j)}\} \quad (\text{ by Definition 24 part 3. }).
\end{aligned}$$

Since the above holds for all $j \in [m]$, and $\varepsilon_j = \gamma_{j,p(j)}$, for $j \in [m]$, this proves invariant (iii).

Given the above, to prove correctness of algorithm it suffices to establish invariant (i). We now show that invariant (i) holds by induction on the iteration $l \ge 1$. Let $\mathcal{L}$,V,p and $\boldsymbol{\varepsilon}$ denote the state at the beginning of some iteration $l \le 1$ for which (i) holds. Note that (i) trivially holds for the base case $l = 1$ since $V = \mathbb{R}^n$. By the reasoning in the previous paragraphs, we also have that invariant (ii) and (iii) hold at the beginning of l. We must now show that (i) holds at the beginning of iteration $l+1$ under these assumptions. Clearly, we may assume that we pass the while loop check $K \cap P_\varepsilon \ne \emptyset$ and $V \ne \emptyset$, since otherwise there is nothing to prove.

Let $j_* \in [m]$ be the index satisfying $\varepsilon_{j_*} > 0$ and $K \cap P_{\boldsymbol{\varepsilon} - (n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}} = \emptyset$ as guaranteed by Lemma 18. This index indeed exists since we already checked that $K \cap P_{\boldsymbol{\varepsilon}} \ne \emptyset$. As argued for (ii), we also know that $p(j_*) < k_{j_*}$, which will ensure we have access to the required inequalities from the valid substitution sequence of $\mathbf{a}_{j_*}\mathbf{x} \le b_{j_*}$. Letting $P'_{\mathcal{L}} := \mathrm{CG}(K \cap P', \mathcal{L})$, to prove that (i) holds for $l+1$, it now suffices to show that

(a) $\mathrm{CG}(P'_{\mathcal{L}}, (\mathbf{a}_{j_*,p(j_*)}, -\mathbf{a}_{j_*,p(j_*)})) \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} = b_{j_*,p(j_*)}\}$,
(b) $\dim(V \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq b_{j_*,p(j_*)}\}) \leq \dim(V) - 1$.

As explained previously, (b) follows directly the maximality assumption in (ii). We may thus focus on (a). To begin, using (i) and (ii) and the same analysis as in (3.11), we see that

$$\begin{aligned} P'_{\mathcal{L}} &\subseteq R\mathbb{B}_1^n \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}'_{j_*}\mathbf{x} \leq b'_{j_*}, \mathbf{a}_{j_*,i}\mathbf{x} = b_{j_*,i}, \forall i \in [p(j_*) - 1]\} \\ &\subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} < b_{j_*,p(j_*)} + 1\}, \end{aligned}$$

where the last containment follows from Definition (24) part 2. In particular,

$$\sup_{\mathbf{x} \in P'_{\mathcal{L}}} \mathbf{a}_{j_*,p(j_*)}\mathbf{x} < b_{j_*,p(j_*)} + 1 \Rightarrow \lfloor \sup_{\mathbf{x} \in P'_{\mathcal{L}}} \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \rfloor \leq b_{j_*,p(j_*)}, \tag{3.13}$$

since $b_{j_*,p(j_*)} \in \mathbb{Z}$. From (3.13), we conclude that

$$\mathrm{CG}(P'_{\mathcal{L}}, \mathbf{a}_{j_*,p(j_*)}) \subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq b_{j_*,p(j_*)}\}. \tag{3.14}$$

From here, again using (i) and (ii), we have that

$$\begin{aligned} P'_{\mathcal{L}} \cap &\{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq b_{j_*,p(j_*)} - 1\} \\ &\subseteq R\mathbb{B}_1^n \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq b_{j,p(j_*)} - 1, \mathbf{a}_{j_*,i}\mathbf{x} = b_{j_*,i}, \forall\ i \in [p(j_*) - 1]\} \\ &\subseteq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*}\mathbf{x} \leq b_{j_*} - n\varepsilon_{j_*}\}, \end{aligned} \tag{3.15}$$

where the last containment follows from Definition (24) part 4 and $\varepsilon_{j_*} = \gamma_{j_*,p(j_*)}$. Noting that

$$P_{\boldsymbol{\varepsilon}} \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*}\mathbf{x} \leq b_{j_*} - n\varepsilon_{j_*}\} = P_{\boldsymbol{\varepsilon} - (n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}},$$

by the guarantees of Lemma 18, invariant (iii) and (3.15), we therefore have that

$$P'_{\mathcal{L}} \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq b_{j_*,p(j_*)} - 1\} \subseteq K \cap P_{\boldsymbol{\varepsilon} - (n+1)\varepsilon_{j_*}\mathbf{e}_{j_*}} = \emptyset.$$

In particular, we must have that

$$\sup_{\mathbf{x} \in P'_{\mathcal{L}}} -\mathbf{a}_{j_*,p(j_*)}\mathbf{x} < -b_{j_*,p(j_*)} + 1 \Rightarrow \lfloor \sup_{\mathbf{x} \in P'_{\mathcal{L}}} -\mathbf{a}_{j_*,p(j_*)}\mathbf{x} \rfloor \leq -b_{j_*,p(j_*)}, \tag{3.16}$$

since $-b_{j_*,p(j_*)} \in \mathbb{Z}$. From (3.16), we conclude that

$$\mathrm{CG}(P'_{\mathcal{L}}, -\mathbf{a}_{j_*,p(j_*)}) \subseteq \{\mathbf{x} \in \mathbb{R}^n : -\mathbf{a}_{j_*,p(j_*)}\mathbf{x} \leq -b_{j_*,p(j_*)}\}. \tag{3.17}$$

Property (a) now follows directly by combining (3.14) and (3.17). This concludes the proof of invariant (i) and the proof of correctness of the algorithm. ◀

3.3 Proof of Theorem 1

Let $N = 10nR, M = (10nR)^{n+2}$. Given $\mathcal{T}$, we construct $\mathcal{T}'$ a labeled binary tree with the same structure as that of $\mathcal{T}$ as described at the beginning of subsection 3.1. Recall that for each internal node $v \in \mathcal{T}$ with associated disjunction $(\mathbf{a}_v, b_v)$, we retrieve a pair of valid substitution sequences from $\texttt{LongToShort}(\mathbf{a}_v, b_v, R, N, M)$, yielding the precision R, N, M sequence $(\mathbf{a}'_v, b'_v, k_v, \mathbf{a}_{v,1}, b_{v,1}, \gamma_{v,1}, \ldots, \mathbf{a}_{v,k}, b_{v,k}, \gamma_{v,k})$ for $\mathbf{a}_v \leq b_v$ and the corresponding flip

(as in the output description of Algorithm 1) for $-\mathbf{a}_v\mathbf{x} \leq -b_v - 1$. For the corresponding node $v' \in \mathcal{T}'$, we create two children v'_l, v'_r and label the left edge (v', v'_l) with $\mathbf{a}'_v\mathbf{x} \leq b'_v$ and the right edge (v', v'_r) with $-\mathbf{a}'_v\mathbf{x} \leq -b'_v - 1$.

From the properties of a valid substitution sequence, we have that $\|\mathbf{a}'_v\|_\infty \leq N^n M^{n+1}$ and $|b'_v| \leq RN^n M^{n+1}$. The choice of $N = 10nR, M = (10nR)^{n+2}$ gives

$$\|\mathbf{a}'_v\|_\infty \leq (10nR)^n (10nR)^{(n+2)(n+1)} = (10nR)^{n^2+4n+2} \quad \text{and} \quad |b'_v| \leq R(10nR)^{n^2+4n+2}.$$

Both of these quantities are upper bounded by $(10nR)^{(n+2)^2}$. For $\mathbf{x} \in \mathbb{Z}^n$, $\langle \mathbf{x} \rangle \leq n + n\log(1 + \|\mathbf{x}\|_\infty)$, so the bit-size of each inequality is $O(n^3)\log(nR)$.

Consider an arbitrary leaf node $v' \in \mathcal{T}'$ with associated leaf node $v \in \mathcal{T}$. Observe that by construction

$$P_{v'} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}'_e\mathbf{x} \leq b'_e, e \in E[P_{\mathcal{T}'}(v')]\}$$

satisfies the hypotheses of Theorem 27, so that there exists a list, say $\mathcal{L}_{v'}$ of integer vectors such that $\mathrm{CG}(K \cap P', \mathcal{L}_{v'}) = \emptyset$. More precisely, $\mathcal{L}_{v'} = (\mathbf{a}_{j_1,p_1}, -\mathbf{a}_{j_1,p_1}, \ldots, \mathbf{a}_{j_l,p_l}, -\mathbf{a}_{j_l,p_l})$, where $l \leq (n+1)$ and $\mathbf{a}_{j_r,p_r}, j_r \in [m_v], p_r \in [k_j - 1]$, $r \in [l]$, are taken from the valid substitution sequences of precision R, N, M of the inequalities in the system $\mathsf{A}_v\mathbf{x} \leq \mathbf{b}_v$, $\mathsf{A}_v \in \mathbb{Q}^{m_v \times n}$, $\mathbf{b}_v \in \mathbb{R}^{m_v \times n}$, defining P_v. Note that by the properties of an R, M, N valid substitution sequence, $\|\mathbf{a}_{j_r,p_r}\|_\infty \leq 11nN^n, |b_{j_r,p_r}| \leq R11nN^n + 1$, $r \in [l]$, and hence via the same argument as above each $(\mathbf{a}_{j,r}, b_{j,r})$ can be described using $O(n^2 \log_2(nR))$ bits.

We now explain how to extend $\mathcal{T}'$ to a valid branching proof. For each leaf node $v' \in \mathcal{T}'$, we will build a branching proof of infeasibility for $K_{v'}$ of length $O(n)$ which simulates the effect of the CG cuts in $\mathcal{L}_{v'}$. By appending these sub-branching proofs to $\mathcal{T}'$ below each leaf node v', the extended $\mathcal{T}'$ clearly becomes a valid branching proof for K having length at most $|\mathcal{T}'| = O(n)|\mathcal{T}|$ by construction.

The construction of the subtree at v' using $\mathcal{L}_{v'} = (\mathbf{a}_{j_1,p_1}, -\mathbf{a}_{j_1,p_1}, \ldots, \mathbf{a}_{j_l,p_l}, -\mathbf{a}_{j_l,p_l})$ (as above) proceeds as follows. Starting from v', we create two children v'_l, v'_r, and label the edge (v', v'_l) with the inequality $\mathbf{a}_{j_1,p_1}\mathbf{x} \leq b_{j_1,p_1}$ and the edge (v', v'_r) with the inequality $\mathbf{a}_{j_1,p_1}\mathbf{x} \geq b_{j_1,p_1} + 1$. Recall that by the definition of a CG cut, the continuous relaxation $K_{v'_r}$ at the right child v'_r is now empty. The construction now proceeds inductively on v'_l using the sublist $(-\mathbf{a}_{j_1,p_1}, \ldots, \mathbf{a}_{j_l,p_l}, -\mathbf{a}_{j_l,p_l})$. Note that for every cut in $\mathcal{L}'$, we add a left and right child to the current left-most leaf of the partially constructed subtree, for which the continuous relaxation of the newly added right child is always empty. At the end of the construction, it is easy to see that the left-most leaf of the constructed subtree has $\mathrm{CG}(K_{v'}, \mathcal{L}')$ as its continuous relaxation, which is empty by assumption. From here, we immediately get that the constructed subtree yields a valid branching proof of infeasibility for $K_{v'}$, and that the number of nodes in the subtree distinct from v' is exactly $2|\mathcal{L}_{v'}| \leq 4(n+1)$. Furthermore, we may bound the bit-size of this subtree by $O(n^3 \log_2(nR))$, since it has $O(n)$ nodes and every edge is labeled with an inequality of bit-size $O(n^2 \log_2(nR))$.

To bound the total bit-size $\langle \mathcal{T}' \rangle$ of the final branching proof $\mathcal{T}'$, we combine the bit-size bound from the subtrees above together with the total bit-size of all the replacement disjunctions of the form $a'_v\mathbf{x} \leq b'_v$ or $\geq b'_v + 1$ (as above) labeling the outgoing edges of nodes in $\mathcal{T}'$ associated with internal nodes of $\mathcal{T}$. Given that each disjunction $a'_v \leq b'_v$ or $\geq b'_v + 1$ requires $O(n^3 \log_2(nR))$ bits as explained above, their total bit-size is bounded by $O(n^3 \log_2(nR)|\mathcal{T}|)$. Furthermore, the bit-size contribution from all the subtrees in $\mathcal{T}'$ associated with leaf nodes of $\mathcal{T}$ is $O(n^3 \log_2(nR)|\mathcal{T}|)$, since the number of these subtrees is bounded by $|\mathcal{T}|$ and each has bit-size $O(n^3 \log_2(nR))$ as explained above. Thus, the total bit-size $\langle \mathcal{T}' \rangle = O(n^3 \log_2(nR)|\mathcal{T}|)$ as needed.

3.4 Proof of Corollary 2

The primary tools for this proof will be the following well-known facts pertaining to the bit-size of linear programs: (see [24] Chapter 10 for a thorough treatment):

▶ **Lemma 28.** *Let* $\mathsf{A} \in \mathbb{Q}^{m\times n}, \mathbf{b} \in \mathbb{Q}^m, \mathbf{c} \in \mathbb{Q}^n$.

1. *For* $\mathbf{c} \in \mathbb{Q}^n$, *if* $\max\{\mathbf{cx} : \mathsf{Ax} \leq \mathbf{b}\}$ *is finite, then it has size at most* $4(\langle \mathsf{A}, \mathbf{b}\rangle + \langle \mathbf{c}\rangle)$.
2. *If the system* $\boldsymbol{\lambda}\mathsf{A} = 0, \boldsymbol{\lambda} \geq 0$ *and* $\boldsymbol{\lambda}\mathbf{b} < 0$ *is feasible, then there exists a solution* $\boldsymbol{\lambda}$ *of with bit-size* $\langle\lambda\rangle = O(n\langle A\rangle)$.

▶ Remark 29. Part 2 of the above lemma in fact corresponds to a bound on the bit-size of a generator $\boldsymbol{\lambda}$ of the relevant extreme ray of the cone $\boldsymbol{\lambda}\mathsf{A} = 0, \boldsymbol{\lambda} \geq 0$, where we note that a Farkas certificate of infeasibility for $\mathsf{Ax} \leq \mathbf{b}$ (if it exists) can always be chosen to be an extreme ray of this cone. This is also the reason why the bit-size bound does not in fact depend on $\langle\mathbf{b}\rangle$.

Proof of Corollary 2. Since $K = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{Cx} \leq \mathbf{d}\}$ is a polytope (i.e., bounded), we may invoke lemma 28 part 1 to conclude that

$$\max_{\mathbf{x}\in P} \|\mathbf{x}\|_1 = \max_{\mathbf{y}\in\{-1,1\}^n} \max\{\mathbf{yx} : \mathsf{Cx} \leq \mathbf{d}\} \leq 2^{4(L+2n)}$$

using that $\langle\mathbf{y}\rangle \leq 2n$, for $\mathbf{y} \in \{-1, 1\}^n$, and that $|a| \leq 2^{\langle a\rangle} \forall a \in \mathbb{Q}$. Therefore, $K \subseteq R\mathbb{B}_1^n$ for $R = 2^{4L+8n}$. Theorem 1 applied with $R = 2^{4L+8n}$ already gives us a $\mathcal{T}'$ with $|\mathcal{T}'| \leq O(n)|\mathcal{T}|$, and the bound on $\|\cdot\|_\infty$ for integer vectors gives the bit-size of every inequality $\langle \mathbf{a}_e\mathbf{x} \leq b_e\rangle \leq O(n^3)\log(n(R+2)) = O(n^3L)$. As a result, the branching proof $\mathcal{T}'$ has size $\langle\mathcal{T}'\rangle = n^{O(1)}L|\mathcal{T}|$. However, $\mathcal{T}'$ is not yet a *certified* branching proof.

We are done if we can add to each leaf node $v' \in \mathcal{T}'$ a Farkas certificate $\boldsymbol{\lambda}_{v'}$ of small size. Recall the continuous relaxation at v' in $\mathcal{T}'$ is $K_{v'} = \{\mathbf{x} \in \mathbb{R}^n : \mathsf{Cx} \leq \mathbf{d}, \mathsf{A}_{v'} \leq \mathbf{b}'\}$. By assumption, we know that $K_{v'} = \emptyset$, and thus by Farkas's Lemma there exists $\boldsymbol{\lambda}_{v'} := (\boldsymbol{\lambda}_{v',1}, \boldsymbol{\lambda}_{v',2}) \geq 0$ such that $\boldsymbol{\lambda}_{v',1}\mathsf{C} + \boldsymbol{\lambda}_{v',2}\mathsf{A}_{v'} = 0$ and $\boldsymbol{\lambda}_{v',1}\mathbf{d} + \boldsymbol{\lambda}_{v',2}\mathbf{b} < 0$. Thus, by lemma 28 part 2, there exists a solution $\boldsymbol{\lambda}_{v'}$ whose bit-complexity is upper bounded by $O(n\langle\mathsf{C}, \mathsf{A}_{v'}\rangle)$. This quantity may be large since we have not controlled the number of rows $\mathsf{A}_{v'}$. By Caratheodory's theorem however, there exists a solution $\boldsymbol{\lambda}_{v'}$ with at most $n+1$ non-zero entries. Therefore, we can restrict our attention to a subset of rows of C and $\mathsf{A}_{v'}$ of cardinality at most $n+1$. As argued above, by Theorem 1 each row of $\mathsf{A}_{v'}$ has bit-size at most $O(n^3\log_2(nR)) = O(n^3L)$, and by assumption $\langle\mathsf{C}\rangle \leq L$. Thus, by restricting to the appropriate sub-system, the bit-length of the non-zero entries of $\boldsymbol{\lambda}_{v'}$ can be bounded by $O(n((n+1)n^3L + L)) = O(n^5L)$, as needed.

Since the number of nodes in $|\mathcal{T}'| = O(n|\mathcal{T}|)$, the combined bit-size of the Farkas certificates above is at most $O(n^6L|\mathcal{T}|)$. This dominates the contribution of the disjunctions to the bit-size of $\mathcal{T}'$, which by Theorem 1 is $O(n^3L|\mathcal{T}|)$. Thus, the certified version of has size $\langle\mathcal{T}'\rangle = O(n^6L|\mathcal{T}|)$, as needed. ◀

4 Simulating Enumerative Branching Proofs by Cutting Planes

In this section, we prove that enumerative branching proofs can be simulated by CP, and give an application to Tseitin formulas (section 4.1).

To begin, we first extend the lifting lemma (Lemma 20) to a sequence of CG cuts. This will allow us to use induction on subtrees of an enumerative branching proof.

▶ **Lemma 30** (Lifting Sequences of CG cuts). *Let $K \subseteq \mathbb{R}^n$ be a non-empty compact set. Let $\mathbf{c} \in \mathbb{Z}^n$, $F := F_K(\mathbf{c})$ and assume that $h_K(\mathbf{c}) \in \mathbb{Z}$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{Z}^n$. Then, there exists $n_1, \ldots, n_k \geq 0$ such that*

$$\mathrm{CG}(K, (\mathbf{a}_1 + n_1\mathbf{c}, \ldots, \mathbf{a}_k + n_k\mathbf{c})) \cap H^{=}_K(\mathbf{c}) = \mathrm{CG}(F, (\mathbf{a}_1, \ldots, \mathbf{a}_k)) \ .$$

Proof. We prove the statement by induction on i. For $i = 0$, there are no CG cuts to apply and the statement becomes $K \cap H^{=}_K(\mathbf{c}) = F$, which follows by definition. Now assume $1 \leq i \leq k$ with induction hypothesis

$$K_{i-1} \cap H^{=}_K(\mathbf{c}) = F_{i-1} \tag{4.1}$$

where

$$K_{i-1} := \mathrm{CG}(K, (\mathbf{a}_1 + n_1\mathbf{c}, \ldots, \mathbf{a}_{i-1} + n_{i-1}\mathbf{c})) \quad \text{and} \quad F_{i-1} := \mathrm{CG}(F, (\mathbf{a}_1, \ldots, \mathbf{a}_{i-1})).$$

We must prove the existence of $n_i \geq 0$ such that (4.1) holds for i. Firstly, if $F_{i-1} = \emptyset$, then regardless of the choice of $n_i \geq 0$, both the sets F_i and $K_i \cap H^{=}_K(\mathbf{c})$ will be empty since they are both contained in $F_{i-1} = \emptyset$. In particular, we may set $n_i = 0$ and maintain the desired equality.

So assume $F_{i-1} \neq \emptyset$. From here, since $\emptyset \neq F_{i-1} \subseteq F$, where we recall that F is the set maximizers of $\mathbf{c}$ in K, and $F_{i-1} \subseteq K_{i-1} \subseteq K$, we have that

$$h_K(\mathbf{c}) \geq h_{K_{i-1}}(\mathbf{c}) \geq h_{F_{i-1}}(\mathbf{c}) = h_K(\mathbf{c}) \in \mathbb{Z}.$$

In particular, $H^{=}_{K_{i-1}}(\mathbf{c}) = H^{=}_K(\mathbf{c})$. Therefore, by the induction hypothesis (4.1)

$$K_{i-1} \cap H^{=}_{K_{i-1}}(\mathbf{c}) = K_{i-1} \cap H^{=}_K(\mathbf{c}) = F_{i-1} \ ,$$

that, F_{i-1} is the set of maximizers of $\mathbf{c}$ in K_{i-1}. Furthermore, since K_{i-1} is the intersection of K with closed halfspaces and K is compact, K_{i-1} is also compact. Therefore, we may apply Lemma 20 to choose $n_i \geq 0$ satisfying

$$H^{\mathrm{cg}}_{K_{i-1}}(\mathbf{a}_i + n_i\mathbf{c}) \cap H^{=}_K(\mathbf{c}) = H^{\mathrm{cg}}_{F_{i-1}}(\mathbf{a}_i) \cap H^{=}_K(\mathbf{c}). \tag{4.2}$$

We use the above n_i to define

$$K_i := K_{i-1} \cap H^{\mathrm{cg}}_{K_{i-1}}(\mathbf{a}_i + n_i\mathbf{c}) = \mathrm{CG}(K, (\mathbf{a}_1 + n_1\mathbf{c}, \ldots, \mathbf{a}_i + n_i\mathbf{c})).$$

Intersecting both sides of (4.2) with K_{i-1}, we conclude that

$$\begin{aligned} H^{\mathrm{cg}}_{K_{i-1}}(\mathbf{a}_i + n_i\mathbf{c}) \cap K_{i-1} \cap H^{=}_K(\mathbf{c}) = H^{\mathrm{cg}}_{F_{i-1}}(\mathbf{a}_i) \cap K_{i-1} \cap H^{=}_K(\mathbf{c}) \Leftrightarrow \\ K_i \cap H^{=}_K(\mathbf{c}) = H^{\mathrm{cg}}_{F_{i-1}}(\mathbf{a}_i) \cap F_{i-1} \Leftrightarrow \\ K_i \cap H^{=}_K(\mathbf{c}) = F_i, \end{aligned}$$

as needed. The lemma thus follows. ◀

We are now ready to prove the main result of this section, which shows that enumerative branching proofs can be simulated by CP.

Proof of Theorem 4. Our procedure for converting enumerative branching proofs to CP proofs is given by Algorithm 3. The proof of correctness of the procedure will yield the theorem:

▷ Claim 31. Given an enumerative branching proof $\mathcal{T}$ of integer infeasibility for a compact convex set $K \subseteq \mathbb{R}^n$, Algorithm 3 correctly outputs a list $\mathcal{L} = (\mathbf{a}_1, \ldots, \mathbf{a}_N) \in \mathbb{Z}^n$ satisfying $\mathrm{CG}(K, \mathcal{L}) = \emptyset$ and $|\mathcal{L}| := N \leq 2|\mathcal{T}| - 1$.

Proof.

Algorithm Outline. We first describe the algorithm at a high level and then continue with a formal proof. The procedure traverses the tree $\mathcal{T}$ in order, visiting the children of each node from right to left. We explain the process starting from the root node $r \in \mathcal{T}$. To begin, we examine its branching direction $\mathbf{a}_r \in \mathbb{Z}^n$ and bounds $l_r \leq u_r$ satisfying

$$\{\mathbf{a}_r\mathbf{x} : \mathbf{x} \in K\} \subseteq [l_r, u_r],$$

recalling that r has a child r_b for each $b \in [l_r, u_r] \cap \mathbb{Z}$.

Starting at r, the procedure adds CG cuts to "chop off" the children of r moving from right to left. In particular, it alternates between adding the CG induced by $\mathbf{a}_r$ to K, which will either make K empty or push the hyperplane $H^{=}_{\mathbf{a}_r,b}$, where $b := h_K(\mathbf{a}_r)$, to the next child of r, and recursively adding CG cuts induced by the subtree $\mathcal{T}_{r_b}$ rooted at the child r_b of r. The cuts computed on the subtree $\mathcal{T}_{r_b}$ will be used to chop off the face $K \cap H^{=}_{\mathbf{a}_r,b}$ from K, which will require lifting cuts from the face to K using Lemma 30. Once the face has been removed, we add the CG cut induced by $\mathbf{a}_r$ to move to the next child. The process continues until all the children have been removed and K is empty.

Algorithm 3 EnumToCP(K, $\mathcal{T}$).

Input: Compact convex set $K \subseteq \mathbb{R}^n$, Enumerative branching proof $\mathcal{T}$ for K.
Output: List $\mathcal{L}$ of CG cuts satisfying $\mathrm{CG}(K, \mathcal{L}) = \emptyset$ and $|\mathcal{L}| \leq 2|\mathcal{T}| - 1$.

1 **initialize** $\mathcal{L} = \emptyset$, $r \leftarrow$ root of $\mathcal{T}$;
2 **if** $K = \emptyset$ **then**
3 **return** $\emptyset$;
4 Retrieve branching direction $\mathbf{a}_r \in \mathbb{Z}^n$ and bounds $l_r \leq u_r$;
5 $K \leftarrow \mathrm{CG}(K, \mathbf{a}_r)$;
6 $\mathcal{L} \leftarrow (\mathbf{a}_r)$;
7 **while** $l_r \leq h_K(\mathbf{a}_r)$ **do**
8 $b \leftarrow h_K(\mathbf{a}_r)$;
9 $\mathcal{T}_{r_b} \leftarrow$ subtree of $\mathcal{T}$ rooted at r_b;
10 $N' \leftarrow$ `EnumToCP`$(F_K(\mathbf{a}_r), \mathcal{T}_{r_b})$;
11 $N \leftarrow$ Lift CG cuts in N' from $F_K(\mathbf{a}_r)$ to K using Lemma 30;
12 $K \leftarrow \mathrm{CG}(K, N)$;
13 $K \leftarrow \mathrm{CG}(K, \mathbf{a}_r)$;
14 Append $N, \mathbf{a}_r$ to $\mathcal{L}$;
15 **return** $\mathcal{L}$;

Analysis. We show correctness by induction on $|\mathcal{T}| \geq 1$. Let $r \in \mathcal{T}$ denote the root node with branching direction $\mathbf{a}_r \in \mathbb{Z}^n$ and bounds $l_r \leq u_r$.

We prove the base case $|\mathcal{T}| = 1$. If $K = \emptyset$, then no CG cuts are needed and clearly $0 \leq 2|\mathcal{T}| - 1 = 1$. If $K \neq \emptyset$, then letting r be the root node, we must have $[l_r, u_r] \cap \mathbb{Z} = \emptyset \Rightarrow \lfloor u_r \rfloor < l_r$. Since $h_K(\mathbf{a}_r) \in [l_r, u_r]$, the initializing CG cut we add on line 5 induced by $\mathbf{a}_r$ will make K empty. This follows since after the cut $h_K(\mathbf{a}_r) \leq \lfloor u_r \rfloor < l_r$. The algorithm thus correctly returns $\mathcal{L} = (\mathbf{a}_r)$, where $|\mathcal{L}| = 1 = 2|\mathcal{T}| - 1$, as needed.

Now assume that $|\mathcal{T}| \geq 2$ and that the algorithm is correct for all smaller trees. If $K = \emptyset$ or we do not enter the while loop on line 7, the algorithm correctly returns by the above analysis. So we now assume that $K \neq \emptyset$ and that the algorithm performs at least one iteration of the while loop.

Let K_0 denote the state of K at the beginning of the algorithm. Let K_i, b_i, $\mathcal{L}_i$, for $i \geq 1$, denote the state of K, b, $\mathcal{L}$ at the beginning of the i^{th} iteration of the while loop on line 7 and let N_i, $i \geq 1$, denote the state of $\mathcal{L}$ at the end the i^{th} iteration. Let $T \geq 1$ denote the last iteration (i.e., which passes the check of the while loop). By design of the algorithm, it is direct to check that $K_i = \mathrm{CG}(K_0, \mathcal{L}_i)$, $\mathcal{L}_{i+1} = (\mathcal{L}_i, N_i, \mathbf{a}_r)$ and that $b_i = h_{K_i}(\mathbf{a}_r)$, $\forall i \in [T]$, where we define $\mathcal{L}_{T+1}$ to be the list of CG cuts returned by the algorithm, $K_{T+1} := \mathrm{CG}(K_0, \mathcal{L}_{T+1}) = \emptyset$ and $b_{T+1} := h_{K_{T+1}}(\mathbf{a}_r) = -\infty$. Note also that $K_i \neq \emptyset$, $\forall i \in [T]$, since otherwise we would have terminated earlier.

To begin, we claim that

$$b_i \in [l_r, u_r] \cap \mathbb{Z}, \forall i \in [T]. \tag{4.3}$$

To see this, note first that for any compact convex set $\mathcal{C}$ either $\mathrm{CG}(\mathcal{C}, \mathbf{a}_r) = \emptyset$ and $h_{\mathrm{CG}(\mathcal{C},\mathbf{a}_r)}(\mathbf{a}_r) = -\infty$ or $\mathrm{CG}(\mathcal{C}, \mathbf{a}_r) \neq \emptyset$ and $h_{\mathrm{CG}(\mathcal{C},\mathbf{a}_r)}(\mathbf{a}_r) = \lfloor h_{\mathcal{C}}(\mathbf{a}_r) \rfloor \in \mathbb{Z}$, where the latter claim follows from convexity and compactness of $\mathrm{CG}(\mathcal{C}, \mathbf{a}_r)$. Since we apply the CG cut induced by $\mathbf{a}_r$ to K directly before the while loop and at the end of every iteration, we immediately get that $b_i = h_{K_i}(\mathbf{a}_r) \in \mathbb{Z}$, $i \in [T]$. Furthermore, since $\emptyset \neq K_i \subseteq K_0$, $\forall i \in [T]$, and $\{\mathbf{a}_r \mathbf{x} : \mathbf{x} \in K_0\} \subseteq [l_r, u_r]$, we must also have $b_i \in [l_r, u_r]$.

Given (4.3), for each $i \in [T]$, we see that r_{b_i} is indeed a child of r. Let $\mathcal{T}_{r_{b_i}}, i \in [T]$ denote the subtree of $\mathcal{T}$ rooted at r_{b_i}. We claim that

$$b_{i+1} \leq b_i - 1, i \in [T], \quad \text{and} \quad |N_i| \leq 2|\mathcal{T}_{r_{b_i}}| - 1, i \in [T]. \tag{4.4}$$

To see this, first recall that $\mathcal{T}_{r_{b_i}}$ is a branching proof for $K_0 \cap H^{=}_{\mathbf{a}_r, b_i}$. In particular, since $K_i \subseteq K_0$, $\mathcal{T}_{r_{b_i}}$ is also a valid branching proof for $K_i \cap H^{=}_{\mathbf{a}_r, b_i} = F_{K_i}(\mathbf{a}_r)$. By the induction hypothesis the call to `EnumToCP` $(F_{K_i}(\mathbf{a}_r), \mathcal{T}_{r_{b_i}})$ on line 10 therefore correctly returns a list N_i' of CG cuts satistyfing $\mathrm{CG}(F_{K_i}(\mathbf{a}_r), N_i') = \emptyset$ and $|N_i'| \leq 2|\mathcal{T}_{r_{b_i}}| - 1$. Furthermore, by Lemma 30, the lifting N_i of N_i' to K computed on line 11 satisfies $|N_i| = |N_i'| \leq 2|\mathcal{T}_{r_i}| - 1$, as needed, and $\mathrm{CG}(K_i, N_i) \cap H^{=}_{\mathbf{a}_r, b_i} = \mathrm{CG}(F_{K_i}(\mathbf{a}_r), N_i') = \emptyset$. Letting $K_i' = \mathrm{CG}(K_i, N_i)$, by compactness of K_i' we therefore must have $h_{K_i'}(\mathbf{a}_r) < b_i$. Recalling that $K_{i+1} = \mathrm{CG}(K_i', \mathbf{a}_r)$, we see that $b_{i+1} = h_{K_{i+1}}(\mathbf{a}_r) \leq \lfloor h_{K_i'}(\mathbf{a}_r) \rfloor \leq b_i - 1$, as needed.

From the above, we see that the procedure clearly terminates in finite time and returns a list $\mathcal{L}_{T+1}$ satisfying $\mathrm{CG}(K_0, \mathcal{L}_{T+1}) = \emptyset$. It remains to bound the size of $|\mathcal{L}_{T+1}|$. Since we add 1 CG cut before the while loop, and at iteration $i \in [T]$, we add $|N_i| + 1$ CG cuts, the total number of cuts is

$$1 + \sum_{i=1}^{T} |N_i| + 1 \leq 1 + \sum_{i=1}^{T} 2|\mathcal{T}_{r_{b_i}}| \leq 2|\mathcal{T}| - 1,$$

where the last inequality follows since the sum is over subtrees rooted at distinct children of r, noting that $|\mathcal{T}| = 1 + \sum_{b \in [l_r, u_r] \cap \mathbb{Z}} |\mathcal{T}_{r_b}|$. This completes the proof. ◁

◀

4.1 Upper Bounds for Tseitin Formulas

Proof of Theorem 3. As explained in the introduction, given Theorem 4, it suffices to show that the Beame et al [5] SP refutation is in fact enumerative. We thus describe their refutation briefly to make clear that this is indeed the case.

We start with a Tseitin formula indexed by a graph $G = (V, E)$, of maximum degree Δ, together with parities $l_v \in \{0, 1\}$, $v \in V$, satisfying $\sum_{v \in V} l_v \equiv 1 \mod 2$. We recall that the variables $\mathbf{x} \in \{0, 1\}^E$ index the corresponding subset of edges where the assignment $\mathbf{x}$ is a satisfying assignment iff $\sum_{e \in E : v \in e} x_e \equiv l_v \mod 2$, $\forall v \in V$.

The Beame et al refutation proceeds as follows. At the root node r, we first divide the vertex set $V = V_1^r \cup V_2^r$ arbitrarily into two parts of near-equal size. We then branch on the number of edges crossing the cut

$$x(E[V_1^r, V_2^r]) := \sum_{\{v_1,v_2\}\in E, v_1\in V_1^r, v_2\in V_2^r} x_{v_1,v_2} \in [0, |E[V_1^r, V_2^r]|].$$

Let c be the child with $x(E[V_1^r, V_2^r]) = b$. c chooses $i_c \in \{1, 2\}$ such that $\sum_{v\in V_{i_c}^r} l_v \neq b \mod 2$, corresponding the set of vertices still containing a contradiction. From here, again c partitions $V_{i_c}^r = V_1^c \cup V_2^c$ into two near-equal pieces. We now branch twice: we first branch on the number of edges crossing the cut $x(E[V_1^c, V_2^c])$, creating corresponding children, and at each such child, we branch on number of edges crossing the cut $x(E[V_1^c, V \setminus V_1^c])$. From here, every child c', two levels down from c, can decide which set of vertices V_1^c or V_2^c still contains a contradiction. The process continues in a similar way until we find a contradicting set corresponding to a single vertex v. At this point, one constructs a complete branching tree on all possible values of the edges outgoing from v. This completes the description.

It is clear from the description, that every branching decision is enumerative. As shown in Beame et al, the above SP refutation has length $2^{\Delta}(n\Delta)^{O(\log n)}$. Theorem 4 shows that one can convert it to a CP refutation of the same length. This completes the proof. ◀

References

1 Karen Aardal and Arjen K Lenstra. Hard equality constrained integer knapsacks. *Mathematics of operations research*, 29(3):724–738, 2004.

2 W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in R^n II: Application of K-convexity. *Discrete and Computational Geometry*, 16:305–311, 1996.

3 Evelyn Martin Lansdowne Beale and John A Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR*, 69(447-454):99, 1970.

4 Paul Beame. Proof complexity. In Steven Rudich and Avi Wigderson, editors, *Computational Complexity Theory*, volume 10 of *IAS/Park City Mathematics Series*, pages 199–246. American Mathematical Society, 2004.

5 Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing planes. In *9th Innovations in Theoretical Computer Science*, volume 94 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 10, 20. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

6 Vasek Chvatal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4):305–337, 1973.

7 W. Cook, C. R. Coullard, and Gy. Turán. On the complexity of cutting-plane proofs. *Discrete Appl. Math.*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.

8 Gérard Cornuéjols and Yanjun Li. Elementary closures for integer programs. *Operations Research Letters*, 28(1):1–8, 2001.

9 Daniel Dadush. *Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation*. PhD thesis, Georgia Institute of Technology, 2012.

10 Daniel Dadush, Santanu S Dey, and Juan Pablo Vielma. On the chvátal–gomory closure of a compact convex set. *Mathematical Programming*, 145(1-2):327–348, 2014.

11 Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

12 András Frank and Éva Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

13 Ralph Gomory. An outline of an algorithm for solving integer programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

14 Miroslav Karamanov and Gérard Cornuéjols. Branching on general disjunctions. *Mathematical Programming*, 128(1-2):403–436, 2011.

15 Jan Krajíček. Discretely ordered modules as a first-order extension of the cutting planes proof system. *The Journal of Symbolic Logic*, 63(4):1582–1596, 1998.

16 Bala Krishnamoorthy and Gábor Pataki. Column basis reduction and decomposable knapsack problems. *Discrete Optimization*, 6(3):242–270, 2009.

17 H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

18 Ashutosh Mahajan and Theodore K Ralphs. Experiments with branching using general disjunctions. In *Operations Research and Cyber-Infrastructure*, pages 101–118. Springer, 2009.

19 Jonathan H Owen and Sanjay Mehrotra. Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. *Computational optimization and applications*, 20(2):159–170, 2001.

20 Gábor Pataki and Mustafa Tural. Basis reduction methods. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

21 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.

22 M. Rudelson. Distances between non-symmetric convex bodies and the MM^*-estimate. *Positivity*, 4(2):161–178, 2000. doi:10.1023/A:1009842406728.

23 Wolfgang M. Schmidt. *Diophantine approximation*, volume 785 of *Lecture Notes in Mathematics*. Springer, Berlin, 1980.

24 Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986.

Geometric Rank of Tensors and Subrank of Matrix Multiplication

Swastik Kopparty
Rutgers University, Piscataway, NJ, USA
swastik.kopparty@gmail.com

Guy Moshkovitz
Institute for Advanced Study, Princeton, NJ, USA
DIMACS, Rutgers University, Piscataway, NJ, USA
guymoshkov@gmail.com

Jeroen Zuiddam
Institute for Advanced Study, Princeton, NJ, USA
jzuiddam@ias.edu

Abstract

Motivated by problems in algebraic complexity theory (e.g., matrix multiplication) and extremal combinatorics (e.g., the cap set problem and the sunflower problem), we introduce the geometric rank as a new tool in the study of tensors and hypergraphs. We prove that the geometric rank is an upper bound on the subrank of tensors and the independence number of hypergraphs. We prove that the geometric rank is smaller than the slice rank of Tao, and relate geometric rank to the analytic rank of Gowers and Wolf in an asymptotic fashion. As a first application, we use geometric rank to prove a tight upper bound on the (border) subrank of the matrix multiplication tensors, matching Strassen's well-known lower bound from 1987.

2012 ACM Subject Classification Mathematics of computing; Theory of computation

Keywords and phrases Algebraic complexity theory, Extremal combinatorics, Tensors, Bias, Analytic rank, Algebraic geometry, Matrix multiplication

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.35

Related Version A full version of the paper is available at `https://arxiv.org/abs/2002.09472`.

Funding *Swastik Kopparty*: Research supported in part by NSF grants CCF-1253886, CCF-1540634, CCF-1814409 and CCF-1412958, and BSF grant 2014359. Some of this research was done while visiting the Institute for Advanced Study.
Guy Moshkovitz: This work was conducted at the Institute for Advanced Study, enabled through support from the National Science Foundation under grant number CCF-1412958, and at DIMACS, enabled through support from the National Science Foundation under grant number CCF-1445755.
Jeroen Zuiddam: Research supported by the National Science Foundation under Grant Number DMS-1638352.

Acknowledgements We would like to thank Avi Wigderson for helpful conversations.

1 Introduction

Tensors play a central role in computer science and mathematics. Motivated by problems in algebraic complexity theory (e.g., the arithmetic complexity of matrix multiplication), extremal combinatorics (e.g., the cap set problem and the Erdős–Szemerédi sunflower problem) and quantum information theory (the resource theory of quantum entanglement), we introduce and study a new tensor parameter called *geometric rank*. Like the many widely studied notions of rank for tensors (rank, subrank, border rank, border subrank, flattening rank, slice rank, analytic rank), geometric rank of tensors generalizes the classical rank of matrices. In this paper, we:

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 35; pp. 35:1–35:21

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


- prove a number of basic properties and invariances of geometric rank,
- develop several tools to reason about, and sometimes exactly compute, the geometric rank,
- show intimate connections between geometric rank and the other important notions of rank for tensors,
- and as a simple application of the above, we answer an old question of Strassen by showing that the (border) subrank of $m \times m$ matrix multiplication is at most $\lceil 3m^2/4 \rceil$ (this is tight for border subrank; previously the border subrank of the matrix multiplication tensor was known to lie between $\frac{3}{4}m^2$ and $(1 - o(1))m^2$).

More generally, we believe that geometric rank provides an interesting new route to prove upper bounds on subrank of tensors (and hence independence numbers of hypergraphs). Such upper bounds are important in complexity theory in the context of matrix multiplication and barriers to matrix multiplication, and combinatorics in the context of specific natural hypergraphs (as in the cap set problem and the Erdős–Szemeredi sunflower problem).

1.1 Geometric rank

We define the geometric rank of a tensor as the codimension of the (possibly reducible) algebraic variety defined by the bilinear forms given by the slices of the tensor. Here we use the standard notions of dimension and codimension of affine varieties from algebraic geometry. That is, for any tensor $T = (T_{i,j,k})_{i,j,k} \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ with coefficients $T_{i,j,k}$ in an algebraically closed field $\mathbb{F}$ (e.g., the complex numbers $\mathbb{C}$) and with 3-slices $M_k = (T_{i,j,k})_{i,j} \in \mathbb{F}^{n_1 \times n_2}$ we define the geometric rank $\mathrm{GR}(T)$ as

$$\mathrm{GR}(T) = \mathrm{codim}\{(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid x^T M_1 y = \cdots = x^T M_{n_3} y = 0\}.$$

Viewing T as the trilinear map $T : \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \times \mathbb{F}^{n_3} \to \mathbb{F} : (x, y, z) \mapsto \sum_{i,j,k} T_{i,j,k}\, x_i y_j z_k$, we can equivalently write the geometric rank of T as

$$\mathrm{GR}(T) = \mathrm{codim}\{(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid \forall z \in \mathbb{F}^{n_3} : T(x, y, z) = 0\}.$$

The definition of geometric rank is expressed asymmetrically in x, y and z. We will see, however, that the codimensions of $\{(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid \forall z : T(x, y, z) = 0\}$, $\{(x, z) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_3} \mid \forall y : T(x, y, z) = 0\}$ and $\{(y, z) \in \mathbb{F}^{n_2} \times \mathbb{F}^{n_3} \mid \forall x : T(x, y, z) = 0\}$ coincide (Theorem 4).

The motivation for this definition is a bit hard to explain right away. We arrived at it while searching for a characteristic 0 analogue of the analytic rank of Gowers and Wolf [19] (see Section 8).

▶ **Example 1.** We give an example of how to compute the geometric rank. Let $T \in \mathbb{F}^{2\times 2\times 2}$ be the tensor with 3-slices

$$M_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

(This is sometimes called the W-tensor). One verifies that the algebraic variety $V = \{(x, y) \in \mathbb{F}^2 \times \mathbb{F}^2 \mid x_1 y_1 = 0, x_2 y_1 + x_1 y_2 = 0\}$ has the three irreducible components $\{(x, y) \in \mathbb{F}^2 \times \mathbb{F}^2 \mid x_1 = 0, x_2 = 0\}$, $\{(x, y) \in \mathbb{F}^2 \times \mathbb{F}^2 \mid x_1 = 0, y_1 = 0\}$ and $\{(x, y) \in \mathbb{F}^2 \times \mathbb{F}^2 \mid y_1 = 0, y_2 = 0\}$. Each irreducible component has dimension 2 and thus V has dimension 2. Hence $\mathrm{GR}(T) = \mathrm{codim}\, V = 4 - 2 = 2$. We will see more examples of geometric rank later (Theorem 17).

1.2 Overview: notions of tensor rank

Before discussing our results we give an introduction to some of the existing notions of rank and their usefulness. Several interesting notions of rank of tensors have been studied in mathematics and computer science, each with their own applications. As a warm-up we first discuss the familiar situation for matrices.

Matrices

For any two matrices $M \in \mathbb{F}^{m_1 \times m_2}$ and $N \in \mathbb{F}^{n_1 \times n_2}$ we write $M \leq N$ if there exist matrices A, B such that $M = ANB$. Defining the matrix rank $\mathrm{R}(M)$ of M as the smallest number r such that M can be written as a sum of r matrices that are outer products $(u_i v_j)_{ij}$ (i.e., rank-1 matrices), we see that in terms of the relation $\leq$ we can write the matrix rank as the minimisation

$$\mathrm{R}(M) = \min\{r \in \mathbb{N} \mid M \leq I_r\},$$

where I_r is the $r \times r$ identity matrix. Matrix rank thus measures the "cost" of M in terms of identity matrices. Let us define the subrank $\mathrm{Q}(M)$ of M as the "value" of M in terms of identity matrices,

$$\mathrm{Q}(M) = \max\{s \in \mathbb{N} \mid I_s \leq M\}.$$

It turns out that subrank equals rank for matrices,

$$\mathrm{Q}(M) = \mathrm{R}(M).$$

Namely, if $\mathrm{R}(M) = r$, then by using Gaussian elimination we can bring M in diagonal form with exactly r nonzero elements on the diagonal, and so $I_r \leq M$. In fact, $M \leq N$ if and only if $\mathrm{R}(M) \leq \mathrm{R}(N)$.

Tensors

For any two tensors $S \in \mathbb{F}^{m_1 \times m_2 \times m_3}$ and $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ we write $S \leq T$ if there are matrices A, B, C such that $S = (A, B, C) \cdot T$ where we define

$$(A, B, C) \cdot T := (\sum_{a,b,c} A_{ia} B_{jb} C_{kc} T_{a,b,c})_{i,j,k}.$$

Thus $(A, B, C) \cdot T$ denotes taking linear combinations of the slices of T in three directions according to A, B and C. Let $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ be a tensor. The tensor rank $\mathrm{R}(T)$ of T is defined as the smallest number r such that T can be written as a sum of r tensors that are outer products $(u_i v_j w_k)_{i,j,k}$. Similarly as for matrices, we can write tensor rank in terms of the relation $\leq$ as the "cost" minimisation

$$\mathrm{R}(T) = \min\{r \in \mathbb{N} \mid M \leq I_r\}$$

where I_r is the $r \times r \times r$ identity tensor (i.e., the diagonal tensor with ones on the main diagonal). Strassen defined the subrank of T as the "value" of T in terms of identity tensors,

$$\mathrm{Q}(T) = \max\{s \in \mathbb{N} \mid I_s \leq M\}.$$

Naturally, since $\leq$ is transitive, we have that value is at most cost: $\mathrm{Q}(T) \leq \mathrm{R}(T)$. Unlike the situation for matrices, however, there exist tensors for which this inequality is strict. One way to see this is using the fact that a random tensor in $\mathbb{F}^{n \times n \times n}$ has tensor rank close to n^2 whereas

its subrank is at most n. Another way to see this is using the ranks $\mathrm{R}^{(i)}(T) \coloneqq \mathrm{R}(T^{(i)})$ of the matrices $T^{(1)} = (T_{i,j,k})_{i,(j,k)} \in \mathbb{F}^{n_1 \times n_2 n_3}$, $T^{(2)} = (T_{i,j,k})_{j,(i,k)} \in \mathbb{F}^{n_2 \times n_1 n_3}$, and $T^{(3)} = (T_{i,j,k})_{k,(i,j)} \in \mathbb{F}^{n_3 \times n_1 n_2}$ obtained from T by grouping two of the three indices together, since

$$\mathrm{Q}(T) \leq \mathrm{R}^{(i)}(T) \leq \mathrm{R}(T).$$

Namely, it is not hard to find tensors T for which $\mathrm{R}^{(1)}(T) < \mathrm{R}^{(2)}(T)$. We will now discuss two upper bounds on the subrank $\mathrm{Q}(T)$ that improve on the flattening ranks $\mathrm{R}^{(i)}(T)$. Then we will discuss connections between subrank and problems in complexity theory and combinatorics.

Slice rank

In the context of the cap set problem, Tao [34] defined the slice rank of any tensor T as the minimum number r such that T can be written as a sum of r tensors of the form $(u_i V_{jk})_{i,j,k}$, $(u_j V_{ik})_{i,j,k}$ or $(u_k V_{ij})_{i,j,k}$ (i.e., an outer product of a vector and a matrix). In other words, $\mathrm{SR}(T) \coloneqq \min\{\mathrm{R}^{(1)}(S_1) + \mathrm{R}^{(2)}(S_2) + \mathrm{R}^{(3)}(S_3) : S_1 + S_2 + S_3 = T\}$. Clearly slice rank is at most any flattening rank, and Tao proved that slice rank upper bounds subrank,

$$\mathrm{Q}(T) \leq \mathrm{SR}(T) \leq \mathrm{R}^{(i)}(T).$$

The lower bound connects slice rank to problems in extremal combinatorics, which we will discuss further in Section 1.3. The slice rank of large Kronecker powers of tensors was studied in [7] and [13], which lead to strong connections with invariant theory and moment polytopes, and with the asymptotic spectrum of tensors introduced by Strassen [32].

Analytic rank

Gowers and Wolf [19] defined the analytic rank of any tensor $T \in \mathbb{F}_p^{n_1 \times n_2 \times n_3}$ over the finite field $\mathbb{F}_p$ for a prime p as $\mathrm{AR}(T) \coloneqq -\log_p \mathrm{bias}(T)$, where the bias of T is defined as $\mathrm{bias}(T) \coloneqq \mathbb{E} \exp(2\pi i\, T(x,y,z)/p)$ with the expectation taken over all vectors $x \in \mathbb{F}_p^{n_1}$, $y \in \mathbb{F}_p^{n_2}$ and $z \in \mathbb{F}_p^{n_3}$. The analytic rank relates to subrank and tensor rank as follows:

$$\mathrm{Q}(T) \leq \frac{\mathrm{AR}(T)}{\mathrm{AR}(I_1)} \leq \mathrm{R}(T)$$

where $\mathrm{AR}(I_1) = -\log_p(1 - (1-1/p)^2)$. The upper bound was proven in [6]. Interestingly, the value of $\mathrm{AR}(T)/\mathrm{AR}(I_1)$ can be larger than $\max_i \mathrm{R}^{(i)}(T)$ for small p. The lower bound is essentially by Lovett [27]. Namely, Lovett proves that $\mathrm{AR}(T)/\mathrm{AR}(I_1)$ upper bounds the size of the largest principal subtensor of T that is diagonal. (We will discuss this further in Section 1.3.) Lovett moreover proved that $\mathrm{AR}(T) \leq \mathrm{SR}(T)$ and he thus proposes analytic rank as an effective upper bound tool for any type of problem where slice rank works well asymptotically. Lovett's result motivated us to study other parameters to upper bound the subrank, which led to geometric rank.

Another line of work has shown upper bounds on $\mathrm{SR}(T)$ in terms of $\mathrm{AR}(T)$. This was first proven by Bhowmick and Lovett [5], with an Ackerman-type dependence. The dependence was later improved significantly by Janzer [22]. Recently, Janzer [23] and Milićević [28] proved polynomial upper bounds of SR in terms of AR. It is not known whether these parameters can be related by a multiplicative constant.

1.3 Connections of subrank to complexity theory and combinatorics

Arithmetic complexity of matrix multiplication and barriers

A well-known problem in computer science concerning tensors is about the arithmetic complexity of matrix multiplication. Asymptotically how many scalar additions and multiplications are required to multiply two $m \times m$ matrices? The answer is known to be between n^2 and $Cn^{2.37\cdots}$, or in other words, the exponent of matrix multiplication ω is known to be between 2 and 2.37... [26]. The complexity of matrix multiplication turns out to be determined by the tensor rank of the matrix multiplication tensors $\langle m, m, m\rangle$ corresponding to taking the trace of the product of three $m \times m$ matrices. Explicitly, $\langle m, m, m\rangle$ corresponds to the trilinear map $\sum_{i,j,k=1}^{m} x_{ij}y_{jk}z_{ki}$. In practice, upper bounds on the rank of the matrix multiplication tensors are obtained by proving a chain of inequalities

$$\langle m, m, m\rangle \leq T \leq I_r$$

for some intermediate tensor T, which is usually taken to be a Coppersmith–Winograd tensor, and an r that is small relatively to m. It was first shown by Ambainis, Filmus and Le Gall [3] that there is a barrier for this strategy to give fast algorithms. This barrier was recently extended and simplified in several works [7, 8, 2, 1, 14] and can be roughly phrased as follows: if the asymptotic subrank of the intermediate tensor $\lim_{n\to\infty} \mathrm{Q}(T^{\otimes n})^{1/n}$ is strictly smaller than the asymptotic rank $\lim_{n\to\infty} \mathrm{R}(T^{\otimes n})^{1/n}$, then one cannot obtain $\omega = 2$ via T. These barriers rely on the fact that the asymptotic subrank of the matrix multiplication tensors is maximal. Summarizing, the rank of the matrix multiplication tensors corresponds to the complexity of matrix multiplication whereas the subrank of any tensor corresponds to the a priori suitability of that tensor for use as an intermediate tensor. The upper bounds on the asymptotic subrank used in the aforementioned results were obtained via slice rank or the related theory of support functionals and quantum functionals [13].

Cap sets, sunflowers and independent sets in hypergraphs

Several well-known problems in extremal combinatorics can be phrased in terms of the independence number of families of hypergraphs. One effective collection of upper bound methods proceeds via the subrank of tensors. (For other upper bound methods, see e.g. the recent work of Filmus, Golubev and Lifshitz [17].) A hypergraph is a a symmetric subset $E \subseteq V \times V \times V$. An independent set of E is any subset $S \subseteq V$ such that S does not induce any edges in E, that is, $E \cap (S \times S \times S) = \emptyset$. The independence number $\alpha(E)$ of E is the largest size of any independence set in E. For any hypergraph $E \subseteq [n] \times [n] \times [n]$, if $T \subseteq \mathbb{F}^{n\times n\times n}$ is any tensor supported on $E \cup \{(i, i, i) : i \in [n]\}$, then

$$\alpha(E) \leq \mathrm{Q}(T).$$

Indeed, for any independent set S of E the subtensor $T|_{S\times S\times S}$ is a diagonal tensor with nonzero diagonal and $T \geq T|_{S\times S\times S}$. For example, the resolution of the cap set problem by Ellenberg and Gijswijt [16], as simplified by Tao [34], can be thought of as upper bounding the subrank of tensors corresponding to strong powers of the hypergraph consisting of the edge $(1, 2, 3)$ and permutations. The Erdős–Szemerédi sunflower problem for three petals was resolved by Naslund and Sawin [29] by similarly considering the strong powers of the hypergraph consisting of the edge $(1, 1, 2)$ and permutations. In both cases slice rank was used to obtain the upper bound. Another result in extremal combinatorics via analytic rank was recently obtained by Briët [10].

1.4 Our results

We establish a number of basic properties of geometric rank. These imply close connections between geometric rank and other notions of rank, and thus bring in a new set of algebraic geometric tools to help reason about the various notions of rank. In particular, our new upper bounds on the (border) subrank of matrix multiplication follow easily from our basic results.

Subrank and slice rank

We prove that the geometric rank $\mathrm{GR}(T)$ is at most the slice rank $\mathrm{SR}(T)$ of Tao [34] and at least the subrank $\mathrm{Q}(T)$ of Strassen [31] (see Theorem 6).

▶ **Theorem 1.** *For any tensor T,*

$$\mathrm{Q}(T) \leq \mathrm{GR}(T) \leq \mathrm{SR}(T).$$

We thus add GR to the collection of tools to upper bound the subrank of tensors Q and in turn the independence number of hypergraphs. We prove these inequalities by proving that GR is monotone under $\leq$, additive under the direct sum of tensors, and has value 1 on the trivial I_1 tensor. We also give a second more direct proof of this inequality (Theorem 23).

Border subrank

We extend our upper bound on subrank to *border subrank*, the (widely studied) approximative version of subrank.

The main ingredient in this extension is the following fact (which itself exploits the algebraic-geometric nature of definition of GR): the set $\{T \in \mathbb{F}^{n\times n\times n} \mid \mathrm{GR}(T) \leq m\}$ is closed in the Zariski topology.[1] In other words, geometric rank is lower-semicontinuous. This implies that the geometric rank also upper bounds the border subrank $\underline{\mathrm{Q}}(T)$ (see Theorem 12).

▶ **Theorem 2.** *For any tensor T,*

$$\underline{\mathrm{Q}}(T) \leq \mathrm{GR}(T).$$

As far as we know, GR is a new tensor parameter. We show that GR is not the same parameter as Q, $\underline{\mathrm{Q}}$ or SR (Remark 20 and Remark 22).

Matrix multiplication

In the study of the complexity of matrix multiplication, Strassen [31] proved that for the matrix multiplication tensors $\langle m,m,m\rangle \in \mathbb{F}^{m^2\times m^2\times m^2}$ the border subrank is lower bounded by $\lceil \frac{3}{4}m^2 \rceil \leq \underline{\mathrm{Q}}(\langle m,m,m\rangle)$. We prove that this lower bound is optimal by proving the following (see Theorem 17).

▶ **Theorem 3.** *For any positive integers $e \leq h \leq \ell$,*

$$\underline{\mathrm{Q}}(\langle e,h,\ell\rangle) = \mathrm{GR}(\langle e,h,\ell\rangle) = \begin{cases} eh - \lfloor \frac{(e+h-\ell)^2}{4} \rfloor & \text{if } e+h \geq \ell, \\ eh & \text{otherwise.} \end{cases}$$

In particular, we have $\mathrm{Q}(\langle m,m,m\rangle) \leq \underline{\mathrm{Q}}(\langle m,m,m\rangle) = \mathrm{GR}(\langle m,m,m\rangle) = \lceil \frac{3}{4}m^2 \rceil$ for any $m \in \mathbb{N}$.

[1] That is, the statement $\mathrm{GR}(T) \leq m$ is characterized by the vanishing of a finite number of polynomials.

Our computation of GR here is a calcluation of the dimension of a variety. We do this by studying the dimension of various sections of that variety, which then reduces to linear algebraic questions about matrices (we are talking about matrix multiplication after all).

Our result improves the previously best known upper bound on the subrank of matrix multiplication of Christandl, Lucia, Vrana and Werner [12], which was $\mathrm{Q}(\langle m,m,m\rangle) \leq m^2 - m + 1$. In fact, our upper bound on $\mathrm{GR}(\langle e,h,\ell\rangle)$ exactly matches the lower bound on $\underline{\mathrm{Q}}(\langle e,h,\ell\rangle)$ of Strassen [31], for any nonnegative integers e, h, and ℓ. We thus solve the problem of determining the exact value of $\underline{\mathrm{Q}}(\langle e,h,\ell\rangle)$.

Analytic rank

Finally, we establish a strong connection between geometric rank and analytic rank.

We prove that for any tensor $T \in \mathbb{Z}^{n_1\times n_2\times n_3} \subseteq \mathbb{C}^{n_1\times n_2\times n_3}$ with integer coefficients, the geometric rank of T equals the *liminf* of the analytic rank of the tensors $T_p \in \mathbb{F}_p^{n_1\times n_2\times n_3}$ obtained from T be reducing all coefficients modulo p and letting p go to infinity over all primes (see Theorem 24).

▶ **Theorem 4.** *For every tensor T over $\mathbb{Z}$ we have*

$$\liminf_{p\to\infty} \mathrm{AR}(T_p) = \mathrm{GR}(T).$$

This result is in fact the source of our definition of geometric rank. The analytic rank of a tensor is defined as the bias of a certain polynomial on random inputs. By simple transformations, computing the analytic rank over $\mathbb{F}_p$ reduces to computing the number of solutions of a system of polynomial equations over $\mathbb{F}_p$. Namely,

$$\mathrm{AR}(T_p) = n_1 + n_2 - \log_p \left|\{(x,y) \in \mathbb{F}_p^{n_1} \times \mathbb{F}_p^{n_2} : T_p(x,y,\cdot) = 0\}\right|.$$

This system of polynomial equations defines a variety, and it is natural to expect that the dimension of the variety roughly determines the number of $\mathbb{F}_p$-points of the variety. This expectation is not true in general, but under highly controlled circumstances something like it is true. This is how we arrived at the definition of geometric rank (which eventually turned out to have very natural properties on its own, without this connection to analytic rank).

Actually establishing the above liminf result is quite roundabout, and requires a number of tools from algebraic geometry and number theory. In particular, we do not know whether this liminf can be replaced by a limit!

We stress that analytic rank is only defined for tensors over prime fields of positive characteristic, whereas geometric rank is defined for tensors over any field. By the aforementioned result, geometric rank over the complex numbers can be thought of as an extension of analytic rank to characteristic 0. Finding an extension of analytic rank beyond finite fields is mentioned as an open problem by Lovett [27, Problem 1.10].

Organization of this paper

In the next section we formally define geometric rank. In Section 3, we give some alternative definitions of geometric rank that help us reason about it. In Section 4 and Section 5 we show the relationship between geometric rank, slice rank, subrank and border subrank. In Section 6 we use the established properties of geometric rank to give a proof of our upper bound on the (border) subrank of matrix multiplication. In Section 7 we give a more direct proof of the inequality between slice rank and geometric rank. Finally, in Section 8 we establish the relationship between geometric and analytic ranks.

2 Geometric rank

In this section we set up some general notation and define geometric rank. Let $\mathbb{F}$ be an algebraically closed field.

Dimension and codimension

The notion of dimension that we use is the standard notion in algebraic geometry, and is defined as follows. Let $V \subseteq \mathbb{F}^n$ be a (possibly reducible) algebraic variety. The codimension $\operatorname{codim} V$ is defined as $n - \dim V$. The dimension $\dim V$ is defined as the length of a maximal chain of irreducible subvarieties of V [21]. In our proofs we will use basic facts about dimension: the dimension of a linear space coincides with the notion from linear algebra, the dimension is additive under the cartesian product, the dimension of a locally open set equals the dimension of its closure and dimension behaves well under projections $(x, y) \mapsto y$.

Notation about tensors

Let $\mathbb{F}^{n_1 \times n_2 \times n_3}$ be the set of all three-dimensional arrays

$$T = (T_{i,j,k})_{i \in [n_1], j \in [n_2], k \in [n_3]}$$

with $T_{i,j,k} \in \mathbb{F}$. We refer to the elements of $\mathbb{F}^{n_1 \times n_2 \times n_3}$ as the $n_1 \times n_2 \times n_3$ *tensors over* $\mathbb{F}$. To any tensor $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ we associate the polynomial $T(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}, z_1, \ldots, z_{n_3})$ in $\mathbb{F}[x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}, z_1, \ldots, z_{n_3}]$ defined by

$$T(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}, z_1, \ldots, z_{n_3}) = \sum_{i \in [n_1]} \sum_{j \in [n_2]} \sum_{k \in [n_3]} T_{i,j,k}\, x_i y_j z_k$$

and the trilinear map $\mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \times \mathbb{F}^{n_3} \to \mathbb{F}$ defined by

$$T(x, y, z) = T(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}, z_1, \ldots, z_{n_3}).$$

Geometric rank

▶ **Definition 2.** *The* geometric rank *of a tensor $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$, written $\operatorname{GR}(T)$, is the codimension of the set of elements $(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2}$ such that $T(x, y, z) = 0$ for all $z \in \mathbb{F}^{n_3}$. That is,*

$$\operatorname{GR}(T) \coloneqq \operatorname{codim}\{(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid \forall z \in \mathbb{F}^{n_3} : T(x, y, z) = 0\}.$$

For any $(x, y) \in \mathbb{F}^{n_2} \times \mathbb{F}^{n_3}$ we define the vector $T(x, y, \cdot) = (T(x, y, e_k))_{k=1}^{n_3}$, where $e_1, \ldots, e_{n_3}$ is the standard basis of $\mathbb{F}^{n_3}$. In this notation the geometric rank is given by

$$\operatorname{GR}(T) = \operatorname{codim}\{(x, y) \mid T(x, y, \cdot) = 0\}.$$

For later use we also define the vectors $T(x, \cdot, z) = T(x, e_j, z)_j$ and $T(\cdot, y, z) = T(e_i, y, z)_i$, and we define the matrices $T(x, \cdot, \cdot) = T(x, e_j, e_k)_{j,k}$, $T(\cdot, y, \cdot) = T(e_i, y, e_k)_{i,k}$ and $T(\cdot, \cdot, z) = T(e_i, e_j, z)_{i,j}$.

We defined the geometric rank of tensors with coefficients in an algebraically closed field. For tensors with coefficients in an arbitrary field we naturally define the geometric rank via the embedding of the field in its algebraic closure.

Computer software

One can compute the dimension of an algebraic variety $V \subseteq \mathbb{F}^n$ using computer software like Macaulay2 [20] or Sage [30]. This allows us to easily compute the geometric rank of small tensors. For example, for Example 1 in the introduction over the field $\mathbb{F} = \mathbb{C}$, one verifies in Macaulay2 with the commands

```
R = CC[x1,x2,y1,y2];
dim ideal(x1*y1, x2*y1 + x1*y2)
```

or in Sage with the commands

```
A.<x1,x2,y1,y2> = AffineSpace(4, CC);
Ideal([x1*y1, x2*y1 + x1*y2]).dimension()
```

that $\dim V = 2$.

Computational complexity

Koiran [24] studied the computational complexity of the problem of deciding whether the dimension of an algebraic variety $V \subseteq \mathbb{C}^n$ is at least a given number. When V is given by polynomial equations over the integers the problem is in PSPACE, and assuming the Generalized Riemann Hypothesis the problem is in the Arthur–Merlin class AM. Thus the same upper bounds apply to computing GR.

In the other direction, Koiran showed that computing dimension of algebraic varieties in general is NP-hard. We know of no hardness results for computing GR.

Higher-order tensors

Our definition of geometric rank extends naturally from the set of 3-tensors $\mathbb{F}^{n_1 \times n_2 \times n_3}$ to the set of k-tensors $\mathbb{F}^{n_1 \times \cdots \times n_k}$ for any $k \geq 2$ by defining the geometric rank of any k-tensor $T \in \mathbb{F}^{n_1 \times \cdots \times n_k}$ as

$$\mathrm{GR}(T) \coloneqq \mathrm{codim}\{(x_1, \ldots, x_{k-1}) \in \mathbb{F}^{n_1} \times \cdots \times \mathbb{F}^{n_{k-1}} \mid \forall x_k \in \mathbb{F}^{n_k} : T(x_1, \ldots, x_{k-1}, x_k) = 0\}.$$

For $k = 2$ geometric rank coincides with matrix rank. Our results extend naturally to k-tensors with this definition, but for clarity our exposition will be in terms of 3-tensors.

3 Alternative descriptions of geometric rank

We give two alternative descriptions of geometric rank that we will use later. The first description relates geometric rank to the matrix rank of the matrices $T(x, \cdot, \cdot) = (T(x, e_j, e_k))_{j,k}$. The second description shows that the geometric rank of $T(x, y, z)$ is symmetric under permuting the variables x, y and z. Both theorems rely on an understanding of the dimension of fibers of a (nice) map.

▶ **Theorem 3.** *For any tensor* $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$,

$$\begin{aligned}\dim\{(x,y) \mid T(x,y,\cdot) = 0\} &= \max_i \dim\big\{x \mid \dim\{y \mid T(x,y,\cdot) = 0\} = i\big\} + i \\ &= \max_i \dim\big\{x \mid \operatorname{corank} T(x,\cdot,\cdot) = i\big\} + i\end{aligned}$$

and therefore

$$\mathrm{GR}(T) = \mathrm{codim}\{(x,y) \mid T(x,y,\cdot) = 0\} = \min_j \mathrm{codim}\big\{x \mid \operatorname{rank} T(x,\cdot,\cdot) = j\big\} + j.$$

Proof. Let $V = \{(x, y) \mid T(x, y, \cdot) = 0\}$. Let $W = \mathbb{F}^{n_1}$. Let $\pi : V \to W$ map (x, y) to x. Define the sets $W_i = \{x \mid \operatorname{corank}(T(x, \cdot, \cdot)) = i\}$. The rank-nullity theorem for matrices gives for any fixed x that $\operatorname{corank}(T(x, \cdot, \cdot)) = \dim\{y \mid T(x, y, \cdot) = 0\}$. The sets W_i are locally closed, that is, each W_i is the intersection of an open set and a closed set. Let $V_i = \pi^{-1}(W_i)$. The set V_i is also locally closed. We have that $W = \cup_i W_i$ and so $V = \cup_i V_i$. Therefore, $\dim V = \max_i \dim V_i$. We claim that $\dim V_i = \dim W_i + i$. From this claim follows $\dim V = \max_i \dim W_i + i$, which finishes the proof.

We prove the claim that $\dim V_i = \dim W_i + i$. For every $x \in W_i$ the fiber dimension $\dim \pi^{-1}(x)$ equals i. Write V_i as a union of irreducible components V_{ij}. Let W_{ij} be the closure of $\pi(V_{ij})$. We now apply Theorem 5 (see the end of this section) with $X = V_i$ and $X_0 = V_{ij}$. For any $p = (x, y) \in X_0$ we have that $\pi^{-1}(\pi(p)) = \{(x, y') \mid T(x, y', \cdot) = 0\}$. The set $\{y' \mid T(x, y', \cdot) = 0\}$ is a linear subspace and thus irreducible. Therefore, $\pi^{-1}(\pi(p))$ is irreducible. Then Theorem 5 gives that $\dim V_{ij} = \dim W_{ij} + i$. We have that $\max_j \dim W_{ij} = \dim W_i$, so taking the j maximising $\dim W_{ij}$ gives $\dim V_i \leq \dim W_i + i$. Also $\max_j \dim V_{ij} = \dim V_i$, so taking the j maximising $\dim V_{ij}$ gives $\dim V_i \geq \dim W_i + i$. ◀

▶ **Theorem 4.** *For any tensor T,*

$$\begin{aligned}\operatorname{GR}(T) = \operatorname{codim}\{(x, y) \mid T(x, y, \cdot) = 0\} = \operatorname{codim}\{(x, z) \mid T(x, \cdot, z) = 0\} \\ = \operatorname{codim}\{(y, z) \mid T(\cdot, y, z) = 0\}.\end{aligned}$$

Proof. We apply Theorem 3 to T and to T after swapping y and z to get that the codimensions of $\{(x, y) \mid T(x, y, \cdot) = 0\}$ and $\{(x, z) \mid T(x, \cdot, z) = 0\}$ are equal to $\min_j \operatorname{codim}\{x \mid \operatorname{rank} T(x, \cdot, \cdot) = j\} + j$. This proves the first equality. The second equality is proven similarly. ◀

▶ **Theorem 5** ([21, special case of Theorem 11.12])**.** *Let $X \subseteq \mathbb{F}^{n_1} \times \mathbb{F}^{n_2}$ be the affine cone over a quasi-projective variety, that is,*

$$X = \{(x, y) \in \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid f_1(x, y) = 0, \ldots, f_k(x, y) = 0, g_1(x, y) \neq 0, \ldots, g_m(x, y) \neq 0\}$$

where the f_i and g_i are homogeneous polynomials. Let $\pi : X \to \mathbb{F}^{n_1}$ map (x, y) to x. Let $X_0 \subseteq X$ be an irreducible component. Suppose that the fiber $\pi^{-1}(\pi(p))$ is irreducible for every $p \in X_0$. Then

$$\dim X_0 = \dim \overline{\pi(X_0)} + \min_{p \in X_0} \dim \pi^{-1}(\pi(p)).$$

4 Geometric rank is between subrank and slice rank

Recall that the subrank $\operatorname{Q}(T)$ of T is the largest number s such that $I_s \leq T$ and the slice rank $\operatorname{SR}(T)$ is the smallest number r such that $T(x, y, z)$ can be written as a sum of r trilinear maps of the form $f(x)g(y, z)$ or $f(y)g(x, z)$ or $f(z)g(x, y)$.

▶ **Theorem 6.** *For any tensor T,*

$$\operatorname{Q}(T) \leq \operatorname{GR}(T) \leq \operatorname{SR}(T).$$

Theorem 6 will follow from the following basic properties of GR. We will give a more direct proof of the inequality $\operatorname{GR}(T) \leq \operatorname{SR}(T)$ in Section 7. Recall from the introduction that for any two tensors $S \in \mathbb{F}^{m_1 \times m_2 \times m_3}$ and $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ we write $S \leq T$ if there are matrices A, B, C such that $S = (A, B, C) \cdot T$ where we define $(A, B, C) \cdot T := (\sum_{a,b,c} A_{ia} B_{jb} C_{kc} T_{a,b,c})_{i,j,k}$.

▶ **Lemma 7.** GR *is* $\leq$*-monotone: if* $S \leq T$*, then* $\mathrm{GR}(S) \leq \mathrm{GR}(T)$.

Proof. Let $T \in \mathbb{F}^{n_1 \times n_2 \times n_3}$. We claim that $\mathrm{GR}((\mathrm{Id}, \mathrm{Id}, C) \cdot T) \leq \mathrm{GR}(T)$ for any $C \in \mathbb{F}^{m_3 \times n_3}$, where Id denotes an identity matrix of the appropriate size. From this claim and the symmetry of GR (Theorem 4), follows the inequalities $\mathrm{GR}((A, \mathrm{Id}, \mathrm{Id}) \cdot T) \leq T$ and $\mathrm{GR}((\mathrm{Id}, B, \mathrm{Id}) \cdot T) \leq \mathrm{GR}(T)$ for any matrices $A \in \mathbb{F}^{m_1 \times n_1}$ and $B \in \mathbb{F}^{m_2 \times n_2}$. Chaining these three inequalities gives that for any two tensors S and T, if $S \leq T$, then $\mathrm{GR}(S) \leq \mathrm{GR}(T)$.

We prove the claim. Let $S = (\mathrm{Id}, \mathrm{Id}, C) \cdot T$. Let $M_k = (T_{i,j,k})_{ij}$ be the 3-slices of T and let $N_k = (S_{i,j,k})_{ij}$ be the 3-slices of S. Since $S = (\mathrm{Id}, \mathrm{Id}, C) \cdot T$, the matrices $N_1, \ldots, N_{m_3}$ are in the linear span of the matrices $M_1, \ldots, M_{n_3}$. Thus $V = \{(x, y) \mid x^T M_1 y = \cdots = x^T M_{n_3} y = 0\}$ is a subset of $W = \{(x, y) \mid x^T N_1 y = \cdots = x^T N_{m_3} y = 0\}$. Therefore, $\dim V \leq \dim W$ and it follows that $\mathrm{GR}(S) = \operatorname{codim} W \leq \operatorname{codim} V = \mathrm{GR}(T)$. ◀

Let $T_1 \in \mathbb{F}^{m_1 \times m_2 \times m_3}$ and $T_2 \in \mathbb{F}^{n_1 \times n_2 \times n_3}$ be tensors with 3-slices A_k and B_k respectively. The direct sum $T_1 \oplus T_2 \in \mathbb{F}^{(m_1+n_1) \times (m_2+n_2) \times (m_3+n_3)}$ is defined as the tensor with 3-slices $A_k \oplus 0_{n_1 \times n_2}$ for $k = 1, \ldots, m_3$ and $0_{m_1 \times m_2} \oplus B_k$ for $k = m_3 + 1, \ldots, m_3 + n_3$ where $0_{a \times b}$ denotes the zero matrix of size $a \times b$. In other words, $T_1 \oplus T_2$ is the block-diagonal tensor with blocks T_1 and T_2.

▶ **Lemma 8.** GR *is additive under direct sums:* $\mathrm{GR}(T_1 \oplus T_2) = \mathrm{GR}(T_1) + \mathrm{GR}(T_2)$.

Proof. Let A_k be the 3-slices of T_1 and let B_k be the 3-slices of T_2. Let $T = T_1 \oplus T_2$ be the direct sum with 3-slices M_k. Then

$$V = \{(x, y) \mid T(x, y, \cdot) = 0\} = \{(x, y) \mid x^T M_1 y = \cdots = x^T M_{m_3+n_3} y = 0\}$$

is the cartesian product of

$$V_1 = \{(x, y) \mid x^T A_1 y = \cdots = x^T A_{m_3} y = 0\}$$

and

$$V_2 = \{(x, y) \mid x^T B_1 y = \cdots = x^T B_{n_3} y = 0\}.$$

Thus $\dim V = \dim V_1 + \dim V_2$ [21, page 138]. Therefore, $\mathrm{GR}(T) = \mathrm{GR}(T_1) + \mathrm{GR}(T_2)$. ◀

▶ **Lemma 9.** GR *is sub-additive under element-wise sums:* $\mathrm{GR}(S + T) \leq \mathrm{GR}(S) + \mathrm{GR}(T)$.

Proof. Note that $S + T \leq S \oplus T$. Thus, $\mathrm{GR}(S + T) \leq \mathrm{GR}(S \oplus T) = \mathrm{GR}(S) + \mathrm{GR}(T)$, where the inequality uses Lemma 7, and the equality uses Lemma 8. ◀

▶ **Lemma 10.** *If* $\mathrm{SR}(T) = 1$*, then* $\mathrm{GR}(T) = 1$.

Proof. It is sufficient to consider a tensor $T \in \mathbb{F}^{1 \times n \times n}$ with one nonzero slice. Then we have that $T(0, \mathbb{F}^n, \mathbb{F}^n) = 0$, and so $\mathrm{GR}(T) = 1 + n - n = 1$. ◀

▶ **Lemma 11.** *For every* $r \in \mathbb{N}$ *we have* $\mathrm{GR}(I_r) = r$.

Proof. We have $\mathrm{SR}(I_1) = 1$ and so $\mathrm{GR}(I_1) = 1$ (Lemma 10). Since I_r is a direct sum of r copies of I_1 and geometric rank is additive under taking the direct sum $\oplus$ (Lemma 9), we find that $\mathrm{GR}(I_r) = r\,\mathrm{GR}(I_1) = r$. ◀

Proof of Theorem 6. We prove that $\mathrm{GR}(T) \leq \mathrm{SR}(T)$. Let $r = \mathrm{SR}(T)$. Then there are tensors $T_1, \ldots, T_r$ so that $T = \sum_{i=1}^{r} T_i$ and $\mathrm{SR}(T_i) = 1$. Then also $\mathrm{GR}(T_i) = 1$ (Lemma 10). Subadditivity of GR under element-wise sums (Lemma 9) gives

$$\mathrm{GR}(T) \leq \sum_{i=1}^{r} \mathrm{GR}(T_i) = r = \mathrm{SR}(T).$$

We prove that $\mathrm{Q}(T) \leq \mathrm{GR}(T)$. Let $s = \mathrm{Q}(T)$. Then $I_s \leq T$. We know $\mathrm{GR}(I_s) = s$ (Lemma 11). By the $\leq$-monotonicity of GR (Lemma 7), we have

$$\mathrm{Q}(T) = s = \mathrm{GR}(I_s) \leq \mathrm{GR}(T).$$ ◀

5 Geometric rank is at least border subrank

In this section we extend the inequality $\mathrm{Q}(T) \leq \mathrm{GR}(T)$ (Theorem 6) to the approximative version of subrank, called border subrank. To define border subrank we first define degeneration $\trianglelefteq$, which is the approximative version of restriction $\leq$. We write $S \trianglelefteq T$, and we say S is a *degeneration* of T, if for some $e \in \mathbb{N}$ we have

$$S + \varepsilon S_1 + \varepsilon^2 S_2 + \cdots + \varepsilon^e S_e = (A(\varepsilon), B(\varepsilon), C(\varepsilon)) \cdot T$$

for some tensors S_i over $\mathbb{F}$ and for some matrices $A(\varepsilon), B(\varepsilon), C(\varepsilon)$ whose coefficients are Laurent polynomials in the formal variable ε. Equivalently, $S \trianglelefteq T$ if and only if S is in the orbit closure $\overline{G \cdot T}$ where G denotes the group $\mathrm{GL}_{n_1} \times \mathrm{GL}_{n_2} \times \mathrm{GL}_{n_3}$, $G \cdot T$ denotes the natural group action that we also used in the definition of $\leq$, and the closure is taken in the Zariski topology [11, Theorem 20.24]. (When $\mathbb{F} = \mathbb{C}$ one may equivalently take the closure in the Euclidean topology.) Recall that the subrank of T is defined as $\mathrm{Q}(T) = \max\{n \in \mathbb{N} \mid I_n \leq T\}$. The *border subrank* of T is defined as

$$\underline{\mathrm{Q}}(T) = \max\{n \in \mathbb{N} \mid I_n \trianglelefteq T\}.$$

Clearly, $\mathrm{Q}(T) \leq \underline{\mathrm{Q}}(T)$.

▶ **Theorem 12.** *For any tensor T,*

$$\underline{\mathrm{Q}}(T) \leq \mathrm{GR}(T).$$

To prove Theorem 12 we use the following theorem on upper-semicontinuity of fiber dimension.

▶ **Theorem 13** ([21, special case of Corollary 11.13])**.** *Let X be the zero set of bi-homogeneous polynomials, that is,*

$$X = \{(a, b) \in \mathbb{F}^{m_1} \times \mathbb{F}^{m_2} \mid f_1(a, b) = \cdots = f_k(a, b) = 0\}$$

where the $f_i(a, b)$ are polynomials that are homogeneous in both a and b. Let $\pi : X \to \mathbb{F}^{m_2}$ map (a, b) to b. Let $Y = \pi(X)$ be its image. For any $q \in Y$, let $\lambda(q) = \dim(\pi^{-1}(q))$. Then $\lambda(q)$ is an upper-semicontinuous function of q, that is, the set $\{q \in Y \mid \lambda(q) \geq m\}$ is Zariski closed in Y.

▶ **Lemma 14.** GR *is lower-semicontinuous: for any $n_i, m \in \mathbb{N}$ the set $\{T \in \mathbb{F}^{n_1 \times n_2 \times n_3} \mid \mathrm{GR}(T) \leq m\}$ is Zariski closed.*

Proof. We define the set

$$X = \{(T, x, y) \in \mathbb{F}^{n_1 \times n_2 \times n_3} \times \mathbb{F}^{n_1} \times \mathbb{F}^{n_2} \mid T(x, y, \mathbb{F}^{n_3}) = 0\}.$$

Let $\pi : X \to \mathbb{F}^{n_1 \times n_2 \times n_3}$ map (T, x, y) to T. Let $Y = \pi(X) = \mathbb{F}^{n_1 \times n_2 \times n_3}$ be the image of π. For any $T \in Y$ let $\lambda(T) \coloneqq \dim(\pi^{-1}(T))$. Then $\lambda(T)$ is an upper-semicontinuous function of T in the Zariski topology on Y by Theorem 13. This means that the set $\{T \in \mathbb{F}^{n_1 \times n_2 \times n_3} \mid \lambda(T) \geq m\}$ is closed for every $m \in \mathbb{N}$. It follows that $\{T \in \mathbb{F}^{n_1 \times n_2 \times n_3} \mid \mathrm{GR}(T) \leq m\}$ is closed for every $m \in \mathbb{N}$. ◀

▶ Remark 15. A well-known example of a lower-semicontinuous function is matrix rank. Indeed, the set of matrices of rank at most m is the zero set of the determinants of all $(m+1) \times (m+1)$ submatrices. For geometric rank we do not know an explicit set of generators for the vanishing ideal of $\{T \in \mathbb{F}^{n_1 \times n_2 \times n_3} \mid \mathrm{GR}(T) \leq m\}$. For slice rank the set $\{T \in \mathbb{F}^{n_1 \times n_2 \times n_3} \mid \mathrm{SR}(T) \leq m\}$ is also known to be Zariski closed and explicit vanishing polynomials for this variety were recently obtained by Bläser, Ikenmeyer, Lysikov, Pandey and Schreyer [9].

▶ **Lemma 16.** GR *is* $\trianglelefteq$*-monotone: if* $S \trianglelefteq T$*, then* $\mathrm{GR}(S) \leq \mathrm{GR}(T)$

Proof. For all $g \in G$ we have $\mathrm{GR}(g \cdot T) = \mathrm{GR}(T)$ by Lemma 7. The set $\{T' \mid \mathrm{GR}(T') \leq \mathrm{GR}(T)\}$ is Zariski closed by Lemma 14. It contains the orbit $G \cdot T$ and hence also its Zariski closure $\overline{G \cdot T}$, that is,

$$\{T' \mid T' \trianglelefteq T\} = \overline{G \cdot T} \subseteq \{T' \mid \mathrm{GR}(T') \leq \mathrm{GR}(T)\}.$$

Therefore, $\mathrm{GR}(S) \leq \mathrm{GR}(T)$. ◀

Proof of Theorem 12. Let $n = \underline{\mathrm{Q}}(T)$. Then $I_n \trianglelefteq T$ by the definition of $\underline{\mathrm{Q}}$, and so $n \leq \mathrm{GR}(T)$ by Lemma 16. This proves the claim. ◀

6 The border subrank of matrix multiplication

In the context of constructing fast matrix multiplication algorithms, Strassen [31, Theorem 6.6] proved that for any positive integers $e \leq h \leq \ell$ the border subrank of the matrix multiplication tensor $\langle e, h, \ell \rangle$ is lower bounded by

$$\underline{\mathrm{Q}}(\langle e, h, \ell \rangle) \geq \begin{cases} eh - \lfloor \frac{(e+h-\ell)^2}{4} \rfloor & \text{if } e + h \geq \ell, \\ eh & \text{otherwise.} \end{cases} \tag{1}$$

Here $\langle e, h, \ell \rangle$ is the tensor that corresponds to taking the trace of the product of an $e \times h$ matrix, an $h \times \ell$ matrix and an $\ell \times e$ matrix. We prove using the geometric rank that this lower bound is optimal.

▶ **Theorem 17.** *For any positive integers* $e \leq h \leq \ell$

$$\underline{\mathrm{Q}}(\langle e, h, \ell \rangle) = \mathrm{GR}(\langle e, h, \ell \rangle) = \begin{cases} eh - \lfloor \frac{(e+h-\ell)^2}{4} \rfloor & \text{if } e + h \geq \ell, \\ eh & \text{otherwise.} \end{cases}$$

In particular, we have $\underline{\mathrm{Q}}(\langle m, m, m \rangle) = \mathrm{GR}(\langle m, m, m \rangle) = \lceil \frac{3}{4} m^2 \rceil$ *for any* $m \in \mathbb{N}$.

Proof. Since $\underline{\mathrm{Q}}(\langle e,h,\ell\rangle) \leq \mathrm{GR}(\langle e,h,\ell\rangle)$ (Theorem 12) and since we have the lower bound in (1), it suffices to show that $\mathrm{GR}(\langle e,h,\ell\rangle)$ is at most $eh - \lfloor (e+h-\ell)^2/4 \rfloor$ if $e+h \geq \ell$ and at most eh otherwise.

Let $T = \langle e,h,\ell\rangle$. Let $V = \{(x,y) \in \mathbb{F}^{eh} \times \mathbb{F}^{h\ell} \mid T(x,y,\cdot) = 0\}$. Then $\mathrm{GR}(T) = eh + h\ell - \dim V$. From Theorem 3 it follows that

$$\dim V = \max_i \dim\{x \in \mathbb{F}^{eh} \mid \dim\{y \in \mathbb{F}^{h\ell} \mid T(x,y,\cdot) = 0\} = i\} + i. \tag{2}$$

We now think of $\mathbb{F}^{eh}$, $\mathbb{F}^{h\ell}$ and $\mathbb{F}^{\ell e}$ as the matrix spaces $\mathbb{F}^{e\times h}$, $\mathbb{F}^{h\times \ell}$ and $\mathbb{F}^{\ell\times e}$. Then T gives the trilinear map $T : \mathbb{F}^{e\times h} \times \mathbb{F}^{h\times\ell} \times \mathbb{F}^{\ell\times e} \to \mathbb{F} : (X,Y,Z) \mapsto \mathrm{Tr}(XYZ)$. Therefore, $T(X,Y,\cdot) = 0$ if and only if $XY = 0$. If the rank of X as an $e \times h$ matrix equals r, then

$$\dim\{Y \in \mathbb{F}^{h\times\ell} \mid T(X,Y,\cdot) = 0\} = (h-r)\ell,$$

since Y is any matrix with columns from $\ker(X)$. We have

$$\dim\{X \in \mathbb{F}^{e\times h} \mid \mathrm{rank}(X) = r\} = er + (h-r)r.$$

Thus the relevant values of i in (2) are of the form $i = (h-r)\ell$ and we have that

$$\begin{aligned}\dim V &= \max_r \dim\{X \in \mathbb{F}^{e\times h} \mid \mathrm{rank}\, X = r\} + (h-r)\ell \\ &= \max_r er + (h-r)r + (h-r)\ell \\ &= \max_r f(r) + h\ell\end{aligned}$$

where $f(r) = r(\Delta - r)$ with $\Delta := e + h - \ell$. Thus,

$$\mathrm{GR}(T) = eh - \max_r f(r).$$

Over the integers, the function f attains its maximum at $\lfloor \frac{\Delta}{2} \rfloor$ (and at $\lceil \frac{\Delta}{2} \rceil$), but this may be outside the interval $[0,e]$ that we want to maximise over (recall $e \leq h \leq l$). Observe that if $\Delta \geq 0$ then $e \geq \Delta/2 \geq 0$, meaning that f does attain its global maximum in the interval $[0,e]$. On the other hand, if $\Delta \leq 0$ then $r(\Delta - r) \leq 0 = f(0)$ for every $r \geq 0$, so the maximum of f in the interval $[0,e]$ is at the endpoint $r = 0$. Summarizing,

$$\max_{0\leq r\leq e} f(r) = \begin{cases} \lfloor \frac{\Delta^2}{4} \rfloor & \text{if } \Delta \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

This completes the proof. ◀

▶ Remark 18. Theorem 17 gives the upper bound $\mathrm{Q}(\langle m,m,m\rangle) \leq \underline{\mathrm{Q}}(\langle m,m,m\rangle) = \lceil \frac{3}{4}m^2 \rceil$ on the subrank of matrix multiplication $\mathrm{Q}(\langle m,m,m\rangle)$. This improves the previously best known upper bound $\mathrm{Q}(\langle m,m,m\rangle) \leq m^2 - m + 1$ from [12, Equation 25].

▶ Remark 19. Geometric rank GR is not sub-multiplicative under the tensor Kronecker product $\otimes$. We give an example. The matrix multiplication tensor $\langle m,m,m\rangle$ can be written as the product $\langle m,m,m\rangle = \langle m,1,1\rangle \otimes \langle 1,m,1\rangle \otimes \langle 1,1,m\rangle$ and $\mathrm{GR}(\langle m,1,1\rangle) = \mathrm{GR}(\langle 1,m,1\rangle) = \mathrm{GR}(\langle 1,1,m\rangle) = 1$ whereas $\mathrm{GR}(\langle m,m,m\rangle) = \lceil \frac{3}{4}m^2 \rceil$ by Theorem 17.

▶ Remark 20. Geometric rank GR is not the same as subrank Q or border subrank $\underline{\mathrm{Q}}$. For example, for the trilinear map $W(x_1,x_2,y_1,y_2,z_1,z_2) = x_1y_1z_2 + x_1y_2z_1 + x_2y_1z_1$ we find $\mathrm{GR}(W) = 2$ (see the example in the introduction), whereas $\mathrm{Q}(W) = \underline{\mathrm{Q}}(W) = 1$. The latter follows from the fact that $\underset{\sim}{\mathrm{Q}}(W) = 1.81...$ [33], where $\underset{\sim}{\mathrm{Q}}(T) := \lim_{n\to\infty} \mathrm{Q}(T^{\otimes n})^{1/n}$ is the asymptotic subrank of T, since $\underline{\mathrm{Q}}(T) \leq \underset{\sim}{\mathrm{Q}}(T)$ [31].

▶ Remark 21. Geometric rank GR is not super-multiplicative under the tensor Kronecker product $\otimes$. Here is an example. Let $\underset{\sim}{\mathrm{SR}}(T) := \lim_{n\to\infty} \mathrm{SR}(T^{\otimes n})^{1/n}$ and let $\underset{\sim}{\mathrm{GR}}(T) := \lim_{n\to\infty} \mathrm{GR}(T^{\otimes n})^{1/n}$, whenever these limits are defined. From the fact that $\mathrm{Q}(T) \leq \mathrm{GR}(T) \leq \mathrm{SR}(T)$ and the fact that $\underset{\sim}{\mathrm{Q}}(W) = \underset{\sim}{\mathrm{SR}}(W) = 1.81...$ [13] it follows that $\underset{\sim}{\mathrm{GR}}(W) = 1.81..$, whereas $\mathrm{GR}(W) = 2$. We conclude that GR is not super-multiplicative. We have seen already in Remark 19 that GR is not sub-multiplicative.

▶ Remark 22. Geometric rank GR is not the same as slice rank SR. For example, for the matrix multiplication tensor $\langle m, m, m\rangle$ we find that $\mathrm{GR}(\langle m, m, m\rangle) = \lceil \frac{3}{4}m^2 \rceil$ (Theorem 17), whereas it was known that $\mathrm{SR}(\langle m, m, m\rangle) = m^2$ [7, Remark 4.9].

7 Geometric rank versus slice rank

In Section 4 we proved, by chaining the basic properties of geometric rank, that geometric rank is at most slice rank, that is, $\mathrm{GR}(T) \leq \mathrm{SR}(T)$. What is the largest gap between $\mathrm{GR}(T)$ and $\mathrm{SR}(T)$? Motivated by this question, and motivated by the analogous question for analytic rank instead of geometric rank that we discussed in the introduction we give a direct proof of the inequality $\mathrm{GR}(T) \leq \mathrm{SR}(T)$.

In fact, we prove a chain of inequalities $\mathrm{GR}(T) \leq \mathrm{ZR}(T) \leq \mathrm{SR}(T)$ where $\mathrm{ZR}(T)$ is defined as follows. We will use the following notation for a tensor $T \in \mathbb{F}^{n_1\times n_2\times n_3}$;

$$\mathbf{V}(T) = \{(x, y) \in \mathbb{F}^{n_1\times n_2} \mid \forall z \in \mathbb{F}^{n_3} : T(x, y, z) = 0\}. \tag{4}$$

Moreover, we use the following standard notation for the variety cut out by polynomials $f_1, \ldots, f_s$;

$$\mathbf{V}(f_1, \ldots, f_s) = \{x \mid f_1(x) = \cdots = f_s(x) = 0\}. \tag{5}$$

Let $\mathbb{F}[\mathbf{x}, \mathbf{y}] = \mathbb{F}[x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}]$ and let

$$\mathbb{F}[\mathbf{x}, \mathbf{y}, \mathbf{z}] = \mathbb{F}[x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}, z_1, \ldots, z_{n_3}].$$

Let $\mathbb{F}[\mathbf{x}, \mathbf{y}]_{\{(0,1),(1,0),(1,1)\}} \subseteq \mathbb{F}[\mathbf{x}, \mathbf{y}]$ be the subset of polynomials that are bi-homogeneous of bi-degree $(0, 1)$, $(1, 0)$ or $(1, 1)$. That is, the set $\mathbb{F}[\mathbf{x}, \mathbf{y}]_{\{(0,1),(1,0),(1,1)\}}$ contains the polynomials in $\mathbb{F}[x_1, \ldots, x_{n_1}]$ that are homogeneous of degree 1, and the polynomials in $\mathbb{F}[y_1, \ldots, y_{n_2}]$ that are homogeneous of degree 1, and the polynomials in $\mathbb{F}[\mathbf{x}, \mathbf{y}]$ that are homogeneous of degree 1 in $x_1, \ldots, x_{n_1}$ and homogeneous of degree 1 in $y_1, \ldots, y_{n_2}$. For any tensor T we define

$$\mathrm{ZR}(T) = \min\big\{s \in \mathbb{N} \mid \exists f_1, \ldots, f_s \in \mathbb{F}[\mathbf{x}, \mathbf{y}]_{\{(0,1),(1,0),(1,1)\}} : \mathbf{V}(f_1, \ldots, f_s) \subseteq \mathbf{V}(T)\big\}.$$

▶ **Theorem 23.** *Let T be a tensor. Then $\mathrm{GR}(T) \leq \mathrm{ZR}(T) \leq \mathrm{SR}(T)$.*

Proof. We prove that $\mathrm{ZR}(T) \leq \mathrm{SR}(T)$. Let $r = \mathrm{SR}(T)$. We view T as a polynomial $T \in \mathbb{F}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$. Write $T = \sum_{i=1}^{r} T_i$ with $\mathrm{SR}(T_i) = 1$ for every i. Then $T_i = f_i g_i$ for some $f_i \in \mathbb{F}[\mathbf{x}, \mathbf{y}]_{\{(0,1),(1,0),(1,1)\}}$ and $g_i \in \mathbb{F}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$. We claim that $\mathbf{V}(f_1, \ldots, f_r) \subseteq \mathbf{V}(T)$. Indeed, if $(x, y) \in \mathbf{V}(f_1, \ldots, f_r)$, then $T_i(x, y, z) = 0$ for every i and every z, and therefore $T(x, y, z) = 0$ for every z. We conclude that $\mathrm{ZR}(T) \leq r = \mathrm{SR}(T)$.

We prove that $\mathrm{GR}(T) \leq \mathrm{ZR}(T)$. Let $s = \mathrm{ZR}(T)$. Then there are s polynomials $f_1, \ldots, f_s \in \mathbb{F}[\mathbf{x}, \mathbf{y}]_{\{(0,1),(1,0),(1,1)\}}$ such that $\mathbf{V}(f_1, \ldots, f_s) \subseteq \mathbf{V}(T)$. We have

$$\mathrm{GR}(T) = \operatorname{codim} \mathbf{V}(T) \leq \operatorname{codim} \mathbf{V}(f_1, \ldots, f_s) \leq s = \mathrm{ZR}(T),$$

where the first inequality follows from the containment $\mathbf{V}(f_1, \ldots, f_s) \subseteq \mathbf{V}(T)$ which implies that $\dim \mathbf{V}(f_1, \ldots, f_s) \leq \dim \mathbf{V}(T)$. ◀

8 Geometric rank as liminf of analytic rank

For a tensor T over $\mathbb{Z}$ and a prime number p, we denote by T_p the 3-tensor over $\mathbb{F}_p$ obtained by reducing all coefficients of T modulo p. In this section we prove the following tight relationship between $\mathrm{AR}(T_p)$ and $\mathrm{GR}(T)$.

▶ **Theorem 24.** *For every tensor T over $\mathbb{Z}$ we have*

$$\liminf_{p\to\infty} \mathrm{AR}(T_p) = \mathrm{GR}(T).$$

The starting point for the proof of Theorem 24 is the important observation that analytic rank can be written in terms of the number of $\mathbb{F}_p$-points of the algebraic variety $\mathbf{V}(T_p)$, that is, for any tensor $T \in \mathbb{Z}^{n_1\times n_2\times n_3}$,

$$\mathrm{AR}(T_p) = n_1 + n_2 - \log_p \left|\mathbf{V}(T_p)(\mathbb{F}_p)\right|.$$

For the proof of Theorem 24 we will need to prove three auxiliary results: that the Bertini–Noether Theorem can be extended to reducible varieties (Theorem 26 below), that prime fields are rich enough infinitely often to contain any finite set of algebraic numbers (Lemma 28 below), and that for any variety satisfying a mild assumption, its number of rational points in a finite field is determined by its dimension (Lemma 31 below).

8.1 Bertini–Noether Theorem

In this subsection we extend the Bertini–Noether Theorem to reducible varieties. The Bertini–Noether Theorem says that, roughly, if an variety is irreducible then applying a homomorphism on the defining equations – for example the modulo-p homomorphism – typically does not change its invariants (see Proposition 10.4.2 in [18]).

▶ **Theorem 25** (Bertini–Noether Theorem [18])**.** *Let $f_1,\dots,f_m \in R[\mathbf{x}]$, where R is an integral domain, such that $V = \mathbf{V}(f_1,\dots,f_m)$ is (absolutely) irreducible. There exists a nonzero $c \in R$ such that for every homomorphism $\phi\colon R \to \mathbb{K}$ into a field $\mathbb{K}$, if $\phi(c) \neq 0$ then $\mathbf{V}(\phi(f_1),\dots,\phi(f_m)) \subseteq \overline{\mathbb{K}}$ is (absolutely) irreducible of dimension $\dim V$ and degree $\deg V$.*[2] [3]

The version of the Berini-Noether Theorem that we need is as follows. We observe that any variety defined over a field $\mathbb{F}$, where $\mathbb{F}$ is the field of fractions of an integral domain R, can also be defined over R, by clearing denominators. For example, any variety defined over the algebraic numbers $\overline{\mathbb{Q}}$ can also be defined over the algebraic integers $\overline{\mathbb{Z}}$.

▶ **Theorem 26** (Extended Bertini–Noether Theorem)**.** *Let $f_1,\dots,f_m \in R[\mathbf{x}]$, where R is an integrally closed domain.*[4] *There exists a nonzero $C \in R$ such that for every homomorphism $\psi\colon R \to \mathbb{K}$ into a field $\mathbb{K}$, if $\psi(C) \neq 0$ then $V^\psi := \mathbf{V}(\psi(f_1),\dots,\psi(f_m)) \subseteq \overline{\mathbb{K}}$ is of dimension $\dim V$ and degree $\deg V$. Moreover, if the irreducible components of $\mathbf{V}(f_1,\dots,f_m)$ are $V_1,\dots,V_k$, where $\mathrm{I}(V_i) = \langle f_{i,j}\rangle_j$ with $f_{i,j} \in R[\mathbf{x}]$, then the irreducible components of V^ψ are $V_1^\psi,\dots,V_k^\psi$, where $V_i^\psi = \mathbf{V}(\psi(f_{i,j})_j)$.*

[2] $\phi(f_i) \in \mathbb{K}[\mathbf{x}]$ is obtained by applying ϕ on each of the coefficients of f_i.
[3] That $\deg V$ remains unchanged follows along similar lines to the proof for $\dim V$ (see Corollary 9.2.2 in [18]).
[4] The field of fractions of the integral domain R is algebraically closed.

For the proof of Theorem 26 we will need some notation and a standard auxiliary result, as follows. Let R be a (commutative) ring. For a ideal I in R, the radical of I (in R) is the ideal $\sqrt{I} = \{f \in R \mid \exists n \in \mathbb{N}\colon f^n \in I\}$. Moreover, for a ring homomorphism $\psi\colon R \to R'$ we denote $\psi(I) = \langle \psi(f) \mid f \in I \rangle$, which is an ideal in R'.

▶ **Lemma 27.** *Let I be an ideal in a ring R, and let $\psi\colon R \to R'$ be a ring homomorphism. Then $\sqrt{\psi(\sqrt{I})} = \sqrt{\psi(I)}$.*

Proof. If $p \in \sqrt{\psi(I)}$ then there is an integer n such that $p^n \in \psi(I) \subseteq \psi(\sqrt{I})$, hence $p \in \sqrt{\psi(\sqrt{I})}$.

Let $p \in \sqrt{\psi(\sqrt{I})}$, meaning there is an integer n such that $p^n \in \psi(\sqrt{I})$. Thus, we have $p^n = \sum_{i=1}^{m} g_i \psi(f_i)$ for some $m \in \mathbb{N}$, $g_i \in R'$ and $f_i \in \sqrt{I}$. Note that for every i there is an integer k_i such that $f_i^{k_i} \in I$. Let $k = \max_{1 \le i \le m} k_i$. Then

$$(p^n)^{km} = \sum_{\substack{d_1,\ldots,d_m \\ d_1+\cdots+d_m = km}} \prod_{i=1}^{m} (g_i \psi(f_i))^{d_i} \ .$$

Observe that every summand has a multiplicand $(g_i\psi(f_i))^{d_i}$ with $d_i \ge k \ge k_i$, which lies in $\psi(I)$ since $\psi(f_i)^{d_i} = \psi(f_i^{d_i})$ and $f_i^{d_i} = f_i^{d_i - k_i} f_i^{k_i} \in I$. We deduce that $p^{nkm} \in \psi(I)$, being a sum of members of the ideal $\psi(I)$. Hence $p \in \sqrt{\psi(I)}$, completing the proof. ◀

Proof of Theorem 26. We begin with some notation. Let $\mathbb{F}$ be the (algebraically closed) field of fractions of R. For any ideal J in $\mathbb{F}[\mathbf{x}]$ we denote by $J^R := J \cap R[\mathbf{x}]$ the corresponding ideal in $R[\mathbf{x}]$. With a slight abuse of notation, we abbreviate $\psi(J) := \psi(J^R)$ (which is an ideal in $\mathbb{K}[\mathbf{x}]$). Furthermore, we take $\sqrt{J^R}$ to mean the radical ideal of J in $R[\mathbf{x}]$. Observe that $\sqrt{J^R} = (\sqrt{J})^R$; indeed, $f \in (\sqrt{J})^R$ iff $f^n \in J$ and $f \in R[\mathbf{x}]$ iff $f \in \sqrt{J^R}$.

Let $I = \langle f_1, \ldots, f_m \rangle$ and $I_i = \langle f_{i,j} \rangle_j$ be ideals in $\mathbb{F}[\mathbf{x}]$. We will show that

$$\sqrt{\psi(I)} = \sqrt{\prod \psi(I_i)} \ . \tag{6}$$

We have $\mathbf{V}(I) = \bigcup_i \mathbf{V}(I_i) = \mathbf{V}(\prod_i I_i)$. By Hilbert's Nullstellensatz, $\sqrt{I} = \sqrt{\prod_i I_i}$. Next, and for the rest of this paragraph, we switch from ideals in $\mathbb{F}[\mathbf{x}]$ to ideals in $R[\mathbf{x}]$. We have

$$\sqrt{I^R} = (\sqrt{I})^R = \left(\sqrt{\prod I_i}\right)^R = \sqrt{\prod I_i^R} \ . \tag{7}$$

We deduce (6) as follows;

$$\sqrt{\psi(I)} = \sqrt{\psi(I^R)} = \sqrt{\psi(\sqrt{I^R})} = \sqrt{\psi(\sqrt{\prod I_i^R})} = \sqrt{\psi(\prod I_i^R)} = \sqrt{\prod \psi(I_i^R)} = \sqrt{\prod \psi(I_i)} \ ,$$

where the second equality follows from Lemma 27, the third follows from (7), the fourth again from Lemma 27, and the fifth using the fact that ψ is a homomorphism. It follows that

$$V^\psi := \mathbf{V}(\psi(I)) = \mathbf{V}(\sqrt{\psi(I)}) = \mathbf{V}\left(\sqrt{\prod \psi(I_i)}\right) = \mathbf{V}\left(\prod \psi(I_i)\right) = \bigcup \mathbf{V}(\psi(I_i)) = \bigcup V_i^\psi \ ,$$

where (6) is used in the third equality.

Recall that V_i is an irreducible variety defined over R. For each i, applying Theorem 25 on any generating set of $\mathrm{I}(V_i)$ in $R[\mathbf{x}]$ and on ψ implies that there is a nonzero $c_i \in R$ such that if $\psi(c_i) \neq 0$ then V_i^ψ is irreducible, of dimension $\dim \mathbf{V}_i^\psi = \dim \mathbf{V}_i$ and degree $\deg V_i^\psi = \deg V_i$. Let $C = \prod_i c_i$. Thus, if $\psi(C) \neq 0$ then $\psi(c_i) \neq 0$ for all i, which implies that $V^\psi = \bigcup_i V_i^\psi$ is a union of irreducible varieties, and moreover,

$$\dim V^\psi = \max_i \dim V_i^\psi = \max_i \dim V_i = \dim V \quad \text{and}$$

$$\deg V^\psi = \sum_i \deg V_i^\psi = \sum_i \deg V_i = \deg V.$$

This completes the proof. ◀

8.2 Modular roots

In this subsection we prove that, intuitively, every finite set of algebraic integers is contained in $\mathbb{F}_p$, for infinitely many primes p. We say that there is a positive density of primes satisfying a property $\mathcal{P} \subseteq \mathbb{P}$ (here $\mathbb{P}$ is the set of prime numbers) if $\lim_{n\to\infty} |\mathcal{P} \cap [n]|/|\mathbb{P} \cap [n]| > 0$.

▶ **Lemma 28.** *For every finite set of algebraic integers S there is a positive density of primes p for which there is a homomorphism from $\mathbb{Z}[S]$ to $\mathbb{F}_p$.*

We will use (a special case of) the Primitive Element Theorem (see, e.g., Section 6.10 in [35]).

▶ **Theorem 29** (Primitive Element Theorem in Characteristic 0 [35])**.** *Let $\mathbb{K}$ be a finite extension of a field $\mathbb{F}$ of characteristic 0. Then $\mathbb{K} = \mathbb{F}(\alpha)$ for some $\alpha \in \mathbb{K}$.*

For example, $\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

We will also rely on the following result (see Berend and Bilu [4], Theorem 2).

▶ **Theorem 30** ([4])**.** *For every polynomial $P \in \mathbb{Z}[x]$ there is a positive density of prime numbers p such that P has a root modulo p.*

Proof of Lemma 28. Consider $\mathbb{Q}(S)$, the field extension of the rationals $\mathbb{Q}$ obtained by adjoining all the elements of S. By the Primitive Element Theorem (Theorem 29) there exists $\alpha \in \mathbb{Q}(S)$ such that $\mathbb{Q}(S) = \mathbb{Q}(\alpha) = \mathbb{Q}[\alpha]$. Thus, for every $\alpha_i \in S$ there is a (univariate) polynomial $f_i \in \mathbb{Q}[x]$ such that $\alpha_i = f_i(\alpha)$. We denote by P be the minimal polynomial of α over $\mathbb{Q}$; by clearing denominators, we assume without loss of generality that $P \in \mathbb{Z}[x]$.

Let p be a prime number such that P has a root a_p modulo p and, moreover, p is larger than the absolute value of the coefficient denominators of every f_i. By Theorem 30, applied on P, there is a positive density of primes satisfying both conditions. Note that $f_i \pmod p$ is a well-defined polynomial in $\mathbb{F}_p[x]$ by our second condition on p. Consider the function ϕ_p that maps each $\alpha_i = f_i(\alpha) \in S$ to $f_i(a_p) \pmod p$. Since every member of $\mathbb{Z}[S]$ is a multivariate polynomial in the variables α_i with integer coefficients, we deduce from our first condition on p that the function ϕ_p extends to a homomorphism $\phi_p \colon \mathbb{Z}[S] \to \mathbb{F}_p$. This completes the proof. ◀

8.3 Putting everything together

We will also need the following asymptotically-tight estimate on the number of rational points in a finite field.

▶ **Lemma 31.** *For every variety V defined over a finite field $\mathbb{F}$, if V has an irreducible component of dimension $\dim V$ that is also defined over $\mathbb{F}$ then*

$$|V(\mathbb{F})| = \Theta_{\deg V,\, n}(|\mathbb{F}|^{\dim V}).$$

The proof of Lemma 31 will follow by combining the Lang-Weil Theorem [25] with a Schwartz-Zippel-type upper bound (see Claim 7.2 in [15]).

▶ **Theorem 32** (Lang–Weil Bound [25])**.** *For every (absolutely) irreducible variety V defined over a finite field $\mathbb{F}$,*

$$|V(\mathbb{F})| = |\mathbb{F}|^{\dim V}(1 + O_{\deg V,\, n}(|\mathbb{F}|^{-1/2})).$$

▶ **Lemma 33** (Generalized Schwartz–Zippel lemma [15])**.** *For every variety V defined over a finite field $\mathbb{F}$, $|V(\mathbb{F})| \leq \deg(V) \cdot |\mathbb{F}|^{\dim V}$.*

Proof of Lemma 31. For the upper bound, apply Lemma 33 on V. For the lower bound, let U be an irreducible component of V of dimension $\dim V$ that is defined over $\mathbb{F}$, as guaranteed by the statement, and apply Theorem 32 on U to obtain $|V(\mathbb{F})| \geq |U(\mathbb{F})| = \Omega_{\deg U,\, n}(|\mathbb{F}|^{\dim U}) = \Omega_{\deg V,\, n}(|\mathbb{F}|^{\dim V})$. ◀

We are now ready to prove the main result of this section.

Proof of Theorem 24. Put $d = \dim \mathbf{V}(T)$ and $r = \deg \mathbf{V}(T)$. We will use the notation in (4) and (5). We will show that $\mathbf{V}(T) \subseteq \overline{\mathbb{Q}}^N$ and $\mathbf{V}(T_p) \subseteq \overline{\mathbb{F}_p}^N$ (here $N = n_1 + n_2$) are related, for infinitely many prime numbers p, in the following sense;

$$|\,\mathbf{V}(T_p)(\mathbb{F}_p)| = \Theta_{r,\, N}(p^d). \tag{8}$$

This would complete the proof since for any such prime p,

$$\mathrm{AR}(T_p) = -\log_p\Big(\frac{|\,\mathbf{V}(T_p)(\mathbb{F}_p)|}{|\mathbb{F}_p|^N}\Big) = N - \log_p|\,\mathbf{V}(T_p)(\mathbb{F}_p)| = \mathrm{GR}(T) - \Theta_{r,\, N}\Big(\frac{1}{\log p}\Big),$$

where the last inequality follows from (8) using the fact that $N - d = \operatorname{codim} \mathbf{V}(T) = \mathrm{GR}(T)$. Thus, proving (8) would imply that $\liminf_{p\to\infty} \mathrm{AR}(T_p) = \mathrm{GR}(T)$, as needed.

Let U be an irreducible component of $\mathbf{V}(T)$ of dimension d. Note that U is defined over some finite extension $\mathbb{Z}[S]$ of the integers, where S is a finite set of algebraic integers. Lemma 28, applied on S, implies that for a positive density of prime numbers p there is a homomorphism $\phi_p\colon \mathbb{Z}[S] \to \mathbb{F}_p$. Thus, if $\mathrm{I}(U) = \mathbf{V}(f_j)_j$ with $f_j \in \mathbb{Z}[S][\mathbf{x}]$ then $U^{\phi_p} := \mathbf{V}(\phi_p(f_j)_j)$ is defined over $\mathbb{F}_p$ (rather than $\overline{\mathbb{F}_p}$). Let p be any such prime. Theorem 26, applied on $R = \overline{\mathbb{Z}}$, $\mathbb{K} = \overline{\mathbb{F}_p}$ and on any extension ψ_p of ϕ_p to a homomorphism from $\overline{\mathbb{Z}}$ to $\overline{\mathbb{F}_p}$, implies that there is $0 \neq C \in \overline{\mathbb{Z}}$ such that for any prime p with $\psi_p(C) \neq 0$, we have that $\dim \mathbf{V}(T_p) = d$, $\deg \mathbf{V}(T_p) = r$, and that $U^{\psi_p} = U^{\phi_p}$ is an irreducible component of $\mathbf{V}(T_p)$ of dimension $d = \dim \mathbf{V}(T_p)$. We claim that the condition $\psi_p(C) \neq 0$ is satisfied for all but finitely many primes p; indeed, since $\psi_p(C)$ is a root modulo p of the minimal polynomial of C over $\mathbb{Z}$, it holds that $\psi_p(C) = 0$ if and only if the constant term c of that polynomial is 0 modulo p, which is never the case for $p > |c|$ (as $c \neq 0$). Lemma 31 therefore implies, together with all of the above, that for a positive density of primes p we have

$$|\,\mathbf{V}(T_p)(\mathbb{F}_p)| = \Theta_{\deg \mathbf{V}(T_p),\, N}(p^{\dim \mathbf{V}(T_p)}) = \Theta_{r,\, N}(p^d).$$

This proves (8), and thus we are done. ◀

References

1 Josh Alman. Limits on the universal method for matrix multiplication. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*, pages 12:1–12:24, 2019. doi:10.4230/LIPIcs.CCC.2019.12.

2 Josh Alman and Virginia Vassilevska Williams. Limits on all known (and some unknown) approaches to matrix multiplication. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2018)*, pages 580–591, 2018. doi:10.1109/FOCS.2018.00061.

3 Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: limitations of the Coppersmith-Winograd method (extended abstract). In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 585–593, 2015. doi:10.1145/2746539.2746554.

4 Daniel Berend and Yuri Bilu. Polynomials with roots modulo every integer. *Proc. Amer. Math. Soc.*, 124(6):1663–1671, 1996. doi:10.1090/S0002-9939-96-03210-8.

5 Abhishek Bhowmick and Shachar Lovett. Bias vs structure of polynomials in large fields, and applications in effective algebraic geometry and coding theory. *arXiv*, 2015. arXiv:1506.02047.

6 Abhishek Bhrushundi, Prahladh Harsha, Pooya Hatami, Swastik Kopparty, and Mrinal Kumar. On multilinear forms: Bias, correlation, and tensor rank. *arXiv*, 2018. arXiv:1804.09124.

7 Jonah Blasiak, Thomas Church, Henry Cohn, Joshua A. Grochow, Eric Naslund, William F. Sawin, and Chris Umans. On cap sets and the group-theoretic approach to matrix multiplication. *Discrete Anal.*, 2017. doi:10.19086/da.1245.

8 Jonah Blasiak, Thomas Church, Henry Cohn, Joshua A Grochow, and Chris Umans. Which groups are amenable to proving exponent two for matrix multiplication? *arXiv*, 2017. arXiv:1712.02302.

9 Markus Bläser, Christian Ikenmeyer, Vladimir Lysikov, Anurag Pandey, and Frank-Olaf Schreyer. Variety membership testing, algebraic natural proofs, and geometric complexity theory. *arXiv*, 2019. arXiv:1911.02534.

10 Jop Briët. Subspaces of tensors with high analytic rank. *arXiv*, 2019. arXiv:1908.04169.

11 Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren Math. Wiss.* Springer-Verlag, Berlin, 1997. doi:10.1007/978-3-662-03338-8.

12 Matthias Christandl, Angelo Lucia, Péter Vrana, and Albert H. Werner. Tensor network representations from the geometry of entangled states. *arXiv*, 2018. arXiv:1809.08185.

13 Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Universal points in the asymptotic spectrum of tensors (extended abstract). In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18)*. ACM, 2018. doi:10.1145/3188745.3188766.

14 Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Barriers for Fast Matrix Multiplication from Irreversibility. In Amir Shpilka, editor, *34th Computational Complexity Conference (CCC 2019)*, volume 137, pages 26:1–26:17, 2019. doi:10.4230/LIPIcs.CCC.2019.26.

15 Zeev Dvir, János Kollár, and Shachar Lovett. Variety evasive sets. *Comput. Complexity*, 23(4):509–529, 2014. doi:10.1007/s00037-013-0073-9.

16 Jordan S. Ellenberg and Dion Gijswijt. On large subsets of $\mathbb{F}_q^n$ with no three-term arithmetic progression. *Ann. of Math. (2)*, 185(1):339–343, 2017. doi:10.4007/annals.2017.185.1.8.

17 Yuval Filmus, Konstantin Golubev, and Noam Lifshitz. High dimensional hoffman bound and applications in extremal combinatorics. *arXiv*, 2019. arXiv:1911.02297.

18 Michael D. Fried and Moshe Jarden. *Field arithmetic*, volume 11 of *Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge. A Series of Modern Surveys in Mathematics*. Springer-Verlag, Berlin, second edition, 2005.

19 W. T. Gowers and J. Wolf. Linear forms and higher-degree uniformity for functions on $\mathbb{F}_p^n$. *Geom. Funct. Anal.*, 21(1):36–69, 2011. doi:10.1007/s00039-010-0106-3.

20 Daniel R. Grayson and Michael E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `http://www.math.uiuc.edu/Macaulay2/`.

21 Joe Harris. *Algebraic geometry: a first course*, volume 133. Springer Science & Business Media, 1992.

22 Oliver Janzer. Low analytic rank implies low partition rank for tensors. *arXiv*, 2018. `arXiv:1809.10931`.

23 Oliver Janzer. Polynomial bound for the partition rank vs the analytic rank of tensors. *arXiv*, 2019. `arXiv:1902.11207`.

24 P. Koiran. Randomized and deterministic algorithms for the dimension of algebraic varieties. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 36–45, October 1997. `doi:10.1109/SFCS.1997.646091`.

25 Serge Lang and André Weil. Number of points of varieties in finite fields. *Amer. J. Math.*, 76:819–827, 1954. `doi:10.2307/2372655`.

26 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC 2014)*, pages 296–303, 2014. `doi:10.1145/2608628.2608664`.

27 Shachar Lovett. The analytic rank of tensors and its applications. *Discrete Analysis*, 2019. `doi:10.19086/da`.

28 Luka Milićević. Polynomial bound for partition rank in terms of analytic rank. *Geometric and Functional Analysis*, 29(5):1503–1530, 2019. `doi:10.1007/s00039-019-00505-4`.

29 Eric Naslund and Will Sawin. Upper bounds for sunflower-free sets. In *Forum of Mathematics, Sigma*, volume 5. Cambridge University Press, 2017.

30 Sage. *SageMath, the Sage Mathematics Software System (Version 8.1)*, 2017. URL: `https://www.sagemath.org`.

31 Volker Strassen. Relative bilinear complexity and matrix multiplication. *J. Reine Angew. Math.*, 375/376:406–443, 1987. `doi:10.1515/crll.1987.375-376.406`.

32 Volker Strassen. The asymptotic spectrum of tensors. *J. Reine Angew. Math.*, 384:102–152, 1988. `doi:10.1515/crll.1988.384.102`.

33 Volker Strassen. Degeneration and complexity of bilinear maps: some asymptotic spectra. *J. Reine Angew. Math.*, 413:127–180, 1991. `doi:10.1515/crll.1991.413.127`.

34 Terence Tao. A symmetric formulation of the Croot-Lev-Pach-Ellenberg-Gijswijt capset bound, 2016. URL: `https://terrytao.wordpress.com/2016/05/18/a-symmetric-formulation-of-the-croot-lev-pach-ellenberg-gijswijt-capset-bound`.

35 B. L. van der Waerden. *Algebra. Vol. I.* Springer-Verlag, New York, 1991. `doi:10.1007/978-1-4612-4420-2`.

Hardness of Bounded Distance Decoding on Lattices in ℓ_p Norms

Huck Bennett
University of Michigan, Ann Arbor, MI, USA
hdbco@umich.edu

Chris Peikert
University of Michigan, Ann Arbor, MI, USA
cpeikert@umich.edu

Abstract

Bounded Distance Decoding $\mathrm{BDD}_{p,\alpha}$ is the problem of decoding a lattice when the target point is promised to be within an α factor of the minimum distance of the lattice, in the ℓ_p norm. We prove that $\mathrm{BDD}_{p,\alpha}$ is NP-hard under randomized reductions where $\alpha \to 1/2$ as $p \to \infty$ (and for $\alpha = 1/2$ when $p = \infty$), thereby showing the hardness of decoding for distances approaching the unique-decoding radius for large p. We also show *fine-grained* hardness for $\mathrm{BDD}_{p,\alpha}$. For example, we prove that for all $p \in [1, \infty) \setminus 2\mathbb{Z}$ and constants $C > 1, \varepsilon > 0$, there is no $2^{(1-\varepsilon)n/C}$-time algorithm for $\mathrm{BDD}_{p,\alpha}$ for some constant α (which approaches $1/2$ as $p \to \infty$), assuming the randomized Strong Exponential Time Hypothesis (SETH). Moreover, essentially all of our results also hold (under analogous non-uniform assumptions) for BDD with *preprocessing*, in which unbounded precomputation can be applied to the lattice before the target is available.

Compared to prior work on the hardness of $\mathrm{BDD}_{p,\alpha}$ by Liu, Lyubashevsky, and Micciancio (APPROX-RANDOM 2008), our results improve the values of α for which the problem is known to be NP-hard for all $p > p_1 \approx 4.2773$, and give the very first fine-grained hardness for BDD (in any norm). Our reductions rely on a special family of "locally dense" lattices in ℓ_p norms, which we construct by modifying the integer-lattice sparsification technique of Aggarwal and Stephens-Davidowitz (STOC 2018).

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

Keywords and phrases Lattices, Bounded Distance Decoding, NP-hardness, Fine-Grained Complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.36

Related Version https://arxiv.org/abs/2003.07903

Acknowledgements We thank the Simons Institute for hosting the Spring 2020 program "Lattices: Algorithms, Complexity, and Cryptography," at which some of this work was completed. We also thank Noah Stephens-Davidowitz for sharing his plot-generating code from [5] with us.

1 Introduction

Lattices in $\mathbb{R}^n$ are a rich source of computational problems with applications across computer science, and especially in cryptography and cryptanalysis. (A lattice is a discrete additive subgroup of $\mathbb{R}^n$, or equivalently, the set of integer linear combinations of a set of linearly independent vectors.) Many important lattice problems appear intractable, and there is a wealth of research showing that central problems like the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP) are NP-hard, even to approximate to within various factors and in various ℓ_p norms [31, 8, 7, 22, 23, 17, 16, 14, 25]. (For the sake of concision, throughout this introduction the term "NP-hard" allows for *randomized* reductions, which are needed in some important cases.)

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 36; pp. 36:1–36:21


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

Bounded Distance Decoding

In recent years, the emergence of lattices as a powerful foundation for cryptography, including for security against quantum attacks, has increased the importance of other lattice problems. In particular, many modern lattice-based encryption schemes rely on some form of the *Bounded Distance Decoding* (BDD) problem, which is like the Closest Vector Problem with a promise. An instance of BDD_α for *relative distance* $\alpha > 0$ is a lattice $\mathcal{L}$ and a target point $\boldsymbol{t}$ whose distance from the lattice is guaranteed to be within an α factor of the lattice's minimum distance $\lambda_1(\mathcal{L}) = \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{\mathbf{0}\}} \|\boldsymbol{v}\|$, and the goal is to find a lattice vector within that distance of $\boldsymbol{t}$; when distances are measured in the ℓ_p norm we denote the problem $\mathrm{BDD}_{p,\alpha}$. Note that when $\alpha < 1/2$ there is a unique solution, but the problem is interesting and well-defined for larger relative distances as well. We also consider *preprocessing* variants of CVP and BDD (respectively denoted CVPP and BDDP), in which unbounded precomputation can be applied to the lattice before the target is available. For example, this can model cryptographic contexts where a fixed long-term lattice may be shared among many users.

The importance of BDD(P) to cryptography is especially highlighted by the Learning With Errors (LWE) problem of Regev [27], which is an average-case form of BDD that has been used (with inverse-polynomial α) in countless cryptosystems, including several that share a lattice among many users (see, e.g., [13]). Moreover, Regev gave a worst-case to average-case reduction from BDD to LWE, so the security of cryptosystems is intimately related to the worst-case complexity of BDD.

Compared to problems like SVP and CVP, the BDD(P) problem has received much less attention from a complexity-theoretic perspective. We are aware of essentially only one work showing its NP-hardness: Liu, Lyubashevsky, and Micciancio [19] proved that $\mathrm{BDD}_{p,\alpha}$ and even $\mathrm{BDDP}_{p,\alpha}$ are NP-hard for relative distances approaching $\min\{1/\sqrt{2}, 1/\sqrt[p]{2}\}$, which is $1/\sqrt{2}$ for $p \geq 2$. A few other works relate BDD(P) to other lattice problems (in both directions) in regimes where the problems are not believed to be NP-hard, e.g., [24, 11, 9]. (Dadush, Regev, and Stephens-Davidowitz [11] also gave a reduction that implies NP-hardness of $\mathrm{BDD}_{2,\alpha}$ for any $\alpha > 1$, which is larger than the relative distance of $\alpha = 1/\sqrt{2} + \varepsilon$ achieved by [19].)

Fine-grained hardness

An important aspect of hard lattice problems, especially for cryptography, is their *quantitative* hardness. That is, we want not only that a problem cannot be solved in polynomial time, but that it cannot be solved in, say, $2^{o(n)}$ time or even $2^{n/C}$ time for a certain constant C. Statements of this kind can be proven under generic complexity assumptions like the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [15] or its variants like Strong ETH (SETH), via *fine-grained* reductions that are particularly efficient in the relevant parameters.

Recently, Bennett, Golovnev, and Stephens-Davidowitz [10] initiated a study of the fine-grained hardness of lattice problems, focusing on CVP; follow-up work extended to SVP and showed more for CVP(P) [5, 2]. The technical goal of these works is a reduction having good *rank efficiency*, i.e., a reduction from k-SAT on n' variables to a lattice problem in rank $n = (C + o(1))n'$ for some constant $C \geq 1$, which we call the reduction's "rank inefficiency." (All of the lattice problems in question can be solved in $2^{n+o(n)}$ time [3, 4, 6], so $C = 1$ corresponds to optimal rank efficiency.) We mention that Regev's BDD-to-LWE reduction [27] has optimal rank efficiency, in that it reduces rank-n BDD to rank-n LWE. However, to date there are no fine-grained NP-hardness results for BDD itself; the prior NP-hardness proof for BDD [19] incurs a large polynomial blowup in rank.

1.1 Our Results

We show improved NP-hardness, and entirely new fine-grained hardness, for Bounded Distance Decoding (and BDD with preprocessing) in arbitrary ℓ_p norms. Our work improves upon the known hardness of BDD in two respects: the relative distance α, and the rank inefficiency C (i.e., fine-grainedness) of the reductions. As p grows, both quantities improve, simultaneously approaching the unique-decoding threshold $\alpha = 1/2$ and optimal rank efficiency of $C = 1$ as $p \to \infty$, and achieving those quantities for $p = \infty$. We emphasize that these are the first fine-grained hardness results of any kind for BDD, for any ℓ_p norm.

Our main theorem summarizing the NP- and fine-grained hardness of BDD (with and without preprocessing) appears below in Theorem 1. For $p \in [1, \infty)$ and $C > 1$, the quantities α_p^* and $\alpha_{p,C}^*$ appearing in the theorem statement are certain positive real numbers that are decreasing in p and C, and approaching $1/2$ as $p \to \infty$ (for any C). See Figure 1 for a plot of their behavior, Equations (3.4) and (3.5) for their formal definitions, and Lemma 27 for quite tight closed-form upper bounds.

▶ **Theorem 1.** *The following hold for* $\mathrm{BDD}_{p,\alpha}$ *and* $\mathrm{BDDP}_{p,\alpha}$ *in rank* n*:*

1. *For every* $p \in [1, \infty)$ *and constant* $\alpha > \alpha_p^*$ *(where* $\alpha_p^* \le \frac{1}{2} \cdot 4.6723^{1/p}$*), and for* $(p, \alpha) = (\infty, 1/2)$*, there is no polynomial-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *(respectively,* $\mathrm{BDDP}_{p,\alpha}$*) unless* NP $\subseteq$ RP *(resp.,* NP $\subseteq$ P/Poly*).*
2. *For every* $p \in [1, \infty)$ *and constant* $\alpha > \min\{\alpha_p^*, \alpha_2^*\}$*, and for* $(p, \alpha) = (\infty, 1/2)$*, there is no* $2^{o(n)}$*-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *unless randomized ETH fails.*
3. *For every* $p \in [1, \infty) \setminus \{2\}$ *and constant* $\alpha > \alpha_p^*$*, and for* $(p, \alpha) = (\infty, 1/2)$*, there is no* $2^{o(n)}$*-time algorithm for* $\mathrm{BDDP}_{p,\alpha}$ *unless non-uniform ETH fails.*
   Moreover, for every $p \in [1, \infty]$ *and* $\alpha > \alpha_2^*$ *there is no* $2^{o(\sqrt{n})}$*-time algorithm for* $\mathrm{BDDP}_{p,\alpha}$ *unless non-uniform ETH fails.*
4. *For every* $p \in [1, \infty) \setminus 2\mathbb{Z}$ *and constants* $C > 1$*,* $\alpha > \alpha_{p,C}^*$*, and* $\epsilon > 0$*, and for* $(p, C, \alpha) = (\infty, 1, 1/2)$*, there is no* $2^{n(1-\epsilon)/C}$*-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *(respectively,* $\mathrm{BDDP}_{p,\alpha}$*) unless randomized SETH (resp., non-uniform SETH) fails.*

Although we do not have closed-form expressions for α_p^* and $\alpha_{p,C}^*$, we do get quite tight closed-form upper bounds (see Lemma 27). Moreover, it is easy to numerically compute close approximations to them, and to the values of p at which they cross certain thresholds. For example, $\alpha_p^* < 1/\sqrt{2}$ for all $p > p_1 \approx 4.2773$, so Item 1 of Theorem 1 improves on the prior best relative distance of any $\alpha > 1/\sqrt{2}$ for the NP-hardness of $\mathrm{BDD}_{p,\alpha}$ in such ℓ_p norms [19].

As a few other example values and their consequences under Theorem 1, we have $\alpha_2^* \approx 1.05006$, $\alpha_{3,2}^* \approx 1.1418$, and $\alpha_{3,5}^* \approx 0.917803$. So by Item 2, BDD in the Euclidean norm for any relative distance $\alpha > 1.05006$ requires $2^{\Omega(n)}$ time assuming randomized ETH. And by Item 4, for every $\varepsilon > 0$ there is no $2^{(1-\varepsilon)n/2}$-time algorithm for $\mathrm{BDD}_{3,1.1418}$, and no $2^{(1-\varepsilon)n/5}$-time algorithm for $\mathrm{BDD}_{3,0.917803}$, assuming randomized SETH.

1.2 Technical Overview

As in prior NP-hardness reductions for SVP and BDD (and fine-grained hardness proofs for the former) [7, 22, 16, 19, 14, 25, 5], the central component of our reductions is a family of rank-n lattices $\mathcal{L} \subset \mathbb{R}^d$ and target points $\boldsymbol{t} \in \mathbb{R}^d$ having a certain "local density" property in a desired ℓ_p norm. Informally, this means that $\mathcal{L}$ has "large" minimum distance $\lambda_1^{(p)}(\mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{\boldsymbol{0}\}} \|\boldsymbol{v}\|_p$, i.e., there are no "short" nonzero vectors, but has many vectors "close" to the target $\boldsymbol{t}$. More precisely, we want $\lambda_1^{(p)}(\mathcal{L}) \ge r$ and $N_p(\mathcal{L}, \alpha r, \boldsymbol{t}) = \exp(n^{\Omega(1)})$ for some relative distance α, where

$$N_p(\mathcal{L}, s, \boldsymbol{t}) := |\{\boldsymbol{v} \in \mathcal{L} : \|\boldsymbol{v} - \boldsymbol{t}\|_p \le s\}|$$

denotes the number of lattice points within distance s of $\boldsymbol{t}$.

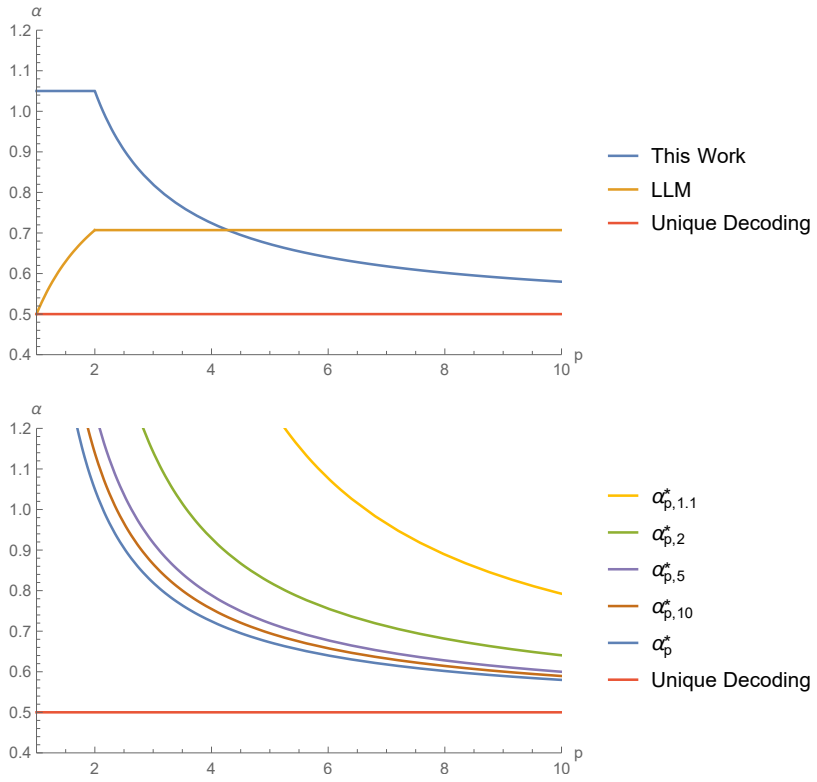


Figure 1 Top: bounds on the relative distances $\alpha = \alpha(p)$ for which $\mathrm{BDD}_{\alpha,p}$ was proved to be NP-hard in the ℓ_p norm, in this work and in [19]; the crossover point is $p_1 \approx 4.2773$. (The plots include results obtained by norm embeddings [28], hence they are maximized at $p = 2$.) Bottom: our bounds $\alpha^*_{p,C}$ on the relative distances $\alpha > \alpha^*_{p,C}$ for which there is no $2^{(1-\varepsilon)n/C}$-time algorithm for $\mathrm{BDD}_{p,\alpha}$ for any $\varepsilon > 0$, assuming randomized SETH.

Micciancio [22] constructed locally dense lattices with relative distance approaching $2^{-1/p}$ in the ℓ_p norm (for every finite $p \geq 1$), and used them to prove the NP-hardness of γ-approximate SVP in ℓ_p for any $\gamma < 2^{1/p}$. Subsequently, Liu, Lyubashevsky, and Micciancio [19] used these lattices to prove the NP-hardness of BDD in ℓ_p for any relative distance $\alpha > 2^{-1/p}$. However, these works observed that the relative distance depends on p in the opposite way from what one might expect: as p grows, so does α, hence the associated NP-hard SVP approximation factors and BDD relative distances *worsen*. Yet using norm embeddings, it can be shown that ℓ_2 is essentially the "easiest" ℓ_p norm for lattice problems [28], so hardness in ℓ_2 implies hardness in ℓ_p (up to an arbitrarily small loss in approximation factor). Therefore, the locally dense lattices from [22] do not seem to provide any benefits for $p > 2$ over $p = 2$, where the relative distance approaches $1/\sqrt{2}$. In addition, the rank of these lattices is a large polynomial in the relevant parameter, so they are not suitable for proving fine-grained hardness.[1]

[1] We mention that Khot [16] gave a different construction of locally dense lattices with other useful properties, but their relative distance is no smaller than that of Micciancio's construction in any ℓ_p norm, and their rank is also a large polynomial in the relevant parameter.

Local density via sparsification

More recently, Aggarwal and Stephens-Davidowitz [5] (building on [10]) proved fine-grained hardness for *exact* SVP in ℓ_p norms, via locally dense lattices obtained in a different way. Because they target exact SVP, it suffices to have local density for relative distance $\alpha = 1$, but for fine-grained hardness they need $N_p(\mathcal{L}, r, \boldsymbol{t}) = 2^{\Omega(n)}$, preferably with a large hidden constant (which determines the rank efficiency of the reduction). Following [21, 12], they start with the integer lattice $\mathbb{Z}^n$ and all-$\frac{1}{2}$s target vector $\boldsymbol{t} = \frac{1}{2}\mathbf{1} \in \mathbb{R}^n$. Clearly, there are 2^n lattice vectors all at distance $r = \frac{1}{2}n^{1/p}$ from $\boldsymbol{t}$ in the ℓ_p norm, but the minimum distance of the lattice is only 1, so the relative distance of the "close" vectors is $\alpha = r$, which is far too large.

To improve the relative distance, they increase the minimum distance to at least $r = \frac{1}{2}n^{1/p}$ using the elegant technique of *random sparsification*, which is implicit in [12] and was first used for proving NP-hardness of approximate SVP in [17, 16]. The idea is to upper-bound the number $N_p(\mathbb{Z}^n, r, \mathbf{0})$ of "short" lattice points of length at most r, by some Q. Then, by taking a random sublattice $\mathcal{L} \subset \mathbb{Z}^n$ of determinant (index) slightly larger than Q, with noticeable probability none of the "short" nonzero vectors will be included in $\mathcal{L}$, whereas roughly $2^n/Q$ of the vectors "close" to $\boldsymbol{t}$ will be in $\mathcal{L}$. So, as long as $Q = 2^{(1-\Omega(1))n}$, there are sufficiently many lattice vectors at the desired relative distance from $\boldsymbol{t}$.

Bounds for $N_p(\mathbb{Z}^n, r, \mathbf{0})$ were given by Mazo and Odlyzko [21], by a simple but powerful technique using the theta function $\Theta_p(\tau) := \sum_{z\in\mathbb{Z}} \exp(-\tau|z|^p)$. They showed (see Proposition 13) that

$$N_p(\mathbb{Z}^n, r, \mathbf{0}) \le \min_{\tau>0} \exp(\tau \cdot r^p) \cdot \Theta_p(\tau)^n = \left(\min_{\tau>0} \exp(\tau/2^p) \cdot \Theta_p(\tau)\right)^n , \tag{1.1}$$

where the equality is by $r = \frac{1}{2}n^{1/p}$. So, Aggarwal and Stephens-Davidowitz need $\min_{\tau>0} \exp(\tau/2^p) \cdot \Theta_p(\tau) < 2$, and it turns out that this is the case for every $p > p_0 \approx 2.1397$. (They also deal with smaller p by using a different target point $\boldsymbol{t}$.)

This work: local density for small relative distance

For the NP- and fine-grained hardness of BDD we use the same basic approach as in [5], but with the different goal of getting local density for as small of a relative distance $\alpha < 1$ as we can manage. That is, we still have 2^n integral vectors all at distance $r = \frac{1}{2}n^{1/p}$ from the target $\boldsymbol{t} = \frac{1}{2}\mathbf{1} \in \mathbb{R}^n$, but we want to "sparsify away" all the nonzero integral vectors of length less than r/α. So, we want the right-hand side of the Mazo-Odlyzko bound (Equation (1.1)) to be at most $2^{(1-\Omega(1))n}$ for as large of a positive hidden constant as we can manage. More specifically, for any $p \ge 1$ and $C > 1$ (which ultimately corresponds to the reduction's rank inefficiency) we can obtain local density of at least $2^{n/C}$ close vectors at any relative distance greater than

$$\alpha^*_{p,C} := \inf\{\alpha^* > 0 : \min_{\tau>0} \exp(\tau/(2\alpha^*)^p) \cdot \Theta_p(\tau) \le 2^{1-1/C}\} .$$

The value of $\alpha^*_{p,C}$ is strictly decreasing in both p and C, and for large C and $p > p_1 \approx 4.2773$ it drops below the relative distance of $1/\sqrt{2}$ approached by the local-density construction of [22] for ℓ_2 (and also ℓ_p by norm embeddings.) This is the source of our improved relative distance for the NP-hardness of BDD in high ℓ_p norms.

We also show that obtaining local density by sparsifying the integer lattice happens to yield a very simple reduction to BDD from the *exact* version of CVP, which is how we obtain fine-grained hardness. Given a CVP instance consisting of a lattice and a target point, we

essentially just take their direct sum with the integer lattice and the $\frac{1}{2}\mathbf{1}$ target (respectively), then sparsify. (See Lemma 21 and Theorem 19 for details.) Because this results in the (sparsified) locally dense lattice having $2^{\Omega(n)}$ close vectors all *exactly* at the threshold of the BDD promise, concatenating the CVP instance either keeps the target within the (slightly weaker) BDD promise, or puts it just outside. This is in contrast to the prior reduction of [19], where the close vectors in the locally dense lattices of [22] are at various distances from the target, hence a reduction from *approximate*-CVP with a large constant factor is needed to put the target outside the BDD promise. While approximating CVP to within any constant factor is known to be NP-hard [8], no fine-grained hardness is known for approximate CVP, except for factors just slightly larger than one [2].

1.3 Discussion and Future Work

Our work raises a number of interesting issues and directions for future research. First, it highlights that there are now two incomparable approaches for obtaining local density in the ℓ_p norm – Micciancio's construction [22], and sparsifying the integer lattice [12, 5] – with each delivering a better relative distance for certain ranges of p. For $p \in [1, p_1 \approx 4.2773]$, Micciancio's construction (with norm embeddings from ℓ_2, where applicable) delivers the better relative distance, which approaches $\min\{1/\sqrt[p]{2}, 1/\sqrt{2}\}$. Moreover, this is essentially optimal in ℓ_2, where $1/\sqrt{2}$ is unachievable due to the Rankin bound, which says that in $\mathbb{R}^n$ we can have at most $2n$ subunit vectors with pairwise distances of $\sqrt{2}$ or more.

A first question, therefore, is whether relative distance less than $1/\sqrt{2}$ can be obtained for all $p > 2$. We conjecture that this is true, but can only manage to prove it via sparsification for all $p > p_1 \approx 4.2773$. More generally, an important open problem is to give a unified local-density construction that subsumes both of the above-mentioned approaches in terms of relative distance, and ideally in rank efficiency as well. In the other direction, another important goal is to give lower bounds on the relative distance in general ℓ_p norms. Apart from the Rankin bound, the only bound we are aware of is the trivial one of $\alpha \geq 1/2$ implied by the triangle inequality, which is essentially tight for ℓ_1 and tight for ℓ_∞ (as shown by [22] and our work, respectively).

More broadly, for the BDD relative distance parameter α there are three regimes of interest: the local-density regime, where we know how to prove NP-hardness; the unique-decoding regime $\alpha < 1/2$; and (at least in some ℓ_p norms, including ℓ_2) the intermediate regime between them. It would be very interesting, and would seem to require new techniques, to show NP-hardness outside the local-density regime. One potential route would be to devise a *gap amplification* technique for BDD, analogous to how SVP has been proved to be NP-hard to approximate to within any constant factor [16, 14, 25]. Gap amplification may also be interesting in the absence of NP-hardness, e.g., for the inverse-polynomial relative distances used in cryptography. Currently, the only efficient gap amplification we are aware of is a modest one that decreases the relative distance by any $(1 - 1/n)^{O(1)}$ factor [20].

A final interesting research direction is related to the *unique* Shortest Vector Problem (uSVP), where the goal is to find a shortest nonzero vector $\boldsymbol{v}$ in a given lattice, under the promise that it is unique (up to sign). More generally, approximate uSVP has the promise that all lattice vectors not parallel to $\boldsymbol{v}$ are a certain factor γ as long. It is known that *exact* uSVP is NP-hard in ℓ_2 [18], and by known reductions it is straightforward to show the NP-hardness of 2-*approximate* uSVP in ℓ_∞. Can recent techniques help to prove NP-hardness of γ-approximate uSVP, for some constant $\gamma > 1$, in ℓ_p for some finite p, or specifically for ℓ_2? Do NP-hard approximation factors for uSVP grow smoothly with p?

2 Preliminaries

For any positive integer q, we identify the quotient group $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ with some set of distinguished representatives, e.g., $\{0, 1, \ldots, q-1\}$. Let $B^+ := (B^t B)^{-1} B^t$ denote the Moore-Penrose pseudoinverse of a real-valued matrix B with full column rank. Observe that $B^+ \boldsymbol{v}$ is the unique coefficient vector $\boldsymbol{c}$ with respect to B of any $\boldsymbol{v} = B\boldsymbol{c}$ in the column span of B.

2.1 Problems with Preprocessing

In addition to ordinary computational problems, we are also interested in (promise) problems with *preprocessing*. In such a problem, an instance (x_P, x_Q) is comprised of a "preprocessing" part x_P and a "query" part x_Q, and an algorithm is allowed to perform unbounded computation on the preprocessing part before receiving the query part.

Formally, a preprocessing problem is a relation $\Pi = \{((x_P, x_Q), y)\}$ of instance-solution pairs, where $\Pi_{\text{inst}} := \{(x_P, x_Q) : \exists\ y \text{ s.t. } ((x_P, x_Q), y) \in \Pi\}$ is the set of problem instances, and $\Pi_{(x_P, x_Q)} := \{y : ((x_P, x_Q), y) \in \Pi\}$ is the set of solutions for any particular instance (x_P, x_Q). If every instance $(x_P, x_Q) \in \Pi_{\text{inst}}$ has exactly one solution that is either YES or NO, then Π is called a decision problem.

▶ **Definition 2.** *A* preprocessing algorithm *is a pair (P, Q) where P is a (possibly randomized) function representing potentially unbounded computation, and Q is an algorithm. The execution of (P, Q) on an input (x_P, x_Q) proceeds in two phases:*

- *first, in the preprocessing phase, P takes x_P as input and produces some preprocessed output σ;*
- *then, in the query phase, Q takes both σ and x_Q as input and produces some ultimate output.*

The running time T of the algorithm is defined to be the time used in the query phase alone, and is considered as a function of the total input length $|x_P| + |x_Q|$. The length of the preprocessed output is defined as $A = |\sigma|$, and is also considered as a function of the total input length. Note that without loss of generality, $A \leq T$.

If (P, Q) is deterministic, we say that it solves preprocessing problem Π if $Q(P(x_P), x_Q) \in \Pi_{(x_P, x_Q)}$ for all $(x_P, x_Q) \in \Pi_{inst}$. If (P, Q) is potentially randomized, we say that it solves Π if

$$\Pr[Q(P(x_P), x_Q) \in \Pi_{(x_P, x_Q)}] \geq \frac{2}{3}$$

for all $(x_P, x_Q) \in \Pi_{inst}$, where the probability is taken over the random coins of both P and Q.[2]

As shown below using a routine quantifier-swapping argument (as in Adleman's Theorem [1]), it turns out that for NP relations and decision problems, any randomized preprocessing algorithm can be derandomized if the length of the query input x_Q is polynomial in the length of the preprocessing input x_P. So for convenience, in this work we allow for randomized algorithms, only switching to deterministic ones for our ultimate hardness theorems.

[2] Note that it could be the case that some preprocessed outputs fail to make the query algorithm output a correct answer on some, or even all, query inputs.

▶ **Lemma 3.** *Let preprocessing problem* Π *be an* NP *relation or a decision problem for which* $|x_Q| = \mathrm{poly}(|x_P|)$ *for all* $(x_P, x_Q) \in \Pi_{inst}$*. If* Π *has a randomized* T*-time algorithm, then it has a deterministic* $T \cdot \mathrm{poly}(|x_P| + |x_Q|)$*-time algorithm with* $T \cdot \mathrm{poly}(|x_P| + |x_Q|)$*-length preprocessed output.*

Proof. Let $q(\cdot)$ be a polynomial for which $|x_Q| \le q(|x_P|)$ for all $(x_P, x_Q) \in \Pi_{\mathrm{inst}}$. Let (P, Q) be a randomized T-time algorithm for Π, which by standard repetition techniques we can assume has probability strictly less than $\exp(-q(|x_P|))$ of being incorrect on any $(x_P, x_Q) \in \Pi_{\mathrm{inst}}$, with only a $\mathrm{poly}(|x_P| + |x_Q|)$-factor overhead in the running time and preprocessed output length. Fix some arbitrary x_P. Then by the union bound over all $(x_P, x_Q) \in \Pi_{\mathrm{inst}}$ and the hypothesis, we have

$$\Pr[\exists\ (x_P, x_Q) \in \Pi_{\mathrm{inst}} : Q(P(x_P), x_Q) \notin \Pi_{(x_P, x_Q)}] < 1.$$

So, there exist coins for P and Q for which $Q(P(x_P), x_Q) \in \Pi_{(x_P, x_Q)}$ for all $(x_P, x_Q) \in \Pi_{\mathrm{inst}}$. By fixing these coins we make P a deterministic function of x_P, and we include the coins for Q along with the preprocessed output $P(x_P)$, thus making Q deterministic as well. The resulting deterministic algorithm solves Π with the claimed resources, as needed. ◀

Reductions for preprocessing problems

We need the following notions of reductions for preprocessing problems. The following generalizes Turing reductions and Cook reductions (i.e., polynomial-time Turing reductions).

▶ **Definition 4.** *A* Turing reduction *from one preprocessing problem* X *to another one* Y *is a pair of algorithms* (R_P, R_Q) *satisfying the following properties:* R_P *is a (potentially randomized) function with access to an oracle* P*, whose output length is polynomial in its input length;* R_Q *is an algorithm with access to an oracle* Q*; and if* (P, Q) *solves problem* Y*, then* (R_P^P, R_Q^Q) *solves problem* X*. Additionally, it is a* Cook reduction *if* R_Q *runs in time polynomial in the total input length of* R_P *and* R_Q*.*

Similarly, the following generalizes mapping reductions and Karp reductions (i.e., polynomial-time mapping reductions) for decision problems.

▶ **Definition 5.** *A* mapping reduction *from one preprocessing decision problem* X *to another one* Y *is a pair* (R_P, R_Q) *satisfying the following properties:* R_P *is a deterministic function whose output length is polynomial in its input length;* R_Q *is a deterministic algorithm; and for any YES (respectively, NO) instance* (x_P, x_Q) *of* X*, the output pair* (y_P, y_Q) *is a YES (resp., NO) instance of* Y*, where* (y_P, y_Q) *are defined as follows:*

- *first,* R_P *takes* x_P *as input and outputs some* (σ', y_P)*, where* σ' *is some "internal" preprocessed output;*
- *then,* R_Q *takes* (σ', x_Q) *as input and outputs some* y_Q*.*

Additionally, it is a Karp reduction *if* R_Q *runs in time polynomial in the total input length of* R_P *and* R_Q*.*

It is straightforward to see that if X mapping reduces to Y, and there is a deterministic polynomial-time preprocessing algorithm (P_Y, Q_Y) that solves Y, then there is also one (P_X, Q_X) that solves X, which works as follows:

1. the preprocessing algorithm P_X, given a preprocessing input x_p, first computes $(\sigma', y_P) = R_P(x_P)$, then computes $\sigma_Y = P_Y(y_P)$ and outputs $\sigma_X = (\sigma', \sigma_Y)$;
2. the query algorithm Q_X, given $\sigma_X = (\sigma', \sigma_Y)$ and a query input x_Q, computes $y_Q = R_Q(\sigma', x_Q)$ and finally outputs $Q_Y(\sigma_Y, y_Q)$.

2.2 Lattices

A *lattice* is the set of all integer linear combinations of some linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$. It is convenient to arrange these vectors as the columns of a matrix. Accordingly, we define a *basis* $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{d \times n}$ to be a matrix with linearly independent columns, and the lattice generated by basis B as

$$\mathcal{L}(B) := \left\{ \sum_{i=1}^{n} a_i \boldsymbol{b}_i : a_1, \ldots, a_n \in \mathbb{Z} \right\} .$$

Let $\mathcal{B}_p^d$ denote the centered unit ℓ_p ball in d dimensions. Given a lattice $\mathcal{L} \subset \mathbb{R}^d$ of rank n, for $1 \le i \le n$ let

$$\lambda_i^{(p)}(\mathcal{L}) := \inf\{r > 0 : \dim(\mathrm{span}(r \cdot \mathcal{B}_p^d \cap \mathcal{L})) \ge i\}$$

denote the ith successive minimum of $\mathcal{L}$ with respect to the ℓ_p norm.

We denote the ℓ_p distance of a vector $\boldsymbol{t}$ to a lattice $\mathcal{L}$ as

$$\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L}} \|\boldsymbol{v} - \boldsymbol{t}\|_p .$$

2.3 Bounded Distance Decoding (with Preprocessing)

The primary computational problem that we study in this work is the Bounded Distance Decoding Problem (BDD), which is a version of the Closest Vector Problem (CVP) in which the target vector is promised to be relatively close to the lattice.

▶ **Definition 6.** *For $1 \le p \le \infty$ and $\alpha = \alpha(n) > 0$, the* α-Bounded Distance Decoding problem in the ℓ_p norm *($\mathrm{BDD}_{p,\alpha}$) is the (search) promise problem defined as follows. The input is (a basis of) a rank-n lattice $\mathcal{L}$ and a target vector $\boldsymbol{t}$ satisfying $\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L}) \le \alpha(n) \cdot \lambda_1^{(p)}(\mathcal{L})$. The goal is to output a lattice vector $\boldsymbol{v} \in \mathcal{L}$ that satisfies $\|\boldsymbol{v} - \boldsymbol{t}\|_p \le \alpha(n) \cdot \lambda_1^{(p)}(\mathcal{L})$.*

The preprocessing (search) promise problem $\mathrm{BDDP}_{p,\alpha}$ is defined analogously, where the preprocessing input is (a basis of) the lattice, and the query input is the target $\boldsymbol{t}$.

We note that in some works, BDD is defined to have the goal of finding a $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p = \mathrm{dist}_p(\boldsymbol{t}, \mathcal{L})$. This formulation is clearly no easier than the one defined above. So, our hardness theorems, which are proved for the definition above, immediately apply to the alternative formulation as well.

We also remark that for $\alpha < 1/2$, the promise ensures that there is a *unique* vector $\boldsymbol{v}$ satisfying $\|\boldsymbol{v} - \boldsymbol{t}\|_p \le \alpha \cdot \lambda_1^{(p)}(\mathcal{L})$. However, BDD is still well defined for $\alpha \ge 1/2$, i.e., above the unique-decoding radius. As in prior work, our hardness results for $\mathrm{BDD}_{p,\alpha}$ are limited to this regime.

To the best of our knowledge, essentially the only previous study of the NP-hardness of BDD is due to [19], which showed the following result.[3]

▶ **Theorem 7** ([19, Corollaries 1 and 2])**.** *For any $p \in [1, \infty)$ and $\alpha > 1/2^{1/p}$, there is no polynomial-time algorithm for $\mathrm{BDD}_{p,\alpha}$ (respectively, with preprocessing) unless $\mathsf{NP} \subseteq \mathsf{RP}$ (resp., unless $\mathsf{NP} \subseteq \mathsf{P/Poly}$).*

[3] Additionally, [11] gave a reduction from CVP to $\mathrm{BDD}_{2,\alpha}$ but only for some $\alpha > 1$. Also, [26, 20] gave a reduction from GapSVP_γ to BDD, but only for large $\gamma = \gamma(n)$ for which GapSVP is not known to be NP-hard.

Regev and Rosen [28] used norm embeddings to show that almost any lattice problem is at least as hard in the ℓ_p norm, for any $p \in [1, \infty]$, as it is in the ℓ_2 norm, up to an arbitrarily small constant-factor loss in the approximation factor. In other words, they essentially showed that ℓ_2 is the "easiest" ℓ_p norm for lattice problems. (In addition, their reduction preserves the rank of the lattice.) Based on this, [19] observed the following corollary, which is an improvement on the factor α from Theorem 7 for all $p > 2$.

▶ **Theorem 8** ([19, Corollary 3])**.** *For any $p \in [1, \infty)$ and $\alpha > 1/\sqrt{2}$, there is no polynomial-time algorithm for $\mathrm{BDD}_{p,\alpha}$ (respectively, with preprocessing) unless $\mathsf{NP} \subseteq \mathsf{RP}$ (resp., unless $\mathsf{NP} \subseteq \mathsf{P/Poly}$).*

Figure 1 shows the bounds from Theorems 7 and 8 together with the new bounds achieved in this work as a function of p.

2.4 Sparsification

A powerful idea, first used in the context of hardness proofs for lattice problems in [17], is that of random lattice sparsification. Given a lattice $\mathcal{L}$ with basis B, we can construct a random sublattice $\mathcal{L}' \subseteq \mathcal{L}$ as

$$\mathcal{L}' = \{\boldsymbol{v} \in \mathcal{L} : \langle \boldsymbol{z}, B^+\boldsymbol{v} \rangle = 0 \pmod q\}$$

for uniformly random $\boldsymbol{z} \in \mathbb{Z}_q^n$, where q is a suitably chosen prime.

▶ **Lemma 9.** *Let q be a prime and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\}$ be arbitrary. Then*

$$\Pr_{\boldsymbol{z} \leftarrow \mathbb{Z}_q^n}[\exists\, i \in [N] \text{ such that } \langle \boldsymbol{z}, \boldsymbol{x}_i \rangle = 0 \pmod q] \le \frac{N}{q}\ .$$

Proof. We have $\Pr[\langle \boldsymbol{z}, \boldsymbol{x}_i \rangle = 0] = 1/q$ for each $\boldsymbol{x}_i$, and the claim follows by the union bound. ◀

The following corollary is immediate.

▶ **Corollary 10.** *Let q be a prime and $\mathcal{L}$ be a lattice of rank n with basis B. Then for all $r > 0$ and all $p \in [1, \infty]$,*

$$\Pr_{\boldsymbol{z} \leftarrow \mathbb{Z}_q^n}[\lambda_1^{(p)}(\mathcal{L}') < r] \le \frac{N_p^o(\mathcal{L} \setminus \{\boldsymbol{0}\}, r, \boldsymbol{0})}{q}\ ,$$

where $\mathcal{L}' = \{\boldsymbol{v} \in \mathcal{L} : \langle \boldsymbol{z}, B^+\boldsymbol{v} \rangle = 0 \pmod q\}$.

▶ **Theorem 11** ([29, Theorem 3.1])**.** *For any lattice $\mathcal{L}$ of rank n with basis B, prime q, and lattice vectors $\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \in \mathcal{L}$ such that $B^+\boldsymbol{x} \neq B^+\boldsymbol{y}_i \pmod q$ for all $i \in [N]$, we have*

$$\frac{1}{q} - \frac{N}{q^2} - \frac{N}{q^{n-1}} \le \Pr_{\boldsymbol{z}, \boldsymbol{c} \leftarrow \mathbb{Z}_q^n}[\langle \boldsymbol{z}, B^+\boldsymbol{x} + \boldsymbol{c} \rangle = 0 \pmod q \wedge \langle \boldsymbol{z}, B^+\boldsymbol{y}_i + \boldsymbol{c} \rangle \neq 0 \pmod q\ \forall\, i \in [N]] \le \frac{1}{q} + \frac{1}{q^n}\ .$$

We will use only the lower bound from Theorem 11, but we note that the upper bound is relatively tight for $q \gg N$.

▶ **Corollary 12.** *For any $p \in [1, \infty]$ and $r \ge 0$, lattice $\mathcal{L}$ of rank n with basis B, vector $\boldsymbol{t}$, prime q, and lattice vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N \in \mathcal{L}$ such that $\|\boldsymbol{v}_i - \boldsymbol{t}\|_p \le r$ for all $i \in [N]$ and such that all the $B^+\boldsymbol{v}_i \bmod q$ are distinct, we have*

$$\Pr_{\boldsymbol{z}, \boldsymbol{c} \leftarrow \mathbb{Z}_q^n}[\mathrm{dist}_p(\boldsymbol{t} + B\boldsymbol{c}, \mathcal{L}') \le r] \ge \frac{N}{q} - \frac{N(N-1)}{q^2} - \frac{N(N-1)}{q^{n-1}}\ ,$$

where $\mathcal{L}' = \{\boldsymbol{v} \in \mathcal{L} : \langle \boldsymbol{z}, B^+\boldsymbol{v} \rangle = 0 \pmod q\}$.

Proof. Observe that for each $i \in [N]$, the events

$$E_i := [\langle \boldsymbol{z}, B^+\boldsymbol{v}_i\rangle = 0 \pmod q \text{ and } \langle \boldsymbol{z}, B^+\boldsymbol{v}_j\rangle \neq 0 \pmod q \text{ for all } j \neq i]$$

are disjoint, and by invoking Theorem 11 with $\boldsymbol{x} = \boldsymbol{v}_i$ and the $\boldsymbol{y}_j$ being the remaining $\boldsymbol{v}_k$ for $k \neq i$, we have

$$\Pr_{\boldsymbol{z},\boldsymbol{c}}[E_i] \geq \frac{1}{q} - \frac{N-1}{q^2} - \frac{N-1}{q^{n-1}} \ .$$

Also observe that if E_i occurs, then $\boldsymbol{v}_i + B\boldsymbol{c} \in \mathcal{L}'$ (also $\boldsymbol{v}_j + B\boldsymbol{c} \notin \mathcal{L}'$ for all $j \neq i$, but we will not need this). Therefore,

$$\mathrm{dist}_p(\boldsymbol{t} + B\boldsymbol{c}, \mathcal{L}') \leq \|\boldsymbol{t} + B\boldsymbol{c} - (\boldsymbol{v}_i + B\boldsymbol{c})\| = \|\boldsymbol{t} - \boldsymbol{v}_i\| \leq r \ .$$

So, the probability in the left-hand side of the claim is at least

$$\Pr_{\boldsymbol{z},\boldsymbol{c}}\Big[\bigcup_{i\in[N]} E_i\Big] = \sum_{i\in[N]} \Pr_{\boldsymbol{z},\boldsymbol{c}}[E_i] \geq \frac{N}{q} - \frac{N(N-1)}{q^2} - \frac{N(N-1)}{q^{n-1}} \ . \qquad \blacktriangleleft$$

2.5 Counting Lattice Points in a Ball

Following [5], for any discrete set A of points (e.g., a lattice, or a subset thereof), we denote the number of points in A contained in the closed and open (respectively) ℓ_p ball of radius r centered at a point $\boldsymbol{t}$ as

$$N_p(A, r, \boldsymbol{t}) := |\{\boldsymbol{y} \in A : \|\boldsymbol{y} - \boldsymbol{t}\|_p \leq r\}| \ , \tag{2.1}$$

$$N_p^o(A, r, \boldsymbol{t}) := |\{\boldsymbol{y} \in A : \|\boldsymbol{y} - \boldsymbol{t}\|_p < r\}| \ . \tag{2.2}$$

Clearly, $N_p^o(A, r, \boldsymbol{t}) \leq N_p(A, r, \boldsymbol{t})$.

For $1 \leq p < \infty$ and $\tau > 0$ define

$$\Theta_p(\tau) := \sum_{z\in\mathbb{Z}} \exp(-\tau|z|^p) \ .$$

We use the following upper bound due to Mazo and Odlyzko [21] on the number of short vectors in the integer lattice. We include its short proof for completeness.

▶ **Proposition 13** ([21]). *For any $p \in [1, \infty)$, $r > 0$, and $n \in \mathbb{N}$,*

$$N_p(\mathbb{Z}^n, r, \boldsymbol{0}) \leq \min_{\tau>0} \exp(\tau r^p) \cdot \Theta_p(\tau)^n \ .$$

Proof. For $\tau > 0$ we have

$$\Theta_p(\tau)^n = \sum_{\boldsymbol{z}\in\mathbb{Z}^n} \exp(-\tau\|\boldsymbol{z}\|_p^p) \geq \sum_{\boldsymbol{z}\in\mathbb{Z}^n \cap r\mathcal{B}_p^n} \exp(-\tau\|\boldsymbol{z}\|_p^p) \geq \exp(-\tau r^p) \cdot N_p(\mathbb{Z}^n, r, \boldsymbol{0}) \ .$$

The result follows by rearranging and taking the minimum over all $\tau > 0$. $\blacktriangleleft$

2.6 Hardness Assumptions

We recall the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [15], and several of its variants. These hypotheses make stronger assumptions about the complexity of the k-SAT problem than the assumption $\mathsf{P} \neq \mathsf{NP}$, and serve as highly useful tools for studying the fine-grained complexity of hard computational problems. Indeed, we will show that strong fine-grained hardness for BDD follows from these hypotheses.

▶ **Definition 14.** *The* (randomized) Exponential Time Hypothesis *((randomized) ETH) asserts that there is no (randomized)* $2^{o(n)}$*-time algorithm for* 3*-SAT on* n *variables.*

▶ **Definition 15.** *The* (randomized) Strong Exponential Time Hypothesis *((randomized) SETH) asserts that for every* $\varepsilon > 0$ *there exists* $k \in \mathbb{Z}^+$ *such that there is no (randomized)* $2^{(1-\varepsilon)n}$*-time algorithm for* k*-SAT on* n *variables.*

For proving hardness of lattice problem with preprocessing, we define (Max-)k-SAT with preprocessing as follows. The preprocessing input is a size parameter n, encoded in unary. The query input is a k-SAT formula ϕ with n variables and m (distinct) clauses, together with a threshold $W \in \{0, \dots m\}$ in the case of Max-k-SAT. For k-SAT, it is a YES instance if ϕ is satisfiable, and is a NO instance otherwise. For Max-k-SAT, it is a YES instance if there exists an assignment to the variables of ϕ that satisfies at least W of its clauses, and is a NO instance otherwise.

Observe that because the preprocessing input is just n, a preprocessing algorithm for (Max-)k-SAT with preprocessing is equivalent to a (non-uniform) family of circuits for the problem *without* preprocessing. Also, for any fixed k, because there are only $O(n^k)$ possible clauses on n variables, the length of the query input for (Max-)k-SAT instances having preprocessing input n is $\mathrm{poly}(n)$, so we get the following corollary of Lemma 3.

▶ **Corollary 16.** *If (Max-)*k*-SAT with preprocessing has a randomized* $T(n)$*-time algorithm, then it has a deterministic* $T(n) \cdot \mathrm{poly}(n)$*-time algorithm using* $T(n) \cdot \mathrm{poly}(n)$*-length preprocessed output.*

Following, e.g., [30, 2], we also define *non-uniform* variants of ETH and SETH, which deal with the complexity of k-SAT with preprocessing. More precisely, non-uniform ETH asserts that no family of size-$2^{o(n)}$ circuits solves 3-SAT on n variables (equivalently, 3-SAT with preprocessing does not have a $2^{o(n)}$-time algorithm), and non-uniform SETH asserts that for every $\varepsilon > 0$ there exists $k \in \mathbb{Z}^+$ such that no family of circuits of size $2^{(1-\varepsilon)n}$ solves k-SAT on n variables (equivalently, k-SAT with preprocessing does not have a $2^{(1-\varepsilon)n}$-time algorithm). These hypotheses are useful for analyzing the fine-grained complexity of preprocessing problems.

One might additionally consider "randomized non-uniform" versions of (S)ETH. However, Corollary 16 says that a randomized algorithm for (Max-)k-SAT with preprocessing can be derandomized with only polynomial overhead, so randomized non-uniform (S)ETH is equivalent to (deterministic) non-uniform (S)ETH, so we only consider the latter.

Finally, we remark that one can define weaker versions of randomized or non-uniform (S)ETH with Max-3-SAT (respectively, Max-k-SAT) in place of 3-SAT (resp., k-SAT). Many of our results hold even under these weaker hypotheses. In particular, the derandomization result in Corollary 16 applies to both k-SAT and Max-k-SAT.

3 Hardness of $\mathrm{BDD}_{p,\alpha}$

In this section, we present our main result by giving a reduction from a known-hard variant GapCVP'_p of the Closest Vector Problem (CVP) to BDD. We peform this reduction in two main steps.

1. First, in Section 3.1 we define a variant of $\mathrm{BDD}_{p,\alpha}$, which we call (S,T)-$\mathrm{BDD}_{p,\alpha}$. Essentially, an instance of this problem is a lattice that may have up to S "short" nonzero vectors of ℓ_p norm bounded by some r, and a target vector that is "close" to – i.e., within distance αr of – at least T lattice vectors. (The presence of short vectors prevents this from being a true $\mathrm{BDD}_{p,\alpha}$ instance.) We then give a reduction, for $S \ll T$, from (S,T)-$\mathrm{BDD}_{p,\alpha}$ to $\mathrm{BDD}_{p,\alpha}$ itself, using sparsification.

2. Then, in Section 3.2 we reduce from GapCVP$'_p$ to (S,T)-BDD$_{p,\alpha}$ for suitable $S \ll T$ whenever α is sufficiently large as a function of p (and the desired rank efficiency), based on analysis given in Section 3.3 and Lemma 27.

3.1 (S,T)-BDD to BDD

We start by defining a special decision variant of BDD. Essentially, the input is a lattice and a target vector, and the problem is to distinguish between the case where there are few "short" lattice vectors but many lattice vectors "close" to the target, and the case where the target is not close to the lattice. There is a gap factor between the "close" and "short" distances, and for technical reasons we count only those "close" vectors having binary coefficients with respect to the given input basis.

▶ **Definition 17.** *Let $S = S(n), T = T(n) \geq 0$, $p \in [1,\infty]$, and $\alpha = \alpha(n) > 0$. An instance of the decision promise problem (S,T)-BDD$_{p,\alpha}$ is a lattice basis $B \in \mathbb{R}^{d\times n}$, a distance $r > 0$, and a target $\boldsymbol{t} \in \mathbb{R}^d$.*

- *It is a YES instance if $N_p^o(\mathcal{L}(B) \setminus \{\mathbf{0}\}, r, \mathbf{0}) \leq S(n)$ and $N_p(B \cdot \{0,1\}^n, \alpha r, \boldsymbol{t}) \geq T(n)$.*
- *It is a NO instance if $\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L}(B)) > \alpha r$.*

The search version is: given a YES instance $(B, r, \boldsymbol{t})$, find a $\boldsymbol{v} \in \mathcal{L}(B)$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \alpha r$.

The preprocessing search and decision problems (S,T)-BDDP$_{p,\alpha}$ are defined analogously, where the preprocessing input is B and r, and the query input is $\boldsymbol{t}$.

We stress that in the preprocessing problems BDDP, the distance r is part of the preprocessing input; this makes the problem no harder than a variant where r is part of the query input. So, our hardness results for the above definition immediately apply to that variant as well. However, our reduction *from* (S,T)-BDDP (given in Lemma 18) critically relies on the fact that r is part of the preprocessing input.

Clearly, there is a trivial reduction from the decision version of (S,T)-BDD$_{p,\alpha}$ to its search version (and similarly for the preprocessing problems): just call the oracle for the search problem and test whether it returns a lattice vector within distance αr of the target. So, to obtain more general results, our reductions involving (S,T)-BDD will be *from* the search version, and *to* the decision version.

Reducing to BDD

We next observe that for $S(n) = 0$ and any $T(n) > 0$, there is *almost* a trivial reduction from (S,T)-BDD$_{p,\alpha}$ to ordinary BDD$_{p,\alpha}$, because YES instances of the former satisfy the BDD$_{p,\alpha}$ promise. (See below for the easy proof.) The only subtlety is that we want the BDD$_{p,\alpha}$ oracle to return a lattice vector that is within distance αr of the target; recall that the definition of BDD$_{p,\alpha}$ only guarantees distance $\alpha \cdot \lambda_1^{(p)}(\mathcal{L}(B))$. This issue is easily resolved by modifying the lattice to *upper bound* its minimum distance by r, which increases the lattice's rank by one. (For the alternative definition of BDD described after Definition 6, the trivial reduction works, and no increase in the rank is needed.)

▶ **Lemma 18.** *For any $T(n) > 0$, $p \in [1,\infty]$, and $\alpha = \alpha(n) > 0$, there is a deterministic Cook reduction from the search version of $(0, T(n))$-BDD$_{p,\alpha}$ (resp., with preprocessing) in rank n to BDD$_{p,\alpha}$ (resp., with preprocessing) in rank $n+1$.*

Proof. The reduction works as follows. On input $(B, r, \boldsymbol{t})$, call the $\mathrm{BDD}_{p,\alpha}$ oracle on

$$B' := \begin{pmatrix} B & 0 \\ 0 & r \end{pmatrix}, \quad \boldsymbol{t}' := \begin{pmatrix} \boldsymbol{t} \\ 0 \end{pmatrix} ,$$

and (without loss of generality) receive from the oracle a vector $\boldsymbol{v}' = (\boldsymbol{v}, zr)$ for some $\boldsymbol{v} \in \mathcal{L}$ and $z \in \mathbb{Z}$. Output $\boldsymbol{v}$.

We analyze the reduction. Let $\mathcal{L} = \mathcal{L}(B)$ and $\mathcal{L}' = \mathcal{L}(B')$. Because the input is a YES instance, we have $N_p^o(\mathcal{L} \setminus \{\mathbf{0}\}, r, \mathbf{0}) = 0$ and hence $\lambda_1^{(p)}(\mathcal{L}) \geq r$, so $\lambda_1^{(p)}(\mathcal{L}') = r$. Moreover, $N_p(B \cdot \{0,1\}^n, \alpha r, \boldsymbol{t}) > 0$ implies that $\mathrm{dist}_p(\boldsymbol{t}', \mathcal{L}') = \mathrm{dist}(\boldsymbol{t}, \mathcal{L}) \leq \alpha r = \alpha \cdot \lambda_1^{(p)}(\mathcal{L}')$. So, $(B', \boldsymbol{t}')$ satisfies the $\mathrm{BDD}_{p,\alpha}$ promise, hence the oracle is obligated to return some $\boldsymbol{v}' = (\boldsymbol{v}, zr) \in \mathcal{L}'$ where $\boldsymbol{v} \in \mathcal{L}$ and $\alpha r = \alpha \lambda_1^{(p)}(\mathcal{L}') \geq \|\boldsymbol{v}' - \boldsymbol{t}'\|_p \geq \|\boldsymbol{v} - \boldsymbol{t}\|_p$. Therefore, the output $\boldsymbol{v}$ of the reduction is a valid solution.

Finally, observe that all of the above also constitutes a valid reduction for the preprocessing problems, because B' depends only on the preprocessing part B, r of the input. ◀

We now present a more general randomized reduction from (S,T)-$\mathrm{BDD}_{p,\alpha}$ to $\mathrm{BDD}_{p,\alpha}$, which works whenever $T(n) \geq 10S(n)$. The essential idea is to sparsify the input lattice, so that with some noticeable probability no short vectors remain, but at least one vector close to the target does remain. In this case, the result will be an instance of $(0,1)$-$\mathrm{BDD}_{p,\alpha}$, which reduces to $\mathrm{BDD}_{p,\alpha}$ as shown above.

We note that the triangle inequality precludes the existence of (S,T)-$\mathrm{BDD}_{p,\alpha}$ instances with $T > S + 1$ and $\alpha \leq 1/2$, so with this approach we can only hope to show hardness of $\mathrm{BDD}_{p,\alpha}$ for $\alpha > 1/2$, i.e., the unique-decoding regime remains out of reach.

▶ **Theorem 19.** *For any $S = S(n) \geq 1$ and $T = T(n) \geq 10S$ that is efficiently computable (for unary n), $p \in [1, \infty]$, and $\alpha = \alpha(n) > 0$, there is a randomized Cook reduction with no false positives from the search version of (S,T)-$\mathrm{BDD}_{p,\alpha}$ (resp., with preprocessing) in rank n to $\mathrm{BDD}_{p,\alpha}$ (resp., with preprocessing) in rank $n + 1$.*

Proof. By Lemma 18, it suffices to give such a reduction to $(0,1)$-$\mathrm{BDD}_{p,\alpha}$ in rank n, which works as follows. On input $(B, r, \boldsymbol{t})$, let $\mathcal{L} = \mathcal{L}(B)$. First, randomly choose a prime q where $10T \leq q \leq 20T$. Then sample $\boldsymbol{z}, \boldsymbol{c} \in \mathbb{Z}_q^n$ independently and uniformly at random, and define

$$\mathcal{L}' := \{\boldsymbol{v} \in \mathcal{L} : \langle \boldsymbol{z}, B^+ \boldsymbol{v} \rangle = 0 \pmod q\} \text{ and } \boldsymbol{t}' := \boldsymbol{t} + B\boldsymbol{c} .$$

Let B' be a basis of $\mathcal{L}'$. (Such a basis is efficiently computable from B, $\boldsymbol{z}$, and q. See, e.g., [29, Claim 2.15].) Invoke the $(0,1)$-$\mathrm{BDD}_{p,\alpha}$ oracle on $(B', r, \boldsymbol{t}')$, and output whatever the oracle outputs.

We now analyze the reduction. We are promised that $(B, r, \boldsymbol{t})$ is a YES instance of (S,T)-$\mathrm{BDD}_{p,\alpha}$, and it suffices to show that $(B', r, \boldsymbol{t}')$ is a YES instance of $(0,1)$-$\mathrm{BDD}_{p,\alpha}$, i.e., $\lambda_1^{(p)}(\mathcal{L}') \geq r$ and $\mathrm{dist}_p(\boldsymbol{t}', \mathcal{L}') \leq \alpha r$, with some positive constant probability. By Corollary 10 we have

$$\Pr[\lambda_1^{(p)}(\mathcal{L}') < r] \leq \frac{N_p^o(\mathcal{L} \setminus \{\mathbf{0}\}, r, \mathbf{0})}{q} \leq \frac{S}{q} \leq \frac{1}{100} .$$

Furthermore, because there are T vectors $\boldsymbol{v}_i \in \mathcal{L}$ for which $\|\boldsymbol{v}_i - \boldsymbol{t}\|_p \leq \alpha r$, and their coefficient vectors $B^+ \boldsymbol{v}_i \in \{0,1\}^n$ are distinct (as integer vectors, and hence also modulo q), by Corollary 12 we have

$$\Pr[\mathrm{dist}_p(\boldsymbol{t}', \mathcal{L}') \leq \alpha r] \geq \frac{T}{q} - \frac{T^2}{q^2} - \frac{T^2}{q^{n-1}} \geq \frac{1}{20} - \frac{1}{400} - \frac{1}{400 q^{n-3}} .$$

Therefore, by the union bound we have

$$\Pr[\lambda_1^{(p)}(\mathcal{L}') \geq r \text{ and } \operatorname{dist}_p(\boldsymbol{t}', \mathcal{L}') \leq \alpha r] \geq \frac{1}{20} - \frac{1}{400} - \frac{1}{400q^{n-3}} - \frac{1}{100} \geq \frac{1}{40}$$

for all $n \geq 3$, as desired.

Finally, the above also constitutes a valid reduction for the preprocessing problems (in the sense of Definition 4), because B' depends only on B from the preprocessing part of the input and the reduction's own random choices (and r remains unchanged). ◀

3.2 GapCVP' to (S, T)-BDD

Here we show that a known-hard variant of the (exact) Closest Vector Problem reduces to (S, T)-BDD (in its decision version).

▶ **Definition 20.** *For $p \in [1, \infty]$, the (decision) promise problem* GapCVP'_p *is defined as follows: an instance consists of a basis $B \in \mathbb{R}^{d \times n}$ and a target vector $\boldsymbol{t} \in \mathbb{R}^d$.*

- *It is a YES instance if there exists $\boldsymbol{x} \in \{0,1\}^n$ such that $\|B\boldsymbol{x} - \boldsymbol{t}\|_p \leq 1$.*
- *It is a NO instance if $\operatorname{dist}_p(\boldsymbol{t}, \mathcal{L}(B)) > 1$.*

The preprocessing (decision) promise problem $\mathrm{GapCVPP}'_p$ *is defined analogously, where the preprocessing input is B and the query input is $\boldsymbol{t}$.*

Observe that for GapCVP'_p the distance threshold is 1 (and not some instance-dependent value) without loss of generality, because we can scale the lattice and target vector. The same goes for $\mathrm{GapCVPP}'_p$, with the caveat that any instance-dependent distance threshold would need to be included in the *preprocessing* part of the input, not the query part. (See Remark 26 below for why this is essentially without loss of generality, under a mild assumption on the $\mathrm{GapCVPP}'_p$ instances.) We remark that some works define these problems with a stronger requirement that in the NO case, $\operatorname{dist}_p(z\boldsymbol{t}, \mathcal{L}(B)) > r$ for all $z \in \mathbb{Z} \setminus \{0\}$. We will not need this stronger requirement, and some of the hardness results for GapCVP' that we rely on are not known to hold with it, so we use the weaker requirement.

We next describe a simple transformation on lattices and target vectors: we essentially take a direct sum of the input lattice with the integer lattice of any desired dimension n and append an all-$\frac{1}{2}$s vector to the target vector.

▶ **Lemma 21.** *For any $n' \leq n$, define the following transformations that map a basis B' of a rank-n' lattice $\mathcal{L}'$ to a basis B of a rank-n lattice $\mathcal{L}$, and a target vector $\boldsymbol{t}'$ to a target vector $\boldsymbol{t}$:*

$$B := \begin{pmatrix} \frac{1}{2}B' & 0 \\ I_{n'} & 0 \\ 0 & I_{n-n'} \end{pmatrix}, \qquad \boldsymbol{t} := \frac{1}{2}\begin{pmatrix} \boldsymbol{t}' \\ \mathbf{1}_{n'} \\ \mathbf{1}_{n-n'} \end{pmatrix}, \tag{3.1}$$

and define

$$s_p = s_p(n) := \tfrac{1}{2}(n+1)^{1/p} \text{ for } p \in [1, \infty), \text{ and } s_\infty := 1/2. \tag{3.2}$$

Then:

1. $N_p^o(\mathcal{L}, r, \mathbf{0}) \leq N_p^o(\mathbb{Z}^n, r, \mathbf{0})$ *for all $r \geq 0$;*
2. *if there exists an $\boldsymbol{x} \in \{0,1\}^{n'}$ such that $\|B'\boldsymbol{x} - \boldsymbol{t}'\|_p \leq 1$, then $N_p(B \cdot \{0,1\}^n, s_p, \boldsymbol{t}) \geq 2^{n-n'}$;*
3. *if $\operatorname{dist}_p(\boldsymbol{t}', \mathcal{L}') > 1$ then $\operatorname{dist}_p(\boldsymbol{t}, \mathcal{L}) > s_p$.*

Proof. Item 1 follows immediately by construction of B, because vectors $\boldsymbol{v}' = (\frac{1}{2}B'\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}) \in \mathcal{L}$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^n$ correspond bijectively to vectors $\boldsymbol{v} = (\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{Z}^n$, and $\|\boldsymbol{v}\|_p \le \|\boldsymbol{v}'\|_p$.

For Item 2, for every $\boldsymbol{y} \in \{0,1\}^{n-n'}$, the vector $\boldsymbol{v} := (\frac{1}{2}B'\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{y}) \in \mathcal{L}$ satisfies

$$\|\boldsymbol{v} - \boldsymbol{t}\|_p^p = \frac{\|B'\boldsymbol{x} - \boldsymbol{t}'\|_p^p}{2^p} + \frac{n}{2^p} \le s_p^p$$

for finite p, and $\|\boldsymbol{v} - \boldsymbol{t}\|_\infty = \max(\frac{1}{2}\|B'\boldsymbol{x} - \boldsymbol{t}'\|_\infty, \frac{1}{2}) = \frac{1}{2} = s_\infty$. The claim follows.

For Item 3, for finite p we have

$$\mathrm{dist}_p(\boldsymbol{t}, \mathcal{L})^p \ge \frac{\mathrm{dist}_p(\boldsymbol{t}', \mathcal{L}')^p}{2^p} + \frac{n}{2^p} > \frac{n+1}{2^p} = s_p^p \ ,$$

and for $p = \infty$ we immediately have $\mathrm{dist}_\infty(\boldsymbol{t}, \mathcal{L}) \ge \frac{1}{2}\,\mathrm{dist}_\infty(\boldsymbol{t}', \mathcal{L}') > \frac{1}{2} = s_\infty$, as needed. ◀

▶ **Corollary 22.** *For any $p \in [1, \infty]$, $\alpha > 0$, and $\mathrm{poly}(n')$-bounded $n \ge n'$, there is a deterministic Karp reduction from GapCVP'_p (resp., with preprocessing) in rank n' to the decision version of (S,T)-$\mathrm{BDD}_{p,\alpha}$ (resp., with preprocessing) in rank n, where $S(n) = N_p^o(\mathbb{Z}^n \setminus \{\mathbf{0}\}, s_p/\alpha, \mathbf{0})$ for s_p as defined in Equation (3.2), and $T(n) = 2^{n-n'}$.*

Proof. Given an input GapCVP'_p instance $(B', \boldsymbol{t}')$, the reduction simply outputs $(B, r = s_p/\alpha, \boldsymbol{t})$, where $B, \boldsymbol{t}$ are as in Equation (3.1). Observe that this is also valid for the preprocessing problems because B and r depend only on B'. Correctness follows immediately by Lemma 21. ◀

3.3 Setting Parameters

We now investigate the relationship among the choice of ℓ_p norm (for finite p), the BDD relative distance α, and the rank ratio $C := n/n'$, subject to the constraint

$$N_p^o(\mathbb{Z}^n, s_p/\alpha, \mathbf{0}) \le 2^{n-n'}/10 = T(n)/10 \ , \tag{3.3}$$

so that the reductions in Corollary 22 and Theorem 19 can be composed. For $p \in [1, \infty)$ and $C > 1$, define

$$\alpha^*_{p,C} := \inf\{\alpha^* > 0 : \min_{\tau > 0} \exp(\tau/(2\alpha^*)^p) \cdot \Theta_p(\tau) \le 2^{1-1/C}\} \ , \tag{3.4}$$

$$\alpha^*_p := \lim_{C \to \infty} \alpha^*_{p,C} = \inf\{\alpha^* > 0 : \min_{\tau > 0} \exp(\tau/(2\alpha^*)^p) \cdot \Theta_p(\tau) \le 2\} \ . \tag{3.5}$$

These quantities are well defined because for any $C > 1$ we have $2^{1-1/C} > 1$, so the inequality in Equation (3.4) is satisfied for sufficiently large τ and α^*. Moreover, it is straightforward to verify that $\alpha^*_{p,C}$ is strictly decreasing in both p and C, and α^*_p is strictly decreasing in p. Although it is not clear how to solve for these quantities in closed form, it is possible to approximate them numerically to good accuracy (see Figure 1), and to get quite tight closed-form upper bounds (see Lemma 27). We now show that to satisfy Equation (3.3) it suffices to take any constant $\alpha > \alpha^*_{p,C}$.

▶ **Corollary 23.** *For any $p \in [1, \infty)$, $C \ge 1$, and constant $\alpha > \alpha^*_{p,C}$ (Equation (3.4)), there is a deterministic Karp reduction from GapCVP'_p (resp., with preprocessing) in rank n' to the decision version of (S,T)-$\mathrm{BDD}_{p,\alpha}$ (resp., with preprocessing) in rank $n = Cn'$, where $S(n) = T(n)/10$ and $T(n) = 2^{(1-1/C)n}$.*

Proof. Recalling that $s_p = \frac{1}{2}(n+1)^{1/p}$, by Proposition 13, $N_p^o(\mathbb{Z}^n, s_p/\alpha, \mathbf{0})$ is at most

$$\begin{aligned} N_p(\mathbb{Z}^n, s_p/\alpha, \mathbf{0}) &\le \min_{\tau>0} \exp(\tau \cdot (s_p/\alpha)^p) \cdot \Theta_p(\tau)^n \\ &= \min_{\tau>0} \exp(\tau \cdot (n+1)/(2\alpha)^p) \cdot \Theta_p(\tau)^n \\ &= \left(\min_{\tau>0} \exp(\tau/(n(2\alpha)^p)) \cdot \exp(\tau/(2\alpha)^p) \cdot \Theta_p(\tau)\right)^n . \end{aligned}$$

Because $\alpha > \alpha^*_{p,C}$, we have that $\min_{\tau>0} \exp(\tau/(2\alpha)^p) \cdot \Theta_p(\tau)$ is a constant strictly less than $2^{1-1/C}$. So, $N_p^o(\mathbb{Z}^n, s_p/\alpha, \mathbf{0}) \le 2^{(1-1/C)n}/10 = T(n)/10$ for all large enough n. The claim follows from Corollary 22. ◀

▶ **Theorem 24.** *For any $p \in [1,\infty)$, $C \ge 1$, and constant $\alpha > \alpha^*_{p,C}$, there is a randomized Cook reduction with no false positives from* GapCVP'_p *(resp., with preprocessing) in rank n' to* $\mathrm{BDD}_{p,\alpha}$ *(resp., with preprocessing) in rank $n = Cn' + 1$. Furthermore, the same holds for $p = \infty$, $C = 1$, $\alpha = 1/2$, and the reduction is deterministic.*

Proof. For finite p, we simply compose the reductions from Corollary 23 and Theorem 19, with the trivial decision-to-search reduction for (S,T)-$\mathrm{BDD}_{p,\alpha}$ in between.

For $p = \infty$, we first invoke the deterministic reduction from Corollary 22, from GapCVP'_∞ in rank n' to (S,T)-$\mathrm{BDD}_{\infty,1/2}$ in rank $Cn' = n'$, where $S = N^o_\infty(\mathbb{Z}^n \setminus \{\mathbf{0}\}, 1, \mathbf{0}) = 0$ and $T = 2^0 > 0$. By Lemma 18, the latter problem reduces deterministically to $\mathrm{BDD}_{\infty,1/2}$ in rank $n' + 1$.

Lastly, all of these reductions work for the preprocessing problems as well, because their component reductions do. ◀

3.4 Putting it all Together

We now combine our reductions from GapCVP' to BDD with prior hardness results for GapCVP' (stated below in Theorem 25) to obtain our ultimate hardness theorems for BDD. We first recall relevant known hardness results for GapCVP'_p and $\mathrm{GapCVPP}'_p$.

▶ **Theorem 25** ([23, 10, 2])**.** *The following hold for* GapCVP'_p *and* $\mathrm{GapCVPP}'_p$ *in rank n:*

1. *For every $p \in [1,\infty]$,* GapCVP'_p *is* NP*-hard, and* $\mathrm{GapCVPP}'_p$ *has no polynomial-time (preprocessing) algorithm unless* $\mathsf{NP} \subseteq \mathsf{P/Poly}$.
2. *For every $p \in [1,\infty]$, there is no $2^{o(n)}$-time randomized algorithm for* GapCVP'_p *unless randomized ETH fails.*
3. *For every $p \in [1,\infty] \setminus \{2\}$, there is no $2^{o(n)}$-time algorithm for* $\mathrm{GapCVPP}'_p$*, and there is no $2^{o(\sqrt{n})}$-time algorithm for* $\mathrm{GapCVPP}'_2$*, unless non-uniform ETH fails.*
4. *For every $p \in [1,\infty] \setminus 2\mathbb{Z}$ and every $\varepsilon > 0$, there is no $2^{(1-\varepsilon)n}$-time randomized algorithm for* GapCVP'_p *(respectively,* $\mathrm{GapCVPP}'_p$*) unless randomized SETH (resp., non-uniform SETH) fails.*

▶ Remark 26. Several of the above results are stated slightly differently from what appears in [23, 10, 2]. First, all of the above results for GapCVP'_p (respectively, $\mathrm{GapCVPP}'_p$) are instead stated for GapCVP_p (resp., $\mathrm{GapCVPP}_p$). However, inspection shows that the reductions are indeed to GapCVP'_p or $\mathrm{GapCVPP}'_p$, so this difference is immaterial.

Second, the above statements ruling out *randomized* algorithms for GapCVP'_p assuming randomized (S)ETH are instead phrased in [10, 2] as ruling out *deterministic* algorithms for GapCVP'_p assuming deterministic (S)ETH. However, because these results are proved via deterministic reductions, randomized algorithms for GapCVP'_p have the consequences claimed above.

Third, the above results for GapCVPP$'_p$ follow from the reductions given in (the proofs of) [23], [2, Theorem 4.3], [10, Theorem 1.4 and Lemma 6.1], and [2, Theorem 4.6]. However, those reductions all prove hardness for the variant of GapCVPP$'_p$ where the distance threshold r is part of the *query* input, rather than the preprocessing input. Inspection of [2, Theorem 4.6] shows that r is fixed in the output instance, so this difference is immaterial in that case. We next describe how to handle this difference for the remaining cases. Below we give, for any $p \in [1, \infty)$, a straightforward rank-preserving mapping reduction (in the sense of Definition 5) from the variant of GapCVPP$'_p$ where the distance threshold r is part of the query input to the variant where it is part of the preprocessing input, assuming that r is always at most some r^* that depends only on B, and whose length $\log r^*$ is polynomial in the length of B. Inspection shows that such an r^* does indeed exist for the reductions given in [23], [2, Theorem 4.3], and [10, Lemma 6.1], which handles the second difference for those cases.

The mapping reduction (R_P, R_Q) in question maps $(B, (\boldsymbol{t}, r)) \mapsto ((B', r^*), \boldsymbol{t}')$ as follows. First, R_P takes B as input, and sets $B' := \left(\begin{smallmatrix} B \\ \mathbf{0}^t \end{smallmatrix}\right)$; it also outputs $\sigma' = r^*$ as side information for R_Q. Then, R_Q takes $(\boldsymbol{t}, r)$ and r^* as input, and outputs $\boldsymbol{t}' := (\boldsymbol{t}, ((r^*)^p - r^p)^{1/p})$. Using the guarantee that $r^* \geq r$, it is straightforward to check that the output instance $((B', r^*), \boldsymbol{t}')$ is a YES instance (respectively, NO instance) if the input instance $(B, (\boldsymbol{t}, r))$ is a YES instance resp., NO instance, as required.

Finally, we again remark that several of the hardness results in Theorem 25 in fact hold under weaker versions of randomized or non-uniform (S)ETH that relate to Max-3-SAT (respectively, Max-k-SAT), instead of 3-SAT (resp. k-SAT). Therefore, it is straightforward to obtain corresponding hardness results for BDD(P) under these weaker assumptions as well.

We can now prove our main theorem, restated from the introduction:

▶ **Theorem 1.** *The following hold for* $\mathrm{BDD}_{p,\alpha}$ *and* $\mathrm{BDDP}_{p,\alpha}$ *in rank* n*:*

1. *For every* $p \in [1, \infty)$ *and constant* $\alpha > \alpha_p^*$ *(where* $\alpha_p^* \leq \frac{1}{2} \cdot 4.6723^{1/p}$*), and for* $(p, \alpha) = (\infty, 1/2)$*, there is no polynomial-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *(respectively,* $\mathrm{BDDP}_{p,\alpha}$*) unless* $\mathsf{NP} \subseteq \mathsf{RP}$ *(resp.,* $\mathsf{NP} \subseteq \mathsf{P/Poly}$*).*
2. *For every* $p \in [1, \infty)$ *and constant* $\alpha > \min\{\alpha_p^*, \alpha_2^*\}$*, and for* $(p, \alpha) = (\infty, 1/2)$*, there is no* $2^{o(n)}$*-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *unless randomized ETH fails.*
3. *For every* $p \in [1, \infty) \setminus \{2\}$ *and constant* $\alpha > \alpha_p^*$*, and for* $(p, \alpha) = (\infty, 1/2)$*, there is no* $2^{o(n)}$*-time algorithm for* $\mathrm{BDDP}_{p,\alpha}$ *unless non-uniform ETH fails.*

   Moreover, for every $p \in [1, \infty]$ *and* $\alpha > \alpha_2^*$ *there is no* $2^{o(\sqrt{n})}$*-time algorithm for* $\mathrm{BDDP}_{p,\alpha}$ *unless non-uniform ETH fails.*
4. *For every* $p \in [1, \infty) \setminus 2\mathbb{Z}$ *and constants* $C > 1$*,* $\alpha > \alpha_{p,C}^*$*, and* $\epsilon > 0$*, and for* $(p, C, \alpha) = (\infty, 1, 1/2)$*, there is no* $2^{n(1-\epsilon)/C}$*-time algorithm for* $\mathrm{BDD}_{p,\alpha}$ *(respectively,* $\mathrm{BDDP}_{p,\alpha}$*) unless randomized SETH (resp., non-uniform SETH) fails.*

Proof. For BDD, each item of the theorem follows from the corresponding item of Theorem 25, followed by Theorem 24 and then (where needed) rank-preserving norm embeddings from ℓ_2 to ℓ_p [28]. (Also, Lemma 27 below provides the upper bound on α_p^*.) The claims for BDDP follow similarly, combined with the well-known fact that $\mathsf{P/Poly} = \mathsf{BPP/Poly}$ and Corollary 16.[4] ◀

[4] In fact, $\mathsf{P/Poly} = \mathsf{BPP/Poly}$ also follows as a corollary of the more general derandomization result in Lemma 3.

3.5 An Upper Bound on $\alpha^*_{p,C}$ and α^*_p

We conclude with closed-form upper bounds on $\alpha^*_{p,C}$ and α^*_p. The main idea is to replace $\Theta_p(\tau)$ with an upper bound of $\Theta_1(\tau)$ (which has a closed-form expression) in Equations (3.4) and (3.5), then directly analyze the value of $\tau > 0$ that minimizes the resulting expressions. This leads to quite tight bounds (and also yields tighter bounds than the techniques used in the proof of [5, Claim 4.4], which bounds a related quantity). For example, $\alpha^*_2 \approx 1.05006$, and the upper bound in Lemma 27 gives $\alpha^*_2 \le 1.08078$; similarly, $\alpha^*_5 \approx 0.672558$ and the upper bound in Lemma 27 gives $\alpha^*_5 \le 0.680575$.

▶ **Lemma 27.** *Define*

$$g(\sigma,\tau) := \exp(\tau/\sigma) \cdot \left(\frac{2}{1-\exp(-\tau)} - 1\right)$$

and $\tau^*(\sigma) := \operatorname{arcsinh}(\sigma) = \ln(\sigma + \sqrt{1+\sigma^2})$. *Let* σ^* *and* σ^*_C *for* $C > 1$ *be the (unique) constants for which* $g(\sigma^*, \tau^*(\sigma^*)) = 2$ *and* $g(\sigma^*_C, \tau^*(\sigma^*_C)) = 2^{1-1/C}$. *Then for any* $p \in [1,\infty)$, *we have*

$$\alpha^*_{p,C} \le \frac{1}{2} \cdot (\sigma^*_C)^{1/p} \quad \text{and} \quad \alpha^*_p \le \frac{1}{2} \cdot (\sigma^*)^{1/p} \le \frac{1}{2} \cdot 4.6723^{1/p} \ .$$

In particular, $\alpha^*_{p,C} \to 1/2$ *as* $p \to \infty$ *for any fixed* $C > 1$, *and therefore* $\alpha^*_p \to 1/2$ *as* $p \to \infty$.

Proof. For any $\tau > 0$, by the definition of $\Theta_p(\tau)$ and the formula for summing geometric series we have

$$\Theta_p(\tau) \le \Theta_1(\tau) = 1 + 2\sum_{i=1}^{\infty} \exp(-\tau)^i = \frac{2}{1-\exp(-\tau)} - 1 \ . \tag{3.6}$$

Define the objective function

$$f(p,\alpha) := \min_{\tau>0} \exp(\tau/(2\alpha)^p) \cdot \Theta_p(\tau)$$

to be the expression that is upper-bounded in Equations (3.4) and (3.5). For any fixed $\alpha > 0$, set $\sigma := (2\alpha)^p$. Applying Equation (3.6), it follows that $f(p,\alpha) \le g(\sigma,\tau)$ for any $\tau > 0$. This implies that if there exists some $\tau > 0$ satisfying $g(\sigma,\tau) \le 2$ then $\alpha^*_p \le \frac{1}{2}\sigma^{1/p}$, and similarly, if $g(\sigma,\tau) \le 2^{1-1/C}$ then $\alpha^*_{p,C} \le \frac{1}{2}\sigma^{1/p}$.

By standard calculus,

$$\frac{\partial g}{\partial \tau} = \frac{e^{\tau/\sigma}}{1-e^{-\tau}} \cdot \left((1+e^{-\tau})/\sigma - 2e^{-\tau}/(1-e^{-\tau})\right) \ .$$

Setting the right-hand side of the above expression equal to 0 and solving for τ yields the single real solution

$$\tau = \tau^*(\sigma) = \operatorname{arcsinh}(\sigma) = \ln(\sigma + \sqrt{1+\sigma^2}) \ ,$$

which is a local minimum, and therefore a global minimum of $g(\sigma,\tau)$ for any fixed $\sigma > 0$.

Define the univariate function $g^*(\sigma) := g(\sigma, \tau^*(\sigma))$. The fact that σ^* and σ^*_C exist and are unique follows by noting that $\lim_{\sigma\to 0^+} g^*(\sigma) = \infty$, that $\lim_{\sigma\to\infty} g^*(\sigma) = 1$, and that $g^*(\sigma)$ is strictly decreasing in $\sigma > 0$. By definition of σ^* (respectively, σ^*_C), it follows that $g^*(\sigma^*) = 2$ for $\alpha = \frac{1}{2}(\sigma^*)^{1/p}$, and $g^*(\sigma^*_C) = 2^{1-1/C}$ for $\alpha = \frac{1}{2}(\sigma^*_C)^{1/p}$, as desired. Moreover, one can check numerically that $\sigma^* \le 4.6723$. ◀

References

1 Leonard M. Adleman. Two theorems on random polynomial time. In *FOCS*, pages 75–83, 1978.

2 Divesh Aggarwal, Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. Fine-grained hardness of CVP(P)— Everything that we can prove (and nothing else), 2019. arXiv:1911.02440.

3 Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete Gaussian sampling: Extended abstract. In *STOC*, pages 733–742, 2015.

4 Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. Solving the closest vector problem in 2^n time – the discrete Gaussian strikes again! In *FOCS*, pages 563–582, 2015.

5 Divesh Aggarwal and Noah Stephens-Davidowitz. (Gap/S)ETH hardness of SVP. In *STOC*, pages 228–238, 2018.

6 Divesh Aggarwal and Noah Stephens-Davidowitz. Just take the average! An embarrassingly simple 2^n-time algorithm for SVP (and CVP). In *Symposium on Simplicity in Algorithms*, volume 61, pages 12:1–12:19, 2018.

7 Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *STOC*, pages 10–19, 1998.

8 Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997. doi:10.1006/jcss.1997.1472.

9 Shi Bai, Damien Stehlé, and Weiqiang Wen. Improved reduction from the bounded distance decoding problem to the unique shortest vector problem in lattices. In *ICALP*, pages 76:1–76:12, 2016.

10 Huck Bennett, Alexander Golovnev, and Noah Stephens-Davidowitz. On the quantitative hardness of CVP. In *FOCS*, 2017.

11 Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *IEEE Conference on Computational Complexity*, pages 98–109, 2014.

12 N. D. Elkies, A. M. Odlyzko, and J. A. Rush. On the packing densities of superballs and other bodies. *Inventiones mathematicae*, 105:613–639, December 1991.

13 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

14 Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. *Theory of Computing*, 8(1):513–531, 2012. Preliminary version in STOC 2007.

15 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

16 Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005. Preliminary version in FOCS 2004.

17 Subhash Khot. Hardness of approximating the shortest vector problem in high ℓ_p norms. *J. Comput. Syst. Sci.*, 72(2):206–219, 2006.

18 Ravi Kumar and D. Sivakumar. On the unique shortest lattice vector problem. *Theor. Comput. Sci.*, 255(1-2):641–648, 2001.

19 Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In *APPROX-RANDOM*, pages 450–461, 2006.

20 Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, pages 577–594, 2009.

21 J. E. Mazo and A. M. Odlyzko. Lattice points in high-dimensional spheres. *Monatshefte für Mathematik*, 110:47–61, March 1990.

22 Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000. Preliminary version in FOCS 1998.

23 Daniele Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Trans. Information Theory*, 47(3):1212–1215, 2001. doi:10.1109/18.915688.

24 Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, pages 84–93, 2008.

25 Daniele Micciancio. Inapproximability of the shortest vector problem: Toward a deterministic reduction. *Theory of Computing*, 8(1):487–512, 2012.

26 Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

27 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

28 Oded Regev and Ricky Rosen. Lattice problems and norm embeddings. In *STOC*, pages 447–456, 2006.

29 Noah Stephens-Davidowitz. Discrete Gaussian sampling reduces to CVP and SVP. In *SODA*, pages 1748–1764, 2016. doi:10.1137/1.9781611974331.ch121.

30 Noah Stephens-Davidowitz and Vinod Vaikuntanathan. SETH-hardness of coding problems. In *FOCS*, pages 287–301, 2019.

31 Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, University of Amsterdam, 1981.

Algebraic Hardness Versus Randomness in Low Characteristic

Robert Andrews
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
rgandre2@illinois.edu

Abstract

We show that lower bounds for explicit constant-variate polynomials over fields of characteristic $p > 0$ are sufficient to derandomize polynomial identity testing over fields of characteristic p. In this setting, existing work on hardness-randomness tradeoffs for polynomial identity testing requires either the characteristic to be sufficiently large or the notion of hardness to be stronger than the standard syntactic notion of hardness used in algebraic complexity. Our results make no restriction on the characteristic of the field and use standard notions of hardness.

We do this by combining the Kabanets-Impagliazzo generator with a white-box procedure to take p^{th} roots of circuits computing a p^{th} power over fields of characteristic p. When the number of variables appearing in the circuit is bounded by some constant, this procedure turns out to be efficient, which allows us to bypass difficulties related to factoring circuits in characteristic p.

We also combine the Kabanets-Impagliazzo generator with recent "bootstrapping" results in polynomial identity testing to show that a sufficiently-hard family of explicit constant-variate polynomials yields a near-complete derandomization of polynomial identity testing. This result holds over fields of both zero and positive characteristic and complements a recent work of Guo, Kumar, Saptharishi, and Solomon, who obtained a slightly stronger statement over fields of characteristic zero.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Polynomial identity testing, hardness versus randomness, low characteristic

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.37

Funding Supported by NSF grant CCF-1755921.

Acknowledgements We would like to thank Michael A. Forbes for many useful comments which helped improve the presentation of this work.

1 Introduction

The interaction between computational hardness and pseudorandomness is a central theme of computational complexity. The goal of this vein of work is to show that a class $\mathcal{C}$ of problems that are solvable by randomized algorithms can in fact be solved by deterministic algorithms which are not much slower than the known randomized algorithm, assuming lower bounds for a related class $\mathcal{D}$. When trying to derandomize BPP, the class of problems solvable in polynomial time by a randomized Turing machine with failure probability at most $1/3$, we understand this problem quite well. A series of works culminated in that of Impagliazzo and Wigderson [20], which showed that BPP = P if there are problems in E which require boolean circuits of exponential size. Subsequent work by Shaltiel and Umans [36] and Umans [40] further tightened the quantitative tradeoffs obtainable for derandomizing BPP.

In this work, we focus on the question of hardness versus randomness in the more restricted computational model of algebraic circuits, which naturally compute multivariate polynomials over a specified base field $\mathbb{F}$. Here, the algorithmic problem of interest is *polynomial identity testing* (PIT), which is the problem of determining if a given algebraic circuit computes the

identically zero polynomial. We typically consider identity testing of circuits whose size and degree are bounded by a polynomial function in the number of variables. This low-degree regime captures polynomials of interest to computer scientists, such as the determinant and permanent, and corresponds to typical algorithmic applications of PIT. In this regime, the problem of PIT is easily solved with randomness by evaluating the circuit at a randomly chosen point of a large enough grid. The correctness of this algorithm follows from the Schwartz-Zippel lemma, which roughly says that a low-degree multivariate polynomial cannot vanish at many points of a sufficiently large grid. To date, no deterministic algorithm for PIT is known that substantially improves on the naïve derandomization of the Schwartz-Zippel lemma.

Polynomial identity testing has widespread applications in theoretical computer science and has led to randomized algorithms for perfect matching [29, 23, 30], primality testing [1, 3], and equivalence testing of read-once branching programs [6], among other problems. In light of the utility of PIT as an algorithmic primitive, it is worth understanding to what extent PIT can be derandomized. There is a large body of work concerned with unconditional derandomization of PIT for various sub-classes of algebraic circuits. For more on this, we refer the reader to the surveys of Shpilka and Yehudayoff [38] and Saxena [34, 35]. In this work, we will focus on conditional derandomization of PIT under suitable hardness assumptions.

1.1 Prior Work

The first instantiation of the hardness-randomness paradigm for polynomial identity testing was given by Kabanets and Impagliazzo [21]. Their work implemented the design-based approach of Nisan and Wigderson [31] in the algebraic setting, showing that lower bounds for an explicit family of multivariate polynomials can be used to derandomize PIT.

Subsequent work by Dvir, Shpilka, and Yehudayoff [13] and Chou, Kumar, and Solomon [12] extended this to the setting of bounded-depth circuits, roughly showing that lower bounds against depth-$(\Delta + O(1))$ circuits suffice to derandomize identity testing of depth-Δ circuits, for any constant Δ. The result of Dvir, Shpilka, and Yehudayoff [13] works with any hard polynomial, but scales poorly with the individual degree of the circuit being tested. Chou, Kumar, and Solomon [12] refined the approach of Dvir, Shpilka, and Yehudayoff [13] and showed that if the family of hard polynomials has sufficiently low degree, then this dependence on the individual degree of the circuit being tested can be avoided. Implementing the hardness-randomness paradigm in low-depth is motivated in part by a host of depth-reduction results in algebraic complexity [4, 24, 39, 18] which show that polynomials computable by small circuits can be computed by non-trivially small low-depth circuits.

Returning to the setting of unrestricted circuits, recent work of Guo, Kumar, Saptharishi, and Solomon [17] uses a stronger hardness assumption than that of Kabanets and Impagliazzo [21] and obtains a stronger derandomization of PIT. Specifically, Guo, Kumar, Saptharishi, and Solomon [17] obtain a polynomial-time derandomization of PIT using lower bounds against an explicit family of constant-variate polynomials. For comparison, Kabanets and Impagliazzo [21] only obtain quasipolynomial-time algorithms for PIT under multivariate hardness assumptions. In Section 6 of this work, we further discuss the relationship between these hardness assumptions and provide evidence for the strength of constant-variate hardness compared to multivariate hardness.

A separate line of work by Agrawal, Ghosh, and Saxena [2] and Kumar, Saptharishi, and Tengse [27] shows that PIT exhibits a "bootstrapping" phenomenon. That is, if one can obtain a barely non-trivial derandomization of PIT for circuits of size and degree which are unbounded in the number of variables, then it follows that there is a near-complete derandomization of PIT for circuits of polynomial size and degree.

From these works, we have a relatively good understanding of what derandomization of PIT is possible under hardness assumptions. However, excluding the bootstrapping results of Agrawal, Ghosh, and Saxena [2] and Kumar, Saptharishi, and Tengse [27], all previous work on hardness-randomness tradeoffs for PIT requires the underlying field to be of zero or large characteristic (for the definition of the characteristic of a field, see Section 2). That is, we can derandomize PIT under hardness assumptions over the complex numbers $\mathbb{C}$ or the finite field of p^m elements $\mathbb{F}_{p^m}$ when p is sufficiently large, but we do not know how to do the same over a field of low characteristic like $\mathbb{F}_{2^m}$.

A partial exception to this deficiency is the work of Kabanets and Impagliazzo [21]. Their results yield derandomization of PIT over a finite field $\mathbb{F}_{p^m}$ assuming an explicit polynomial which is hard to compute *as a function* over $\mathbb{F}_{p^m}$. Over infinite fields, two polynomials are equal if and only if they compute the same function. However, this no longer holds over finite fields. For example, over $\mathbb{F}_2$, the polynomial $x^2 - x$ computes the zero function but is decidedly not the zero polynomial. It is more common in the study of algebraic circuits to prove lower bounds on the task of computing a polynomial as a syntactic object, not as a function. Functional lower bounds imply syntactic lower bounds, but the reverse direction does not hold, which makes proving functional lower bounds a harder task.

If one inspects the proof of Kabanets and Impagliazzo [21], the functional hardness assumption can be replaced with a slightly weaker, albeit non-standard, syntactic hardness assumption. Namely, it suffices to assume the existence of an explicit family of n-variate polynomials $\{f_n : n \in \mathbb{N}\}$ such that $f_n^{p^k}$ is hard in the syntactic sense for $1 \leqslant p^k \leqslant 2^{O(n)}$. Over characteristic zero fields, the factoring algorithm of Kaltofen [22] implies that if f is hard to compute, then f^d is comparably hard to compute as long as d is not too large. Over fields of characteristic p, it is not clear if hardness of f^p is implied by hardness of f. For example, it is consistent with our current state of knowledge that the $n \times n$ permanent $\mathrm{perm}_n(\overline{x})$ is $2^{\Omega(n)}$-hard over $\mathbb{F}_3$, but that $\mathrm{perm}_n(\overline{x})^3$ is computable by circuits of size $O(n^2)$ over $\mathbb{F}_3$. Understanding the relationship between the complexity of f and f^p over fields of characteristic $p > 0$ in general remains a challenging open problem.

For further exposition on hardness-randomness tradeoffs for PIT, see the recent survey of Kumar and Saptharishi [26].

1.2 Identity Testing in Low Characteristic

Before describing our contributions, we take a detour to look more closely at the question of derandomizing PIT over fields of low characteristic. Known techniques for derandomizing PIT over fields of small characteristic under hardness assumptions fail due to the fact that over a field of positive characteristic, the derivative of a non-constant polynomial may be zero. For example, over $\mathbb{F}_2$, we have $\frac{\partial}{\partial x}(x^2) = 2x = 0$, since $2 = 0$ in $\mathbb{F}_2$. Thus, techniques which are in some sense "analytic" break in low characteristic. Given that the problem of polynomial identity testing is entirely algebraic, it would be nice to find an "algebraic" approach that does not succumb to this flaw. Indeed, derandomizing PIT in low characteristic fields under hardness assumptions is listed as an open problem in the recent survey of Kumar and Saptharishi [26] on algebraic derandomization.

The problem of derandomizing PIT in low characteristic fields also has interesting algorithmic applications. Consider, for example, the randomized algorithm of Lovász [29] to detect whether a bipartite graph has a perfect matching. Let $G = (V_1 \sqcup V_2, E)$ be a balanced bipartite graph on $2n$ vertices with partite sets V_1 and V_2. We form the $n \times n$ symbolic matrix A given by

$$A_{i,j} = \begin{cases} x_{i,j} & \{i,j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

It is not hard to see that $\det(A) \neq 0$ if and only if G has a perfect matching. We can then check if G has a perfect matching by evaluating A at a random point chosen from a suitably large grid of integers.

In evaluating $\det(A)$, we may encounter large numbers of size $\Omega(n!)$. Arithmetic on such numbers is expensive, requiring at least $\Omega(n \log n)$ time. We could instead implement this algorithm over a finite field of size $\text{poly}(n)$. As the determinant is a polynomial of degree n, the Schwartz-Zippel lemma guarantees that this modification yields an algorithm with low error probability. What we have gained is the fact that elements of such a finite field can be represented in $O(\log n)$ bits, so our arithmetic becomes more efficient. In principle, one could choose the field so that the characteristic is large enough for the the hardness-randomness paradigm to apply, but there may be other considerations which motivate picking, say, an extension field of $\mathbb{F}_2$. Derandomizing such an algorithm (under hardness assumptions) requires extending the hardness-randomness paradigm to fields of low characteristic.

Alternatively, one can reduce the bit complexity by using a derandomized polynomial identity testing algorithm over the rational numbers, but with the arithmetic performed modulo a small prime number. This approach also achieves logarithmic bit complexity. However, we are now in the position of having to derandomize the selection of the prime number. It is not obvious how to do this much faster than brute force, so the benefits of reducing the bit complexity are negated by the need to try many different primes.

While the previous example may seem somewhat artificial, we remark that there are instances of algorithms which explicitly rely on polynomial identity testing over fields of low characteristic. For example, the randomized algorithm of Williams [41] for the k-path problem makes use of polynomial identity testing over fields of characteristic 2. If one wanted to derandomize this algorithm under a hardness assumption, prior work on hardness-randomness tradeoffs for PIT would not suffice.

1.3 Our Results

In this work, we instantiate the hardness-randomness paradigm for PIT over fields of low characteristic under standard syntactic hardness assumptions. That is, we obtain derandomization of PIT from the existence of an explicit family of hard polynomials $\{f_n : n \in \mathbb{N}\}$ without assuming hardness of p^{th} powers of f_n. At the heart of our results is a new technique for computing the map $f^p \mapsto f$ over $\mathbb{F}[\overline{x}]$ when the polynomial f^p is given by an algebraic circuit. When f depends on a small number of variables, the circuit computing f is not too much larger than the circuit which computes f^p.

▶ **Lemma 1.1** (informal version of Corollary 3.6). *Suppose $f(\overline{x})^p$ is a polynomial on $O(1)$ variables and can be computed by a circuit of size s over a field of characteristic $p > 0$. Then $f(\overline{x})$ can be computed by a circuit of size $O(s)$.*

Using this, we are able to extend the techniques of Kabanets and Impagliazzo [21] to fields of low characteristic. To do so, we need stronger hardness assumptions than those made by Kabanets and Impagliazzo [21] for the case of zero characteristic fields. In algebraic complexity, lower bounds are typically proved for families of polynomials parameterized by the number of variables, as this captures the regime of interest for algorithmic applications. To prove our results, we assume lower bounds against a family of constant-variate polynomials which are parameterized by degree.

For the sake of exposition, we focus on the case of lower bounds for univariate polynomials. A univariate polynomial of degree d can easily be computed by circuits of size $O(d)$ using Horner's rule. It is not hard to show that every such polynomial also requires size $\Omega(\log d)$

to compute. However, improving on this $\Omega(\log d)$ lower bound for an explicit family of polynomials is a long-standing open problem. Standard dimension arguments show that most univariate polynomials of degree d require circuits of size $d^{\Omega(1)}$ to compute.

When comparing statements regarding degree d univariates and degree $n^{O(1)}$ multivariate polynomials on n variables, it is instructive to think of n and $\log d$ as comparable. In this sense, our results achieve the same hardness-randomness tradeoffs as those of Kabanets and Impagliazzo [21], but require translating their hardness assumptions to the comparable statement for univariate polynomials.

Using Lemma 1.1, we can extend the analysis of Kabanets and Impagliazzo to work over fields of low characteristic. We now give two concrete examples of the derandomization we can obtain using this extension.

▶ **Theorem 1.2** (informal version of Theorem 4.3 and Corollary 4.5)**.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let $\{f_d(x) : d \in \mathbb{N}\}$ be an explicit family of univariate polynomials which cannot be computed by circuits of size less than $s(d)$ over $\mathbb{F}$.*

1. *If $s(d) = \log^{\omega(1)} d$, then there is a deterministic algorithm for identity testing of polynomial-size, polynomial-degree circuits over $\mathbb{F}$ in n variables which runs in time $2^{n^{o(1)}}$.*
2. *If $s(d) = 2^{\log^{\Omega(1)} d}$, then there is a deterministic algorithm for identity testing of polynomial-size, polynomial-degree circuits over $\mathbb{F}$ in n variables which runs in time $2^{\log^{O(1)} n}$.*

For comparison, from an $n^{\omega(1)}$ lower bound against a family of explicit multilinear polynomials, Kabanets and Impagliazzo [21] give a deterministic algorithm for PIT over fields of characteristic zero which runs in time $2^{n^{o(1)}}$. If instead one has a $2^{n^{\Omega(1)}}$ lower bound, then their techniques yield a deterministic algorithm which runs in time $2^{\log^{O(1)} n}$. Viewing $\log d$ and n as (roughly) equivalent, we see that our derandomization obtains the same tradeoff between hardness and pseudorandomness as Kabanets and Impagliazzo [21], modulo the difference between univariate and multivariate lower bounds.

It is not hard to show that lower bounds in the constant-variate regime imply comparable lower bounds in the multivariate regime (see Lemma 2.6), but the reverse implication is not known. In Section 6, we investigate the possibility of using known techniques to prove univariate lower bounds from multivariate lower bounds.

As the assumption of a hard univariate family seems strong, it raises the question of whether or not one can obtain a stronger derandomization of PIT over fields of positive characteristic under a univariate hardness assumption. There is evidence this can be done, as Guo, Kumar, Saptharishi, and Solomon [17] use univariate lower bounds to obtain a complete derandomization of PIT over fields of characteristic zero. With a more careful instantiation of the Kabanets-Impagliazzo result, we are able to derandomize PIT in a way that suffices for the bootstrapping results of Agrawal, Ghosh, and Saxena [2] and Kumar, Saptharishi, and Tengse [27] to take effect. This allows us to prove nearly-optimal hardness-randomness tradeoffs for PIT over fields of positive characteristic, which comes close to matching the characteristic zero result of Guo, Kumar, Saptharishi, and Solomon [17]. More concretely, we prove the following.

▶ **Theorem 1.3** (informal version of Theorem 5.3)**.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let $\{f_d(x) : d \in \mathbb{N}\}$ be an explicit family of univariate polynomials which cannot be computed by circuits of size less than d^δ for some constant $\delta > 0$. Then there is a deterministic algorithm for identity testing of polynomial-size, polynomial-degree algebraic circuits in n variables over $\mathbb{F}$ which runs in time $n^{\exp \circ \exp(O(\log^\star n))}$.*

The rest of this work is organized as follows. In Section 2, we establish notation, definitions, and relevant background necessary to state and prove our results. In Section 3, we prove our main technical lemma on computing p^{th} roots of algebraic circuits over fields of characteristic

$p > 0$. We then use this in Section 4 to extend the work of Kabanets and Impagliazzo to the low characteristic setting. We combine our techniques with the bootstrapping results to obtain near-complete derandomization of PIT over fields of positive characteristic in Section 5. Section 6 investigates the relationship between univariate and multivariate circuit lower bounds. We conclude in Section 7 with a collection of problems left open by this work.

2 Preliminaries

For $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$ and $[\![n]\!] \coloneqq \{0, \ldots, n-1\}$. If A is an $n \times m$ matrix, we write $A_{i,\bullet}$ and $A_{\bullet,j}$ for the i^{th} row and j^{th} column of A, respectively. We abbreviate a vector of variables $(x_1, \ldots, x_n)$, numbers $(a_1, \ldots, a_n)$, or field elements $(\alpha_1, \ldots, \alpha_n)$ by $\overline{x}$, $\overline{a}$, and $\overline{\alpha}$, respectively, where the length is usually clear from context. We also abbreviate the product $\prod_{i=1}^{n} x_i^{a_i} =: \overline{x}^{\overline{a}}$. Given a polynomial $f(\overline{x}) = \sum_{\overline{a}} \alpha_{\overline{a}} \overline{x}^{\overline{a}}$, we write $\deg(f)$ and $\operatorname{ideg}(f)$ for the *total degree* and *individual degree* of f, respectively. The total degree of f is given by $\deg(f) \coloneqq \max\{\|\overline{a}\|_1 : \alpha_{\overline{a}} \neq 0\}$, while the individual degree of f is given by $\operatorname{ideg}(f) \coloneqq \max\{\|\overline{a}\|_\infty : \alpha_{\overline{a}} \neq 0\}$.

For a field $\mathbb{F}$, the *characteristic* of $\mathbb{F}$, denoted $\operatorname{char}\mathbb{F}$, is the smallest positive integer p such that $p \cdot 1 = 0$ in $\mathbb{F}$. In the case that there is no such p, we say that $\mathbb{F}$ has characteristic zero. Alternatively, $\operatorname{char}\mathbb{F}$ is the number p such that the ring homomorphism $\mathbb{Z} \to \mathbb{F}$ induced by $1 \mapsto 1$ has kernel $p\mathbb{Z}$. The set $\mathcal{C}_{\mathbb{F}}(s, n, d) \subseteq \mathbb{F}[\overline{x}]$ denotes the set of all n-variate degree d polynomials which can be computed by an algebraic circuit of size at most s over $\mathbb{F}$.

2.1 Algebraic Computation and Polynomial Identity Testing

We assume familiarity with the models of algebraic circuits, formulae, and branching programs. When we refer to the *size* of a circuit, formula, or branching program, we mean the number of nodes in the computational device. An introduction to this area can be found in the survey of Shpilka and Yehudayoff [38]. Throughout this work, we analyze our algorithms under the assumption that arithmetic over the base field $\mathbb{F}$ can be performed in constant time.

We now collect basic definitions and results needed for the study of deterministic black-box algorithms for polynomial identity testing. More in-depth exposition is available in the recent survey of Kumar and Saptharishi [26].

We start with the notion of a hitting set, the basic object used to construct deterministic black-box algorithms for polynomial identity testing.

▶ **Definition 2.1.** *Let $\mathcal{C} \subseteq \mathbb{F}[\overline{x}]$ be a set of n-variate polynomials. We say that a set $\mathcal{H} \subseteq \mathbb{F}^n$ is a* hitting set for $\mathcal{C}$ *if for every non-zero $f(\overline{x}) \in \mathcal{C}$, there is a point $\overline{\alpha} \in \mathcal{H}$ such that $f(\overline{\alpha}) \neq 0$. If $\mathcal{H}$ can be computed in $t(n)$ time, then we say that $\mathcal{H}$ is* $t(n)$-explicit.

We now introduce hitting set generators, the analogue of pseudorandom generators in the context of algebraic derandomization.

▶ **Definition 2.2.** *Let $\mathcal{C} \subseteq \mathbb{F}[\overline{x}]$ be a set of n-variate polynomials. Let $\mathcal{G} : \mathbb{F}^m \to \mathbb{F}^n$ be a mapping given by*

$$\mathcal{G}(\overline{y}) = (\mathcal{G}_1(\overline{y}), \ldots, \mathcal{G}_n(\overline{y})),$$

where $\mathcal{G}_i \in \mathbb{F}[\overline{y}]$. We say that $\mathcal{G}$ is a hitting set generator for $\mathcal{C}$ *if for every non-zero $f(\overline{x}) \in \mathcal{C}$, we have $f(\mathcal{G}(\overline{y})) \neq 0$. The* seed length *of $\mathcal{G}$ is m. The* degree *of $\mathcal{G}$ is $\max_{i \in [n]} \deg(\mathcal{G}_i)$. We say $\mathcal{G}$ is* $t(n)$-explicit *if, given $\overline{\alpha} \in \mathbb{F}^m$, we can compute $\mathcal{G}(\overline{\alpha})$ in $t(n)$ time.*

It is a well-known result that an explicit, low-degree hitting set generator for $\mathcal{C}$ with small seed length yields an explicit hitting set for $\mathcal{C}$ of small size. The hitting set is constructed by evaluating the generator on a grid of large enough size. Correctness follows from the Schwartz-Zippel lemma.

▶ **Lemma 2.3.** *Let $\mathcal{C}$ be a set of n-variate degree d polynomials. Let $\mathcal{G} : \mathbb{F}^m \to \mathbb{F}^n$ be a $t(n)$-explicit hitting set generator for $\mathcal{C}$ of degree D. Then there is a $(dD+1)^m t(n)$-explicit hitting set $\mathcal{H}$ for $\mathcal{C}$ of size $(dD+1)^m$.*

We also need a notion of explicitness for a family of polynomials. In previous works on hardness-randomness tradeoffs for polynomial identity testing, a family of n-variate polynomials $\{f_n \in \mathbb{F}[\overline{x}] : n \in \mathbb{N}\}$ is considered explicit if f_n is computable in $\exp(O(n))$ time. However, we will need a slightly different notion of explicitness. Instead of an exponential-time algorithm to compute f_n, we require an exponential-time algorithm to compute the coefficient of a given monomial in f_n. This different notion of explicitness will be used to transition between the constant-variate and multivariate regimes later on in Section 4 and Section 5.

▶ **Definition 2.4.** *Let $\{f_{n,d}(\overline{x}) \in \mathbb{F}[\overline{x}] : n, d \in \mathbb{N}\}$ be a family of n-variate degree d polynomials. We say that this family is* strongly $t(n,d)$-explicit *if there is an algorithm which on input $(n, d, \overline{a})$ outputs the coefficient of $\overline{x}^{\overline{a}}$ in $f_{n,d}(\overline{x})$ in $t(n,d)$ time.*

▶ Remark 2.5. The preceding definition is reminiscent of Valiant's criterion for membership in VNP. Briefly, Valiant's criterion says that if the coefficient of $\overline{x}^{\overline{a}}$ can be computed in #P/poly, then the polynomial $f(\overline{x})$ is in VNP, an algebraic analogue of NP. We refer the reader to Bürgisser [8, Chapters 1 and 2] for further exposition on VNP and Valiant's criterion.

We will repeatedly build explicit families of hard multivariate polynomials out of explicit families of hard constant-variate polynomials. By "a family of hard multivariate polynomials," we mean a family of polynomials $\{f_n(\overline{x}) \in \mathbb{F}[\overline{x}] : n \in \mathbb{N}\}$, where f_n is an n-variate polynomial of degree $n^{O(1)}$. When we say "a family of hard constant-variate polynomials," we mean a family $\{f_d(\overline{x}) \in \mathbb{F}[\overline{x}] : d \in \mathbb{N}\}$, where f_d is a degree d polynomial on $k = O(1)$ variables. That is, when we consider multivariate polynomials, we parameterize the family by the number of variables and primarily consider families of small degree; when we look at constant-variate polynomials, we fix the number of variables in all polynomials and parameterize the family by the degree of the polynomial.

To illustrate how we can obtain hard multivariate polynomials from hard constant-variate polynomials, suppose $g_d(x) = \sum_{i=0}^{d} \alpha_i x^i$ is a hard degree d univariate polynomial. We will define a new polynomial $f_n(\overline{y})$ on $n := \lfloor \log d \rfloor + 1$ variables, where the monomials of f_n correspond to writing each term of g_d "in base 2." More precisely, for each $\overline{e} \in \{0,1\}^n$, let $j(\overline{e})$ be the number whose representation in binary corresponds to $\overline{e}$. We assign the coefficient $\alpha_{j(\overline{e})}$ to the monomial $\overline{y}^{\overline{e}}$ in f_n. To show that f_n is hard, we show the contrapositive: a small circuit for f_n implies a small circuit for g_d, which contradicts the hardness of g_d. The proof of this is relatively straightforward, as we simply find a way to substitute powers of x for each y_i so that the monomial $\overline{y}^{\overline{e}}$ is mapped to $x^{j(\overline{e})}$.

In the case where g_d is a polynomial in multiple variables, we simultaneously write each variable appearing in g_d "in base 2." We remark that there is nothing *a priori* special about our use of base 2. However, doing so yields polynomials which are multilinear, a fact which will be useful later on.

We now make the preceding sketch precise, showing that lower bounds in the constant-variate regime imply comparable lower bounds in the multivariate regime.

▶ **Lemma 2.6.** *Let $g_{m,d}(\overline{x}) = \sum_{\overline{a}} \alpha_{\overline{a}} \overline{x}^{\overline{a}}$ be a strongly $t(m,d)$-explicit m-variate degree d polynomial which requires circuits of size s to compute. Let $j : \{0,1\}^{\lfloor \log d \rfloor + 1} \to [\![2^{\lfloor \log d \rfloor + 1}]\!]$ be given by $j(\overline{e}) = \sum_{i=1}^{\lfloor \log d \rfloor + 1} \overline{e}_i 2^{i-1}$, that is, $j(\overline{e})$ is the number whose binary representation corresponds to $\overline{e}$. Let $\overline{y} = (y_{1,1}, \ldots, y_{1,\lfloor \log d \rfloor + 1}, \ \cdots \ , y_{m,1}, \ldots, y_{m,\lfloor \log d \rfloor + 1})$ and define*

$$f_{m,d}(\overline{y}) = \sum_{\overline{e} \in \{0,1\}^{m \times \lfloor \log d \rfloor + 1}} \alpha_{(j(\overline{e}_{1,\bullet}),\ldots,j(\overline{e}_{m,\bullet}))} \overline{y}^{\overline{e}}.$$

Then $f_{m,d}$ is a strongly $t(m,d)$-explicit multilinear polynomial on $m(\lfloor \log d \rfloor + 1)$ variables which requires circuits of size $s - \Theta(m \log d)$ to compute.

Proof. The fact that $f_{m,d}$ is multilinear is clear from the definition.

To see that $f_{m,d}$ is hard to compute, suppose Φ is a circuit of size t which computes $f_{m,d}$. By applying the Kronecker substitution $y_{i,j} \mapsto x_i^{2^j}$, we can recover a circuit which computes $g_{m,d}(\overline{x})$. This mapping can be computed in size $\Theta(m \log d)$ by repeated squaring, so we obtain a circuit for $g_{m,d}$ of size $t + \Theta(m \log d)$. By assumption, $t + \Theta(m \log d) \geqslant s$, so $t \geqslant s - \Theta(m \log d)$, which proves the lower bound on the circuit complexity of $f_{m,d}$.

Finally, remark that the binary description of a monomial in $f_{m,d}$ is exactly the same as the binary description of a monomial in $g_{m,d}$. This implies we can use the $t(m,d)$-time algorithm to compute the coefficients of $f_{m,d}$, so $f_{m,d}$ inherits the explicitness of $g_{m,d}$. ◀

Whether lower bounds in the multivariate regime imply lower bounds in the constant-variate regime is an open question. In Section 6, we give complexity-theoretic evidence that suggests the technique used to prove the preceding lemma does not suffice to prove constant-variate lower bounds from multivariate lower bounds.

In Section 5, we will run into some technical issues concerning circuits which are defined over a low-degree extension of the base field $\mathbb{F}$. The next lemma says that whenever a circuit Φ is defined over an extension $\mathbb{K} \supseteq \mathbb{F}$ of low degree, such a circuit can in fact be defined over $\mathbb{F}$ without increasing its size too much. A related result was proved in Bürgisser, Clausen, and Shokrollahi [10, §4.3], where the authors considered extensions $\mathbb{K} \supseteq \mathbb{F}$ such that circuits defined over $\mathbb{K}$ have no computational advantage compared to circuits defined over $\mathbb{F}$ when computing a polynomial in $\mathbb{F}[\overline{x}]$.

▶ **Lemma 2.7** ([8, Proposition 4.1(iii)], [19], see also [10, §4.3])**.** *Let $\mathbb{F}$ be a field and let $\mathbb{K} \supseteq \mathbb{F}$ be an extension of degree k. Suppose $f(\overline{x})$ can be computed by a circuit of size s over $\mathbb{K}$. Then there is a circuit of size $O(k^3 s)$ which computes f over $\mathbb{F}$.*

We conclude our preliminaries on algebraic complexity by quoting a celebrated result of Kaltofen which shows that algebraic circuits may be factored without a large increase in size.

▶ **Theorem 2.8** ([22])**.** *Let $f(\overline{x}) \in \mathbb{F}[\overline{x}]$ be a polynomial of degree d computable by an algebraic circuit of size s. Let $g(\overline{x}) \in \mathbb{F}[\overline{x}]$ be a factor of $f(\overline{x})$. Then there is an algebraic circuit of size $s' \leqslant O((snd)^4)$ which computes*

1. *$g(\overline{x})$, in the case that $\operatorname{char} \mathbb{F} = 0$, and*
2. *$g(\overline{x})^{p^k}$ where $k \geqslant 0$ is the largest integer such that $g(\overline{x})^{p^k}$ divides $f(\overline{x})$, in the case that $\operatorname{char} \mathbb{F} = p > 0$.*

2.2 Combinatorial Designs

We will make use of the designs of Nisan and Wigderson [31], specifically as they are used by Kabanets and Impagliazzo [21] to prove hardness-randomness tradeoffs for polynomial identity testing. Nisan and Wigderson [31] gave two constructions of designs: one via

Reed-Solomon codes, and one via a greedy algorithm. We first quote their construction using Reed-Solomon codes, which was also recently described in work by Kumar, Saptharishi, and Tengse [27].

▶ **Lemma 2.9** ([31], see also [27]). *Let $c \geqslant 2$ be a positive integer, and let $n, m, \ell, r \in \mathbb{N}$ be such that (i) $\ell = m^c$, (ii) $r \leqslant m$, (iii) m is a prime power, and (iv) $n \leqslant m^{(c-1)r}$. Then there is a collection of sets $S_1, \ldots, S_n \subseteq [\ell]$ such that*

- *for each $i \in [n]$, we have $|S_i| = m$; and*
- *for all distinct $i, j \in [n]$, we have $|S_i \cap S_j| \leqslant r$.*

Additionally, such a family can be deterministically constructed in $\mathrm{poly}(n)$ *time.*

We now cite the designs obtained by Nisan and Wigderson [31] via a greedy algorithm. In the regime where $m = O(\log n)$, this improves on the previous construction by taking the size ℓ of the ground set to be $O(\log n)$ as opposed to $O(\log^2 n)$.

▶ **Lemma 2.10** ([31]). *Let n and m be integers such that $n < 2^m$. There exists a family of sets $S_1, \ldots, S_n \subseteq [\ell]$ such that*

1. *$\ell = O(m^2/\log(n))$,*
2. *for each $i \in [n]$, we have $|S_i| = m$; and*
3. *for all distinct $i, j \in [n]$, we have $|S_i \cap S_j| \leqslant \log(n)$.*

Such a family of sets can be deterministically constructed in time $\mathrm{poly}(n, 2^\ell)$.

In extending the analysis of the Kabanets-Impagliazzo generator to low characteristic fields, we will make use of Lemma 2.10. Our use of Lemma 2.9 will arise when we combine the hardness versus randomness paradigm with the bootstrapping phenomenon. In that setting, we will apply Lemma 2.9 with $c = O(1)$ and $r = O(1)$. Compared to Lemma 2.10, this yields sets with much smaller intersection size, though the number of sets is only $m^{O(1)}$ as opposed to 2^m.

2.3 Field Theory

To cleanly state some of our results, we need the notion of a perfect field. Namely, given a circuit Φ which computes $f(\overline{x})^p \in \mathbb{F}[\overline{x}]$, we will construct in Section 3 a circuit Ψ which computes $f(\overline{x})$. This construction takes p^{th} roots of field elements $\alpha \in \mathbb{F}$, which are not always guaranteed to exist in $\mathbb{F}$. To ensure Ψ is defined over the base field $\mathbb{F}$, we require that $\mathbb{F}$ is closed under taking p^{th} roots, which is equivalent to requiring that $\mathbb{F}$ is perfect.

▶ **Definition 2.11.** *A field $\mathbb{F}$ is called* perfect *if either $\mathbb{F}$ has characteristic 0 or $\mathbb{F}$ has characteristic $p > 0$ and the map $\alpha \mapsto \alpha^p$ is an automorphism of $\mathbb{F}$. If $\mathbb{F}$ has characteristic $p > 0$, then the* perfect closure *of $\mathbb{F}$, denoted $\mathbb{F}^{p^{-\infty}}$, is the smallest field containing $\mathbb{F}$ which is closed under taking p^{th} roots.*

It is a basic fact that perfect closures exist.

▶ **Fact 2.12.** *Every field $\mathbb{F}$ of characteristic $p > 0$ has a perfect closure $\mathbb{F}^{p^{-\infty}}$.*

Informally, one can prove this by adjoining "enough" p^{th} roots to the field $\mathbb{F}$. That is, for each $\alpha \in \mathbb{F}$, we introduce a countable collection of new field elements denoted by (α, n) for $n \in \mathbb{N}$, where the element (α, n) is meant to represent $\alpha^{p^{-n}}$. We then take a quotient by a suitable equivalence relation; for example, if $\alpha^p = \beta$, then we regard (α, n) and $(\beta, n+1)$ as equivalent for all $n \in \mathbb{N}$. One must then verify that the resulting object is in fact a field and is (up to isomorphism) the perfect closure of $\mathbb{F}$. More formally, the perfect closure can be constructed as the *direct limit* of a particular *direct system* of fields. We refer the reader to Bourbaki [7, Chapter 5, §1] for the details of this construction.

Examples of perfect fields of positive characteristic include all finite fields and all algebraically closed fields of positive characteristic. A non-example is given by $\mathbb{F}_{p^m}(\overline{x})$, the field of rational functions in n variables with coefficients in $\mathbb{F}_{p^m}$, where $\mathbb{F}_{p^m}$ is the finite field of size p^m. The field $\mathbb{F}_{p^m}(\overline{x})$ fails to be perfect due to the fact that $x_1^{1/p} \notin \mathbb{F}_{p^m}(\overline{x})$, so x_1 is not in the image of the map $\alpha \mapsto \alpha^p$.

For more details on perfect fields, we refer the reader to any text on field theory, e.g., Roman [33, Chapter 3].

3 p^{th} Roots of Algebraic Computation

Suppose $\mathbb{F}$ is a field of characteristic $p > 0$ and Φ is a circuit which computes $f(\overline{x})^p$ for a polynomial $f(\overline{x})$. If we want to obtain a circuit which computes $f(\overline{x})$, then Theorem 2.8 does not suffice. In this section, we will describe a simple transformation of Φ which yields a circuit computing $f(\overline{x})$. This is the main technical step that will allow us to obtain hardness-randomness tradeoffs over fields of low characteristic.

In general, this transformation will incur an exponential blow-up in the size of Φ. If the original circuit computes a polynomial on n variables, then the new circuit we build will be larger in size by a factor of about p^{2n}. In particular, if our input is a circuit on a constant number of variables, then we only increase the size of the circuit by a constant factor. The fact that this transformation is efficient in the constant-variate regime is exactly the reason we need to use hardness of constant-variate families of polynomials as opposed to a family of hard multilinear polynomials.

Before describing the construction for circuits on an arbitrary number of variables, we first examine the case of univariate polynomials. Let $\mathbb{F}$ be a field of characteristic $p > 0$ and let $f(x) \in \mathbb{F}[x]$ be a univariate polynomial. We start by grouping the monomials of f by their degree modulo p, which allows us to write

$$f(x) = \sum_{i=0}^{p-1} \widetilde{f}_i(x) x^i,$$

where each $\widetilde{f}_i(x)$ is a univariate polynomial in x which is only supported on p^{th} powers of x. That is, the term $\widetilde{f}_i(x)x^i$ corresponds exactly to the monomials in $f(x)$ whose degree in x is congruent to i modulo p. Recall that over a field of characteristic $p > 0$, we have the identity $(a+b)^p = a^p + b^p$. Since $\widetilde{f}_i(x)$ is a sum of p^{th} powers of x, we can write

$$\widetilde{f}_i(x) = \sum_{j=0}^{d_i} \alpha_{i,j} x^{jp} = \left(\sum_{j=0}^{d_i} \alpha_{i,j}^{1/p} x^j \right)^p.$$

This expresses $\widetilde{f}_i(x)$ as a p^{th} power of the polynomial $f_i(x) \coloneqq \sum_{j=0}^{d_i} \alpha_{i,j}^{1/p} x^j$. In general, f_i may not be well-defined over $\mathbb{F}$, as the coefficients $\alpha_{i,j}^{1/p}$ may not exist in $\mathbb{F}$. However, $\alpha_{i,j}^{1/p} \in \mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$, so f_i is well-defined over $\mathbb{F}^{p^{-\infty}}$.

With this, we can write

$$f(x) = \sum_{i=0}^{p-1} f_i(x)^p x^i.$$

We refer to such an expression as the mod-p decomposition of f. This motivates the following definition, which generalizes this decomposition to the case of multivariate polynomials.

▶ **Definition 3.1.** *Let $f(\overline{x}) \in \mathbb{F}[\overline{x}]$. The* mod-$p$ decomposition of $f(\overline{x})$ *is the collection of polynomials $\{f_{\overline{a}}(\overline{x}) : \overline{a} \in [\![p]\!]^n\}$ such that*

$$f(\overline{x}) = \sum_{\overline{a} \in [\![p]\!]^n} f_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}}.$$

Over a perfect field $\mathbb{F}$ of characteristic $p > 0$, the existence of the mod-p decomposition follows from the fact that any polynomial of the form $\sum_{\overline{a}} \alpha_{\overline{a}} \overline{x}^{p \cdot \overline{a}}$ has a p^{th} root, given by $\sum_{\overline{a}} \alpha_{\overline{a}}^{1/p} \overline{x}^{\overline{a}}$. Here, we use the fact that $\mathbb{F}$ is perfect to guarantee the constants $\alpha_{\overline{a}}^{1/p}$ exist in $\mathbb{F}$. Uniqueness of the decomposition follows from the fact that the monomials $\{\overline{x}^{\overline{a}} : \overline{a} \in \mathbb{N}^n\}$ form a basis for $\mathbb{F}[\overline{x}]$. We record this observation as a lemma.

▶ **Lemma 3.2.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$ and let $f, g \in \mathbb{F}[\overline{x}]$. Let $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ and $\{g_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decompositions of f and g, respectively. Then $f = g$ if and only if $f_{\overline{a}} = g_{\overline{a}}$ for all $\overline{a} \in [\![p]\!]^n$.*

The utility of the mod-p decomposition becomes apparent when $f(\overline{x})$ is itself a p^{th} power. In this case, f itself is a sum of p^{th} powers of monomials in the variables $x_1, \ldots, x_n$, so we have $f(\overline{x}) = f_{\overline{0}}(\overline{x})^p$. Given a circuit Φ which computes f, suppose we could transform Φ into a new circuit Ψ which computes the mod-p decomposition of f. Then to compute $f(\overline{x})^{1/p}$, we simply construct the circuit Ψ and set $f_{\overline{0}}(\overline{x}) = f(\overline{x})^{1/p}$ to be the output.

Before continuing on, we record a straightforward lemma about how the mod-p decomposition behaves with respect to addition and multiplication.

▶ **Lemma 3.3.** *Let $\mathbb{F}$ be a perfect field of characteristic $p > 0$. Let $f, g \in \mathbb{F}[\overline{x}]$, and let $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ and $\{g_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decompositions of f and g, respectively. Let $h = \alpha f + \beta g$ and $q = \gamma f g$ for $\alpha, \beta, \gamma \in \mathbb{F}$. Let $\{h_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ and $\{q_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decompositions of h and q. Then for all $\overline{a} \in [\![p]\!]^n$, we have*

$$h_{\overline{a}} = \alpha^{1/p} f_{\overline{a}} + \beta^{1/p} g_{\overline{a}}$$

and

$$q_{\overline{a}} = \gamma^{1/p} \sum_{\substack{\overline{b}, \overline{c} \in [\![p]\!]^n \\ \overline{b} + \overline{c} \equiv \overline{a} \bmod p}} f_{\overline{b}} g_{\overline{c}} \overline{x}^{\frac{\overline{b} + \overline{c} - \overline{a}}{p}},$$

where the sum and congruence $\overline{b} + \overline{c} \equiv \overline{a} \bmod p$ are performed component-wise.

Proof. By expanding the equality $h = \alpha f + \beta g$ in the mod-p decomposition and using the fact that $(a + b)^p = a^p + b^p$, we obtain

$$\begin{aligned} \sum_{\overline{a} \in [\![p]\!]^n} h_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}} &= \alpha \sum_{\overline{a} \in [\![p]\!]^n} f_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}} + \beta \sum_{\overline{a} \in [\![p]\!]^n} g_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}} \\ &= \sum_{\overline{a} \in [\![p]\!]^n} (\alpha^{1/p} f_{\overline{a}}(\overline{x}) + \beta^{1/p} g_{\overline{a}}(\overline{x}))^p \overline{x}^{\overline{a}}. \end{aligned}$$

Lemma 3.2 implies that $h_{\overline{a}} = \alpha^{1/p} f_{\overline{a}} + \beta^{1/p} g_{\overline{a}}$ as claimed.

For $q(\overline{x})$, we again expand the equality $q = \gamma f g$ in the mod-p decomposition to obtain

$$\begin{aligned}\sum_{\overline{a}\in[\![p]\!]^n} q_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}} &= \gamma \left(\sum_{\overline{a}\in[\![p]\!]^n} f_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}}\right)\left(\sum_{\overline{a}\in[\![p]\!]^n} g_{\overline{a}}(\overline{x})^p \overline{x}^{\overline{a}}\right) \\ &= \gamma \sum_{\overline{b},\overline{c}\in[\![p]\!]^n} f_{\overline{b}}(\overline{x})^p g_{\overline{c}}(\overline{x})^p \overline{x}^{\overline{b}+\overline{c}} \\ &= \sum_{\overline{a}\in[\![p]\!]^n} \left(\gamma^{1/p} \sum_{\substack{\overline{b},\overline{c}\in[\![p]\!]^n \\ \overline{b}+\overline{c}\equiv\overline{a} \bmod p}} f_{\overline{b}}(\overline{x}) g_{\overline{c}}(\overline{x}) \overline{x}^{\frac{\overline{b}+\overline{c}-\overline{a}}{p}}\right)^p \overline{x}^{\overline{a}}.\end{aligned}$$

Once more, Lemma 3.2 implies that

$$q_{\overline{a}} = \gamma^{1/p} \sum_{\substack{\overline{b},\overline{c}\in[\![p]\!]^n \\ \overline{b}+\overline{c}\equiv\overline{a} \bmod p}} f_{\overline{b}} g_{\overline{c}} \overline{x}^{\frac{\overline{b}+\overline{c}-\overline{a}}{p}}$$

as claimed. ◀

3.1 Circuits

We start by implementing the strategy outlined above in the case of algebraic circuits. Throughout this and subsequent sections, Φ and Ψ will denote algebraic circuits, formulae, or branching programs, and v, u, and w will denote gates in these circuits. We will frequently refer to the polynomial computed at a gate v, which we denote by $\hat{v}$. For $\overline{a} \in [\![p]\!]^n$, we write $\hat{v}_{\overline{a}}$ for the part of the mod-p decomposition of $\hat{v}$ indexed by $\overline{a}$.

▶ **Lemma 3.4.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic circuit of size s which computes a polynomial $f(\overline{x}) \in \mathbb{F}[\overline{x}]$ and let $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decomposition of f. Then there is a circuit Ψ of size $3sp^{2n} + 2^n$ which simultaneously computes $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. To construct the desired circuit Ψ, we will split each gate v of Φ into pieces $\{(v, \overline{a}) : \overline{a} \in [\![p]\!]^n\}$ and wire Ψ so that $(v, \overline{a})$ computes $\hat{v}_{\overline{a}}$. As Φ computes $f(\overline{x})$, this implies that Ψ will contain gates computing $f_{\overline{a}}(\overline{x})$ for all $\overline{a} \in [\![p]\!]^n$. To wire each gate $(v, \overline{a})$ in Ψ, we consider the type of the gate v in Φ.

- First, suppose v is an input gate in Φ labeled by a constant $\alpha \in \mathbb{F}$. In this case, we set $(v, \overline{0}) = \alpha^{1/p}$ and $(v, \overline{a}) = 0$ for $\overline{a} \neq \overline{0}$. By definition, $\mathbb{F}^{p^{-\infty}}$ contains $\alpha^{1/p}$, so this is valid over $\mathbb{F}^{p^{-\infty}}$.
  It follows from the definition of $\hat{v}_{\overline{a}}$ that $(v, \overline{a})$ correctly computes $\hat{v}_{\overline{a}}$.
- If v is an input gate labeled by the variable x_i, let $\overline{e}_i$ denote the vector with a 1 in the i^{th} slot and zero elsewhere. We set $(v, \overline{e}_i) = 1$ and $(v, \overline{a}) = 0$ for $\overline{a} \neq \overline{e}_i$.
  Again, it follows immediately from the definition of $\hat{v}_{\overline{a}}$ that $(v, \overline{a})$ correctly computes $\hat{v}_{\overline{a}}$.
- Suppose now that v is an addition gate in Φ with children u and w with incoming edges labeled α_u and α_w. For each $\overline{a} \in [\![p]\!]^p$, we set $(v, \overline{a}) = \alpha_u^{1/p} \cdot (u, \overline{a}) + \alpha_w^{1/p} \cdot (w, \overline{a})$.
  By induction, $(u, \overline{a})$ and $(w, \overline{a})$ correctly compute $\hat{u}_{\overline{a}}$ and $\hat{w}_{\overline{a}}$, respectively. Lemma 3.3 then implies that $(v, \overline{a})$ correctly computes $\hat{v}_{\overline{a}}$.

- Finally, we consider the case where v is a multiplication gate in Φ with children u and w with incoming edges labeled α_u and α_w. For $\overline{a} \in [\![p]\!]^n$, we set

$$(v, \overline{a}) = \alpha_u^{1/p} \alpha_w^{1/p} \sum_{\substack{\overline{b}, \overline{c} \in [\![p]\!]^n \\ \overline{b} + \overline{c} \equiv \overline{a} \pmod{p}}} (u, \overline{b}) \cdot (w, \overline{c}) \cdot \overline{x}^{\frac{\overline{b} + \overline{c} - \overline{a}}{p}},$$

  where vector addition and congruence of vectors is performed coordinate-wise. Note that since $\overline{b} + \overline{c} \equiv \overline{a} \bmod p$, the vector $\frac{1}{p}(\overline{b} + \overline{c} - \overline{a})$ is in fact an integer vector. Moreover, since $\overline{b} + \overline{c} \in \{0, \ldots, 2(p-1)\}^n$, it follows that $\overline{b} + \overline{c} - \overline{a} \in \{0, p\}^n$, so $\frac{1}{p}(\overline{b} + \overline{c} - \overline{a}) \in \{0, 1\}^n$ is a zero-one vector.
  Via induction, $(u, \overline{b})$ and $(w, \overline{c})$ correctly compute $\hat{u}_{\overline{b}}$ and $\hat{w}_{\overline{c}}$, respectively. From this and Lemma 3.3, it follows that $(v, \overline{a})$ correctly computes $\hat{v}_{\overline{a}}$.

As previously remarked, since Φ computes $f(\overline{x})$, for every $\overline{a} \in [\![p]\!]^n$ there is a gate in Ψ which computes $f_{\overline{a}}(\overline{x})$, so Ψ correctly computes all components of the mod-p decomposition of f. It remains to bound the size of Ψ.

For every gate in Φ, we construct p^n gates of the form $(v, \overline{a})$ in Ψ. In the case that v is a multiplication gate, we need extra intermediate hardware to compute the summation $(v, \overline{a}) = \sum_{\overline{b} + \overline{c} \equiv \overline{a} \pmod{p}} (u, \overline{b}) \cdot (w, \overline{c}) \cdot \overline{x}^{\frac{\overline{b} + \overline{c} - \overline{a}}{p}}$. This can be done with p^n summation gates and $2p^n$ multiplication gates. We also need 2^n gates to compute the products $\overline{x}^{\overline{e}}$ for $\overline{e} \in \{0, 1\}^n$. Since Ψ is a circuit, we only need to pay for these gates once, as we can reuse them for all the multiplication computations. In total, each multiplication gate incurs an extra cost of $3p^n$ gates.

This implies each gate in Φ gives rise to at most $3p^{2n}$ gates in Ψ. As there are s gates in Φ, there are at most $3sp^{2n} + 2^n$ gates in Ψ. ◀

▶ Remark 3.5. In the above construction, rather than using the perfect closure, the resulting circuit can be defined over an extension $\mathbb{K} \supseteq \mathbb{F}$ of finite degree. This can be done by adjoining to $\mathbb{F}$ all p^{th} roots of constants which appear in Φ. The degree of this extension may be exponential in s in the worst case.

We can now use the construction of Lemma 3.4 to take p^{th} roots of circuits which compute a p^{th} power over a field of characteristic p.

▶ **Corollary 3.6.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic circuit of size s which computes a polynomial $f(\overline{x})^p \in \mathbb{F}[\overline{x}]$. Then there is a circuit Ψ of size $3sp^{2n} + 2^n$ which computes $f(\overline{x})$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. By Lemma 3.4, there is a circuit Ψ of the claimed size which computes $(f(\overline{x})^p)_{\overline{0}}$. It follows from the definition of the mod-p decomposition that $f(\overline{x}) = (f(\overline{x})^p)_{\overline{0}}$, so Ψ computes $f(\overline{x})$ as desired. ◀

▶ Remark 3.7. If $n = O(\log_p s)$, then Corollary 3.6 shows that if f^p is computable in size s, then f is computable in size $s^{O(1)}$. While the log-variate regime may appear as a somewhat artificial intermediary between the constant-variate and full multivariate regimes, it is a meaningful setting to study due to various corollaries of the bootstrapping results. For example, Forbes, Ghosh, and Saxena [14] recently studied the problem of designing explicit hitting sets for log-variate depth-three diagonal circuits.

3.2 Formulae

It is natural to ask if the mod-p decomposition allows us to efficiently take p^{th} roots in other models of algebraic computation. We address this question first in the case of algebraic formulae, and subsequently for algebraic branching programs. For the reader who is solely interested in the application of the mod-p decomposition and Corollary 3.6 to hardness-randomness tradeoffs, it is safe to skip ahead to Section 4. Before continuing on, we make an important remark regarding formulae and branching programs for univariate polynomials.

▶ Remark 3.8. In the univariate regime, our results (as stated) for formulae and branching programs are not as meaningful as the result for circuits. A formula or ABP of size s can only compute a polynomial of degree $d \leqslant s$, so any formula or ABP computing a degree d univariate polynomial must have size at least d. For univariate polynomials, Horner's rule supplies a matching $O(d)$ upper bound. Thus, the p^{th} root of a univariate polynomial which has complexity s can be computed by a device of size s/p, which is much stronger than what we will obtain in Corollary 3.10 and Corollary 3.12.

However, if one modifies the model of formulae (or branching programs) to allow leaves (or edges) labeled by a power of a variable x_i^j, then the trivial $\Omega(d)$ lower bound no longer holds. Our techniques can be adapted to this stronger model with little modification, where the upper bounds we obtain are less trivial.

We now show how one can compute the mod-p decomposition of an algebraic formula. We essentially do this by applying the transformation of Lemma 3.4 and arguing that we can convert the resulting circuit into a formula without increasing its size too much. To do this, we need some additional bookkeeping to ensure that the underlying graph of the resulting computation is a tree. We borrow this style of bookkeeping from Raz [32], who used it for improved homogenization and multilinearization of formulae. Alternatively, one can use the fact that formulae of size s can be rebalanced to have depth $O(\log s)$ and then analyze the increase in depth incurred in the proof of Lemma 3.4.

▶ **Lemma 3.9.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic formula of size s and product depth d which computes a polynomial $f(\overline{x}) \in \mathbb{F}[\overline{x}]$ and let $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decomposition of f. Then there is a formula Ψ of size $3snp^{n(d+3)}$ and product depth $d + \lceil \log n \rceil$ which simultaneously computes $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. As in Lemma 3.4, we will split each gate v of Φ into pieces which compute components of the mod-p decomposition of $\hat{v}$. However, we will need a much larger number of copies of v to ensure that the resulting circuit Ψ is in fact a formula.

We first set up some notation, borrowing heavily from Raz [32]. For a gate v in Φ, let $\text{path}(v)$ denote the set of all vertices on the path from v to the root of Φ, including v itself. Let N_v denote the set of all functions $T : \text{path}(v) \to [\![p]\!]^n$ such that for all $u, w \in \text{path}(v)$ where u is a sum gate with child w, we have $T(u) = T(w)$. Informally, the map T encodes the progression of types in the mod-p decomposition seen as the computation progresses through the formula.

For each gate v in Φ, we create a collection of gates $\{(v, \overline{a}, T) : \overline{a} \in [\![p]\!]^n, T \in N_v, T(v) = \overline{a}\}$. We will wire the gates of Ψ so that $(v, \overline{a}, T)$ computes $\hat{v}_{\overline{a}}$. As before, to wire the gates of Ψ correctly, we consider what type of gate v is in Φ. The construction only differs meaningfully from that of Lemma 3.4 in the case of multiplication gates.

- If v is an input gate in Φ labeled by $\alpha \in \mathbb{F}$, then we set $(v, \overline{0}, T) = \alpha^{1/p}$ and $(v, \overline{a}, T) = 0$ for $\overline{a} \neq \overline{0}$. As $\alpha^{1/p} \in \mathbb{F}^{p^{-\infty}}$, this produces a valid circuit over $\mathbb{F}^{p^{-\infty}}$.

  It is immediate from the definition that $(v, \overline{a}, T)$ correctly computes $\hat{v}_{\overline{a}}$.

- If v is an input gate labeled by the variable x_i, let $\overline{e}_i$ denote the vector with a 1 in the i^{th} slot and zero elsewhere. We set $(v, \overline{e}_i, T) = 1$ and $(v, \overline{a}, T) = 0$ for $\overline{a} \neq \overline{e}_i$.
  Once more, it is an immediate consequence of the definition that $(v, \overline{a}, T)$ correctly computes $\hat{v}_{\overline{a}}$.
- Suppose now that v is an addition gate with children u and w with incoming edges labeled α_u and α_w. For each $\overline{a} \in \{0, \ldots, p-1\}^n$ and $T \in N_v$, we set $(v, \overline{a}, T) = \alpha_u^{1/p} \cdot (u, \overline{a}, T_u) + \alpha_w^{1/p} \cdot (w, \overline{a}, T_w)$, where $T_u \in N_u$ and $T_w \in N_w$ extend T and satisfy $T(v) = T_u(u) = T_w(w)$.
  By induction, $(u, \overline{a}, T_u)$ and $(w, \overline{a}, T_w)$ correctly compute $\hat{u}_{\overline{a}}$ and $\hat{w}_{\overline{a}}$, respectively. By Lemma 3.3, it follows that $(v, \overline{a}, T)$ correctly computes $\hat{v}_{\overline{a}}$.
- Finally, consider the case when v is a multiplication gate with children u and w with incoming edges labeled α_u and α_w. We set

$$(v, \overline{a}, T) = \alpha_u^{1/p} \alpha_w^{1/p} \sum_{\overline{b}+\overline{c} \equiv \overline{a} \pmod{p}} (u, \overline{b}, T_{u,\overline{b}}) \cdot (w, \overline{c}, T_{w,\overline{c}}) \cdot \overline{x}^{\frac{\overline{b}+\overline{c}-\overline{a}}{p}},$$

  where $T_{u,\overline{b}}$ (respectively $T_{w,\overline{c}}$) extends T and satisfies $T_{u,\overline{b}}(u) = \overline{b}$ (respectively $T_{w,\overline{c}}(w) = \overline{c}$).
  By induction, $(u, \overline{b}, T_{u,\overline{b}})$ and $(w, \overline{c}, T_{w,\overline{c}})$ compute $\hat{u}_{\overline{b}}$ and $\hat{w}_{\overline{c}}$, respectively. Lemma 3.3 implies that $(v, \overline{a}, T)$ correctly computes $\hat{v}_{\overline{a}}$.

By construction, Ψ correctly computes $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$. It remains to bound the size and product depth of Ψ and show that Ψ is indeed a formula.

Each gate v in Φ yields $p^n |N_v|$ gates of the form $(v, \overline{a}, T)$ in Ψ. If v is a multiplication gate with children u and w, we need to implement the sum over the children $(u, \overline{b}, T_u)$ and $(w, \overline{c}, T_w)$. For a given $\overline{e} \in \{0,1\}^n$, we can compute $\overline{x}^{\overline{e}}$ using a subformula of size at most n. To compute $(v, \overline{a}, T)$, we need p^n summation gates and $2p^n$ multiplication gates in addition to the gates computing $(u, \overline{b}, T_u)$, $(w, \overline{c}, T_w)$, and $\overline{x}^{\overline{e}}$. This implies that we can compute $(v, \overline{a}, T)$ using at most $3np^n$ extra gates. Thus, for every gate v in Φ, we create at most $3np^{2n}|N_v|$ gates in Ψ.

To bound the size of N_v, note that a function $T \in N_v$ can only change values along path(v) at multiplication gates. Since there are at most d multiplication gates along path(v), we can specify T by a $(d+1)$-tuple of elements of $[\![p]\!]^n$, corresponding to the values taken by T between successive multiplication gates. This implies $|N_v| \leqslant p^{n(d+1)}$. Thus Ψ contains at most $3snp^{n(d+3)}$ gates.

It follows from the definition of Ψ that the product depth of Ψ is $d + \lceil \log n \rceil$, as the number of product gates on any path from a leaf to the root increases by at most an additive $\lceil \log n \rceil$. This arises from the need to implement a product of the form $\overline{x}^{\overline{e}}$ at gates of Ψ which correspond to multiplication gates in Φ. As we need to compute a product of this form at most once along every path from the root to a leaf, we only incur an additive $\lceil \log n \rceil$ increase in product depth as opposed to a multiplicative increase.

To see that Ψ is a formula, consider the edges leaving the gate $(u, \overline{a}, T)$. Let v denote the parent of u in Ψ. If v is an addition gate, then only $(v, \overline{a}, T_v)$ receives an edge from $(u, \overline{a}, T)$ where $T_v \in N_v$ agrees with T on path(v). If v is a multiplication gate, then only $(v, T(v), T_v)$ receives an edge from $(u, \overline{a}, T)$ where $T_v \in N_v$ agrees with T on path(v). In both cases, the fan-out of the gate u is 1, so Ψ is in fact a formula. ◀

As with circuits, we can use Lemma 3.9 to compute p^{th} roots of formulae which compute a p^{th} power over a field of characteristic $p > 0$.

▶ **Corollary 3.10.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic formula of size s and product depth d which computes a polynomial $f(\overline{x})^p \in \mathbb{F}[\overline{x}]$. Then there is a formula Ψ of size $3snp^{n(d+3)}$ and product depth $d + \lceil \log n \rceil$ which computes $f(\overline{x})$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. Analogous to the proof of Corollary 3.6. ◀

3.3 Algebraic Branching Programs

We now consider the task of taking p^{th} roots of algebraic branching programs. We consider the model of branching programs where edges may only be labeled by a constant $\alpha \in \mathbb{F}$ or a multiple of a variable αx_i. Some authors allow the edges of a branching program to be labeled by an affine form $\ell(\overline{x}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$. Such a branching program can be converted to one whose edges are labeled by field constants or multiples of a variable. This transformation increases the number of vertices by a factor of $O(n)$, which is small compared to the increase in size we will incur by taking a p^{th} root. We begin by computing the mod-p decomposition of an algebraic branching program.

▶ **Lemma 3.11.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic branching program on s vertices with edges labeled by variables or field constants which computes a polynomial $f(\overline{x}) \in \mathbb{F}[\overline{x}]$ and let $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ be the mod-p decomposition of f. Then there is an algebraic branching program Ψ on sp^n vertices which simultaneously computes $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. For each node v in Φ, we create a collection of nodes $\{(v, \overline{a}) : \overline{a} \in [\![p]\!]^n\}$ in Ψ. We will wire the nodes of Ψ so that $(v, \overline{a})$ computes $\hat{v}_{\overline{a}}$.

For a pair of vertices u and v, let $\ell(u, v)$ denote the label of the edge between u and v. Let $N^{\text{in}}(v)$ denote the set of vertices w such that the edge (w, v) is present in Φ.

Let u and v be two nodes in Φ and suppose there is an edge from u to v in Φ. We consider two cases, depending on whether this edge is labeled by a constant $\alpha \in \mathbb{F}$ or a multiple of a variable αx_i.

- Suppose the edge from u to v is labeled by $\alpha \in \mathbb{F}$. For all $\overline{a} \in [\![p]\!]^n$, we add an edge between $(u, \overline{a})$ and $(v, \overline{a})$ labeled by $\alpha^{1/p}$. Since $\alpha^{1/p} \in \mathbb{F}^{p^{-\infty}}$, this construction is valid over the perfect closure $\mathbb{F}^{p^{-\infty}}$ of $\mathbb{F}$.
- Suppose the edge from u to v is labeled by αx_i, where $\alpha \in \mathbb{F}$. Denote by $\overline{e}_i$ the vector which has a 1 in the i^{th} slot and zeroes elsewhere. For all $\overline{a} \in [\![p]\!]^n$, we add an edge between $(u, \overline{a})$ and $(v, \overline{a} + \overline{e}_i)$, where the addition $\overline{a} + \overline{e}_i$ is performed modulo p. If $\overline{a}_i < p - 1$, we label this edge with $\alpha^{1/p}$. If $\overline{a}_i = p - 1$, we label this edge with $\alpha^{1/p} x_i$. Again, $\alpha^{1/p} \in \mathbb{F}^{p^{-\infty}}$ by definition, so this construction is valid.

To see that this construction is correct, let v be a node in Φ. By the definition of an algebraic branching program, we have

$$\hat{v} = \sum_{u \in N^{\text{in}}(v)} \ell(u, v) \cdot \hat{u}.$$

Repeatedly applying the addition case of Lemma 3.3 yields, for each $\overline{a} \in [\![p]\!]^n$,

$$\hat{v}_{\overline{a}} = \sum_{u \in N^{\text{in}}(v)} (\ell(u, v) \cdot \hat{u})_{\overline{a}}.$$

If $\ell(u,v) = \alpha \in \mathbb{F}$, then we have $(\ell(u,v) \cdot \hat{u})_{\overline{a}} = \alpha^{1/p} \hat{u}_{\overline{a}}$. If $\ell(u,v) = \alpha x_i$, then if $\overline{a}_i > 0$, we have $(\ell(u,v) \cdot \hat{u})_{\overline{a}} = \alpha^{1/p} \hat{u}_{\overline{a}-\overline{e}_i}$. Otherwise, $\overline{a}_i = 0$, so $(\ell(u,v) \cdot \hat{u})_{\overline{a}} = \alpha^{1/p} \hat{u}_{\overline{a}-\overline{e}_i} x_i$, where the subtraction $\overline{a} - \overline{e}_i$ is done modulo p.

By induction, $(u, \overline{a})$ correctly computes $\hat{u}_{\overline{a}}$. From our construction of Ψ, if (u,v) is an edge in Φ, then $(v, \overline{a})$ has an incoming edge which computes $(\ell(u,v) \cdot \hat{u})_{\overline{a}}$. This implies that $(v, \overline{a})$ computes the polynomial $\sum_{u \in N^{\text{in}}(v)} (\ell(u,v) \cdot \hat{u})_{\overline{a}} = \hat{v}_{\overline{a}}$, which is what we want.

Thus, Ψ simultaneously computes $\{f_{\overline{a}} : \overline{a} \in [\![p]\!]^n\}$. Every node in Φ corresponds to p^n nodes in Ψ. Unlike the cases of circuits and formulae, we do not need extra hardware to implement intermediate calculations, so Ψ consists of sp^n nodes as claimed. ◀

Again, as in the case of circuits and formulae, this immediately yields a way to compute p^{th} roots of algebraic branching programs which compute a p^{th} power over a field of characteristic $p > 0$.

▶ **Corollary 3.12.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let Φ be an algebraic branching program on s vertices with edges labeled by variables or field constants which computes a polynomial $f(\overline{x})^p \in \mathbb{F}[\overline{x}]$. Then there is an algebraic branching program Ψ on sp^n vertices which computes $f(\overline{x})$ over $\mathbb{F}^{p^{-\infty}}$, the perfect closure of $\mathbb{F}$.*

Proof. Analogous to the proof of Corollary 3.6. ◀

4 Extending the Kabanets-Impagliazzo Generator

With our main technical tool in hand, we move on to our first application. The hitting set generator of Kabanets and Impagliazzo [21] was the first to provide hardness-randomness tradeoffs for polynomial identity testing over fields of characteristic zero. Over fields of characteristic $p > 0$, Kabanets and Impagliazzo obtain hardness-randomness tradeoffs under non-standard hardness assumptions. Namely, they require an explicit family of polynomials $\{f_n : n \in \mathbb{N}\}$ such that $f_n^{p^k}$ is hard to compute for $1 \leqslant p^k \leqslant 2^{O(n)}$, though they do not state their results in this way. Rather, they use the assumption of a family of polynomials which are hard to compute as functions, which implies hardness of p^{th} powers over finite fields.

It is more common in algebraic complexity to prove lower bounds on the task of computing polynomials as syntactic objects. Over infinite fields, this is equivalent to computing a polynomial as a function. However, the two notions differ over finite fields. For example, the polynomial $x^2 - x$ is non-zero as a polynomial over $\mathbb{F}_2$, but computes the zero function over $\mathbb{F}_2$. It is interesting to note that examples of functional lower bounds over finite fields are known. The works of Grigoriev and Karpinski [15], Grigoriev and Razborov [16], and Kumar and Saptharishi [25] prove lower bounds against constant-depth circuits over finite fields which functionally compute an explicit polynomial.

In this section, we will extend the Kabanets-Impagliazzo generator to all perfect fields of characteristic $p > 0$ under syntactic hardness assumptions for a single family of polynomials. The perfect fields of characteristic p include all finite fields and all algebraically closed fields of positive characteristic. To do this, we need a stronger (but still syntactic) hardness assumption. In their work, Kabanets and Impagliazzo use the existence of an explicit family of hard multilinear polynomials to derandomize polynomial identity testing. Here, we need lower bounds against an explicit family of constant-variate polynomials of arbitrarily high degree. Such an assumption appears to be stronger than the assumption of a hard family of multilinear polynomials. We discuss the relationship between these hypotheses in more detail in Section 6.

4.1 The Kabanets-Impagliazzo Generator

We first describe the construction of the Kabanets-Impagliazzo generator.

▶ **Construction 4.1** ([21]). *Let n and m be integers satisfying $n < 2^m$. Let $g \in \mathbb{F}[\overline{x}]$ be a polynomial on m variables. Let $S_1, \ldots, S_n \subseteq [\ell]$ be a Nisan-Wigderson design as in Lemma 2.10. The Kabanets-Impagliazzo generator $\mathcal{G}_{\mathrm{KI},g}(\overline{z}) : \mathbb{F}^\ell \to \mathbb{F}^n$ is the polynomial map given by*

$$\mathcal{G}_{\mathrm{KI},g}(\overline{z}) \coloneqq (g(\overline{z}|_{S_1}), \ldots, g(\overline{z}|_{S_n})),$$

where $\overline{z}|_{S_i}$ denotes the restriction of $\overline{z}$ to the variables with indices in S_i.

We now quote the main lemma used by Kabanets and Impagliazzo in the analysis of their generator.

▶ **Lemma 4.2** ([21]). *Let $\mathbb{F}$ be any field and $n, m \in \mathbb{N}$ such that $n < 2^m$. Let $f \in \mathbb{F}[y_1, \ldots, y_n]$ and $g \in \mathbb{F}[x_1, \ldots, x_m]$ be non-zero polynomials of degree d_f and d_g, respectively. Let $f(\overline{y})$ be computable by an algebraic circuit of size s. Let $S \subseteq \mathbb{F}$ be any set of size at least $d_f d_g + 1$ and let $\ell = O(m^2/\log n)$ be as in Lemma 2.10. Let $\mathcal{G}_{\mathrm{KI},g}$ be as in Construction 4.1.*

Suppose that $f(\mathcal{G}_{\mathrm{KI},g}(\overline{\alpha})) = 0$ for all $\overline{\alpha} \in S^\ell$. Then there is an algebraic circuit Φ of size $s' \leqslant \mathrm{poly}(n, m, d_f, d_g, s, (1+\mathrm{ideg}\, g)^{\log n})$ which computes the following. If $\mathbb{F}$ has characteristic zero, then Φ computes $g(\overline{x})$. If $\mathbb{F}$ has characteristic $p > 0$, then Φ computes $g(\overline{x})^{p^k}$ for some $k \in \mathbb{N}$ such that $p^k \leqslant d_f$.

If $f(\mathcal{G}_{\mathrm{KI},g}(\overline{z})) = 0$, then using Lemma 4.2, we can reconstruct a circuit for g using the circuit for f. By taking g from a family of hard polynomials, we obtain a contradiction if there is a small circuit which computes f. This proves that $\mathcal{G}_{\mathrm{KI},g}$ is a hitting set generator for the class of small circuits. The explicitness of $\mathcal{G}_{\mathrm{KI},g}$ follows from the explicitness of the family from which g is taken. The hardness-randomness tradeoffs of Kabanets and Impagliazzo [21] then follow by setting parameters according to the hardness of g.

Over a field of characteristic $p > 0$, Lemma 4.2 provides a circuit computing $g(\overline{x})^{p^k}$. Suppose we are working over $\mathbb{F}_q$, the finite field of $q = p^a$ elements. By taking p^{th} powers of $g(\overline{x})^{p^k}$ if necessary, we can obtain a circuit which computes $g(\overline{x})^{p^{ar}} = g(\overline{x})^{q^r}$ for some $r \in \mathbb{N}$. The map $\alpha \mapsto \alpha^q$ is the identity over $\mathbb{F}_q$, so the circuit which computes $g(\overline{x})^{q^r}$ in fact computes the same function as $g(\overline{x})$. This is why, without further work, we need a polynomial which is hard to compute as a function to obtain hardness-randomness tradeoffs over finite fields.

If we could factor the circuit for $g(\overline{x})^{p^k}$ to obtain a not-too-much-larger circuit for $g(\overline{x})$, then we could derive hardness-randomness tradeoffs from the assumption of an explicit family of multilinear polynomials which are hard to compute. It remains an open problem to show that if $g(\overline{x})^p$ has a small circuit, then $g(\overline{x})$ has a small circuit. However, in the constant-variate regime, Corollary 3.6 resolves this problem in the affirmative. This is the main fact which drives our extension of the Kabanets-Impagliazzo generator.

4.2 Extension to Fields of Low Characteristic

We now show how to use the Kabanets-Impagliazzo generator to obtain hardness-randomness tradeoffs over all perfect fields of characteristic $p > 0$. Recall that $\mathcal{C}_{\mathbb{F}}(s, n, d)$ denotes the set of n-variate degree d polynomials computable by circuits of size at most s.

▶ **Theorem 4.3.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$ and let $c, k \in \mathbb{N}$ be positive constants. Let $\{g_d(\overline{x}) : d \in \mathbb{N}\}$ be a strongly $t(k,d)$-explicit family of k-variate degree d polynomials. Let $s : \mathbb{N} \to \mathbb{N}$ be a function such that g_d cannot be computed by algebraic circuits of size smaller than $s(d)$ over $\mathbb{F}^{p^{-\infty}}$. Then there is a hitting set generator $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ for $\mathcal{C}_\mathbb{F}(n^c, n, n^c)$ which*

1. *is $\left(\mathrm{poly}(n, 2^\ell) + t(k, n^{3ck+\Omega(c)}) \cdot s^{-1}(n^{3ck+\Omega(c)})^{O(k)}\right)$-explicit,*
2. *has seed length $\ell = O\left(\frac{k^2 \log^2(s^{-1}(n^{3ck+O(c)}))}{\log n}\right)$, and*
3. *has degree $O(k \log(s^{-1}(n^{3ck+O(c)})))$.*

Proof. We will obtain our generator by using $\{g_d : d \in \mathbb{N}\}$ to construct a family of hard multilinear polynomials. We then set parameters and instantiate the Kabanets-Impagliazzo generator with this hard multilinear family.

By Lemma 2.6, there is a strongly $t(k,d)$-explicit family of multilinear polynomials $h_d(\overline{y})$ on $m := k(\lfloor \log d \rfloor + 1)$ variables such that any circuit which computes h_d must be of size $s(d) - O(k \log d)$. The construction of h_d also yields the identity

$$g_d(\overline{x}) = h_d(x_1^{2^0}, x_1^{2^1}, \ldots, x_1^{2^{\lfloor \log d \rfloor}}, \ \ldots \ , x_k^{2^0}, x_k^{2^1}, \ldots, x_k^{2^{\lfloor \log d \rfloor}}),$$

which allows us to obtain a circuit for g_d from a circuit for h_d. As h_d is multilinear, we have $\deg(h_d) \leqslant m$ and $\mathrm{ideg}(h_d) = 1$.

Set $d = s^{-1}(n^e)$ for a large enough constant $e \geqslant 1$ to be specified later. Since g_d is a k-variate degree d polynomial, we trivially have $s(d) \leqslant d^{O(k)}$, so $s^{-1}(d) \geqslant d^{\Omega(1/k)}$. This gives us

$$2^m \geqslant d^k = s^{-1}(n^e)^k \geqslant (n^{\Omega(e/k)})^k = n^{\Omega(e)}.$$

Taking e to be large enough guarantees $2^m > n$. Let $S_1, \ldots, S_n \subseteq [\ell]$ be the Nisan-Wigderson design guaranteed by Lemma 2.10. Our generator $\mathcal{G} : \mathbb{F}^\ell \to \mathbb{F}^n$ is given by instantiating the Kabanets-Impagliazzo generator with h_d. That is,

$$\mathcal{G}(\overline{z}) := \mathcal{G}_{\mathrm{KI},h_d}(\overline{z}) = (h_d(\overline{z}|_{S_1}), \ldots, h_d(\overline{z}|_{S_n})).$$

We now verify the claimed properties of $\mathcal{G}$.

Correctness. To see that $\mathcal{G}$ is indeed a hitting set generator for $\mathcal{C}_\mathbb{F}(n^c, n, n^c)$, suppose there is some non-zero $f \in \mathcal{C}_\mathbb{F}(n^c, n, n^c)$ such that $f(\mathcal{G}(\overline{z})) = 0$. Then by Lemma 4.2, there is a circuit of size

$$s' \leqslant \mathrm{poly}(n, m, n^c, 2^{\log n}) \leqslant n^{O(c)}$$

which computes $h_d(\overline{y})^{p^a}$ for $p^a \leqslant \deg(f) \leqslant n^c$. Via the Kronecker substitution $y_{i,j} \mapsto x_i^{2^j}$, we obtain a circuit of size $s' + O(k \log d) \leqslant n^{O(c)}$ which computes $g_d(\overline{x})^{p^a}$. We now apply Corollary 3.6 a total of a times to obtain a circuit which computes $g_d(\overline{x})$ and has size $s'' \leqslant (3 \cdot 2^k \cdot p^{2k})^a n^{O(c)}$. Since $p^a \leqslant n^c$ and $2 \leqslant p$, we obtain $s'' \leqslant n^{3kc+O(c)}$. By setting $e = 3ck + \Theta(c)$ where the hidden constant on the $\Theta(c)$ term is large enough, we obtain a contradiction as follows. By assumption, any circuit which computes g_d must be of size at least $s(d) = n^e$. However, we have a circuit of size $n^{3ck+O(c)} \ll n^e = s(d)$ which computes g_d, a contradiction. Thus, it must be the case that $f(\mathcal{G}(\overline{z})) \neq 0$. Hence $\mathcal{G}$ is a hitting set generator for $\mathcal{C}_\mathbb{F}(n^c, n, n^c)$.

Explicitness. Given a point $\overline{\alpha} \in \mathbb{F}^\ell$, we can evaluate $\mathcal{G}$ as follows. First, we construct the Nisan-Wigderson design $S_1, \ldots, S_n \subseteq [\ell]$ in time $\mathrm{poly}(n, 2^\ell)$. We then compute all $d^{O(k)}$ coefficients of h_d, each in $t(k,d)$ time. Finally, for each $i \in [\ell]$, we evaluate h_d on $\overline{\alpha}|_{S_i}$ in time $d^{O(k)}$. Using the fact that $d = s^{-1}(n^{3ck+O(c)})$, we can evaluate $\mathcal{G}$ in $\mathrm{poly}(n, 2^\ell) + t(k, n^{3ck+O(c)}) \cdot s^{-1}(n^{3ck+O(c)})^{O(k)}$ time as claimed.

Seed length. It follows from Lemma 2.10 that $\mathcal{G}$ has seed length $\ell = O(m^2/\log n) = O\left(\frac{k^2 \log^2 d}{\log n}\right)$. By our choice of $d = s^{-1}(n^{3ck+O(c)})$, we obtain the claimed seed length of $O\left(\frac{k^2 \log^2(s^{-1}(n^{3ck+O(c)}))}{\log n}\right)$.

Degree. By construction, $\mathcal{G}$ is a map of degree $\deg(h_d) \leqslant m = k(\lfloor \log d \rfloor + 1)$. Once more, plugging in our choice of d yields the claimed bound of $O(k \log(s^{-1}(n^{3ck+O(c)})))$. ◀

By applying Lemma 2.3, we obtain the following construction of explicit hitting sets for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$.

▶ **Corollary 4.4.** *Assume the setup of Theorem 4.3. Let T, ℓ, and Δ be the explicitness, seed length, and degree of the generator of Theorem 4.3, respectively. Then there is a hitting set $\mathcal{H}$ for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$ which*

1. *has size $|\mathcal{H}| = (n^c\Delta + 1)^\ell$, and*
2. *has explicitness $|\mathcal{H}| \cdot T = (n^c\Delta + 1)^\ell \cdot T$.*

Proof. This is Lemma 2.3 applied to Theorem 4.3. ◀

We conclude this section with some concrete hardness-randomness tradeoffs obtainable via Theorem 4.3 and Corollary 4.4. Recall that for constant k, a k-variate polynomial of degree d consists of at most $\binom{k+d}{k} \leqslant d^{O(k)}$ monomials. In this regime, a polynomial which is strongly $d^{O(k)}$-explicit is "exponential time explicit," as the description of a single monomial consists of $O(k \log d)$ bits.

▶ **Corollary 4.5.** *Let $\mathbb{F}$ be a field of characteristic $p > 0$. Let $c, k \in \mathbb{N}$ be fixed constants. Let $\{g_d(\overline{x}) : d \in \mathbb{N}\}$ be a strongly $d^{O(k)}$-explicit family of k-variate degree d polynomials which cannot be computed by circuits of size smaller than $s(d)$ over $\mathbb{F}^{p^{-\infty}}$. Then the following results hold regarding hitting sets for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$.*

1. *If $s(d) = \log^{\omega(1)} d$, then there is a $2^{n^{o(1)}}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$ of size $2^{n^{o(1)}}$.*
2. *If $s(d) = 2^{\log^{\Omega(1)} d}$, then there is a $2^{\log^{O(1)} n}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$ of size $2^{\log^{O(1)} n}$.*
3. *If $s(d) = d^{\Omega(1)}$, then there is a $n^{O(\log n)}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$ of size $n^{O(\log n)}$.*

Proof. Each statement follows by setting parameters in Theorem 4.3 and Corollary 4.4 and using the fact that c and k are fixed constants independent of n and d. We omit the straightforward calculations. ◀

5 Bootstrapping from Constant-Variate Hardness

Given that we use the seemingly stronger assumption of constant-variate hardness in our extension of the Kabanets-Impagliazzo generator, one may wonder if we can push the hardness-randomness connection further and obtain a better derandomization of identity testing for $\mathcal{C}_{\mathbb{F}}(n^c, n, n^c)$. Perhaps surprisingly, this is possible by going through the recent development of "bootstrapping" for hitting sets.

5.1 A Non-Trivial Hitting Set from Constant-Variate Hardness

Let n be a constant and let s be arbitrarily large. Suppose we have an explicit, slightly non-trivial hitting set for $\mathcal{C}_{\mathbb{F}}(s, n, s)$. Then we can "bootstrap" the advantage this hitting set has over the trivial one in order to obtain an explicit hitting set of very small size for $\mathcal{C}_{\mathbb{F}}(s, s, s)$. That is, in order to almost completely derandomize polynomial identity testing

for the class of polynomials of polynomial degree computed by polynomial-size circuits, it suffices to find a non-trivial derandomization of polynomial identity testing for circuits on a constant number of variables but of arbitrary size and degree.

We remark that, throughout this section, one should read $\mathcal{C}_{\mathbb{F}}(s,s,s)$ as a stand-in for $\mathcal{C}_{\mathbb{F}}(n^c,n,n^c)$, where c is a fixed constant. This follows by taking $s = n^c$ and noting that $\mathcal{C}_{\mathbb{F}}(n^c,n,n^c) \subseteq \mathcal{C}_{\mathbb{F}}(n^c,n^c,n^c) = \mathcal{C}_{\mathbb{F}}(s,s,s)$. While the following results are stated for $\mathcal{C}_{\mathbb{F}}(s,s,s)$, changing s by at most a polynomial factor will not qualitatively affect the results we obtain.

We now formally state the bootstrapping result. Let $\log^\star s$ denote the iterated logarithm of s. That is,

$$\log^\star s := \begin{cases} 1 + \log^\star(\log s) & s > 1 \\ 0 & s \leqslant 1. \end{cases}$$

This version of the bootstrapping theorem is due to Kumar, Saptharishi, and Tengse [27] and improves upon the initial work of Agrawal, Ghosh, and Saxena [2]. Note that this theorem holds over all fields, including those of positive characteristic.

▶ **Theorem 5.1** ([27]). *Let $\mathbb{F}$ be any field and let $\varepsilon > 0$ and $n \geqslant 2$ be constants. Suppose that for all sufficiently large s, there is an $s^{O(n)}$-explicit hitting set of size $s^{n-\varepsilon}$ for $\mathcal{C}_{\mathbb{F}}(s,n,s)$. Then there is an $s^{\exp\circ\exp(O(\log^\star s))}$-explicit hitting set of size $s^{\exp\circ\exp(O(\log^\star s))}$ for $\mathcal{C}_{\mathbb{F}}(s,s,s)$.*

In this section, we will use Theorem 5.1 to obtain a stronger derandomization of polynomial identity testing over fields of characteristic $p > 0$ under appropriate hardness assumptions. Suppose $\{g_d(\overline{x}) : d \in \mathbb{N}\}$ is a family of strongly $d^{O(k)}$-explicit k-variate degree d polynomials which require algebraic circuits of size $d^{\Omega(k)}$. Using Corollary 4.5, we can obtain a $s^{O(\log s)}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(s,s,s)$ of size $s^{O(\log s)}$. By a more careful instantiation of the Kabanets-Impagliazzo generator, we can use the hardness assumption on g_d to design an explicit hitting set which satisfies the hypotheses of Theorem 5.1. This yields an explicit hitting set for $\mathcal{C}_{\mathbb{F}}(s,s,s)$ of size $s^{\exp\circ\exp(O(\log^\star s))}$, which greatly improves upon the size $s^{O(\log s)}$ hitting set of Corollary 4.5.

Our argument also works for fields of characteristic zero, giving us a general theorem which converts near-optimal constant-variate hardness into near-optimal derandomization of polynomial identity testing for $\mathcal{C}_{\mathbb{F}}(s,s,s)$.

First, we need a technical lemma regarding lower bounds against constant-variate polynomials. Roughly, we will show that d^δ lower bounds against degree d constant-variate polynomials can be magnified to d^c lower bounds against constant-variate polynomials for arbitrary $\delta, c > 0$.

▶ **Lemma 5.2.** *Let $\mathbb{F}$ be any field. Let $k \in \mathbb{N}$ and $c, \delta > 0$ be fixed constants. Let $\{g_d(\overline{x}) : d \in \mathbb{N}\}$ be a strongly $d^{O(k)}$-explicit family of k-variate polynomials of degree d. Suppose that for d sufficiently large, g_d cannot be computed by algebraic circuits of size smaller than d^δ over $\mathbb{F}$. Then there is a constant $m \in \mathbb{N}$ and a family $\{h_\Delta(\overline{y}) : \Delta \in \mathbb{N}\}$ of strongly $\Delta^{O(m)}$-explicit m-variate degree Δ polynomials such that for Δ sufficiently large, h_Δ cannot be computed by algebraic circuits of size smaller than Δ^c over $\mathbb{F}$.*

Proof. We follow the approach of Lemma 2.6, but in base $d^{\delta/2c}+1$ as opposed to base 2.

Without loss of generality, assume that $\delta \leqslant 1 \leqslant c$. Let $m := \frac{2ck}{\delta}$ and let $\overline{y} = (y_{1,1}, \ldots, y_{k,2c/\delta})$. Let $\sigma(y_{i,j}) = x_i^{(d^{\delta/2c}+1)^j}$. We will take $h_\Delta(\overline{y})$ to be the polynomial of individual degree $d^{\delta/2c}$ which satisfies the equation $h(\sigma(\overline{y})) = g_d(\overline{x})$. More explicitly, let

$g_d(\overline{x}) = \sum_{\overline{a} \in \mathbb{N}^k} \alpha_{\overline{a}} \overline{x}^{\overline{a}}$ be the expression of g_d as a sum of monomials. Let $\varphi : [\![d^{\delta/2c}+1]\!]^{2c/\delta} \to [\![d+1]\!]$ be the map which takes the base-$(d^{\delta/2c}+1)$ expansion of a number $t \in [\![d+1]\!]$ and returns t. Then we define $h_\Delta(\overline{y})$ as

$$h_\Delta(\overline{y}) = \sum_{A \in [\![d^{\delta/2c}+1]\!]^{k \times 2c/\delta}} \alpha_{\varphi(A_{1,\bullet}),\ldots,\varphi(A_{k,\bullet})} \prod_{i,j \in [\![d^{\delta/2c}+1]\!]} y_{i,j}^{A_{i,j}}.$$

It is clear from the construction of h_Δ that $h_\Delta(\sigma(\overline{y})) = g_d(\overline{x})$. The polynomial h_Δ is of individual degree at most $d^{\delta/2c}$, so $\Delta \coloneqq \deg(h_\Delta)$ can be bounded as

$$\Delta \leqslant m d^{\delta/2c} = \frac{2ckd^{\delta/2c}}{\delta}.$$

Since k and δ are fixed constants, for d large enough, we obtain $\Delta \leqslant d^{2\delta/3c}$.

To show that h_Δ has the claimed hardness, suppose we are given a circuit of size s which computes h_Δ. By repeated squaring, we may compute the map $\sigma(\overline{y})$ using a circuit of size $O(k \log d) = O(m \log \Delta) = O(\log \Delta)$. This yields a circuit of size $s' \leqslant s + O(\log \Delta)$ which computes g_d. By the assumed hardness of g_d, we have $s' \geqslant d^\delta$. Putting things together gives us

$$s \geqslant d^\delta - O(\log \Delta).$$

Since $\Delta \leqslant d^{2\delta/3c}$ for d large enough, we obtain

$$s \geqslant \Delta^{3c/2} - O(\log \Delta).$$

For Δ (and hence d) large enough, we have $s \geqslant \Delta^c$, which yields the desired lower bound on h_Δ.

It remains to verify the explicitness of h_Δ. We can compute a coefficient of h_Δ by computing the corresponding coefficient of g_d, so h_Δ inherits the strong $d^{O(k)}$-explicitness of g_d. We need to show that $d^{O(k)} \leqslant \Delta^{O(m)}$ in order to conclude that h_Δ is strongly $\Delta^{O(m)}$-explicit. By writing h_Δ as a sum of monomials, there is a circuit of size $\Delta^{O(m)}$ which computes h_Δ. Combined with the argument above, this yields a circuit of size $\Delta^{O(m)} + O(\log \Delta) = \Delta^{O(m)}$ which computes g_d. Since any circuit which computes g_d must have size d^δ, we obtain $\Delta^{O(m)} \geqslant d^\delta$. As c, k, δ, and m are all fixed constants, this yields $d^{O(k)} \leqslant \Delta^{O(m)}$ as desired. ◀

Now we are ready to state and prove our hardness-randomness tradeoff.

▶ **Theorem 5.3.** *Let $\mathbb{F}$ be any field and let $k \in \mathbb{N}$ and $\delta > 0$ be fixed constants. Let $\mathbb{K} = \mathbb{F}^{p^{-\infty}}$ if $\operatorname{char} \mathbb{F} = p > 0$ and $\mathbb{K} = \mathbb{F}$ otherwise. Let $\{g_d(\overline{x}) \in \mathbb{F}[\overline{x}] : d \in \mathbb{N}\}$ be a family of strongly $d^{O(k)}$-explicit k-variate degree d polynomials. Suppose that for all d sufficiently large, g_d cannot be computed by algebraic circuits of size smaller than d^δ over $\mathbb{K}$. Then for all sufficiently large s, there is an $s^{\exp \circ \exp(O(\log^\star s))}$-explicit hitting set of size $s^{\exp \circ \exp(O(\log^\star s))}$ for $\mathcal{C}_\mathbb{F}(s, s, s)$.*

Proof. Using Lemma 5.2, we may assume without loss of generality that $\delta \geqslant 30$.

By Theorem 5.1, it suffices to provide an explicit hitting set of size $s^{n-\varepsilon}$ for $\mathcal{C}_\mathbb{F}(s, n, s)$ for constants ε, n and all s sufficiently large. We will instantiate the Kabanets-Impagliazzo generator with g_d as the hard polynomial, using the finer-grained designs of Lemma 2.9.

Let s be given. By adding auxiliary variables if necessary, we may assume that k is a prime power. Note there is always a power of 2 between k and $2k$, so this at most doubles the number of variables in g_d. We set parameters as follows:

- $c := 3$,
- $n := 2k^{c+1} = 2k^4$,
- $r := 2$, and
- $d := s^k$.

By Lemma 2.9, we can construct in $\text{poly}(n)$ time a collection of sets $S_1, \ldots, S_n \subseteq [k^c]$ such that $|S_i| = k$ and $|S_i \cap S_j| \leqslant r$.

Consider the generator $\mathcal{G} : \mathbb{F}^{k^c} \to \mathbb{F}^n$ given by

$$\mathcal{G}(\overline{z}) = (g_d(\overline{z}|_{S_1}), \ldots, g_d(\overline{z}|_{S_n})).$$

By construction, $\mathcal{G}$ has seed length k^c and degree $d = s^k$. Since g_d is strongly $d^{O(k)}$-explicit, we can evaluate $\mathcal{G}$ by constructing the design $S_1, \ldots, S_n$, computing the coefficients of g_d, and evaluating each of the n copies of g_d. Constructing the design takes $n^{O(1)}$ time and computing the coefficients of g_d takes $d^{O(k)}$ time. To evaluate g_d, we use the expression of g_d as a sum of monomials, which requires $d^{O(k)}$ time for each of the n evaluations. In total, we can evaluate $\mathcal{G}$ in time

$$n^{O(1)} \cdot d^{O(k)} = n^{O(1)} \cdot s^{O(k^2)} = n^{O(1)} \cdot s^{O(\sqrt{n})},$$

so $\mathcal{G}$ is $s^{O(\sqrt{n})}$-explicit for s sufficiently large.

If $\mathcal{G}$ is in fact a hitting set generator for $\mathcal{C}_{\mathbb{F}}(s, n, s)$, then using Lemma 2.3, we obtain a hitting set $\mathcal{H}$ for $\mathcal{C}_{\mathbb{F}}(s, n, s)$ of size

$$(s \cdot d)^{k^c} = (s^{k+1})^{k^3} = s^{k^4+k^3} \leqslant s^{2k^4-\varepsilon} = s^{n-\varepsilon}$$

for some $\varepsilon > 0$ when s is large enough. Moreover, $\mathcal{H}$ is $s^{O(\sqrt{n})} \cdot |\mathcal{H}| \leqslant s^{O(n)}$-explicit. We now apply Theorem 5.1 to obtain the claimed $s^{\exp \circ \exp(O(\log^\star s))}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(s, s, s)$ of size $s^{\exp \circ \exp(O(\log^\star s))}$. It remains to show that $\mathcal{G}$ is indeed a hitting set generator for $\mathcal{C}_{\mathbb{F}}(s, n, s)$.

To show this, suppose for the sake of contradiction that $\mathcal{G}$ is not a hitting set generator for $\mathcal{C}_{\mathbb{F}}(s, n, s)$. Then there is some $f(\overline{y}) \in \mathcal{C}_{\mathbb{F}}(s, n, s)$ such that $f(\overline{y}) \neq 0$ and $f(\mathcal{G}(\overline{z})) = 0$. We define the hybrid polynomials $f_0, \ldots, f_n$ by

$$\begin{aligned}
f_0(\overline{y}, \overline{z}) &= f(y_1, \ldots, y_n) \\
f_1(\overline{y}, \overline{z}) &= f(g_d(\overline{z}|_{S_1}), y_2, \ldots, y_n) \\
&\vdots \\
f_{n-1}(\overline{y}, \overline{z}) &= f(g_d(\overline{z}|_{S_1}), \ldots, g_d(\overline{z}|_{S_{n-1}}), y_n) \\
f_n(\overline{y}, \overline{z}) &= f(g_d(\overline{z}|_{S_1}), \ldots, g_d(\overline{z}|_{S_n})) = f(\mathcal{G}(\overline{z})).
\end{aligned}$$

Since $f_0 \neq 0$ and $f_n = 0$, there is some $i \in [n]$ such that $f_{i-1} \neq 0$ and $f_i = 0$. Assuming $|\mathbb{F}| > sd \geqslant \deg(f_i)$, we can find an assignment to the variables $\{y_j : j \neq i\}$ and $\{z_j : j \notin S_i\}$ such that f_i remains non-zero under this partial evaluation. If $\mathbb{F}$ is too small, we may find such an assignment using values from some finite extension $\mathbb{F}' \supseteq \mathbb{F}$ of size at least $sd + 1$ (and hence degree $O(\log(sd))$). After renaming variables, denote this non-zero restriction of f_i by $\overline{f}(z_1, \ldots, z_k, y)$.

We can compute $\overline{f}$ by composing the circuit for f with at most $n - 1$ copies of the partial evaluation of $g_d(\overline{z}|_{S_j})$ for $j < i$. By assumption, we can compute f with a circuit of size s. Since $|S_j \cap S_i| \leqslant 2$ for $j \neq i$, at most 2 variables in $\overline{z}|_{S_j}$ are unset. This implies each restriction of $g_d(\overline{z}|_{S_j})$ is a polynomial of degree d on 2 variables and thus can be computed

by a depth-two circuit of size at most $d \cdot (d+1)^2$. This yields a circuit for $\overline{f}$ of size at most $s + nd \cdot (d+1)^2$. Note that the degree of $\overline{f}$ is bounded by sd, since $\overline{f}$ is the composition of two polynomials of degrees at most s and d.

By assumption, we have that $\overline{f}(z_1, \ldots, z_k, y) \neq 0$ and $\overline{f}(z_1, \ldots, z_k, g_d(\overline{z})) = 0$. This implies that $y - g_d(\overline{z})$ is a factor of $\overline{f}$. We now apply Theorem 2.8 to factor the circuit for $\overline{f}$.

- If $\operatorname{char} \mathbb{F} = p > 0$, we obtain a circuit for $(y - g_d(\overline{z}))^{p^t} = y^{p^t} - g_d(\overline{z})^{p^t}$ for some $t \in \mathbb{N}$. Since $y^{p^t} - g_d(\overline{z})^{p^t}$ is a factor of $\overline{f}(z_1, \ldots, z_k, y)$, we must have

  $$dp^t = \deg(y^{p^t} - g_d(\overline{z})^{p^t}) \leqslant \deg(\overline{f}) \leqslant sd.$$

  This implies $p^t \leqslant s$. Since $\overline{f}$ has degree sd and is computable in size $s + O(nd^3)$, the circuit computing $y^{p^t} - g_d(\overline{z})^{p^t}$ has size at most $O((nsd)^{12})$. By setting $y = 0$ and negating the output of the circuit, we obtain a circuit for $g_d(\overline{z})^{p^t}$ of size $O((nsd)^{12})$.
  We now apply Corollary 3.6 a total of t times. This produces a circuit which computes $g_d(\overline{z})$ and has size $O((nsd)^{12} p^{2kt} 2^{kt} 3^t) = O((nsd)^{12} s^{3k+2})$. Here we use the fact that $p \geqslant 2$, so $2^{kt} \leqslant p^{kt} \leqslant s^k$ and $3^t \leqslant 4^t \leqslant p^{2t} \leqslant s^2$.
  In the case where $|\mathbb{F}| > sd$, the circuit for $\overline{f}$ was defined over $\mathbb{F}$, so the circuit for g_d is defined over $\mathbb{K} = \mathbb{F}^{p^{-\infty}}$. If instead $|\mathbb{F}| \leqslant sd$, the circuit for $\overline{f}$ was defined over a finite extension $\mathbb{F}' \supseteq \mathbb{F}$ of degree $O(\log(sd))$. As $\mathbb{F}'$ is a finite field, $\mathbb{F}'$ is perfect, so the circuit obtained from Corollary 3.6 is defined over $\mathbb{F}'$. We apply Lemma 2.7 to simulate this circuit over $\mathbb{F}$, incurring an extra $O(\log^3(sd))$ factor in the circuit size.
  In total, we now have a circuit which computes g_d over $\mathbb{K} = \mathbb{F}^{p^{-\infty}}$ and has size bounded by $O((nsd)^{12} s^{3k+2} \log^3(sd))$.
- If $\operatorname{char} \mathbb{F} = 0$, the previous case applies, but without the need to take a p^{th} root or simulate a field extension. This yields a circuit which computes $g_d(\overline{z})$ over $\mathbb{K} = \mathbb{F}$ and has size $O((nsd)^{12})$.

In both cases, we obtain a circuit which computes $g_d(\overline{z})$ over $\mathbb{K}$ and has size at most $O((nsd)^{12} s^{3k+2} \log^3(sd))$. Restating in terms of k and d, we have a circuit for g_d of size

$$O((nsd)^{12} s^{3k+2} \log^3(sd)) = O(k^{48} s^{14+3k} d^{12} \log^3(d)) = O(k^{48} d^{15+14/k} \log^3(d)).$$

Since $k \geqslant 1$ and k is a constant, we can bound the size of the circuit computing g_d by $O(d^{29} \log^3(d))$. This contradicts the fact that g_d requires circuits over $\mathbb{K}$ of size $d^\delta \geqslant d^{30}$ for sufficiently large d. Hence $\mathcal{G}$ is in fact a hitting set generator for $\mathcal{C}_{\mathbb{F}}(s, n, s)$. ◀

5.2 Comparison to Characteristic Zero

Over fields of characteristic zero, the recent work of Guo, Kumar, Saptharishi and Solomon [17] obtained what is currently the best-known derandomization of polynomial identity testing for $\mathcal{C}_{\mathbb{F}}(s, s, s)$ under a hardness assumption. From an explicit family of k-variate degree d polynomials of hardness $d^{\Omega(1)}$, they obtain an explicit hitting set for $\mathcal{C}_{\mathbb{F}}(s, s, s)$ of size $s^{O(1)}$. Specifically, they prove the following theorem.

▶ **Theorem 5.4** ([17]). *Let $\mathbb{F}$ be a field of characteristic zero. Let $k \in \mathbb{N}$ be large enough and let $\delta > 0$ be a fixed constant. Suppose $\{P_{k,d} \in \mathbb{F}[\overline{x}] : d \in \mathbb{N}\}$ is a family of $d^{O(k)}$-explicit k-variate polynomials of degree d such that $P_{k,d}$ cannot be computed by algebraic circuits of size smaller than d^δ. Then there is an $s^{(k/\delta)^{O(1)}}$-explicit hitting set for $\mathcal{C}_{\mathbb{F}}(s, s, s)$ of size $s^{O(k^2/\delta^2)}$.*

We remark that Guo, Kumar, Saptharishi, and Solomon [17] do not define the notion of explicitness they use in their result, but it is enough for $P_{k,d}$ to be computable by a uniform algorithm which runs in time $d^{O(k)}$. This is slightly different from our notion of strong

explicitness, where we require the coefficients of $P_{k,d}$ to be computable in $d^{O(k)}$ time. It is clear that one can pass from strong explicitness to the standard notion of explicitness by computing a polynomial as a sum of monomials. Via polynomial interpolation, one can show that polynomials which are "evaluation-explicit" are strongly explicit. In both cases, the explicitness parameter may degrade considerably, as the number of terms in a polynomial may be exponentially larger than the amount of time required to compute the polynomial or one of its coefficients. In general, one cannot hope to do better than this: in one direction, the coefficients of the permanent are easy to compute, but the permanent is widely conjectured to be hard to compute; in the other direction, there are examples of polynomials which are easy to compute but which have the permanent of a large matrix embedded in their coefficients (see, for example, Bürgisser [8, §2.3]).

In the context of Theorem 5.3 and Theorem 5.4, however, the two notions of explicitness coincide. When working with k-variate polynomials of degree d, we incur an overhead of $d^{O(k)}$ in passing between the two notions of explicitness. As the hypotheses of these theorems are already in the regime of (strong) $d^{O(k)}$-explicitness, the explicitness parameter changes by a polynomial factor, which is small enough to not affect the asymptotics of the results obtained.

The fact that the underlying field has characteristic zero is used in a key part of the proof of Theorem 5.4, and it is not clear how to adapt the proof to fields of positive characteristic. The generator used to design the hitting set in the conclusion of Theorem 5.4 is notably not a variation on the Kabanets-Impagliazzo generator, but instead a new generator whose construction is more algebraic than combinatorial in flavor.

Note that Theorem 5.3 and Theorem 5.4 require the same hardness assumption. This gives a second proof of derandomization of polynomial identity testing from an explicit family of hard constant-variate polynomials, although the derandomization we obtain is slightly weaker compared to Theorem 5.4. However, our construction does not require the characteristic of the underlying field to be zero. It is tempting to conjecture that one can recover the conclusion of Theorem 5.4 in positive characteristic by improving the bootstrapping process used to prove Theorem 5.1. It is unclear whether such a result is possible.

6 Relating Constant-Variate and Multivariate Lower Bounds

This work and the work of Guo, Kumar, Saptharishi, and Solomon [17] have shown that lower bounds against (strongly) explicit constant-variate polynomials yield very strong derandomizations of polynomial identity testing. We are able to give an explicit hitting set of size $s^{\exp \circ \exp(O(\log^\star s))}$ for $\mathcal{C}_{\mathbb{F}}(s, s, s)$ for any field $\mathbb{F}$ (this is Theorem 5.3), while Guo, Kumar, Saptharishi, and Solomon [17] obtain explicit hitting sets of size $s^{O(1)}$ for the same class when $\operatorname{char} \mathbb{F} = 0$. However, if one instead assumes the existence of a (strongly) explicit family of maximally-hard multivariate polynomials of low degree (specifically, degree $n^{O(1)}$ where n is the number of variables), it is not clear how to obtain similar derandomization results. The best-known derandomization from multivariate lower bounds is that of Kabanets and Impagliazzo [21], who gave an explicit hitting set of size $s^{O(\log s)}$ for $\mathcal{C}_{\mathbb{F}}(s, s, s)$.

The fact that we can obtain strong derandomizations of polynomial identity testing from constant-variate hardness raises the question of whether or not such derandomization is possible under multivariate hardness assumptions. A natural first approach to this would be to show that lower bounds for a (strongly) explicit family of multivariate polynomials imply comparable lower bounds against a (strongly) explicit family of constant-variate polynomials. Such an implication is known in the setting of non-commutative circuits and is due to Carmosino, Impagliazzo, Lovett, and Mihajlin [11].

It is not hard to show a connection in the other direction; that is, lower bounds against strongly explicit families of constant-variate polynomials can be translated into comparable lower bounds against strongly explicit families of multivariate polynomials. An easy way to do this is via the approach of Lemma 2.6.

In this section, we investigate to what extent a converse to Lemma 2.6 may hold. Unconditionally refuting the converse of Lemma 2.6 requires proving circuit lower bounds that seem far out of reach, so we have little hope to fully resolve this question. However, we can give some complexity-theoretic evidence which shows a converse to Lemma 2.6 is unlikely to hold. To do this, we take a detour into the arithmetic complexity of integers.

6.1 Complexity of Computing Integers

We start by defining the model we use to compute sequences of integers.

▶ **Definition 6.1.** *For a natural number $n \in \mathbb{N}$, let $\tau(n)$ denote the size of the smallest circuit which computes n using the constant 1 and the operations of addition, subtraction, and multiplication. Let $(a_n)_{n\in\mathbb{N}}$ be a sequence of natural numbers. If $\tau(a_n) \leqslant \log^{O(1)} n$, then we say $(a_n)_{n\in\mathbb{N}}$ is* easy to compute*. Otherwise, we say $(a_n)_{n\in\mathbb{N}}$ is* hard to compute*.*

As an example, the sequence $(2^n)_{n\in\mathbb{N}}$ is easy to compute, as we can compute 2^n in $O(\log n)$ arithmetic steps by repeated squaring. A major open problem in this area is to understand $\tau(n!)$, the complexity of the sequence of factorials. The following conjecture regarding $\tau(n!)$ appears to be folklore.

▶ **Conjecture 6.2.** The sequence of factorials $(n!)_{n\in\mathbb{N}}$ is hard to compute.

Prior work has established relationships between Conjecture 6.2 and other prominent conjectures in computational complexity. Blum, Cucker, Shub, and Smale [5, page 126] gave an argument that shows if $\tau(n!) \leqslant \log^{O(1)} n$, then there are circuits of $\log^{O(1)} n$ size to factor n. A related work by Shamir [37] reduces factorization to computing factorials, albeit in a slightly different model. Bürgisser [9] showed that Conjecture 6.2 implies that the $n \times n$ permanent cannot be computed by constant-free division-free algebraic circuits of size $n^{O(1)}$. Work by Lipton [28] shows that average-case hardness of factoring implies a slightly weaker form of Conjecture 6.2; namely, that the polynomial $\prod_{i=1}^{n}(x-i)$ is hard to compute by constant-free algebraic circuits.

Before moving on to address the question of a converse to Lemma 2.6, we present a reduction due to Shamir [37] which reduces the task of computing $n!$ to the task of computing $\binom{2n}{n}$.

▶ **Lemma 6.3** ([37])**.** *If $(\binom{2n}{n})_{n\in\mathbb{N}}$ is easy to compute, then $(n!)_{n\in\mathbb{N}}$ is easy to compute.*

Proof. Suppose $\tau\big(\binom{2n}{n}\big) \leqslant O(\log^c n)$. Recall the identity

$$n! = \begin{cases} ((n/2)!)^2 \cdot \binom{n}{n/2} & n \text{ is even} \\ n \cdot ((\frac{n-1}{2})!)^2 \cdot \binom{n-1}{(n-1)/2} & n \text{ is odd.} \end{cases}$$

This implies

$$\tau(n!) \leqslant \tau(n) + \tau((\lfloor n/2 \rfloor !)^2) + \tau\left(\binom{2 \cdot \lfloor n/2 \rfloor}{\lfloor n/2 \rfloor}\right).$$

Expanding out the recurrence and using the fact that $\tau((\lfloor n/2 \rfloor !)^2) \leqslant \tau(\lfloor n/2 \rfloor !) + 1$, we get

$$\begin{aligned}\tau(n!) &\leqslant \sum_{i=1}^{\log n}\left[\tau(\lfloor n/2^i\rfloor)+\tau\left(\binom{2\cdot\lfloor n/2^{i+1}\rfloor}{\lfloor n/2^{i+1}\rfloor}\right)\right]+1\\ &\leqslant \log n\cdot(O(\log n)+O(\log^c n)+1)\\ &\leqslant O(\log^{c+1} n).\end{aligned}$$

Hence $(n!)_{n\in\mathbb{N}}$ is easy to compute. ◀

6.2 The Inverse Kronecker Map and Constant-Free Circuits

Here, we show that two forms of a converse to Lemma 2.6 refute Conjecture 6.2 to varying degrees. Our first argument shows that a straightforward converse of Lemma 2.6 implies that Conjecture 6.2 fails infinitely often. That is, suppose $g(x)$ is a univariate degree d polynomial and $f(\overline{y})$ is a multilinear polynomial which simplifies to $g(x)$ under the mapping $y_i \mapsto x^{2^i}$. Lemma 2.6 says that hardness of $g(x)$ implies hardness of $f(\overline{y})$. The following conjecture, which we wish to conditionally refute, says that hardness of $f(\overline{y})$ implies hardness of $g(x)$.

▶ **Conjecture 6.4.** Let $g_{m,d}(\overline{x}) = \sum_{\overline{a}} \alpha_{\overline{a}}\overline{x}^{\overline{a}}$ be an m-variate degree d polynomial. Let $j : \{0,1\}^{\lfloor \log d\rfloor+1} \to [\![2^{\lfloor \log d\rfloor+1}]\!]$ be given by $j(\overline{e}) = \sum_{i=1}^{\lfloor \log d\rfloor+1} \overline{e}_i 2^{i-1}$. That is, $j(\overline{e})$ is the number whose binary representation corresponds to $\overline{e}$. Let

$$\overline{y} = (y_{1,1},\ldots,y_{1,\lfloor \log d\rfloor+1},\ \ldots\ ,y_{m,1},\ldots,y_{m,\lfloor \log d\rfloor+1})$$

and define

$$f_{m,d}(\overline{y}) = \sum_{\overline{e}\in\{0,1\}^{m\times\lfloor \log d\rfloor+1}} \alpha_{(j(\overline{e}_{1,\bullet}),\ldots,j(\overline{e}_{m,\bullet}))}\overline{y}^{\overline{e}}.$$

Suppose $f_{m,d}$ requires constant-free circuits of size s to compute. Then $g_{m,d}$ requires constant-free circuits of size $s^{\Omega(1)} - \Theta(m\log d)$ to compute.

We now show that Conjecture 6.4 implies the factorials are easy to compute infinitely often.

▶ **Theorem 6.5.** *Suppose Conjecture 6.4 holds over $\mathbb{Q}$. Then the sequence of factorials $(n!)_{n\in\mathbb{N}}$ is easy to compute infinitely often.*

Proof. It is easy to see that $\sum_{i=0}^{2^n}\binom{2^n}{i}x^i = (x+1)^{2^n}$ is computable by a constant-free algebraic circuit of size $O(n)$ via repeated squaring. Let

$$f_n(\overline{y}) = \sum_{\overline{e}\in\{0,1\}^{n+1}} \binom{2^n}{j(\overline{e})}\overline{y}^{\overline{e}}.$$

The contrapositive of Conjecture 6.4 yields a constant-free circuit of size $O(n^c)$ which computes f_n for some absolute constant c. Let $a_{n-1} = 1$ and $a_0 = \cdots = a_{n-2} = a_n = 0$. Then $f_n(\overline{a}) = \binom{2^n}{2^{n-1}} + 1$. By evaluating the circuit for f_n at $\overline{a}$ and subtracting 1, we obtain a circuit of size $O(n^c)$ which computes $\binom{2^n}{2^{n-1}}$.

We now follow the argument of Lemma 6.3 to construct circuits of size $O(n^{c+1})$ to compute $(2^n!)_{n\in\mathbb{N}}$. By definition, we have

$$\begin{aligned}
2^n! &= \binom{2^n}{2^{n-1}}(2^{n-1}!)^2 \\
&= \binom{2^n}{2^{n-1}}\binom{2^{n-1}}{2^{n-2}}^2(2^{n-2}!)^4 \\
&\vdots \\
&= \prod_{i=0}^{n-1}\binom{2^{n-i}}{2^{n-i-1}}^{2^i}.
\end{aligned}$$

Using the fact that we fact that we can compute $\binom{2^n}{2^{n-1}}$ by a circuit of size $O(n^c)$, we obtain

$$\tau(2^n!) \leqslant \sum_{i=0}^{n-1}\tau\left(\binom{2^{n-i}}{2^{n-i-1}}^{2^i}\right) \leqslant \sum_{i=0}^{n-1} O(n^{c+1}) \leqslant O(n^{c+2}).$$

Hence the factorials are easy to compute infinitely often. ◀

It is unclear whether there is meaningful evidence to suggest that the factorials are not easy to compute at numbers of the form 2^n. Because of this, Theorem 6.5 may be best viewed as evidence that if Conjecture 6.4 is true, the proof will not be straightforward.

Conjecture 6.4 can be seen as a base-two converse to Lemma 2.6. Instead, we might consider the following strengthening of Conjecture 6.4 to all number bases.

▶ **Conjecture 6.6.** Let $g_{m,d}(\overline{x}) = \sum_{\overline{a}} \alpha_{\overline{a}}\overline{x}^{\overline{a}}$ be an m-variate degree d polynomial. Let $k \in \mathbb{N}$ and let $j : [\![k]\!]^{\lfloor \log_k d\rfloor+1} \to [\![k^{\lfloor \log_k d\rfloor+1}]\!]$ be given by $j(\overline{e}) = \sum_{i=1}^{\lfloor \log_k d\rfloor+1}\overline{e}_i k^{i-1}$, that is, $j(\overline{e})$ is the number whose base-k representation corresponds to $\overline{e}$. Let $\overline{y} = (y_{1,1},\ldots,y_{1,\lfloor \log_k d\rfloor+1},\ \cdots\ ,y_{m,1},\ldots,y_{m,\lfloor \log_k d\rfloor+1})$ and define

$$f_{m,d}(\overline{y}) = \sum_{\overline{e}\in[\![k]\!]^{m\times\lfloor \log_k d\rfloor+1}} \alpha_{(j(\overline{e}_{1,\bullet}),\ldots,j(\overline{e}_{m,\bullet}))}\overline{y}^{\overline{e}}.$$

Suppose $f_{m,d}$ requires constant-free circuits of size s to compute. Then $g_{m,d}$ requires constant-free circuits of size $s^{\Omega(1)} - \Theta(m\log d)$ to compute.

We can show that this stronger conjecture is less likely to hold than Conjecture 6.4.

▶ **Theorem 6.7.** *Suppose Conjecture 6.6 holds over $\mathbb{Q}$. Then $(n!)_{n\in\mathbb{N}}$ is easy to compute.*

Proof. By Lemma 6.3, it suffices to show that the central binomial coefficients $\binom{2n}{n}_{n\in\mathbb{N}}$ are easy to compute. Let $n \in \mathbb{N}$ be given. There is constant-free circuit of size $O(\log n)$ which computes $g(x) = (x+1)^{2n}$. Consider the polynomial

$$f(y_1, y_n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\binom{2n}{i+jn}y_1^i y_n^j,$$

where by convention $\binom{n}{k} = 0$ when $n < k$. Note that

$$f(x, x^n) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\binom{2n}{i+jn}x^{i+jn} = \sum_{k=0}^{n^2-1}\binom{2n}{k}x^k = \sum_{k=0}^{2n}\binom{2n}{k}x^k = (x+1)^{2n}.$$

The contrapositive of Conjecture 6.6 implies that f is computable by a constant-free circuit of size $O(\log^c n)$ for some absolute constant c. We now evaluate $f(0,1)$ to obtain

$$f(0,1) = \sum_{j=0}^{n-1} \binom{2n}{jn} = \binom{2n}{0} + \binom{2n}{n} + \binom{2n}{2n} = \binom{2n}{n} + 2.$$

By computing $f(0,1) - 2$, we obtain a constant-free circuit of size $O(\log^c n)$ which computes $\binom{2n}{n}$. Hence the central binomial coefficients are easy to compute. ◀

Note that the results of this section only give evidence that Conjecture 6.4 and Conjecture 6.6 do not hold over fields of characteristic zero. Over fields of positive characteristic, it is unclear whether these conjectures are likely to be true or false. This is somewhat interesting, as if Conjecture 6.4 holds over fields of positive characteristic, then we can replace constant-variate hardness with multivariate hardness in our extension of the Kabanets-Impagliazzo generator to fields of small characteristic.

7 Conclusion and Open Problems

In this work, we gave the first instantiation of the algebraic hardness-randomness paradigm over fields of small characteristic. Our main tool was the mod-p decomposition, which we used to efficiently compute p^{th} roots of circuits which depend on a small number of variables. This allowed us to extend known hardness-randomness tradeoffs due to Kabanets and Impagliazzo [21] to fields of small characteristic under seemingly stronger hardness assumptions. We also constructed a hitting set generator which, under suitable hardness assumptions, provides a near-complete derandomization of polynomial identity testing. As our hardness assumptions are somewhat atypical, we compared them to more standard hardness assumptions and gave a conditional result which says that our hardness assumptions are not implied by standard ones.

A number of problems in low-characteristic derandomization remain open, some of which we have pointed out earlier in this work. Here, we mention some challenges which our techniques are not able to resolve.

1. Is it possible to obtain hardness-randomness tradeoffs over fields of small characteristic using a strongly explicit family of hard multilinear polynomials as opposed to constant-variate polynomials?
2. Let $\mathbb{F}$ be a field of characteristic $p > 0$, where p is some fixed constant. Suppose $f(\overline{x})^p \in \mathbb{F}[\overline{x}]$ is an n-variate polynomial which can be computed by a circuit of size s over $\mathbb{F}$. Is there a circuit of size $s^{O(1)}$ which computes $f(\overline{x})$ in the case that $n = \omega(\log s)$?
3. In the conclusion of Theorem 5.1, is it possible to obtain a hitting set of size $s^{O(1)}$? If so, this would give a construction of a hitting set generator over low characteristic fields which qualitatively matches the parameters of the generator of Guo, Kumar, Saptharishi, and Solomon [17].
4. Is it possible to lift lower bounds from the multivariate regime to the constant-variate regime? It seems like the answer may be "no," but our evidence thus far only applies to constant-free circuits over fields of characteristic zero. What can we say if we remove the constant-free restriction? What about fields of positive characteristic?

References

1 Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, 2003. Preliminary version in the *40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999)*. doi:10.1145/792538.792540.

2 Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. *Proc. Natl. Acad. Sci. USA*, 116(17):8107–8118, 2019. Preliminary version in the *50th Annual ACM Symposium on Theory of Computing (STOC 2018)*. doi:10.1073/pnas.1901272116.

3 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004. doi:10.4007/annals.2004.160.781.

4 Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. doi:10.1109/FOCS.2008.32.

5 Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998. With a foreword by Richard M. Karp. doi:10.1007/978-1-4612-0701-6.

6 Manuel Blum, Ashok K. Chandra, and Mark N. Wegman. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Inform. Process. Lett.*, 10(2):80–82, 1980. doi:10.1016/S0020-0190(80)90078-2.

7 Nicolas Bourbaki. *Algebra. II. Chapters 4–7*. Elements of Mathematics (Berlin). Springer-Verlag, Berlin, 1990. Translated from the French by P. M. Cohn and J. Howie.

8 Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2000. doi:10.1007/978-3-662-04179-6.

9 Peter Bürgisser. On defining integers and proving arithmetic circuit lower bounds. *Comput. Complexity*, 18(1):81–103, 2009. Preliminary version in the *24th Symposium on Theoretical Aspects of Computer Science (STACS 2007)*. doi:10.1007/s00037-009-0260-x.

10 Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig. doi:10.1007/978-3-662-03338-8.

11 Marco L. Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness amplification for non-commutative arithmetic circuits. In *Proceedings of the 33rd Annual Computational Complexity Conference (CCC 2018)*, volume 102 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.12.

12 Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Hardness vs randomness for bounded depth arithmetic circuits. In *Proceedings of the 33rd Annual Computational Complexity Conference (CCC 2018)*, volume 102 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.13.

13 Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*. doi:10.1137/080735850.

14 Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.54.

15 Dima Grigoriev and Marek Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 577–582. ACM, New York, 1998.

16 Dima Grigoriev and Alexander Razborov. Exponential lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Appl. Algebra Engrg. Comm. Comput.*, 10(6):465–487, 2000. Preliminary version in the *39th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1998)*. doi:10.1007/s002009900021.

17 Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. Derandomization from algebraic hardness: Treading the borders. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2019)*, pages 147–157, 2019. doi:10.1109/FOCS.2019.00018.

18 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: a chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. Preliminary version in the *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*. doi:10.1137/140957123.

19 Pavel Hrubeš and Amir Yehudayoff. Arithmetic complexity in ring extensions. *Theory of Computing*, 7(8):119–129, 2011. doi:10.4086/toc.2011.v007a008.

20 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC 1997)*, pages 220–229. ACM, New York, 1997.

21 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*. doi:10.1007/s00037-004-0182-6.

22 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Advances in Computing Research*, 5, 1989.

23 Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in Random NC. *Combinatorica*, 6(1):35–48, 1986. Preliminary version in the *17th Annual ACM Symposium on Theory of Computing (STOC 1985)*. doi:10.1007/BF02579407.

24 Pascal Koiran. Arithmetic circuits: the chasm at depth four gets wider. *Theoret. Comput. Sci.*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

25 Mrinal Kumar and Ramprasad Saptharishi. An exponential lower bound for homogeneous depth-5 circuits over finite fields. In *Proceedings of the 32nd Annual Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–30:30. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.31.

26 Mrinal Kumar and Ramprasad Saptharishi. Hardness-randomness tradeoffs for algebraic computation. *Bull. Eur. Assoc. Theor. Comput. Sci.*, 129:56–87, 2019.

27 Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. Near-optimal bootstrapping of hitting sets for algebraic circuits. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 639–646. SIAM, Philadelphia, PA, 2019. doi:10.1137/1.9781611975482.40.

28 Richard J. Lipton. Straight-line complexity and integer factorization. In *Algorithmic number theory (Ithaca, NY, 1994)*, volume 877 of *Lecture Notes in Comput. Sci.*, pages 71–79. Springer, Berlin, 1994. doi:10.1007/3-540-58691-1_45.

29 László Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of computation theory (Proc. Conf. Algebraic, Arith. and Categorical Methods in Comput. Theory, Berlin/Wendisch-Rietz, 1979)*, volume 2 of *Math. Res.*, pages 565–574. Akademie-Verlag, Berlin, 1979.

30 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. Preliminary version in the *19th Annual ACM Symposium on Theory of Computing (STOC 1987)*. doi:10.1007/BF02579206.

31 Noam Nisan and Avi Wigderson. Hardness vs. randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

32 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):Art. 40, 15, 2013. Preliminary version in the *42nd Annual ACM Symposium on Theory of Computing (STOC 2010)*. doi:10.1145/2535928.

33 Steven Roman. *Field theory*, volume 158 of *Graduate Texts in Mathematics*. Springer, New York, 2 edition, 2006.

34 Nitin Saxena. Progress on polynomial identity testing. *Bull. Eur. Assoc. Theor. Comput. Sci.*, 99:49–79, 2009.

35 Nitin Saxena. Progress on polynomial identity testing ii. In *Proceedings of the Workshop celebrating Somenath Biswas' 60th Birthday*, pages 131–146, 2014.

36 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. Preliminary version in the *42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*. doi:10.1145/1059513.1059516.

37 Adi Shamir. Factoring numbers in $O(\log n)$ arithmetic steps. *Inform. Process. Lett.*, 8(1):28–31, 1979. doi:10.1016/0020-0190(79)90087-5.

38 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: a survey of recent results and questions. *Found. Trends Theor. Comput. Sci.*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

39 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inform. and Comput.*, 240:2–11, 2015. doi:10.1016/j.ic.2014.09.004.

40 Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. System Sci.*, 67(2):419–440, 2003. Preliminary version in the *34th Annual ACM Symposium on Theory of Computing (STOC 2002)*. doi:10.1016/S0022-0000(03)00046-1.

41 Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inform. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

Sum of Squares Bounds for the Ordering Principle

Aaron Potechin
University of Chicago, IL, USA
potechin@uchicago.edu

Abstract

In this paper, we analyze the sum of squares hierarchy (SOS) on the ordering principle on n elements (which has $N = \Theta(n^2)$ variables). We prove that degree $O(\sqrt{n}log(n))$ SOS can prove the ordering principle. We then show that this upper bound is essentially tight by proving that for any $\epsilon > 0$, SOS requires degree $\Omega(n^{\frac{1}{2}-\epsilon})$ to prove the ordering principle.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases sum of squares hierarchy, proof complexity, ordering principle

Digital Object Identifier 10.4230/LIPIcs.CCC.2020.38

Funding This work was supported by the Knut and Alice Wallenberg Foundation, the European Research Council, and the Swedish Research Council.

Acknowledgements The author would like to thank Marc Vinyals for proposing this problem. The author would also like to thank Prahladh Harsha, Toniann Pitassi, and anonymous reviewers for helpful comments on this paper.

1 Introduction

In proof complexity, we study how easy or difficult it is to prove or refute various statements. Proof complexity is an extremely rich field, so we will not attempt to give an overview of proof complexity here (for a recent survey of proof complexity, see [15]). Instead, we will only describe the particular proof system and the particular statement we are considering, namely the sum of squares hierarchy (SOS), and the ordering principle.

SOS can be described in terms of sum of squares/Positivstellensatz proofs (which we write as SOS proofs for brevity). SOS proofs have the following nice properties:

1. SOS proofs are broadly applicable as they are complete for systems of polynomial equations over $\mathbb{R}$. In other words, for any system of polynomial equations over $\mathbb{R}$ which is infeasible, there is an SOS proof that it is infeasible [16].
2. SOS proofs are surprisingly powerful. In particular, SOS captures both spectral methods and powerful inequalities such as Cauchy-Schwarz and variants of hypercontractivity [2], which means that much of our mathematical reasoning can be captured by SOS proofs.
3. In some sense, SOS proofs are simple. In particular, SOS proofs only use polynomial equalities and the fact that squares are non-negative over $\mathbb{R}$.

SOS has been extensively studied, so we will not give an overview of what is known about SOS here. To learn more about SOS, see the following survey on SOS [3] and the following recent seminars/courses on SOS [4, 9, 11, 12].

The ordering principle (which has $N = \Theta(n^2)$ variables) states that if we have elements $a_1, \ldots, a_n$ which have an ordering and no two elements are equal then some element a_i must be minimal. The ordering principle is a very interesting example in proof complexity because for several proof systems, it has a small size proof but any proof must have high width/degree.

The ordering principle was first considered by Krishnamurthy [10] who conjectured that it was hard for the resolution proof system. This conjecture was refuted by Stalmark [17], who showed that the ordering principle has a polynomial size resolution proof based on induction.

35th Computational Complexity Conference (CCC 2020).
Editor: Shubhangi Saraf; Article No. 38; pp. 38:1–38:37

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


However, any resolution proof of the ordering principle must have width $\Omega(n) = \Omega(\sqrt{N})$. While this is a trivial statement because it takes width n to even describe the ordering principle, Bonet and Galesi [6] showed that the ordering principle can be modified to give a statement which can be described with constant width but resolution still requires width $\Omega(n) = \Omega(\sqrt{N})$ to prove it. This showed that the width/size tradeoff of Ben-Sasson and Wigderson [5] (which was based on the degree/size tradeoff shown for polynomial calculus by Impagliazzo, Pudlák, and Sgall [8]) is essentially tight.

Later, $\Omega(n)$ degree lower bounds for the ordering principle were also shown for polynomial calculus [7] and for the Sherali-Adams hierarchy. However, non-trivial SOS degree bounds for the ordering principle were previously unknown. Thus, a natural question is, does SOS also require degree $\Omega(n)$ to prove the ordering principle or is there an SOS proof of the ordering principle which has smaller degree?

1.1 Our Results

In this paper, we show almost tight upper and lower SOS degree bounds for the ordering principle. In particular, we show the following theorems:

▶ **Theorem 1.** *Degree $O(\sqrt{n}log(n))$ SOS can prove the ordering principle on n elements.*

▶ **Theorem 2.** *For any constant $\epsilon > 0$ there is a constant $C_\epsilon > 0$ such that for all $n \in \mathbb{N}$, degree $C_\epsilon n^{\frac{1}{2}-\epsilon}$ SOS cannot prove the ordering principle on n elements.*

Theorem 1 shows that looking at degree, SOS is more powerful than resolution, polynomial calculus, and the Sherali-Adams hierarchy for proving the ordering principle. This also implies that the ordering principle is not a tight example for the size/degree trade-off for SOS which was recently shown by Atserias and Hakoniemi [1]. In particular, Atserias and Hakoniemi show [1] that if the variables are Boolean and there is an SOS proof of size S then there must be an SOS proof of degree $O(\sqrt{Nlog(S)} + k)$ where N is the number of variables and k is the maximum degree of an axiom (which is usually constant). According to this bound, a statement such as the ordering principle which has a polynomial size proof could still require degree $\tilde{\Omega}(\sqrt{N})$ to prove. However, Theorem 1 shows that the ordering principle has an SOS proof of degree $\tilde{O}(\sqrt[4]{N})$, so there is a considerable gap. On the other hand, Theorem 2 shows that Theorem 1 is essentially tight and thus the ordering principle does still give an example where there is a small SOS proof yet any SOS proof must have high degree.

▶ Remark 3. We should note that our encoding of the ordering principle is tailored to SOS and differs from previous encodings of the ordering principle. In particular, we use non-Boolean auxiliary variables, so the size/degree trade-off for SOS doesn't actually apply to our encoding. That said, our upper bound also applies to previous encodings of the ordering principle (for which the size/degree trade-off for SOS does apply) and with additional work, we can also prove a slightly worse SOS lower bound for an encoding of the ordering principle with only Boolean variables (for technical reasons which are likely an artefact of the proof, the number of variables is increased to $N = O(n^{\frac{5}{2}})$ so the final lower bound is $\Omega(N^{\frac{1}{5}-\epsilon})$). This is discussed in Section 2.2.1 and Appendix A.

1.2 Outline

The remainder of the paper is organized as follows. In Section 2 we give some prelimiaries. In Section 3, we give natural pseudo-expectation values for the ordering principle. In Section 4, we prove our SOS upper bound. In Sections 5–9, we describe how to prove our SOS lower bound.

2 Preliminaries

In this section, we recall the definitions of SOS proofs and pseudo-expectation values and describe the encoding of the ordering principle which we will analyze.

2.1 Sum of Squares/Positivstellensatz Proofs and Pseudo-expectation Values

▶ **Definition 4.** *Given a system of polynomial equations* $\{s_i = 0\}$ *over* $\mathbb{R}$*, a degree* d *SOS proof of infeasibility is an equality of the form*

$$-1 = \sum_i f_i s_i + \sum_j g_j^2$$

where

1. $\forall i, deg(f_i) + deg(s_i) \leq d$
2. $\forall j, deg(g_j) \leq \frac{d}{2}$.

▶ Remark 5. This is a proof of infeasibility because if the equations $\{s_i = 0\}$ were satisfied we would have that $\forall i, f_i s_i = 0$ and for all $\forall j, g_j^2 \geq 0$, so we cannot possibly have that $\sum_i f_i s_i + \sum_j g_j^2 < 0$.
In order to prove that there is no degree d SOS proof of infeasibility for a system of polynomial equations over $\mathbb{R}$, we use degree d pseudo-expectation values, which are defined as follows.

▶ **Definition 6.** *Given a system of polynomial equations* $\{s_i = 0\}$ *over* $\mathbb{R}$*, degree* d *pseudo expectation values are a linear map* $\tilde{E}$ *from polynomials of degree at most* d *to* $\mathbb{R}$ *such that*

1. $\tilde{E}[1] = 1$
2. $\forall f, i, \tilde{E}[f s_i] = 0$ *whenever* $deg(f) + deg(s_i) \leq d$
3. $\forall g, \tilde{E}[g^2] \geq 0$ *whenever* $deg(g) \leq \frac{d}{2}$.

As shown by the following proposition, these conditions are a weakening of the constraint that we have expected values over an actual distribution of solutions. Thus, intuitively, pseudo-expectations look like they are the expected values (up to degree d) over a distribution of solutions, but they may not actually correspond to a distribution of solutions.

▶ **Proposition 7.** *If* Ω *is an actual distribution of solutions to the equations* $\{s_i = 0\}$ *over* $\mathbb{R}$ *then*

1. $E_\Omega[1] = 1$
2. $\forall f, i, E_\Omega[f s_i] = 0$
3. $\forall g, E_\Omega[g^2] \geq 0$.

▶ **Proposition 8.** *For a given system of polynomial equations* $\{s_i = 0\}$ *over* $\mathbb{R}$*, there cannot be both degree* d *pseudo-expectation values and a degree* d *SOS proof of infeasibility.*

Proof. Assume we have both degree d pseudo-epxectation values and a degree d SOS proof of infeasibility. Applying the pseudo-expectation values to the SOS proof gives the following contradiction:

$$-1 = \tilde{E}[-1] = \sum_i \tilde{E}[f_i s_i] + \sum_j \tilde{E}[g_j^2] \geq 0.$$ ◀

2.2 Equations for the ordering principle

For our SOS bounds, we analyze the following system of infeasible equations which corresponds to the negation of the ordering principle.

▶ **Definition 9** (Ordering principle equations). *The negation of the ordering principle says that it is possible to have distinct ordered elements $\{a_1, \ldots, a_n\}$ such that no a_j is the minimum element. We encode this with the folloing equations:*

1. *We have variables x_{ij} where we want that $x_{ij} = 1$ if $a_i < a_j$ and $x_{ij} = 0$ if $a_i > a_j$. We also have auxiliary variables $\{z_j : j \in [n]\}$.*
2. *$\forall i \neq j, x_{ij}^2 = x_{ij}$ and $x_{ij} = 1 - x_{ji}$ (ordering)*
3. *For all distinct i, j, k, $x_{ij}x_{jk}(1 - x_{ik}) = 0$ (transitivity)*
4. *$\forall j, \sum_{i \neq j} x_{ij} = 1 + z_j^2$ (for all $j \in [n]$, a_j is not the minimum element of $\{a_1, \ldots, a_n\}$)*

We call this system of equations the ordering principle equations.

▶ Remark 10. In this encoding of the negation of the ordering principle, we use the auxiliary variables $\{z_j : j \in [n]\}$ so that we can express the statement that $\forall j, \exists i \neq j : x_{ij} = 1$ as polynomial equalities of low degree.

2.2.1 Relationship to other encodings of the negation of the ordering principle

The equations in Definition 9 are tailored for SOS, so they are not the same as the encodings of the negation of ordering principle which were studied in prior works [6, 7]. We now discuss this difference and how it affects our results.

For resolution, the following axioms encode the negation of the ordering principle:

1. We have variables x_{ij} where we want that x_{ij} is true if $a_i < a_j$ and x_{ij} is false if $a_i > a_j$.
2. $\forall i \neq j, x_{ij} \vee x_{ji}$ and $\neg x_{ij} \vee \neg x_{ji}$ (ordering)
3. For all distinct i, j, k, $\neg x_{ij} \vee \neg x_{jk} \vee x_{ik} = 0$ (transitivity)
4. $\forall j, \bigvee_{i \neq j} x_{ij}$ (for all $j \in [n]$, a_j is not the minimum element of $\{a_1, \ldots, a_n\}$)

Translating this into polynomial equations, this gives us the following equations for polynomial calculus:

1. We have variables x_{ij} where we want that $x_{ij} = 1$ if $a_i < a_j$ and $x_{ij} = 0$ if $a_i > a_j$.
2. $\forall i \neq j, x_{ij}^2 = x_{ij}$ and $x_{ij} = 1 - x_{ji}$ (ordering)
3. For all distinct i, j, k, $x_{ij}x_{jk}(1 - x_{ik}) = 0$ (transitivity)
4. $\forall j, \prod_{i \neq j} (1 - x_{ij}) = 0$ (for all $j \in [n]$, a_j is not the minimum element of $\{a_1, \ldots, a_n\}$)

However, as noted in the introduction, a width/degree lower bound of $\Omega(n)$ is trivial for these encodings as the axioms already have width/degree n. To handle this, Bonet and Galesi [6] used auxiliary variables to break up the axioms into constant width axioms. Galesi and Lauria [7] instead considered the following ordering principle on graphs.

▶ **Definition 11** (Ordering principle on graphs). *Given a graph G with $V(G) = [n]$, the ordering principle on G says that if each vertex $i \in [n]$ has a value a_i and all of the values are distinct then there must be some vertex whose value is less than its neighbors' values.*

When we take the negation of the ordering principle on a graph G, this changes our axioms/equations as follows:

1. For resolution, instead of the axioms $\forall j, \bigvee_{i \neq j} x_{ij}$ we have the axioms $\forall j, \bigvee_{i:(i,j) \in E(G)} x_{ij}$.
2. For polynomial calculus, instead of the equations $\forall j, \prod_{i \neq j} (1 - x_{ij}) = 0$ we have the equations $\forall j, \prod_{i:(i,j) \in E(G)} (1 - x_{ij}) = 0$.

▶ Remark 12. If $G = K_n$ then the ordering principle on G is just the ordering principle Galesi and Lauria [7] showed that if G is an expander then polynomial calculus requires degree $\Omega(n)$ to refute these equations.

The ordering principle on graphs is a weaker statement than the ordering principle, so we would expect that its negation would be easier to refute. This is indeed the case. As shown by the following lemma, we can recover the equation $\sum_{i \neq j} x_{ij} = 1 + z_j^2$ from the equation $\prod_{i:(i,j) \in E(G)} (1 - x_{ij}) = 0$, except that z_j^2 is replaced by a sum of squares.

▶ **Lemma 13.** *Given Boolean variables $\{x_i : i \in [k]\}$ (i.e. $\forall i \in [k], x_i^2 = x_i$) and the equation $\prod_{i=1}^k (1 - x_i) = 0$, we can deduce that $\left(\sum_{i=1}^k x_i\right) - 1$ is a sum of squares.*

Proof. Observe that modulo the axioms $x_i^2 = x_i$,

$$\sum_{i=1}^k x_i - 1 = -\left(\prod_{i=1}^k (1 - x_i)\right) + \sum_{J \subseteq [k]: J \neq \emptyset} (|J| - 1) \left(\prod_{i \in J} x_i\right) \left(\prod_{i \in [k] \setminus J} (1 - x_i)\right)$$

$$= -\left(\prod_{i=1}^k (1 - x_i)\right) + \sum_{J \subseteq [k]: J \neq \emptyset} (|J| - 1) \left(\prod_{i \in J} x_i^2\right) \left(\prod_{i \in [k] \setminus J} (1 - x_i)^2\right).$$

To see this, observe that for any non-empty $J \subseteq [k]$, if $x_i = 1$ for all $i \in J$ and $x_i = 0$ for all $i \notin J$ then the left and right sides are both $|J| - 1$. Similarly, if all of the x_i are 0 then the left and right sides are both -1. ◀

This implies that our SOS upper bound holds for the graph ordering principle as well as the ordering principle. However, our SOS lower bound does not apply to the ordering principle on expander graphs. Part of the reason is that our SOS lower bound relies heavily on symmetry under permutations of $[n]$.

There is one way in which the ordering principle equations are unsatisfactory for our purposes. We want to show that the size/degree tradeoffs for SOS [1] cannot be improved too much further. However, the auxiliary variables in the ordering principle equations are not Boolean and this tradeoff only applies when all of the variables are Boolean. To fix this, we show that we can modify the ordering principle equations so that we only have Boolean variables but our SOS upper and lower bounds still hold. For details, see Appendix A.

3 Pseudo-expectation values for the ordering principle

In this section, we give natural candidate pseudo-expectation values $\tilde{E}_n$ for the ordering principle equations. In fact, $\tilde{E}_n$ is essentially the only possible symmetric pseudo-expectation values. In particular, in section 4 we will show that if $\tilde{E}_n$ fails at degree d then there is an SOS proof of degree at most $2d + 2$ that these equations are infeasible.

3.1 The candidate pseudo-expectation values $\tilde{E}_n$

As noted in Section 2, intuitively, pseudo-expectation values should look like the expected values over a distribution of solutions. Also, as shown by the following lemma, since the problem is symmetric under permutations of $[n]$, we can take $\tilde{E}$ to be symmetric as well.

▶ **Lemma 14.** *If $\{s_i = 0\}$ is a system of polynomial equations which is symmetric under permutations of $[n]$ then if there are degree d pseudo-expectation values $\tilde{E}$ then there are degree d pseudo-expectation values $\tilde{E}'$ which are symmetric under permutations of $[n]$.*

Proof. Given degree d pseudo-expectation values $\tilde{E}$, take $\tilde{E}'$ to be the linear map from polynomials of degree at most d to $\mathbb{R}$ such that for all monomials p, $\tilde{E}'[p] = E_{\pi \in S_n}[\tilde{E}[\pi(p)]]$. Now observe that

1. $\tilde{E}'[1] = E_{\pi \in S_n}[\tilde{E}[1]] = 1$
2. $\forall f, i : deg(f) + deg(s_i) \leq d, \tilde{E}'[f s_i] = E_{\pi \in S_n}[\tilde{E}[\pi(f)\pi(s_i)]] = 0$ because the system of equations $\{s_i = 0\}$ is symmetric under permutations of $[n]$.
3. $\forall g : deg(g) \leq \frac{d}{2}, \tilde{E}'[g^2] = E_{\pi \in S_n}[\tilde{E}[\pi(g)^2]] \geq 0$. ◀

Guided by this, we take the expected values over the uniform distribution over orderings of $x_1, \ldots, x_n$. These orderings are not solutions to the equations because each ordering causes one equation $\sum_{i \neq j} x_{ij} = 1 + z_j^2$ to fail. However, a random ordering makes each individual equation $\sum_{i \neq j} x_{ij} = 1 + z_j^2$ true with high probability, so the intuition is that low degree SOS will not be able to detect that there is always one equation which fails.

▶ **Definition 15.** *We define U_n to be the uniform distribution over orderings of $x_1, \ldots, x_n$, i.e. for each permutation $\pi : [n] \to [n]$ we have that with probability $\frac{1}{n!}$, $x_{\pi(1)} < x_{\pi(2)} < \ldots < x_{\pi(n)}$ and thus for all $i < j$, $x_{\pi(i)\pi(j)} = 1$ and $x_{\pi(j)\pi(i)} = 0$.*

▶ **Definition 16.** *Given a polynomial $f(\{x_{ij} : i \neq j\})$, we define*

$$\tilde{E}_n[f(\{x_{ij} : i \neq j\})] = E_{U_n}[f].$$

▶ **Example 17.** $\forall i \neq j, \tilde{E}_n[x_{ij}] = \frac{1}{2}$ because there is a $\frac{1}{2}$ chance that i comes before j in a random ordering.

▶ **Example 18.** For all distinct i, j, k, $\tilde{E_n}[x_{ij}x_{jk}] = \frac{1}{6}$ because there is a $\frac{1}{6}$ chance that $i < j < k$ in a random ordering.

However, in order to fully define $\tilde{E}_n$ we have to define $\tilde{E}_n[p]$ for monomials p involving the z variables. We can do this as follows.

▶ **Definition 19** (Candidate pseudo-expectation values)**.**

1. *For all monomials $p(\{x_{ij} : i, j \in [n], i \neq j\})$, we take $\tilde{E}_n[p] = E_{U_n}[p]$.*
2. *For all monomials $p(\{x_{ij} : i, j \in [n], i \neq j\})$, we take $\tilde{E}_n\left[\left(\prod_{j \in A} z_j\right) p\right] = 0$ whenever $A \subseteq [n]$ is non-empty because each z_j could be positive or negative.*
3. *For all monomials $p(\{x_{ij} : i, j \in [n], i \neq j\}, \{z_j : j \in [n]\})$ and all $j \in [n]$, we take $\tilde{E}_n[z_j^2 p] = \tilde{E}_n\left[\left(\sum_{i \neq j} x_{ij} - 1\right) p\right]$ because we have that for all j, $z_j^2 = \sum_{i \neq j} x_{ij} - 1$.*

3.2 Checking if $\tilde{E}_n$ are pseudo-expectation values

We now discuss what needs to be checked in order to determine whether our candidate pseudo-expectation values $\tilde{E}_n$ are actually degree d pseudo-expectation values. To analyze $\tilde{E}_n$, it is convenient to create a new variable w_j which is equal to z_j^2.

▶ **Definition 20.** *Define $w_j = \sum_{i \neq j} x_{ij} - 1$.*

Observe that viewing everything in terms of the variables $\{x_{ij}\}$ and $\{w_j\}$, $\tilde{E}_n$ is the expected values over a distribution of solutions. This implies that the polynomial equalities obtained by multiplying one of the ordering principle equations in Definition 9 by a monomial will be satisfied at all degrees, not just up to degree d. However, each w_j is supposed to be a square but this is not actually the case for this distribution, so $\tilde{E}_n$ may fail to give valid pseudo-expectation values. In fact, this is the only way in which $\tilde{E}_n$ can fail to give valid pseudo-expectation values. We make this observation precise with the following lemma:

▶ **Lemma 21.** *If* $\tilde{E}_n\left[\left(\prod_{j\in A} w_j\right) g_A((\{x_{ij} : i,j \in [n], i \neq j\})^2\right] \geq 0$ *whenever* $A \subseteq [n]$ *and* $|A| + deg(g_A) \leq \frac{d}{2}$ *then* $\tilde{E}_n$ *gives degree* d *pseudo-expectation values.*

Proof. We first check that $\tilde{E}_n[g^2] \geq 0$ whenever $deg(g) \leq \frac{d}{2}$ as this is the more interesting part. Given a polynomial g of degree at most $\frac{d}{2}$, decompose g as

$$g = \sum_{A\subseteq[n]} \left(\prod_{j\in A} z_j\right) g_A(x_1, \ldots, x_n)$$

where for all $A \subseteq [n]$, $|A| + deg(g_A) \leq \frac{d}{2}$. Now observe that

$$\begin{aligned}\tilde{E}_n[g^2] &= \tilde{E}_n\left[\sum_{A,A'\subseteq[n]} \left(\prod_{j\in A} z_j \prod_{j\in A'} z_j\right) g_A(x_1, \ldots, x_n) g_{A'}(x_1, \ldots, x_n)\right] \\ &= \sum_{A\subseteq[n]} \tilde{E}_n\left[\left(\prod_{j\in A} w_j\right) g_A^2(x_1, \ldots, x_n)\right] \geq 0.\end{aligned}$$

We now check that the polynomial equalities obtained by multiplying one of the ordering principle equations in Definition 9 by a monomial are satisfied. By the definition of $\tilde{E}_n$, we have that for all monomials p and all $j \in [n]$,

$$\tilde{E}_n[(\sum_{i\neq j} x_{ij} - 1 - z_j^2)p] = \tilde{E}_n[(\sum_{i\neq j} x_{ij} - 1)p] - \tilde{E}_n[(\sum_{i\neq j} x_{ij} - 1)p] = 0$$

To prove the other polynomial equalities, we use induction on the total degree of the $\{z_j\}$ variables. For the base case, observe that

1. $\tilde{E}_n[1] = E_{U_n}[1] = 1$
2. For all monomials $p(\{x_{ij} : i,j \in [n], i \neq j\})$ and for all ordering or transitivity constraints $s_i = 0$, $\tilde{E}_n[ps_i] = E_{U_n}[ps_i] = 0$.

For the inductive step, if p is a monomial which is divisible by z_j^2 for some j then write $p = z_j^2 p'$. By the inductive hypothesis, for all ordering or transitivity constraints $s_i = 0$,

$$\tilde{E}_n[ps_i] = \tilde{E}_n[z_j^2 p' s_i] = \tilde{E}_n[(\sum_{i\neq j} x_{ij} - 1)p' s_i] = 0.$$

Finally, if p is a monomial of the form $p = \left(\prod_{j\in A} z_j\right) p'(\{x_{ij} : i,j \in [n], i \neq j\})$ where $A \neq \emptyset$ then for all ordering or transitivity constraints $s_i = 0$,

$$\tilde{E}_n[ps_i] = \tilde{E}_n\left[\left(\prod_{j\in A} z_j\right) p' s_i\right] = 0.$$ ◀

4 $O(\sqrt{n}\log(n))$ Degree SOS Upper Bound

In this section, we prove theorem 1 by constructing a degree $O(\sqrt{n}log(n))$ proof of the ordering principle. To construct this proof, we first find a polynomial g of degree $O(\sqrt{n}log(n))$ such that $\tilde{E}_n[g^2] < 0$. We then show that there is an SOS proof (which in fact uses only polynomial equalities) that $E_{\pi\in S_n}[\pi(g)^2] = \tilde{E}_n[g^2] < 0$.

4.1 Failure of $\tilde{E}_n$

We now show that $\tilde{E}_n$ fails to give valid pseudo-expectation values at degree $O(\sqrt{n}log(n))$.

▶ **Theorem 22.** *For all $n \geq 4$ there exists a polynomial $g(w_1)$ of degree $\lceil \frac{1}{2}\sqrt{n}(log_2(n)+1)\rceil$ such that $\tilde{E}[(z_1 g(w_1))^2] = E_{U_n}[w_1 g^2(w_1)] < 0$.*

Proof. Observe that over the uniform distribution of orderings, w_1 is equally likely to be any integer in $[-1, n-2]$. To make $E_{U_n}[w_1 g^2(w_1)]$ negative, we want $g(w_1)$ to have high magnitude at $w_1 = -1$ and small magnitude on $[1, n-2]$. For this, we can use Chebyshev polynomials. From Wikipedia [18],

▶ **Definition 23.** *The mth Chebyshev polynomial can be expressed as*

1. $T_m(x) = cos(mcos^{-1}(x))$ *if* $|x| \leq 1$
2. $T_m(x) = \frac{1}{2}\left(\left(x+\sqrt{x^2-1}\right)^m + \left(x-\sqrt{x^2-1}\right)^m\right)$ *if* $|x| \geq 1$.

▶ **Theorem 24.** *For all integers $m \geq 0$ and all $x \in [-1,1]$, $|T_m(x)| \leq 1$.*
We now take $g(w_1) = T_m(-1+\frac{2w_1}{n})$ where $m = \lceil \frac{1}{2}\sqrt{n}(log_2(n)+1)\rceil$ and analyze g.

▶ **Lemma 25.** *Taking $g(w_1) = T_m(-1+\frac{2w_1}{n})$ where $m = \lceil \frac{1}{2}\sqrt{n}(log_2(n)+1)\rceil$,*

1. $|g(-1)| \geq n$
2. *For all $w_1 \in [0, n-2]$, $|g(w_1)| \leq 1$.*

Proof. The second statement follows immediately from Theorem 24 as when $w_1 \in [0, n-2]$, $-1+\frac{2w_1}{n} \in [-1,1]$ so $|g(w_1)| = |T_m(-1+\frac{2w_1}{n})| \leq 1$. For the first statement, let $\Delta = \frac{2}{n}$ and observe that when $x = -1-\Delta$,

1. $\sqrt{x^2-1} = \sqrt{(1+\Delta)^2-1} \geq \sqrt{2\Delta}$. Thus, $|x-\sqrt{x^2-1}| \geq 1+\sqrt{2\Delta}$.
2. $x+\sqrt{x^2-1} < 0$ and $x-\sqrt{x^2-1} < 0$ so

$$|T_m(x)| = \frac{1}{2}\left(\left|x+\sqrt{x^2-1}\right|^m + \left|x-\sqrt{x^2-1}\right|^m\right) \geq \frac{(1+\sqrt{2\Delta})^m}{2}.$$

We now use the following proposition.

▶ **Proposition 26.** *For all $y \in [0,1]$ and all $m \geq 0$, $(1+y)^m \geq 2^{ym}$.*

Proof. Observe that $(1+y)^m = \left((1+y)^{\frac{1}{y}}\right)^{ym}$ and if $y \leq 1$ then $(1+y)^{\frac{1}{y}} \geq 2$. ◀

Since $n \geq 4$, $\Delta = \frac{2}{n} \leq \frac{1}{2}$ and $\sqrt{2\Delta} \leq 1$. Applying Proposition 26 with $y = \sqrt{2\Delta}$ and recalling that $m = \lceil \frac{1}{2}\sqrt{n}(log_2(n)+1)\rceil$,

$$|g(-1)| = |T_m(-1-\Delta)| \geq \frac{(1+\sqrt{2\Delta})^m}{2} \geq \frac{2^{\sqrt{2\Delta}m}}{2} = \frac{2^{\frac{2m}{\sqrt{n}}}}{2} \geq 2^{log_2(n)+1}2 = n. \quad ◀$$

We can now complete the proof of Theorem 22. By Lemma 25,

$$E_{U_n}[w_1 {g'}^2(w_1)] \leq \frac{1}{n-1}(-n^2 + \sum_{j=0}^{n-2} j) < 0. \quad ◀$$

4.2 Constructing an SOS proof of infeasibility

We now show that from the failure of $\tilde{E}_n$, we can construct an SOS proof that the ordering principle equations are infeasible.

▶ **Theorem 27.** *If there exists a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_n[g^2] < 0$ then there exists an SOS proof of degree at most $2d+2$ that the ordering principle equations are infeasible.*

Proof. To prove this theorem, we will show that for any monomial $p(\{x_{i,j} : i,j \in [n], i \neq j\})$ of degree at most d, there is a proof of degree at most $2d+2$ that $\frac{1}{n!}\sum_{\pi \in S_n} \pi(p) = \tilde{E}_n[p]$ which uses only polynomial equalities. To prove this, we observe that given arbitrary indices $i_1, \ldots, i_k$, we can split things into cases based on the order of $a_{i_1}, \ldots, a_{i_k}$.

▶ **Lemma 28.** *Given the ordering and transitivity axioms, for all $r \in \mathbb{N}$, tuples of distinct indices $(i_1, \ldots, i_{r+1})$, and permutations $\pi \in S_r$,*

$$\begin{aligned}\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}} &= x_{i_{r+1}i_{\pi(1)}} \prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}} \\ &+ \sum_{k=1}^{r-1} \left(\prod_{j=1}^{k-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right) x_{i_{\pi(k)}i_r} x_{i_r i_{\pi(k+1)}} \left(\prod_{j=k+1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right) \\ &+ \prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}} x_{i_{\pi(r)}i_{r+1}}\end{aligned}$$

and there is a degree $r+1$ proof of this fact which uses only polynomial equalities.

▶ Remark 29. The idea behind this lemma is that we have found the correct ordering for $i_1, \ldots, i_r$ and we are inserting i_{r+1} into the correct place.

Proof. Using the ordering and transitivity axioms,

1. $\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}} = x_{i_{r+1}i_{\pi(1)}} \left(\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right) + x_{i_{\pi(1)}i_{r+1}} \left(\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right)$
2. For all $k \in [r-1]$,

$$\begin{aligned}&x_{i_{\pi(k)}i_{r+1}} \prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}} \\ &= (x_{i_{\pi(k)}i_{r+1}} x_{i_{r+1}i_{\pi(k+1)}} + x_{i_{\pi(k)}i_{r+1}} x_{i_{\pi(k+1)}i_{r+1}}) \left(\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right) \\ &= (x_{i_{\pi(k)}i_{r+1}} x_{i_{r+1}i_{\pi(k+1)}} + x_{i_{\pi(k+1)}i_{r+1}}) \left(\prod_{j=1}^{r-1} x_{i_{\pi(j)}i_{\pi(j+1)}}\right)\end{aligned}$$

   where the second equality follows because of the transitivity axiom

$$x_{i_{\pi(k)}i_{\pi(k+1)}} x_{i_{\pi(k+1)}i_{r+1}} (1 - x_{i_{\pi(k)}i_{r+1}}) = 0$$

   which implies that $x_{i_{\pi(k)}i_{\pi(k+1)}} x_{i_{\pi(k+1)}i_{r+1}} x_{i_{\pi(k)}i_{r+1}} = x_{i_{\pi(k)}i_{\pi(k+1)}} x_{i_{\pi(k+1)}i_{r+1}}$.
3. For all $k \in [r-1]$, we have the transitivity axiom

$$x_{i_{\pi(k)}i_{r+1}} x_{i_{r+1}i_{\pi(k+1)}} (1 - x_{i_{\pi(k)}i_{\pi(k+1)}}) = 0$$

which implies that $x_{i_{\pi(k)}i_{r+1}}x_{i_{r+1}i_{\pi(k+1)}}x_{i_{\pi(k)}i_{\pi(k+1)}} = x_{i_{\pi(k)}i_{r+1}}x_{i_{r+1}i_{\pi(k+1)}}$. Thus,

$$x_{i_{\pi(k)}i_{r+1}}x_{i_{r+1}i_{\pi(k+1)}}\prod_{j=1}^{r-1}x_{i_{\pi(j)}i_{\pi(j+1)}}$$
$$=\left(\prod_{j=1}^{k-1}x_{i_{\pi(j)}i_{\pi(j+1)}}\right)x_{i_{\pi(k)}i_r}x_{i_ri_{\pi(k+1)}}\left(\prod_{j=k+1}^{r-1}x_{i_{\pi(j)}i_{\pi(j+1)}}\right).$$

The result follows by combining all of these equalities. ◀

▶ **Corollary 30.** *Given the ordering and transitivity axioms, for all k and all sets of k distinct indices $\{i_1,\dots,i_k\}$,*

$$1=\sum_{\pi\in S_k}\prod_{j=1}^{k-1}x_{i_{\pi(j)}i_{\pi(j+1)}}$$

and there is a degree $k+1$ proof of this fact which uses only polynomial equalities.

▶ **Corollary 31.** *For any monomial $p(\{x_{i,j}:i,j\in[n],i\neq j\})$ of degree d whose variables contain a total of k indices $i_1,\dots,i_k$, there is a proof of degree at most $d+k+1$ that $\frac{1}{n!}\sum_{\pi\in S_n}\pi(p)=\tilde{E}_n[p]$ which uses only polynomial equalities.*

Proof sketch. By Corollary 30,

$$p=\sum_{\pi\in S_k}\prod_{j=1}^{k-1}x_{i_{\pi(j)}i_{\pi(j+1)}}p$$

and there is a proof of this fact of degree at most $d+k+1$ which uses only polynomial equalities. Using the transitivity axioms, we can prove that

$$\sum_{\pi\in S_k}\prod_{j=1}^{k-1}x_{i_{\pi(j)}i_{\pi(j+1)}}p=\sum_{\pi\in S_k:p=1 \text{ when } x_{i_{\pi(1)}}<\dots<x_{i_{\pi(k)}}}\prod_{j=1}^{k-1}x_{i_{\pi(j)}i_{\pi(j+1)}}.$$

Using Corollary 30 again, this implies that there is a proof of degree at most $d+k+1$ which uses only polynomial equations that

$$\frac{1}{n!}\sum_{\pi\in S_n}\pi(p)=Pr_{\pi\in S_n}[p=1 \text{ when } x_{\pi(1)}<\dots<x_{\pi(n)}]=E_{U_n}[p]=\tilde{E}_n[p].$$ ◀

Now note that given a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_n[g^2]<0$, g^2 is a polynomial of degree at most d in the variables $\{x_{ij}:i,j\in[n],i\neq j\}$ and the variables of every monomial of g^2 contain a total of at most d indices. Thus, there is a proof of degree at most $2d+2$ that $\frac{1}{n!}\sum_{\pi\in S_n}\pi(g^2)=\tilde{E}_n[g^2]<0$ which uses only polynomial equalities and this immediately gives us an SOS proof of degree at most $2d+2$ that the ordering principle equations are infeasible. ◀

Combining Theorem 22 and Theorem 27, we obtain an SOS proof of degree $O(\sqrt{n}log(n))$ that the equations corresponding to the negation of the ordering principle are infeasible, which proves Theorem 1.

5 Lower Bound Overview

Proving the lower bound is surprisingly subtle. We proceed as follows.

▶ **Definition 32.** *We define $\Omega_{n,d}$ to be the distribution on a variable u with support $[0, n-d] \cap \mathbb{Z}$ and the following probabilities:*

$$Pr[u = k] = \frac{\binom{n-k-1}{d-1}}{\binom{n}{d}}.$$

1. In Section 6, we show that to prove our sum of squares lower bound, it is sufficient to show that for all polynomials g_* of degree at most d, for some $d_2 \geq 2d$, $E_{\Omega_{n,d_2}}[(u-1)g_*(u)^2] \geq 0$. Equivalently,

$$\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} k g_*^2(k+1) \geq g_*^2(0).$$

   This reduces the problem to analyzing a distribution on one variable. For the precise statement of this result, see Theorem 33.
2. In Section 7, we observe that an approximation to the above statement is the statement that for some small $\Delta > 0$ (we will take $\Delta = \frac{2d_2}{n}$), taking $g_2(x) = g_*\left(\frac{x}{\Delta} + 1\right)$,

$$\int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx \geq \Delta^2 g_2^2(-\Delta).$$

   We then prove this approximate statement. For the precise statement of this result, see Theorem 50.
3. In Section 8, we analyze the difference between $\Delta \sum_{k=0}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta)$ and $\int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx$ and show that it is small. For the precise statement of this result, see Theorem 67.
4. In Section 9, we analyze the difference between $\Delta \sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} (k\Delta) g_2^2(k\Delta)$ and $\Delta \sum_{k=0}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta)$. For the precise statement of this result, see Theorem 70.

In Section 10, we put all of these pieces together to prove our SOS lower bound.

6 Reducing Checking $\tilde{E}$ to Analyzing a Single-Variable Distribution

Recall that $\Omega_{n,d}$ is the distribution on a variable u with support $[0, n-d] \cap \mathbb{Z}$ and the following probabilities:

$$Pr[u = k] = \frac{\binom{n-k-1}{d-1}}{\binom{n}{d}}.$$

In this section, we show that to check that our candidate pseudo-expectation values $\tilde{E}_{2n}$ are valid, it is sufficient to analyze the distribution $\Omega_{n,d}$. In particular, we prove the following theorem:

▶ **Theorem 33.** *For all $d, d_2, n \in \mathbb{N}$ such that $2d \leq d_2 \leq n$, if there is a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$ then there is a polynomial $g_* : \mathbb{R} \to \mathbb{R}$ of degree at most d such that $E_{\Omega_{n,d_2}}[(u-1)g_*(u)^2] < 0$. Equivalently,*

$$\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} k g_*^2(k+1) < g_*^2(0).$$

To see why this statement is equivalent, observe that

$$E_{\Omega_{n,d_2}}[(u-1)g_*(u)^2] = \sum_{k=0}^{n-d_2} \frac{\binom{n-k-1}{d_2-1}}{\binom{n}{d_2}}(k-1)g_*^2(k)$$
$$= \left(\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n}{d_2}} k g_*^2(k+1)\right) - \frac{\binom{n-1}{d_2-1}}{\binom{n}{d_2}} g_*^2(0).$$

Thus, $E_{\Omega_{n,d_2}}[(u-1)g_*(u)^2] < 0$ if and only if

$$\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n}{d_2}} k g_*^2(k+1) < \frac{\binom{n-1}{d_2-1}}{\binom{n}{d_2}} g_*^2(0).$$

Multiplying both sides of this inequality by $\frac{\binom{n}{d_2}}{\binom{n-1}{d_2-1}}$, this is equivalent to

$$\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} k g_*^2(k+1) < g_*^2(0).$$

In the remainder of this section, we prove this theorem by starting with the polynomial g and constructing the polynomial g_*.

6.1 Distinguished Indices of g

We first use symmetry to argue that we can take g to be symmetric under permutations of all but d distinguished indices. For this, we use Theorem 4.1 in [13], which is essentially implied by Corollary 2.6 of [14].

▶ **Definition 34.** *The index degree of a polynomial g is the maximum number of indices mentioned in any monomial of g.*

▶ **Example 35.** $g = x_{12}x_{13} + x_{45}^4$ has index degree 3 and degree 4.

▶ **Theorem 36.** *If $\tilde{E}$ is a linear map from polynomials to $\mathbb{R}$ which is symmetric with respect to permutations of $[1,n]$ then for any polynomial g, we can write*

$$\tilde{E}[g^2] = \sum_{I \subseteq [1,n], j : |I| \leq indexdeg(g)} \tilde{E}[g_{Ij}^2]$$

where for all I, j,
1. *g_{Ij} is symmetric with respect to permutations of $[1,n] \setminus I$*
2. *$indexdeg(g_{Ij}) \leq indexdeg(g)$ and $deg(g_{Ij}) \leq deg(g)$*
3. *$\forall i \in I, \sum_{\pi \in S_{[1,n]\setminus(I\setminus\{i\})}} \pi(g_{Ij}) = 0$.*

▶ Remark 37. The statement that $deg(g_{Ij}) \leq deg(g)$ is not in Theorem 4.1 as stated in [13] but it follows from the proof.

By Theorem 36, if there is a polynomial g_0 of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g_0^2] < 0$ then there is a polynomial g of degree at most $\frac{d}{2}$ such that
1. $\tilde{E}_{2n}[g^2] < 0$
2. g is symmetric under permutations of $[2n] \setminus I$ for some $I \subseteq [2n]$ such that $|I| \leq indexdeg(g_0) \leq 2deg(g_0) \leq d$,

where $indexdeg(g_0) \leq 2deg(g_0)$ because all of our variables mention at most two indices.

6.2 Decomposing g Based on z_j Variables

We now show that we can choose g to be a polynomial of the form $g = \left(\prod_{j\in A} z_j\right) g_A$ where $A \subseteq [n]$ and g_A is a polynomial in the x_{ij} variables. To do this, just as in Section 3.2, we decompose g as $g = \sum_{A\subseteq[n]:|A|\leq \frac{d}{2}} \left(\prod_{j\in A} z_j\right) g_A$ where each g_A is a polynomial in the x_{ij} variables and observe that

$$\tilde{E}_{2n}[g^2] = \tilde{E}_{2n}\left[\sum_{A,A'\subseteq[n]} \left(\prod_{j\in A} z_j \prod_{j\in A'} z_j\right) g_A g_{A'}\right]$$
$$= \sum_{A\subseteq[2n]} \tilde{E}_{2n}\left[\left(\prod_{j\in A} z_j^2\right) g_A^2\right].$$

If $\tilde{E}_{2n}[g^2] < 0$ then there must be an $A \subseteq [2n]$ such that $\tilde{E}_{2n}[\left(\prod_{j\in A} z_j^2\right) g_A^2] < 0$. Thus, we can take $g = \left(\prod_{j\in A} z_j\right) g_A$. Note that $g = \left(\prod_{j\in A} z_j\right) g_A$ is symmetric under permutations of $[2n] \setminus I'$ where $I' = I \cup A$ and thus $|I'| \leq 2d$.

6.3 Choosing an Ordering on the Distinguished Indices and Changing Variables

We now further decompose $\tilde{E}_{2n}[g^2]$ by observing that for any set of indices $I'' = \{i_1, \ldots, i_m\}$,

$$\tilde{E}_{2n}[g^2] = \tilde{E}_{2n}\left[\left(\sum_{\pi\in S_m} \prod_{j=1}^{m-1} x_{i_{\pi(j)} i_{\pi(j+1)}}\right) g^2\right].$$

Since $\tilde{E}_{2n}[g^2] < 0$, there must be a $\pi \in S_m$ such that $\tilde{E}_{2n}\left[\left(\sum_{\pi\in S_m} \prod_{j=1}^{m-1} x_{i_{\pi(j)} i_{\pi(j+1)}}\right) g^2\right] < 0$. Thus, we can restrict our attention to $\tilde{E}_{2n}\left[\left(\sum_{\pi\in S_m} \prod_{j=1}^{m-1} x_{i_{\pi(j)} i_{\pi(j+1)}}\right) g^2\right]$ which effectively imposes the ordering $x_{i_{\pi(1)}} < \ldots < x_{i_{\pi(m)}}$.

For technical reasons, we take I'' to be $I' = I \cup A$ plus some additional indices so that $|I''| = d_2$. Without loss of generality, we can assume that $I'' = [d_2]$ and π is the identity, giving the ordering $x_1 < x_2 < \ldots < x_{d_2}$.

We now observe that under this ordering, for all $j \in [d_2]$,

$$z_j^2 = (j-2) + \sum_{i\in[2n]\setminus[d_2]} x_{ij}^2.$$

Thus, for all $j \in [2, d_2]$, z_j^2 is a sum of squares so $\left(\prod_{j\in A\setminus\{1\}} z_j^2\right) g_A^2$ is a sum of squares. This implies that $1 \in A$ as otherwise $\tilde{E}_{2n}[g^2] \geq 0$. Following similar logic as before, there is a polynomial $g_{\{1\}}$ in the x_{ij} variables of degree at most $\frac{d}{2} - 1$ such that

$$\tilde{E}_{2n}\left[\left(\prod_{i=1}^{d_2-1} x_{i(i+1)}\right) z_1^2 g_{\{1\}}^2\right] < 0.$$

Now observe that by symmetry, under the ordering $x_1 < x_2 < \ldots < x_{d_2}$, we can express $g_{\{1\}}$ in terms of the following new variables:

▶ **Definition 38.** *For $i \in [d_2] \cup \{0\}$, we define the variable u_i so that*

1. $u_0 = \sum_{j \in [n] \setminus [d_2]} x_{j1}$ *is the number of elements before* a_1.
2. *For* $i \in [d_2 - 1]$, $u_i = \sum_{j \in [n] \setminus [d_2]} x_{ij} x_{j(i+1)}$ *is the number of elements between* a_i *and* a_{i+1}.
3. $u_{d_2} = \sum_{j \in [n] \setminus [d_2]} x_{d_2 j}$ *is the number of elements after* a_{d_2}.

With these new variables,

$$\tilde{E}_{2n}\left[\left(\prod_{i=1}^{d_2-1} x_{i(i+1)}\right) z_1^2 g_{\{1\}}^2\right] = \frac{1}{d_2!} E_{u_0,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\}: \sum_{j=0}^{d_2} u_j = 2n - d_2}[(u_0 - 1) g_{\{1\}}^2(u_0, \ldots, u_{d_2})] < 0$$

where the $\frac{1}{d_2!}$ term appears because the probability of having the ordering $x_1 < x_2 < \ldots < x_{d_2}$ is $\frac{1}{d_2!}$.

6.4 Reducing to a Single Variable

We now complete the proof of Theorem 33 by constructing $g_*(u_0)$ and proving that it has the needed properties. To construct g_*, we take

$$g_*(u_0) = E_{u_1,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\}: \sum_{j=1}^{d_2} u_j = 2n - d_2 - u_0}[g_{\{1\}}(u_0, \ldots, u_{d_2})^2].$$

We first need to check that $g_*(u_0)$ is indeed a polynomial of degree at most d in u_0. This follows from the following lemma:

▶ **Lemma 39.** *For all $d_2 \in \mathbb{N}$ and any polynomial $p(u_1, \ldots, u_{d_2})$ of degree at most d,*

$$E_{u_1,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\}: \sum_{j=1}^{d_2} u_j = n'} p(u_1, \ldots, u_{d_2})$$

is a polynomial of degree at most d in n' (for $n' \in \mathbb{N} \cup \{0\}$).

Proof sketch. We illustrate why this lemma is true by computing these expected values for a few monomials in the variables $\{u_1, \ldots, u_{d_2}\}$. The ideas used in these computations can be generalized to any monomial. The idea is to consider placing $d_2 - 1$ dividing lines among n' labeled balls in a random order.

▶ **Example 40.** With 2 balls and 2 bins, the possibilities are as follows:

1. 12|: Balls 1 and 2 are in the first bin in the order $1, 2$.
2. 21|: Balls 1 and 2 are in the first bin in the order $2, 1$.
3. 1|2: Ball 1 is in the first bin and ball 2 is in the second bin.
4. 2|1: Ball 2 is in the first bin and ball 1 is in the second bin.
5. |12: Balls 1 and 2 are in the second bin in the order $1, 2$.
6. |21: Balls 1 and 2 are in the second bin in the order $2, 1$.

To analyze monomials in the variables $\{u_1, \ldots, u_{d_2}\}$, we write $u_j = \sum_{i=1}^{n'} t_{ij}$ where $t_{ij} = 1$ if ball i is in bin j and $t_{ij} = 0$ otherwise.

1. By symmetry, the probability that a given ball is put into the first bin is $\frac{1}{d_2}$. Thus, $E[u_1] = n' E[t_{i1}] = \frac{n'}{d_2}$.
2. If we consider balls i and j where $i \neq j$, the probability that ball i is put into the first bin is $\frac{1}{d_2}$. If ball i is placed into the first bin, this effectively splits the first bin into two bins, the part before ball i and the part after ball i. For ball j, the probability that it is put into one of these parts is $\frac{2}{d_2+1}$ and the probability that it is put into the second bin is $\frac{1}{d_2+1}$. Thus, $E[t_{i1}t_{j1}] = \frac{2}{d_2(d_2+1)}$ and $E[t_{i1}t_{j2}] = \frac{1}{d_2(d_2+1)}$. This implies that $E[u_1 u_2] = n'(n'-1)E[t_{i1}t_{j2}] = \frac{n'(n'-1)}{d_2(d_2+1)}$ and $E[u_1^2] = n' E[t_{i1}] + n'(n'-1)E[t_{i1}t_{j1}] = \frac{n'}{d_2} + \frac{2n'(n'-1)}{d_2(d_2+1)}$.

Following similar ideas, we can analyze any monomial of degree at most d and show that its expected value is a polynomial in n' of degree at most d. ◀

To complete the proof of Theorem 33, we need one more technical lemma.

▶ **Lemma 41.** *For all $n, k, d_2 \in \mathbb{N}$ such that $k \leq n - d_2$,*

$$\frac{\binom{n-k-1}{d_2-1}}{\binom{n-1}{d_2-1}} \leq \left(\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2.$$

Proof. Observe that for all k', n' such that $0 < k' \leq n'$, $\left(\frac{2n'-k'}{2n'}\right)^2 = 1 - \frac{k'}{n'} + \frac{k'^2}{4n^2} > 1 - \frac{k'}{n'}$.
Now observe that

$$\left(\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2 = \prod_{j=1}^{d_2-1}\left(\frac{2n-k-j}{2n-j}\right)^2 \geq \prod_{j=1}^{d_2-1}\left(\frac{2n-k-2j}{2n-2j}\right)^2$$
$$\geq \prod_{j=1}^{d_2-1}\frac{n-k-j}{n-j} = \frac{\binom{n-k-1}{d_2-1}}{\binom{n-1}{d_2-1}}.$$ ◀

We now complete the proof of Theorem 33. Recall that Ω_{n,d_2} is the distribution on a variable u with support $[0, n-d_2] \cap \mathbb{Z}$ and the following probabilities

$$Pr[u = k] = \frac{\binom{n-k-1}{d_2-1}}{\binom{n}{d_2}}.$$

We have the following facts:

1. $E_{\Omega_{2n,d_2}}[(u-1)g_*(u)] = E_{u_0,\ldots,u_{d_2}\in\mathbb{N}\cup\{0\}:\sum_{j=0}^{d_2} u_j = 2n-d_2}[(u_0-1)g^2_{\{1\}}(u_0,\ldots,g_{d_2})] < 0$
2. For all $u_0 \in [0, 2n-d_2] \cap \mathbb{Z}$, $g_*(u) \geq 0$

▶ Remark 42. Intuitively, g_* should already be a sum of squares. However, we are not sure how to prove this, so we instead show that g_*^2 is sufficient for our purposes.
Since $E_{\Omega_{2n,d_2}}[(u-1)g_*(u)] < 0$,

$$\sum_{k=1}^{2n-d_2}\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n}{d_2}}(k-1)g_*(k) < \frac{\binom{2n-1}{d_2-1}}{\binom{2n}{d_2}}g_*(0)$$

which implies that

$$\sum_{k=1}^{2n-d_2}\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}(k-1)\frac{g_*(k)}{g_*(0)} < 1.$$

In turn, this implies that

$$\sum_{k=1}^{2n-d_2}\left(\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2(k-1)\frac{g_*^2(k)}{g_*^2(0)} \leq \sum_{k=1}^{2n-d_2}\left(\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2(k-1)^2\frac{g_*^2(k)}{g_*^2(0)} < 1.$$

Using Lemma 41,

$$\sum_{k=1}^{n-d_2}\frac{\binom{n-k-1}{d_2-1}}{\binom{n-1}{d_2-1}}(k-1)\frac{g_*^2(k)}{g_*^2(0)} \leq \sum_{k=1}^{2n-d_2}\left(\frac{\binom{2n-k-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2(k-1)\frac{g_*^2(k)}{g_*^2(0)} < 1.$$

Multiplying both sides by $g_*^2(0)$,

$$\sum_{k=1}^{n-d_2}\frac{\binom{n-k-1}{d_2-1}}{\binom{n-1}{d_2-1}}(k-1)g_*^2(k) = \sum_{k=0}^{n-d_2-1}\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}kg_*^2(k+1) < g_*^2(0).$$

This implies that $E_{\Omega_{n,d_2}}[(u-1)g_*^2(u)] < 0$, as needed.

7 Approximate Analysis for Ω_{n,d_2}

To prove our SOS lower bound, we need to show that for any polynomial g_* of degree at most d, $E_{\Omega_{n,d_2}}[(u-1)g_*^2(u)] \geq 0$. Equivalently, we need to show that for any polynomial g_* of degree at most d,

$$\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} k g_*^2(k+1) \geq g_*^2(0).$$

7.1 Approximation by an Integral

The expression $\sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} k g_*^2(k+1)$ is hard to analyze, so we approximate it by an integral. Observe that as long as $k << n$, $kd_2^2 << n^2$ and $k^2 d_2 << n$,

$$\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} = \prod_{j=1}^{d_2-1} \left(1-\frac{k}{n}\right) \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \approx \left(1-\frac{k}{n}\right)^{d_2-1} \approx e^{-\frac{d_2 k}{n}}$$

as

$$1 - \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} = \frac{(n-k)(n-j) - n(n-j-k-1)}{(n-k)(n-j)} = \frac{jk+n}{(n-k)(n-j)}$$

which is small. Taking $\Delta = \frac{d_2}{n}$ and $g_2(x) = g_*\left(\frac{x}{\Delta}+1\right)$, approximately what we need to show is that for all polynomials g_2 of degree at most d,

$$\frac{1}{\Delta} \sum_{j=0}^{\infty} (j\Delta) e^{-j\Delta} g_2^2(j\Delta) \geq g_2^2(-\Delta).$$

In turn, this statement is approximately the same as the statement that for all polynomials g_2 of degree at most d_2,

$$\int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx \geq \Delta^2 g_2^2(-\Delta).$$

In the remainder of this section, we prove this statement when $dd_2 << n$ by analyzing the distribution $\mu(x) = xe^{-x}$. In Sections 8 and 9, we will then analyze how to bound the difference between this statement and the statement which we actually need to prove.

▶ Remark 43. For technical reasons, we will actually take $\Delta = \frac{2d_2}{n}$ rather than $\Delta = \frac{d_2}{n}$. For details, see Section 9.

▶ Remark 44. We might think that the probability that x is much more than $log(n)$ is very small and can be ignored. If so, than using Chebyshev polynomials would cause this statement to fail at degree $\tilde{O}\left(\sqrt{\frac{n}{d}}\right)$ which is much less than $\sqrt{n}$. However, this is not correct. Intuitively, since we are considering polynomials of degree up to d, we should consider the point where $x^d e^{-x}$ becomes negligible, which is when x is a sufficiently large constant times $dlog(d)$.

Based on this, we can only ignore u_0 which are a sufficiently large constant times $dlog(d)\frac{n}{d_2}$. Roughly speaking, we will want to ignore all $u_0 > \frac{n}{4}$, so we want d_2 to be at least $Cdlog(d)$ for some sufficiently large constant C. For details, see Section 9.

7.2 Orthonormal Basis for $\mu(x) = xe^{-x}$

In order to analyze $\int_{x=0}^{\infty} g^2(x)xe^{-x}dx$, it is very useful to find the unique orthonormal basis $\{h_k : k \in \mathbb{N} \cup \{0\}\}$ for the distribution $\mu(x) = xe^{-x}$ such that h_k has degree k and the leading coefficient of h_k is positive. In this subsection, we find this orthonormal basis.

▶ **Definition 45.** *Given two polynomials f and g, we define $f \cdot g = \int_{x=0}^{\infty} f(x)g(x)xe^{-x}dx$.*

▶ **Definition 46.** *We define h_k to be the degree k polynomial such that the leading coefficient of h_k is positive, $h_k \cdot h_k = 1$, and for all $j < k$, $h_j \cdot h_k = 0$.*

▶ **Lemma 47.**

$$h_k(x) = \frac{1}{\sqrt{k!(k+1)!}} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} x^j$$

Proof.

▶ **Proposition 48.** $x^p \cdot x^q = (p+q+1)!$

Computing directly using Gram-Schmidt, the first few polynomials in the orthonormal basis are

1. $h_0 = 1$
2. $h_1 = \frac{1}{\sqrt{2}}(x-2)$
3. $h_2 = \frac{1}{\sqrt{12}}(x^2 - 6x + 6)$
4. $h_3 = \frac{1}{\sqrt{144}}(x^3 - 12x^2 + 36x - 24)$
5. $h_4 = \frac{1}{\sqrt{2880}}(x^4 - 20x^3 + 120x^2 - 240x + 120)$.

To check the general pattern, we need to check that for all $i \in [0, k-1]$, $h_k \cdot x^i = 0$ and $h_k \cdot h_k = 1$. To see this, observe that for all $i \geq 0$,

$$\begin{aligned} h_k \cdot x^i &= \frac{1}{\sqrt{k!(k+1)!}} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} (i+j+1)! \\ &= \frac{1}{\sqrt{k!(k+1)!}} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \binom{i+j+1}{j+1} (k+1)!i!. \end{aligned}$$

Now observe that for all k and all functions $f(j)$,

$$\sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} f(j) = (\Delta^k f)(0)$$

where $(\Delta f)(x) = f(x+1) - f(x)$.

▶ **Proposition 49.** *If $f = x^i$ then $\Delta^k f = 0$ if $i < k$ and $\Delta^k f = k!$ if $i = k$.*

Viewing $\binom{i+j+1}{j+1}$ as a polynomial in j,

$$\binom{i+j+1}{j+1} = \frac{(i+j+1)!}{i!(j+1)!} \frac{j^i}{i!} + \text{ lower order terms}$$

Putting everything together,

1. $h_k \cdot x^i = 0$ whenever $i \leq k$.
2. $h_k \cdot h_k = \frac{1}{\sqrt{k!(k+1)!}}(h_k \cdot x^k) = \frac{k!(k+1)!}{k!(k+1)!}(\Delta^k \binom{k+j+1}{j+1}(0)) = 1$. ◀

7.3 Proof of the Approximate Statement

Now that we have the orthonormal basis for $\mu(x) = xe^{-x}$, we prove the approximate statement we need.

▶ **Theorem 50.** *For all $d \in \mathbb{N}$ and all $\Delta > 0$ such that $10(d+1)^2\Delta^2 e^{2d\Delta} \leq 1$, for any polynomial g_2 of degree at most d,*

$$\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \geq 10\Delta^2 g_2^2(-\Delta).$$

Proof. Given a polynomial g_2 of degree at most d, write $g_2 = \sum_{k=0}^{d} a_k h_k$. Since $\{h_k\}$ is an orthonormal basis for $\mu(x) = xe^{-x}$,

$$\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx = \sum_{k=0}^{d} a_k^2.$$

Using Cauchy Schwarz, we have that

$$\sum_{k=0}^{d} |a_k| \leq \sqrt{\left(\sum_{k=0}^{d} a_k^2\right)\left(\sum_{k=0}^{d} 1\right)} = \sqrt{(d+1)}\sqrt{\sum_{k=0}^{d} a_k^2}$$

which implies that $\sum_{k=0}^{d} a_k^2 \geq \frac{\left(\sum_{k=0}^{d}|a_k|\right)^2}{d+1}$.

In order to upper bound $|g_2(-\Delta)|$, we need to bound $h_k(x)$ near $x = 0$. For this, we use the following lemma:

▶ **Lemma 51.** *For all $k \in \mathbb{N}$ and all $x \in \mathbb{R}$,*

$$|h_k(x)| \leq \sqrt{k+1}e^{k|x|}.$$

Proof. Observe that

$$\begin{aligned} h_k(x) &\leq \frac{1}{\sqrt{k!(k+1)!}} \sum_{j=0}^{k} \binom{k}{j} \frac{(k+1)!}{(j+1)!}|x|^j \\ &\leq \sqrt{k+1}\sum_{j=0}^{k} \frac{(k|x|)^j}{j!(j+1)!} \leq \sqrt{k+1}e^{k|x|}. \end{aligned}$$ ◀

By Lemma 51,

$$|g_2(-\Delta)| \leq \sum_{k=0}^{d} |a_k|\sqrt{k+1}e^{k\Delta} \leq \sqrt{d+1}e^{d\Delta}\sum_{k=0}^{d}|a_k|.$$

Thus, $g_2^2(-\Delta) \leq (d+1)e^{2d\Delta}\left(\sum_{k=0}^{d}|a_k|\right)^2$.

Putting everything together, as long as $10(d+1)^2\Delta^2 e^{2d\Delta} \leq 1$, $\int_{x=0}^{\infty} g^2(x)xe^{-x}dx \geq 10\Delta^2 g^2(-\Delta)$, as needed. ◀

8 Handling Numerical Integration Error

In this section, we show how to bound the difference between $\Delta\sum_{j=0}^{\infty}(j\Delta)e^{-j\Delta}g_2^2(j\Delta)$ and $\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx$.

8.1 Bounding Numerical Integration Error via Higher Derivatives

In this subsection, we describe how the numerical integration error can be bounded using higher derivatives.

▶ **Lemma 52.** *For any $\Delta > 0$ and any differentiable function $f : [0, \infty) \to \mathbb{R}$,*

$$\left| \int_{x=0}^{\infty} f(x)dx - \Delta \sum_{j=0}^{\infty} f(j\Delta) \right| \leq \Delta \int_{x=0}^{\infty} |f'(x)|dx.$$

Proof. This result follows by summing the following proposition over all $j \in \mathcal{N} \cup \{0\}$ and using the fact that $|a + b| \leq |a| + |b|$.

▶ **Proposition 53.** *For all $j \in \mathcal{N} \cup \{0\}$,*

$$\left| \int_{x=j\Delta}^{(j+1)\Delta} (f(x) - f(j\Delta))dx \right| \leq \Delta \int_{x=j\Delta}^{(j+1)\Delta} |f'(x)|dx.$$

Proof. Observe that for all $j \in \mathcal{N} \cup \{0\}$, for all $x \in [j\Delta, (j+1)\Delta]$,

$$|f(x) - f(j\Delta)| \leq \int_{x=j\Delta}^{(j+1)\Delta} |f'(x)|dx.$$

Thus,

$$\left| \int_{x=j\Delta}^{(j+1)\Delta} (f(x) - f(j\Delta))dx \right| \leq \Delta \int_{x=j\Delta}^{(j+1)\Delta} |f'(x)|dx.$$ ◀

◀

Using higher derivatives, we can get better bounds on the error.

▶ **Lemma 54.** *For any $\Delta > 0$ and any twice differentiable function $f : [0, \infty) \to \mathbb{R}$,*

$$\left| \int_{x=0}^{\infty} f(x)dx - \Delta \sum_{j=0}^{\infty} f(j\Delta) + \frac{\Delta}{2} f(0) \right| \leq \Delta^2 \int_{x=0}^{\infty} |f''(x)|dx.$$

Proof. This result follows from summing the following lemma over all $j \in \mathcal{N} \cup \{0\}$ and using the fact that $|a + b| \leq |a| + |b|$.

▶ **Lemma 55.** *For all $j \in \mathcal{N} \cup \{0\}$,*

$$\left| \int_{x=j\Delta}^{(j+1)\Delta} f(x)dx - \frac{\Delta}{2}(f(j\Delta) + f((j+1)\Delta)) \right| \leq \Delta^2 \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|dx.$$

Proof. We prove this lemma using the following estimate of $f(x)$ for $x \in [j\Delta, (j+1)\Delta]$

▶ **Proposition 56.** *For all $x \in [j\Delta, (j+1)\Delta]$,*

$$|f(x) - f(j\Delta) - (x - j\Delta)f'(j\Delta)| \leq (x - j\Delta) \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|dx.$$

Proof. Observe that for all $j \in \mathcal{N} \cup \{0\}$, for all $x \in [j\Delta, (j+1)\Delta]$,

$$|f'(x) - f'(j\Delta)| \leq \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|.$$

Taking the integral of this equation from $j\Delta$ to x and using the fact that $|a+b| \leq |a| + |b|$,

$$|f(x) - f(j\Delta) - (x - j\Delta)f'(j\Delta)| \leq (x - j\Delta) \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|. \quad \blacktriangleleft$$

We now make the following observations:

1. By Proposition 56,

$$|f(j\Delta)+\Delta f'(j\Delta)-f((j+1)\Delta)| = |f((j+1)\Delta)-f(j\Delta)-\Delta f'(j\Delta)| \leq \Delta \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|dx.$$

2. Taking the integral of Proposition 56 from $j\Delta$ to $(j+1)\Delta$ and using the fact that $|a+b| \leq |a| + |b|$,

$$\left| \int_{x=j\Delta}^{(j+1)\Delta} f(x)dx - \Delta f(j\Delta) - \frac{\Delta^2}{2} f'(j\Delta) \right| \leq \frac{\Delta^2}{2} \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|$$

Adding $\frac{\Delta}{2}$ times the first equation to the second equation, we have that

$$\left| \int_{x=j\Delta}^{(j+1)\Delta} f(x)dx - \frac{\Delta}{2}(f(j\Delta) + f((j+1)\Delta)) \right| \leq \Delta^2 \int_{x=j\Delta}^{(j+1)\Delta} |f''(x)|dx$$

as needed. ◀

◀

We now generalize this argument to (t+1)th derivatives.

▶ **Definition 57.** *For all $t \in \mathbb{N}$, we define M_t to be the $(t+1) \times (t+1)$ matrix with entries $(M_t)_{ab} = (b-1)^{(a-1)}$ where $(M_t)_{11} = 1$. Note that M_t is a Vandermonde matrix and is thus invertible.*

▶ **Definition 58.** *For all $t \in \mathbb{N}$, we define v_t to be the vector of length $t+1$ with entries $(v_t)_a = \frac{t^{a-1}}{a}$ and we define $c_t = M_t^{-1} v_t$.*

▶ **Lemma 59.** *For all $t \in \mathbb{N}$, for any $\Delta > 0$ and any function $f : [0, \infty) \to \mathbb{R}$ which can be differentiated $t+1$ times,*

$$\left| \frac{1}{t} \left(\sum_{b=0}^{t-1} \int_{x=b\Delta}^{\infty} f(x)dx \right) - \Delta \sum_{j=0}^{\infty} f(j\Delta) + \Delta \sum_{j=0}^{t-1} \left(\sum_{b=j+1}^{t} (c_t)_{b+1} \right) f(j\Delta) \right|$$
$$\leq (t\Delta)^{t+1} \left(\frac{t}{(t+1)!} + \frac{\sum_{b=1}^{t} |(c_t)_{b+1}|}{t!} \right) \int_{x=0}^{\infty} |f^{(t+1)}(x)|dx.$$

Proof. This result follows from summing the following lemma over all $j \in \mathcal{N} \cup \{0\}$:

▶ **Lemma 60.** *For all $j \in \mathcal{N} \cup \{0\}$,*

$$\left|\frac{1}{t}\int_{x=j\Delta}^{(j+t)\Delta} f(x)dx - \Delta\sum_{b=0}^{t}(c_t)_{b+1}f((j+b)\Delta)\right|$$
$$\leq (t\Delta)^{t+1}\left(\frac{1}{(t+1)!} + \frac{\sum_{b=1}^{t}|(c_t)_{b+1}|}{t(t!)}\right)\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx.$$

Proof. We prove this lemma using the following estimate of $f(x)$ for $x \in [j\Delta, (j+t)\Delta]$.

▶ **Proposition 61.** *For all $x \in [j\Delta, (j+t)\Delta]$,*

$$\left|f(x) - \sum_{a=0}^{t}\frac{(x-j\Delta)^a}{a!}f^{(a)}(j\Delta)\right| \leq \frac{(x-j\Delta)^t}{t!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx.$$

Proof. Observe that for all $j \in \mathcal{N} \cup \{0\}$, for all $x \in [j\Delta, (j+t)\Delta]$,

$$|f^{(t)}(x) - f^{(t)}(j\Delta)| \leq \int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|.$$

Taking the integral of this equation from $j\Delta$ to x t times and using the fact that $|a+b| \leq |a|+|b|$,

$$\left|f(x) - \sum_{a=0}^{t}\frac{(x-j\Delta)^a}{a!}f^{(a)}(j\Delta)\right| \leq \frac{(x-j\Delta)^t}{t!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx. \quad ◀$$

We now make the following observations:

1. By Proposition 61, for all $b \in [t]$,

$$\left|\sum_{a=0}^{t}\frac{(b\Delta)^a}{a!}f^{(a)}(j\Delta) - f(j\Delta+b)\right| \leq \frac{(b\Delta)^t}{t!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|.$$

2. Taking the integral of Proposition 61 from $j\Delta$ to $(j+t)\Delta$ and using the fact that $|a+b| \leq |a|+|b|$,

$$\left|\int_{x=j\Delta}^{(j+t)\Delta}f(x)dx - \sum_{a=0}^{t}\frac{(t\Delta)^{a+1}}{(a+1)!}f^{(a)}(j\Delta)\right| \leq \frac{(t\Delta)^{t+1}}{(t+1)!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx.$$

Adding $(c_t)_{b+1}$ times the first equation for each $b \in [t]$ to $\frac{1}{t}$ times the second equation, we have that

$$\frac{1}{t}\int_{x=j\Delta}^{(j+t)\Delta}f(x)dx - \sum_{a=0}^{t}\Delta^{a+1}\left(\frac{t^a}{(a+1)!} - \sum_{b=1}^{t}\frac{b^a(c_t)_{b+1}}{a!}\right)f^{(a)}(j\Delta) - \sum_{b=1}^{t}\Delta(c_t)_{b+1}f(j\Delta+b)$$
$$\leq \frac{(t\Delta)^{t+1}}{(t+1)!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx + \sum_{b=1}^{t}\frac{\Delta|(c_t)_{b+1}|(b\Delta)^t}{t!}\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx$$
$$\leq (t\Delta)^{t+1}\left(\frac{1}{(t+1)!} + \frac{\sum_{b=1}^{t}|(c_t)_{b+1}|}{t(t!)}\right)\int_{x=j\Delta}^{(j+t)\Delta}|f^{(t+1)}(x)|dx.$$

Thus, it is sufficient to show the following:

1. If $a = 0$ then $\frac{t^a}{(a+1)!} - \sum_{b=1}^{t}\frac{b^a(c_t)_{b+1}}{a!} = (c_t)_1$.
2. If $a \in [t]$ then $\frac{t^a}{(a+1)!} - \sum_{b=1}^{t}\frac{b^a(c_t)_{b+1}}{a!} = 0$.

To see these statements, observe that

$$\sum_{b=0}^{t} b^a (c_t)_{b+1} = \sum_{b=1}^{t+1} (M_t)_{(a+1),b}(c_t)_b = (v_t)_{a+1} = \frac{t^a}{a+1}.$$

Thus, for all $a \in [t] \cup \{0\}$

$$\frac{t^a}{(a+1)!} - \sum_{b=0}^{t} \frac{b^a (c_t)_{b+1}}{a!} = 0.$$

When $a = 0$, $\sum_{b=1}^{t} \frac{b^a (c_t)_{b+1}}{a!} = \sum_{b=0}^{t} \frac{b^a (c_t)_{b+1}}{a!} - (c_t)_1$ so $\frac{t^a}{(a+1)!} - \sum_{b=1}^{t} \frac{b^a (c_t)_{b+1}}{a!} = (c_t)_1$.
When $a > 0$, $\sum_{b=1}^{t} \frac{b^a (c_t)_{b+1}}{a!} = \sum_{b=0}^{t} \frac{b^a (c_t)_{b+1}}{a!}$ so $\frac{t^a}{(a+1)!} - \sum_{b=1}^{t} \frac{b^a (c_t)_{b+1}}{a!} = 0$. ◀

◀

8.2 Bounds on h_k

In order to use our tools, we need bounds on the integrals of the functions h_k.

▶ **Lemma 62.** *For all $j, j' \in \mathbb{N} \cup \{0\}$, $\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| x e^{-x} dx \leq 1$.*

Proof. Observe that

$$\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| x e^{-x} dx \leq \int_{x=0}^{\infty} \frac{h_j^2(x) + h_{j'}^2(x)}{2} x e^{-x} dx = 1.$$ ◀

▶ **Lemma 63.** *For all $j \in \mathbb{N} \cup \{0\}$, $\int_{x=0}^{\infty} h_j^2(x) e^{-x} dx \leq j + 8$.*

Proof. The cases where $j = 0$ and $j = 1$ can be computed directly. For $j > 1$, observe that by Lemma 51,

$$\begin{aligned}
\int_{x=0}^{\infty} h_j^2(x) e^{-x} dx &= \int_{x=0}^{\frac{1}{j}} h_j^2(x) e^{-x} dx + \int_{x=\frac{1}{j}}^{\infty} h_j^2(x) e^{-x} dx \\
&\leq \int_{x=0}^{\frac{1}{j}} (j+1) e^{2jx} e^{-x} dx + j \int_{x=\frac{1}{j}}^{\infty} h_j^2(x) x e^{-x} dx \\
&\leq \frac{(j+1)}{2j-1}(e^2 - 1) + j \leq j + 8.
\end{aligned}$$ ◀

▶ **Corollary 64.** *For all $j, j' \in \mathbb{N} \cup \{0\}$, $\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| e^{-x} dx \leq \sqrt{(j+8)(j'+8)}$.*

Proof. Observe that by Lemma 63,

$$\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| e^{-x} dx \leq \int_{x=0}^{\infty} \frac{\sqrt{\frac{j'+8}{j+8}} h_j^2(x) + \sqrt{\frac{j+8}{j'+8}} h_{j'}^2(x)}{2} e^{-x} dx \leq \sqrt{(j+8)(j'+8)}.$$ ◀

8.3 Derivative of h_k

We also need to analyze what happens when we take the derivative of h_k. Calculating directly, the first few derivatives are:

1. $\frac{d(1)}{dx} = 0$
2. $\frac{d(x-2)}{dx} = 1$
3. $\frac{d(x^2-6x+6)}{dx} = 2x - 6 = 2(x-2) - 2$

4. $\frac{d(x^3-12x^2+36x-24)}{dx} = 3x^2 - 24x + 36 = 3(x^2 - 6x + 6) - 6(x - 2) + 6$
5. $$\frac{d(x^4 - 20x^3 + 120x^2 - 240x + 120)}{dx} = 4x^3 - 60x^2 + 240x - 240$$
$$= 4(x^3 - 12x^2 + 36x - 24) - 12(x^2 - 6x + 6) + 24(x - 2) - 24.$$

The general pattern is as follows:

▶ **Lemma 65.** $h'_k(x) = \sum_{k'=0}^{k-1} (-1)^{k-1-k'} \frac{k!}{k'!} \frac{\sqrt{k'!(k'+1)!}}{\sqrt{k!(k+1)!}} h_j(x)$

Proof. To prove this lemma, we need to show that the derivative of

$$\sqrt{k!(k+1)!}h_k = \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} x^j$$

is

$$\sum_{k'=0}^{k-1} (-1)^{k-1-k'} \frac{k!}{k'!} \left(\sum_{j=0}^{k'} (-1)^{k'-j} \binom{k'}{j} \frac{(k'+1)!}{(j+1)!} x^j \right) = \sum_{k'=0}^{k-1} (-1)^{k-1-k'} \frac{k!}{k'!} \left(\sqrt{k'!(k'+1)!} h_{k'} \right).$$

To prove this, we use the following proposition.

▶ **Proposition 66.** *For all n and all j,*

$$\sum_{k=j}^{n-1} \binom{k}{j} = \binom{n}{j+1}.$$

Proof. Observe that choosing $j+1$ objects out of n objects is equivalent to choosing the position $k+1$ of the last object and then choosing the remaining j objects from the first k objects. ◀

With this proposition in hand, we observe that

$$\sum_{k'=0}^{k-1} (-1)^{k-1-k'} \frac{k!}{k'!} \left(\sum_{j=0}^{k'} (-1)^{k'-j} \binom{k'}{j} \frac{(k'+1)!}{(j+1)!} x^j \right)$$
$$= \sum_{j=0}^{k-1} (-1)^{k-1-j} \frac{k!}{(j+1)!} \left(\sum_{k'=j}^{k-1} (k'+1) \binom{k'}{j} \right) x^j$$
$$= \sum_{j=0}^{k-1} (-1)^{k-1-j} \frac{k!}{j!} \left(\sum_{k'=j}^{k-1} \binom{k'+1}{j+1} \right) x^j$$
$$= \sum_{j=0}^{k-1} (-1)^{k-1-j} \frac{k!}{j!} \binom{k+1}{j+2} x^j$$
$$= \sum_{j=1}^{k} (-1)^{k-j} \frac{k!}{j!} \binom{k+1}{j+1} (jx^{j-1}) = \sum_{j=1}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} (jx^{j-1})$$
$$= \frac{d\left(\sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} x^j \right)}{dx}.$$ ◀

8.4 Bounding the Numerical Integration Error

In this subsection, we use our tools to bound the numerical integration error

$$\Delta \sum_{j=0}^{\infty} (j\Delta) e^{-j\Delta} g_2^2(j\Delta) - \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx.$$

▶ **Theorem 67.** *For all $t \in \mathbb{N}$, there exist constants $C_{t1}, C_{t2} > 0$ such that for all $\Delta > 0$, $d \in \mathbb{N}$, and polynomials g_2 of degree at most d,*

$$\left| \Delta \sum_{j=0}^{\infty} (j\Delta) e^{-j\Delta} g_2^2(j\Delta) - \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx \right| \leq \left(C_{t1} (d\Delta)^2 e^{2td\Delta} + C_{t2} d (d\Delta)^{t+1} \right) \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx.$$

Proof. To prove this, we bound $\int_{x=0}^{\infty} \left| \frac{d^{t+1}\left(g_2^2(x) x e^{-x}\right)}{dx^{t+1}} \right| dx$.

▶ **Lemma 68.** *For all $t, d \in \mathbb{N}$ If $g_2 = \sum_{i=0}^{d} a_i h_i$ is a polynomial of degree at most d then*

$$\int_{x=0}^{\infty} \left| \frac{d^{t+1}\left(g_2(x)^2 x e^{-x}\right)}{dx^{t+1}} \right| dx \leq (t+4)(d+8)(d+1)(3d)^t \left(\sum_{i=0}^{d} a_i^2 \right).$$

Proof. Observe that

$$\frac{d^{t+1}\left(g_2(x)^2 x e^{-x}\right)}{dx^{t+1}} = \sum_{j_1=0}^{t+1} \sum_{j_2=0}^{t+1-j_1} \frac{(t+1)!(-1)^{t+1-j_1-j_2}}{j_1! j_2! (t+1-j_1-j_2)!} \frac{d^{j_1} g_2(x)}{dx^{j_1}} \frac{d^{j_2} g_2(x)}{dx^{j_2}} x e^{-x} + \sum_{j_1=0}^{t} \sum_{j_2=0}^{t-j_1} \frac{(t+1)!(-1)^{t-j_1-j_2}}{j_1! j_2! (t-j_1-j_2)!} \frac{d^{j_1} g_2(x)}{dx^{j_1}} \frac{d^{j_2} g_2(x)}{dx^{j_2}} e^{-x}.$$

By Lemma 65, if $f = \sum_{i=0}^{d} b_i h_i$ is a polynomial of degree at most d then writing $\frac{df}{dx} = \sum_{i=0}^{d-1} b_i' h_i$, $\sum_{i=0}^{d-1} |b_i'| \leq d \sum_{i=0}^{d} |b_i|$. By Lemma 62 and Corollary 64, we have that for all $j, j' \in [0, d]$:

1. For all $j, j' \in \mathbb{N} \cup \{0\}$, $\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| x e^{-x} dx \leq 1$.
2. For all $j, j' \in \mathbb{N} \cup \{0\}$, $\int_{x=0}^{\infty} |h_j(x) h_{j'}(x)| e^{-x} dx \leq \sqrt{(j+8)(j'+8)}$.

Putting these facts together, if $g_2 = \sum_{i=0}^{d} a_i h_i$ then

$$\int_{x=0}^{\infty} \left| \frac{d^{t+1}\left(g_2^2(x) x e^{-x}\right)}{dx^{t+1}} \right| dx \leq \left(3^{t+1} d^{t+1} + (t+1) 3^t d^t (d+8)\right) \left(\sum_{i=0}^{d} |a_i| \right)^2 \leq (t+4)(d+8)(d+1)(3d)^t \left(\sum_{i=0}^{d} a_i^2 \right). \quad \blacktriangleleft$$

With this bound in hand, we now apply Lemma 59 with $f(x) = g_2^2(x) x e^{-x}$. For convenience, we recall the statement of Lemma 59 here:

For all $t \in \mathbb{N}$, for any $\Delta > 0$ and any function $f : [0, \infty) \to \mathbb{R}$ which can be differentiated $t+1$ times,

$$\left| \frac{1}{t} \left(\sum_{b=0}^{t-1} \int_{x=b\Delta}^{\infty} f(x)dx \right) - \Delta \sum_{j=0}^{\infty} f(j\Delta) + \Delta \sum_{j=0}^{t-1} \left(\sum_{b=j+1}^{t} (c_t)_{b+1} \right) f(j\Delta) \right|$$
$$\leq (t\Delta)^{t+1} \left(\frac{t}{(t+1)!} + \frac{\sum_{b=1}^{t} |(c_t)_{b+1}|}{t!} \right) \int_{x=0}^{\infty} |f^{(t+1)}(x)|dx.$$

To use this bound, we need to bound $f(x) = g_2^2(x)xe^{-x}$ when $x \in [0, t\Delta]$.

▶ **Lemma 69.** *If $g_2 = \sum_{i=0}^{d} a_i h_i$ then for all $x \in [0, t\Delta]$,*

$$f(x) = g_2(x)^2 xe^{-x} \leq t\Delta(d+1)^2 e^{2dt\Delta} \left(\sum_{i=0}^{d} a_i^2 \right).$$

Proof. By Lemma 51, for all $x \in \mathbb{R}$ and all $j \in \mathbb{N}$, $|h_j(x)| \leq \sqrt{j+1}e^{j|x|}$. Thus, for all $x \in [0, t\Delta]$,

$$|g_2(x)| = \left| \sum_{i=0}^{d} a_i h_i(x) \right| \leq \sqrt{d+1} e^{dt\Delta} \left(\sum_{i=0}^{d} |a_i| \right)$$

which implies that for all $x \in [0, t\Delta]$

$$g_2(x)^2 xe^{-x} \leq t\Delta(d+1)e^{2dt\Delta} \left(\sum_{i=0}^{d} |a_i| \right)^2 \leq t\Delta(d+1)^2 e^{2dt\Delta} \left(\sum_{i=0}^{d} a_i^2 \right). \quad \blacktriangleleft$$

Using Lemma 69, we make the following observations:

1. $\left| \int_0^{\infty} f(x)dx - \frac{1}{t} \left(\sum_{b=0}^{t-1} \int_{x=b\Delta}^{\infty} f(x)dx \right) \right| \leq (t\Delta)^2(d+1)^2 e^{2dt\Delta} \left(\sum_{i=0}^{d} a_i^2 \right)$
2. $\left| \Delta \sum_{j=0}^{t-1} \left(\sum_{b=j+1}^{t} (c_t)_{b+1} \right) f(j\Delta) \right| \leq (t\Delta)^2(d+1)^2 e^{2dt\Delta} \left(\sum_{b=1}^{t} |(c_t)_{b+1}| \right) \left(\sum_{i=0}^{d} a_i^2 \right)$.

Putting everything together, there exist constants C_{t1} and C_{t2} such that

$$\left| \Delta \sum_{j=0}^{\infty} (j\Delta) e^{-j\Delta} g_2^2(j\Delta) - \int_{x=0}^{\infty} g_2^2(x) xe^{-x} dx \right| \leq \left(C_{t1}(d\Delta)^2 e^{2td\Delta} + C_{t2} d(d\Delta)^{t+1} \right) \left(\sum_{i=0}^{d} a_i^2 \right).$$

Since $\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx = \sum_{i=0}^{d} a_i^2$, the result follows. ◀

9 Handling the Difference Between Distributions

In this section, we prove the following theorem:

▶ **Theorem 70.** *For all $d, d_2, n \in \mathbb{N}$ such that $(4d+2)ln(d_2) + 2ln(20) \leq d_2 \leq \frac{\sqrt{n}}{4}$, for all polynomials g_2 of degree at most d, taking $\Delta = \frac{2d_2}{n}$,*

$$\Delta \sum_{k=0}^{n-d_2-1} (k\Delta) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) \geq \frac{\Delta}{2} \sum_{k=0}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) - \frac{1}{10} \int_{x=0}^{\infty} g_2^2(x) xe^{-x} dx.$$

Proof. To prove this theorem, we prove the following two statements:

1. $\Delta \sum_{k=0}^{\lceil \frac{n}{4} \rceil - 1} \left(\frac{k\Delta}{2}\right) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) \geq \frac{\Delta}{4} \sum_{k=0}^{\lceil \frac{n}{4} \rceil - 1} (k\Delta) e^{-k\Delta} g_2^2(k\Delta)$
2. $\Delta \sum_{k=\lceil \frac{n}{4} \rceil}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) \leq \frac{1}{5} \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx.$

Assuming these two statements, we have that

$$\begin{aligned}
\Delta \sum_{k=0}^{n-d_2-1} (k\Delta) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) &\geq \Delta \sum_{k=0}^{\lceil \frac{n}{4} \rceil - 1} (k\Delta) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) \\
&\geq \frac{\Delta}{2} \sum_{k=0}^{\lceil \frac{n}{4} \rceil - 1} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) \\
&= \frac{\Delta}{2} \left(\sum_{k=0}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) - \sum_{k=\lceil \frac{n}{4} \rceil}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) \right) \\
&\geq \frac{\Delta}{2} \sum_{k=0}^{\infty} (k\Delta) e^{-k\Delta} g_2^2(k\Delta) - \frac{1}{10} \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx.
\end{aligned}$$

We now prove these two statements. The first statement follows immediately from the following lemma:

▶ **Lemma 71.** *For all $k, d_2, n \in \mathbb{N}$ such that $d_2 \leq \frac{\sqrt{n}}{4}$ and $k \leq \frac{n}{4}$, taking $\Delta = \frac{2d_2}{n}$,*

$$\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} \geq \frac{1}{2} e^{-k\Delta}.$$

Proof. Observe that

$$\begin{aligned}
\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} &= \prod_{j=1}^{d_2-1} \frac{n-j-k-1}{n-j} \\
&= \prod_{j=1}^{d_2-1} \left(e^{-\frac{2k}{n}} \cdot \frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}} \cdot \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \right) \\
&\geq e^{-k\Delta} \left(\frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}} \right)^{d_2-1} \left(\prod_{j=1}^{d_2-1} \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \right).
\end{aligned}$$

Thus, to prove this result, it is sufficient to lower bound $\left(\frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}}\right)^{d_2-1}$ and $\prod_{j=1}^{d_2-1} \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}}$.

▶ **Proposition 72.** *For all $k \in \mathbb{N}$ such that $k \leq \frac{n}{4}$, $\frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}} \geq 1$.*

Proof. Observe that for all $x \geq 0$, $e^{-x} \leq 1 - x + \frac{x^2}{2}$. Taking $x = \frac{2k}{n}$, if $k \leq \frac{n}{4}$ then

$$e^{-\frac{2k}{n}} \leq 1 - \frac{2k}{n} + \frac{2k^2}{n^2} \leq 1 - \frac{2k}{n} + \frac{k}{2n} \leq 1 - \frac{k}{n}$$

and thus $\frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}} \geq 1$. ◀

To bound $\prod_{j=1}^{d_2-1} \frac{\frac{n-j-k}{n-j}}{1-\frac{k-1}{n}}$, we prove the following lemma.

▶ **Lemma 73.** *For all $j, k, n \in \mathbb{N}$ such that $j \leq \frac{n}{8}$ and $k \leq \frac{n}{4}$,*

$$\frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \geq \left(1-\frac{2}{n}\right)\left(1-\frac{2jk}{n^2}\right).$$

Proof. Observe that

$$1-\frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} = \frac{(n-k)(n-j)-n(n-j-k-1)}{(n-k)(n-j)} = \frac{jk+n}{(n-k)(n-j)}$$
$$\leq \frac{2jk+\frac{32n}{21}}{n^2} = \frac{2jk}{n^2}+\frac{2}{n}-\frac{11}{21n} \leq \frac{2jk}{n^2}+\frac{2}{n}-\frac{4jk}{n^3}.$$

Rearranging this, we have that

$$\frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \geq 1-\frac{2jk}{n^2}-\frac{2}{n}+\frac{4jk}{n^3}$$

as needed. ◀

Combining this lemma with the following proposition, we have the following corollary.

▶ **Proposition 74.** *For all $x \in [0,1]$ and all $k \in \mathbb{N}$, $(1-x)^k \geq 1-kx$.*

▶ **Corollary 75.** *For all $d_2, k, n \in \mathbb{N}$ such that $d_2 \leq \frac{n}{8}$ and $k \leq \frac{n}{4}$,*

$$\prod_{j=1}^{d_2-1} \frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}} \geq \left(1-\frac{2d_2}{n}\right)\left(1-\frac{2d_2^2k}{n^2}\right).$$

Since $d_2 \leq \frac{\sqrt{n}}{4} \leq \frac{n}{8}$ and $k \leq \frac{n}{4}$, combining Proposition 72 and Corollary 75 with the inequality

$$\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} \geq e^{-k\Delta}\left(\frac{1-\frac{k}{n}}{e^{-\frac{2k}{n}}}\right)^{d_2-1}\left(\prod_{j=1}^{d_2-1}\frac{\frac{n-j-k-1}{n-j}}{1-\frac{k}{n}}\right)$$

we have that

$$\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} \geq e^{-k\Delta}\left(1-\frac{2d_2}{n}\right)\left(1-\frac{2d_2^2k}{n^2}\right) \geq \frac{1}{2}e^{-k\Delta}$$

which completes the proof of Lemma 71. ◀

We now prove the second statement needed to prove Theorem 70.

▶ **Lemma 76.** *For all $d, d_2, n \in \mathbb{N}$ such that $d_2 \geq 4dln(d_2)+2ln(10n)$, for any polynomial g_2 of degree at most d, taking $\Delta = \frac{2d_2}{n}$*

$$\Delta\sum_{k=\lceil\frac{n}{4}\rceil}^{\infty}(k\Delta)e^{-k\Delta}g_2^2(k\Delta) \leq 2n^2\left(\frac{3}{4}\right)^{d_2-1}\left(\frac{4n}{d_2}\right)^d\int_{x=0}^{\infty}g_2^2(x)xe^{-x}dx.$$

Proof. To prove this, we upper bound $|h_k(x)|$ for large x.

▶ **Lemma 77.** *For all $k \in \mathbb{N}$ and all $x \geq 1$, $|h_k(x)| \leq (2x)^k$.*

Proof. Recall that

$$h_k(x) = \frac{1}{\sqrt{k!(k+1)!}} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} \frac{(k+1)!}{(j+1)!} x^j.$$

Now observe that

$$\sum_{j=0}^{k} \binom{k}{j} \frac{1}{(j+1)!} \leq \sum_{j=0}^{k} \frac{k^j}{j!2^j} = e^{\frac{k}{2}}.$$

Thus, for all $k \in \mathbb{N}$ and all $x \geq 1$, $|h_k(x)| \leq \sqrt{k+1} e^{\frac{k}{2}} x^k$.

If $k \geq 5$ then $\sqrt{k+1} e^{\frac{k}{2}} \leq 2^k$ and we are done. For $k \in [1,4]$ we check the polynomials directly.

1. $|h_1(x)| = \left|\frac{1}{\sqrt{2}}(x-2)\right| \leq \max\{\frac{x}{\sqrt{2}}, \frac{1}{\sqrt{2}}\} < 2x$
2. $|h_2(x)| = \left|\frac{1}{\sqrt{12}}(x^2-6x+6)\right| \leq \max\{\frac{x^2}{\sqrt{12}}, \frac{5x}{\sqrt{12}}\} < 4x^2$
3. $|h_3(x)| = \left|\frac{1}{\sqrt{144}}(x^3-12x^2+36x-24)\right| \leq \max\{\frac{25x^3}{\sqrt{144}}, \frac{12x^2}{\sqrt{144}}\} < 8x^3$
4. $|h_4(x)| = \left|\frac{1}{\sqrt{2880}}(x^4-20x^3+120x^2-240x+120)\right| \leq \max\{\frac{101x^4}{\sqrt{2880}}, \frac{139x^3}{\sqrt{2880}}\} < 16x^4.$ ◀

▶ **Corollary 78.** *For all $d \in \mathbb{N}$, if g_2 is a polynomial of degree at most d then for all $y \geq 1$,*

$$g_2^2(y) \leq 2(2y)^{2d} \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx.$$

Proof. Writing $g_2 = \sum_{i=0}^{d} a_i h_i$, we have that $\int_{x=0}^{\infty} g_2^2(x)^2 x e^{-x} dx = \sum_{i=0}^{d} a_i^2$ and

$$\begin{aligned} g_2^2(y) &\leq \sum_{i=0}^{d} \sum_{i'=0}^{d} a_i a_{i'} (2y)^{i+i'} \\ &\leq \sum_{i=0}^{d} \sum_{i'=0}^{d} \left(\frac{1}{2^{d-i'+1}} a_i^2 + \frac{1}{2^{d-i+1}} a_{i'}^2 \right) (2y)^{2d} \\ &\leq \left(\sum_{i=0}^{d} a_i^2 + \sum_{i'=0}^{d} a_{i'}^2 \right) (2y)^{2d} = 2(2y)^{2d} \sum_{i=0}^{d} a_i^2. \end{aligned}$$ ◀

With this bound, we can now prove Lemma 76. By Corollary 78,

$$g_2^2(y) \leq 2(2y)^{2d} \int_{x=0}^{\infty} g_2^2(x) x e^{-x} dx$$

Applying this with $y = k\Delta$, since $d_2 \geq (4d+2)ln(d_2) + 2ln(20)$,

$$\begin{aligned}
&\Delta \sum_{k=\lceil \frac{n}{4} \rceil}^{\infty} (k\Delta)e^{-k\Delta} g_2^2(k\Delta) \leq \Delta \sum_{k=\lceil \frac{n}{4} \rceil}^{\infty} (2k\Delta)e^{-k\Delta}(2k\Delta)^{2d} \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \\
&\leq \left(\int_{x=(\lceil \frac{n}{4} \rceil - 1)\Delta}^{\infty} (2x)^{2d+1} e^{-x} dx \right) \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \\
&\leq 2^{2d+1} \sum_{j=0}^{2d+1} \frac{(2d+1)!}{(2d+1-j)!} \left(\left(\left\lceil \frac{n}{4} \right\rceil - 1 \right) \Delta \right)^{2d+1-j} e^{-(\lceil \frac{n}{4} \rceil - 1)\Delta} \left(\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \right) \\
&\leq 2^{2d+2} \left(\left(\left\lceil \frac{n}{4} \right\rceil - 1 \right) \Delta \right)^{2d+1} e^{-(\lceil \frac{n}{4} \rceil - 1)\Delta} \left(\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \right) \\
&\leq 2e^{\Delta} {d_2}^{2d+1} e^{-\frac{d_2}{2}} \left(\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \right) \\
&\leq 4e^{(2d+1)ln(d_2) - \frac{d_2}{2}} \left(\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \right) \\
&\leq \frac{1}{5} \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.
\end{aligned}$$

where the second inequality holds because $(2x)^{2d+1}e^{-x}$ is a decreasing function whenever $x \geq 2d+1$. ◀

◀

10 Putting Everything Together

In this section, we put everything together to prove our SOS lower bound.

10.1 Lower bounding our sum with an integral

We first combine Theorems 67 and 70 to lower bound our sum with an integral.

▶ **Theorem 79.** *For all $d, d_2, t, n \in \mathbb{N}$, taking $\Delta = \frac{2d_2}{n}$, if the following conditions hold:*

1. $(4d+2)ln(d_2) + 2ln(20) \leq d_2 \leq \frac{\sqrt{n}}{4}$.
2. *Letting C_{t1} and C_{t2} be the constants given by Theorem 67,*

$$C_{t1}(d\Delta)^2 e^{2td\Delta} + C_{t2}d(d\Delta)^{t+1} \leq \frac{1}{2}$$

then for any polynomial g_2 of degree at most d,

$$\Delta \sum_{k=0}^{n-d_2-1} (k\Delta) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) \geq \frac{3}{20} \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.$$

Proof. By Theorem 70, for all $d, d_2, n \in \mathbb{N}$ such that $4dln(d_2) + 2ln(10n) \leq d_2 \leq \frac{\sqrt{n}}{4}$, for all polynomials g_2 of degree at most d, taking $\Delta = \frac{2d_2}{n}$,

$$\Delta \sum_{k=0}^{n-d_2-1} (k\Delta) \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}} g_2^2(k\Delta) \geq \frac{\Delta}{2} \sum_{k=0}^{\infty} (k\Delta)e^{-k\Delta} g_2^2(k\Delta) - \frac{1}{10} \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.$$

By Theorem 67, for all $\Delta > 0$, $d \in \mathbb{N}$, and polynomials g_2 of degree at most d,

$$\begin{aligned}&\left|\Delta\sum_{k=0}^{\infty}(k\Delta)e^{-k\Delta}g_2^2(j\Delta) - \int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx\right| \\ &\leq \left(C_{t1}(d\Delta)^2e^{2td\Delta} + C_{t2}d(d\Delta)^{t+1}\right)\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \\ &\leq \frac{1}{2}\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.\end{aligned}$$

Thus,

$$\Delta\sum_{k=0}^{\infty}(k\Delta)e^{-k\Delta}g_2^2(j\Delta) \geq \frac{1}{2}\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.$$

Combining these statements, $\Delta\sum_{k=0}^{n-d_2-1}(k\Delta)\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}g_2^2(k\Delta) \geq \frac{3}{20}\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx$, as needed. ◀

10.2 Proof of the SOS lower bound

We now prove our SOS lower bound.

▶ **Theorem 80.** *For all $d, d_2, t, n \in \mathbb{N}$, taking $\Delta = \frac{2d_2}{n}$, if the following conditions hold:*

1. $(4d+2)ln(d_2) + 2ln(20) \leq d_2 \leq \frac{\sqrt{n}}{4}$.
2. $10(d+1)^2\Delta^2e^{2d\Delta} \leq 1$.
3. *Letting C_{t1} and C_{t2} be the constants given by Theorem 67,*

$$C_{t1}(d\Delta)^2e^{2td\Delta} + C_{t2}d(d\Delta)^{t+1} \leq \frac{1}{2}$$

then there is no polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$.

Proof. We recall the following results.

1. By Theorem 33, since $2d \leq d_2 \leq n$, if there is a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$ then there is a polynomial $g_* : \mathbb{R} \to \mathbb{R}$ of degree at most d such that $E_{\Omega_{n,d_2}}[(u-1)g_*(u)^2] < 0$. Equivalently,

$$\sum_{k=0}^{n-d_2-1}\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}kg_*^2(k+1) < g_*^2(0)$$

Taking $g_2(x) = g_*\left(\frac{x}{\Delta}+1\right)$,

$$\Delta\sum_{k=0}^{n-d_2-1}\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}(k\Delta)g_2^2(k\Delta) < \Delta^2g_2^2(-\Delta).$$

2. By Theorem 79, under the given conditions,

$$\Delta\sum_{k=0}^{n-d_2-1}(k\Delta)\frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}g_2^2(k\Delta) \geq \frac{3}{20}\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx.$$

3. By Theorem 50, since $10(d+1)^2\Delta^2e^{2d\Delta} \leq 1$, for any polynomial g_2 of degree at most d,

$$\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx \geq 10\Delta^2g_2^2(-\Delta).$$

Putting everything together, if there is a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$ then there is a polynomial $g_2 : \mathbb{R} \to \mathbb{R}$ of degree at most d such that

$$\frac{3}{2}\Delta^2 g_2^2(-\Delta) \leq \Delta \sum_{k=0}^{n-d_2-1} \frac{\binom{n-k-2}{d_2-1}}{\binom{n-1}{d_2-1}}(k\Delta)g_2^2(k\Delta) \leq \frac{3}{20}\int_{x=0}^{\infty} g_2^2(x)xe^{-x}dx < \Delta^2 g_2^2(-\Delta)$$

which is impossible. Thus, there is no polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$. ◀

▶ **Corollary 81.** *For all $\epsilon > 0$, there exists a constant C_ϵ such that for all $n \in \mathbb{N}$, degree $C_\epsilon n^{\frac{1}{2}-\epsilon}$ sum of squares cannot prove the ordering principle on n elements.*

11 Conclusion

In this paper, we analyzed the performance of SOS for proving the ordering principle, showing that SOS requires degree roughly $\sqrt{n}$ to prove the ordering principle on n elements. This shows that in terms of degree, SOS is more powerful than resolution, polynomial caluclus, and the Sherali-Adams hierarchy, but SOS still requires high degree to prove the ordering principle. While this mostly resolves the question of how powerful SOS is for proving the ordering principle, there are several open questions remaining including the following:

1. Can we find a tight example for the size/degree trade-off for SOS which was recently shown by Atserias and Hakoniemi [1]?
2. Can we prove SOS lower bounds for the graph ordering principle on expanders?

References

1 Albert Atserias and Tuomas Hakoniemi. Size-degree trade-offs for sums-of-squares and positivstellensatz proofs. In *Computational Complexity Conference*, 2019.

2 Boaz Barak, Fernando G.S.L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 307–326, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2213977.2214006.

3 Boaz Barak and David Steurer. Sum-of-squares proofs and the quest toward optimal algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:59, 2014.

4 Boaz Barak and David Steurer. Proofs, beliefs, and algorithms through the lens of sum-of-squares, 2016. URL: https://www.sumofsquares.org/public/index.html.

5 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, March 2001. doi:10.1145/375827.375835.

6 Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Comput. Complex.*, 10(4):261–276, May 2002. doi:10.1007/s000370100000.

7 Nicola Galesi and Massimo Lauria. Optimality of size-degree tradeoffs for polynomial calculus. *ACM Trans. Comput. Logic*, 12(1), November 2010. doi:10.1145/1838552.1838556.

8 Russell Impagliazzo, Pavel Pudlák, and Jirí Sgall. Lower bounds for the polynomial calculus and the gröbner basis algorithm. *computational complexity*, 8:127–144, 1999.

9 Pravesh Kothari and David Steurer. Sum-of-squares: proofs, beliefs, and algorithms, 2016. URL: https://sos16.dsteurer.org/.

10 Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Inf.*, 22(3):253–275, August 1985.

11 Massimo Lauria. Dd3013 sum of squares and integer programming relaxations, 2014. URL: http://www.csc.kth.se/~lauria/sos14/.

12 Aaron Potechin. Cmsc 39600 1 (autumn 2018) topics in theoretical computer science: The sum of squares hierarchy, 2018. URL: https://canvas.uchicago.edu/courses/17604.

13 Aaron Potechin. Sum of squares lower bounds from symmetry and a good story. In *ITCS*, 2019.

14 Annie Raymond, James Saunderson, Mohit Singh, and Rekha R. Thomas. Symmetric sums of squares over k-subset hypercubes. *Math. Program.*, 167(2):315–354, February 2018. doi:10.1007/s10107-017-1127-6.

15 Alexander Razborov. Guest column: Proof complexity and beyond. *SIGACT News*, 47(2):66–86, June 2016. doi:10.1145/2951860.2951875.

16 Gilbert Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207:87–97, 1974.

17 Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Inf.*, 33(3):277–280, May 1996. doi:10.1007/s002360050044.

18 Wikipedia. Chebyshev polynomials, Accessed May 30, 2020. URL: https://en.wikipedia.org/wiki/Chebyshev_polynomials.

A Analyzing the Ordering Principle with Boolean Variables

In this appendix, we describe how to modify the ordering principle equations so that they only have Boolean variables. We then describe how to modify the pseudo-expectation values and the SOS lower bound proof for these equations.

A.1 Equations for the ordering principle with Boolean auxiliary variables

To encode the negation of the ordering principle using only Boolean variables, we simply replace each z_j^2 with a sum of squares of Boolean auxiliary variables. This gives us the following equations for the negation of the ordering principle:

1. We have variables x_{ij} where we want that $x_{ij} = 1$ if $a_i < a_j$ and $x_{ij} = 0$ if $a_i > a_j$. We also have auxiliary variables $\{z_{jk} : j \in [n], k \in [m]\}$ where $m \geq n - 2$.
2. $\forall i \neq j, x_{ij}^2 = x_{ij}$ and $\forall j \in [n], \forall k \in [m], z_{jk}^2 = z_{jk}$ (variables are Boolean)
3. $\forall i \neq j, x_{ij} = 1 - x_{ji}$ (ordering)
4. For all distinct i, j, k, $x_{ij}x_{jk}(1 - x_{ik}) = 0$ (transitivity)
5. $\forall j, \sum_{i \neq j} x_{ij} = 1 + \sum_{k=1}^{m} z_{jk}^2$ (for all $j \in [n]$, a_j is not the minimum element of $\{a_1, \ldots, a_n\}$)

A.2 Pseudo-expectation values with Boolean auxiliary variables

In order to give pseudo-expectation values for these equations, we need to give pseudo-expectation values for polynomials involving the auxiliary variables. The idea for this is as follows. Letting $w_j = \left(\sum_{i \neq j} x_{ij}\right) - 1$, we want that w_j of the auxiliary variables $\{z_{jk} : k \in [m]\}$ are 1. If $w_j \in [0, m] \cap \mathbb{Z}$, if we choose which of these auxiliary variables are 1 at random,

1. $Pr(z_{j1} = 1) = \frac{w_j}{m}$,
2. $Pr(z_{j1} = 1, z_{j2} = 1) = \frac{w_j(w_j-1)}{m(m-1)}$
3. More generally, for all $K \subseteq [m]$, $Pr(\forall k \in K, z_{jk} = 1) = \frac{\prod_{a=0}^{|K|-1}(w_j - a)}{\prod_{a=0}^{|K|-1}(m-a)}$.

Note that these expressions are still defined for other w_j including $w_j = -1$ (though in this case they aren't actual probabilities over a distribution of solutions). Based on this, we have the following candidate pseudo-expectation values:

▶ **Definition 82** (Candidate pseudo-expectation values with Boolean auxiliary varialbes)**.**

1. *For all polynomials* $p(\{x_{ij} : i, j \in [n], i \neq j\})$*, we take* $\tilde{E}_n[p] = E_{U_n}[p]$*.*
2. *For all* $j \in [n]$*, for all* $K \subseteq [m]$ *and all polynomials* p *which do not contain any of the auxiliary variables* $\{z_{jk} : k \in [m]\}$*, we take*

$$\tilde{E}\left[\left(\prod_{k\in K} z_{jk}\right) p\right] = \frac{\tilde{E}\left[\left(\prod_{a=0}^{|K|-1}(w_j - a)\right) p\right]}{\prod_{a=0}^{|K|-1}(m-a)}.$$

A.3 Reducing to one variable with Boolean auxiliary variables

Unfortunately, our lower bound for the ordering principle equations in Definition 9 does not directly imply a lower bound for the ordering principle equations with Boolean auxiliary variables. That said, we can still reduce the problem to one variable by using the same techniques we used to prove Theorem 33. The resulting theorem is similar but not quite the same as Theorem 33.

▶ **Theorem 83.** *For all* $d, d_2, n, m \in \mathbb{N}$ *such that* $2d \leq d_2 \leq n$ *and* $m \geq 15nd$*, if there is a polynomial* g *of degree at most* $\frac{d}{2}$ *such that* $\tilde{E}_{2n}[g^2] < 0$ *then there is a polynomial* $g_* : \mathbb{R} \to \mathbb{R}$ *of degree at most* d *and a* $j \in [d]$ *such that*

$$\sum_{u=1}^{n-d_2} \frac{\binom{n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}} \left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right) g_*^2(u) < 1.2 g_*^2(0).$$

Proof sketch. Having Boolean auxiliary variables affects each part of the proof of Theorem 33 as follows:

1. Since the equations and pseudo-expectation values are still symmetric under permutations of $[2n]$, the argument in Section 6.1 that we can reduce to the case when g is symmetric under permutations of $[2n] \setminus I$ for some subset $I \subseteq [2n]$ where $|I| \leq d$ still applies.
2. In Section 6.2, we decomposed $\tilde{E}_{2n}[g^2]$ as

$$\tilde{E}_{2n}[g^2] = \sum_{A\subseteq[2n]} \tilde{E}_{2n}\left[\left(\prod_{j\in A} z_j^2\right) g_A^2\right].$$

Here we can do a similar decomposition but it is somewhat more complicated.

▶ **Definition 84.** *Given a* $j \in [n]$ *and a nonempty* $K \subseteq [m]$*, define*

$$y_{jK} = \prod_{k\in K} z_{jk} - \frac{\prod_{a=0}^{|K|-1}(w_j - a)}{\prod_{a=0}^{|K|-1}(m-a)}.$$

▶ **Proposition 85.** *For any* $j \in [2n]$*, any nonempty* $K \subseteq [m]$*, and any polynomial* p *which does not depend on the auxiliary variables* $\{z_{jk} : k \in [m]\}$*,* $\tilde{E}_{2n}[y_{jK}p] = 0$

With this proposition in mind, we decompose g as $g = \sum_{A\subseteq[2n]} g_A$ where

$$g_A = \sum_{\{K_j : j\in A\}} \left(\prod_{j\in A} y_{jK_j}\right) p_{\{K_j : j\in A\}} (\text{where each } K_j \text{ is a nonempty subset of } [m])$$

where each K_j is a nonempty subset of $[m]$. With this decomposition, we have that

$$\tilde{E}_{2n}[g^2] = \sum_{A\subseteq[2n]} \tilde{E}_{2n}[g_A^2].$$

Note that unlike before, here we have the auxiliary variables be part of g_A. That said, this allows us to assume that there are no auxiliary variables z_{jk} where $j \notin A$ and that everything is symmetric under permutations of $[2n] \setminus I'$ where $|I'| \leq 2d$.

3. In Section 6.3, we restricted ourselves to a single ordering for the distinguished indices and expressed everything in terms of the new variables $u_0, \ldots, u_{d_2}$. We can still do this with Boolean auxiliary variables, but this no longer removes all of the auxiliary variables. What we get is a polynomial $g_{\{1\}}(u_0, \ldots, u_{d_2}, \{z_{j'k} : j' \in [d_2]\})$ of degree at most $\frac{d}{2}$ such that

$$\tilde{E}_{2n}\left[\left(\prod_{i=1}^{d_2-1} x_{i(i+1)}\right) g_{\{1\}}^2\right]$$
$$= \frac{1}{d_2!} E_{u_0,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\} : \sum_{j=0}^{d_2} u_j = 2n-d_2} \left[\tilde{E'}_{u_0,\ldots,u_{d_2}}[g_{\{1\}}^2(u_0, \ldots, u_{d_2}, \{z_{j'k} : j' \in [d_2]\})]\right] < 0$$

where $\tilde{E'}_{u_0,\ldots,u_{d_2}}$ gives the pseudo-expectation values of the auxiliary variables for given values of $u_0, \ldots, u_{d_2}$.

4. In Section 6.4, we took

$$g_*(u_0) = E_{u_1,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\} : \sum_{j=1}^{d_2} u_j = 2n-d_2-u_0} [g_{\{1\}}(u_0, \ldots, u_{d_2})^2].$$

Before we can do this here, we need to remove the auxiliary variables $\{z_{1k} : k \in [m]\}$. We can do this as follows:

   a. Observe that looking at the auxiliary variables $\{z_{1k} : k \in [m]\}$, $\tilde{E'}_{u_0,\ldots,u_{d_2}}$ (and thus $\tilde{E}_{2n}$) is symmetric under permutations of $[m]$. Using Theorem 36, we can assume that $g_{\{1\}}$ is symmetric (as far as the auxiliary variables $\{z_{1k} : k \in [m]\}$ are concerned) under permutations of $[m] \setminus K$ for some $K \subseteq [m]$ where $|K| \leq d$.
   b. Breaking things into cases based on the values of the auxiliary variables $\{z_{1k} : k \in K\}$, we can assume that

$$g_{\{1\}}(u_0, \ldots, u_{d_2}, \{z_{j'k} : j' \in [d_2]\}) =$$
$$\left(\prod_{k \in K_1} z_{1k}\right)\left(\prod_{k \in K_2} (1 - z_{1k})\right) p_{\{1\}}(u_0, \ldots, u_{d_2}, \{z_{j'k} : j' \in [2, d_2]\})$$

   for some $K_1, K_2 \subseteq [m]$ such that $K_1 \cap K_2 = \emptyset$ and $|K_1 \cup K_2| \leq d$.

We now take

$$g_* = E_{u_1,\ldots,u_{d_2} \in \mathbb{N} \cup \{0\} : \sum_{j=1}^{d_2} u_j = 2n-d_2-u_0} \left[\tilde{E'}_{u_0,\ldots,u_{d_2}}[p_{\{1\}}^2(u_0, \ldots, u_{d_2}, \{z_{j'k} : j' \in [2, d_2]\})]\right]$$

and we have that

   a. $g_*(u_0)$ is a polynomial of degree at most d in u_0.
   b. For all $u_0 \in [0, 2n - d_2] \cap \mathbb{Z}$, $g_*(u_0) \geq 0$.
   c. $E_{\Omega_{2n,d_2}}\left[\left(\prod_{a=1}^{|K_1|}(u-a)\right)\left(\prod_{a=1}^{|K_2|}(m+2-u-a)\right) g_*(u)\right] < 0$.

Equivalently, taking $j = |K_1|$ and $j_2 = |K_2|$,

$$\sum_{u=0}^{2n-d_2} \frac{\binom{2n-u-1}{d_2-1}}{\binom{2n}{d_2}} \left(\prod_{a=1}^{j}(u-a)\right)\left(\prod_{a=1}^{j_2}(m+2-u-a)\right) g_*(u) < 0.$$

Manipulating this gives

$$\sum_{u=1}^{2n-d_2} \frac{\binom{2n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}} \left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right)\left(\prod_{a=1}^{j_2}\frac{m+2-u-a}{m+2-a}\right)\left(\frac{g_*(u)}{g_*(0)}\right) < 1,$$

which implies that

$$\sum_{u=1}^{2n-d_2} \left(\frac{\binom{2n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2 \left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right)^2 \left(\prod_{a=1}^{j_2}\frac{m+2-u-a}{m+2-a}\right)^2 \left(\frac{g_*(u)}{g_*(0)}\right)^2 < 1.$$

We now make the following observations:

a. By Lemma 41, for all $u \in \mathbb{N}$ such that $u \leq n - d_2$, $\left(\frac{\binom{2n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}}\right)^2 \geq \frac{\binom{n-u-1}{d_2-1}}{\binom{n-1}{d_2-1}}$.

b. For all $u \in \mathbb{N}$, $\left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right)^2 \geq \frac{\prod_{a=1}^{j}(u-a)}{j!}$.

c. Since $m \geq 15nd$, for all $u \in \mathbb{N}$ such that $u \leq n - d_2$,

$$\left(\prod_{a=1}^{j_2}\frac{m+2-u-a}{m+2-a}\right)^2 \geq \left(\frac{14nd-n}{14nd}\right)^{2d} \geq \frac{12}{14}.$$

Putting everything together,

$$\sum_{u=1}^{n-d_2} \frac{\binom{n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}} \left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right) g_*^2(u) < 1.2 g_*^2(0)$$

as needed. ◀

A.4 SOS lower bound with Boolean auxiliary variables

When we have Boolean auxiliary variables, our SOS lower bound is modified as follows:

▶ **Theorem 86.** *For all $d, d_2, t, n, m \in \mathbb{N}$ such that $m \geq 15nd$, if the following conditions hold for all $j \in [d]$:*

1. $(4d+2)ln(d_2) + 2ln(20) \leq d_2 \leq \frac{\sqrt{n'}}{4}$
2. $\frac{(n-1)!(n'-d_2)!}{(n'-1)!(n-d_2)!\binom{2j-1}{j}}\Delta^2(d+1)^2 e^{2d(2j-1)\Delta} \leq \frac{1}{10}$
3. *Letting C_{t1} and C_{t2} be the constants given by Theorem 67,*

$$C_{t1}(d\Delta)^2 e^{2td\Delta} + C_{t2} d(d\Delta)^{t+1} \leq \frac{1}{2}$$

where $n' = n - 2d + 2$ and $\Delta = \frac{2d_2}{n'}$ then there is no polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$.

▶ Remark 87. We believe the condition on m is an artefact of the proof and that we should have essentially the same lower bound as long as $m \geq n - 2$, though proving this would require modifying the analysis further.

Proof. Assume there is a polynomial g of degree at most $\frac{d}{2}$ such that $\tilde{E}_{2n}[g^2] < 0$. By Theorem 83, since $2d \leq d_2 \leq n$, there is a polynomial $g_* : \mathbb{R} \to \mathbb{R}$ of degree at most d and a $j \in [d]$ such that

$$\sum_{u=1}^{n-d_2} \frac{\binom{n-u-1}{d_2-1}}{\binom{2n-1}{d_2-1}} \left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right) g_*^2(u) < 1.2 g_*^2(0).$$

We transform this left side of this equation into the same form as the left hand side of Theorem 79 using the following lemma.

▶ **Lemma 88.** *For all* $j, u \in \mathbb{N}$, $\frac{1}{j!}\left(\prod_{a=1}^{j}(u-a)\right) \geq \binom{2j-1}{j}(u-2j+1)$.

Proof. We prove this lemma by induction. When $u \leq 2j-1$, the result is trivial. When $u = 2j$,

$$\frac{1}{j!}\left(\prod_{a=1}^{j}(u-a)\right) = \binom{2j-1}{j} = \binom{2j-1}{j}(u-2j+1).$$

Now assume the result is true for $u = k$ where $k \geq 2j$ and consider the case when $u = k+1$. By the inductive hypothesis,

$$\begin{aligned}\frac{1}{j!}\left(\prod_{a=1}^{j}((k+1)-a)\right) &= \frac{k}{k-j}\left(\frac{1}{j!}\prod_{a=1}^{j}(k-a)\right) \geq \frac{k}{k-j}\left(\binom{2j-1}{j-1}(k-2j+1)\right)\\ &\geq \frac{k-2j+2}{k-2j+1}\left(\binom{2j-1}{j-1}(k-2j+1)\right) = \binom{2j-1}{j}(k-2j+2).\end{aligned}$$ ◀

Applying Lemma 88, we have that

$$\begin{aligned}1.2g_*^2(0) > \sum_{u=1}^{n-d_2}\frac{\binom{n-u-1}{d_2-1}}{\binom{n-1}{d_2-1}}\left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right)g_*^2(u) &\geq \sum_{u=2j-1}^{n-d_2}\frac{\binom{n-u-1}{d_2-1}}{\binom{n-1}{d_2-1}}\left(\frac{\prod_{a=1}^{j}(u-a)}{j!}\right)g_*^2(u)\\ &\geq \sum_{u=2j-1}^{n-d_2}\frac{\binom{n-u-1}{d_2-1}}{\binom{n-1}{d_2-1}}\binom{2j-1}{j}(u-2j+1)g_*^2(u).\end{aligned}$$

Taking $k = u-2j+1$, $n' = n-2j+2$, $\Delta = \frac{2d_2}{n'}$, and $g_2(x) = g_*\left(\frac{x}{\Delta}+2j-1\right)$,

$$\frac{1.2(n-1)!(n'-d_2)!}{(n'-1)!(n-d_2)!\binom{2j-1}{j}}\Delta^2 g_2^2(-(2j-1)\Delta) > \Delta\sum_{k=0}^{n'-d_2-1}\frac{\binom{n'-k-2}{d_2-1}}{\binom{n'-1}{d_2-1}}(k\Delta)g_2^2(k\Delta)$$

By Theorem 79, under the given conditions,

$$\Delta\sum_{k=0}^{n'-d_2-1}\frac{\binom{n'-k-2}{d_2-1}}{\binom{n'-1}{d_2-1}}(k\Delta)g_2^2(k\Delta) \geq \frac{3}{20}\int_{x=0}^{\infty}g_2^2(x)xe^{-x}dx.$$

Thus,

$$\frac{3}{20}\int_{x=0}^{\infty}g_2^2(x)xe^{-x}dx < \frac{1.2(n-1)!(n'-d_2)!}{(n'-1)!(n-d_2)!\binom{2j-1}{j}}\Delta^2 g_2^2(-(2j-1)\Delta).$$

Decomposing g_2 as $g_2 = \sum_{i=0}^{d} c_i h_i$, observe that

1. $\frac{3}{20}\int_{x=0}^{\infty}g_2^2(x)xe^{-x}dx = \frac{3}{20}\left(\sum_{i=0}^{d}c_i^2\right)$.
2. By Cauchy-Schwarz,

$$g_2^2(-(2j-1)\Delta) = \left(\sum_{i=0}^{d}c_i h_i(-(2j-1)\Delta)\right)^2 \leq \left(\sum_{i=0}^{d}c_i^2\right)\left(\sum_{i=0}^{d}h_i(-(2j-1)\Delta)^2\right).$$

By Lemma 51, for all $i \in \mathbb{N}$ and all $x \in \mathbb{R}$, $|h_i(x)| \leq \sqrt{i+1}e^{i|x|}$. Thus,

$$g_2^2(-(2j-1)\Delta) \leq (d+1)^2 e^{2d(2j-1)\Delta}.$$

Putting these pieces together,

$$\frac{3}{20} < \frac{1.2(n-1)!(n'-d_2)!}{(n'-1)!(n-d_2)!\binom{2j-1}{j}}\Delta^2(d+1)^2e^{2d(2j-1)\Delta}.$$

However, $\frac{(n-1)!(n'-d_2)!}{(n'-1)!(n-d_2)!\binom{2j-1}{j}}\Delta^2(d+1)^2e^{2d(2j-1)\Delta} \leq \frac{1}{10}$ so this gives $\frac{3}{20} = .15 < .12$, which is a contradiction. ◀