37th Computational Complexity Conference

CCC 2022, July 20–23, 2022, Philadelphia, PA, USA

Edited by

Shachar Lovett

Editors

Shachar Lovett
University of California San Diego, La Jolla, CA, US
slovett@cs.ucsd.edu

ACM Classification 2012
Theory of computation

ISBN 978-3-95977-241-9

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/978-3-95977-241-9.

Publication date
July, 2022

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.2022.0

ISBN 978-3-95977-241-9 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

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

Papers

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

Preface

The papers in this volume were accepted for presentation at the 37th Computational Complexity Conference (CCC 2022), held between July 20–23, 2021, in Philadelphia, USA. 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 89 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 for their advice and assistance; the Local Arrangements Committee chair Anindya De; Dean Doron, Rahul Santhanam, and Gilles Zemor for their invited talks; and Michael Wagner for coordinating the production of these proceedings.

Shachar Lovett
Program Committee Chair, on behalf of the Program Committee

Awards

The program committee of the 37th Computational Complexity Conference is very pleased to present the **Best Paper Award** to Scott Aaronson, Devon Ingram and William Kretschmer for their paper

The Acrobatics of BQP.

The **Best Student Paper Award** is shared between Xin Lyu for his paper

Improved Pseudorandom Generators for AC^0 Circuits

and Oliver Korten for his paper

Derandomization from Time-Space Tradeoffs.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

Conference Organization

Program Committee

Benny Applebaum, Tel Aviv University
Eshan Chattopadhyay, Cornell University
Gil Cohen, Tel Aviv University
Johan Håstad, KTH
Antonina Kolokolova, Memorial University of Newfoundland
Pravesh Kothari, CMU
Mrinal Kumar, IIT Bombay
Marvin Kunnemann, ETH Zurich
Shachar Lovett (Chair), UC San Diego
Nitin Saxena, IIT Kanpur
Osamu Watanabe, Tokyo Institute of Technology
John Wright, UT Austin

Local Arrangements Committee

Anindya De (Chair), University of Pennsylvania

Board of Trustees

Amit Chakrabarti, Dartmouth College
Venkatesan Guruswami, Carnegie Mellon University
Valentine Kabanets (President), Simon Fraser University
Michal Koucký, Charles University
Nutan Limaye, IIT Bombay
Meena Mahajan, The Institute of Mathematical Sciences
Pierre McKenzie, Université de Montréal
Ashwin Nayak, University of Waterloo
Ronen Shaltiel, University of Haifa
Ryan Williams, Massachusetts Institute of Technology

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
COMPUTATIONAL COMPLEXITY CONFERENCE

External Reviewers

Josh Alman
Per Austrin
Vikraman Arvind
Greg Bodwin
Andrej Bogdanov
Joshua Brakensiek
Cs Bhargav
Michael Chapman
Abhranil Chatterjee
Marco Carmosino
Radu Curticapean
Sagnik Dutta
Uriel Feige
Mohit Jayanti Gurumukhani
Jesse Goodman
Sumanta Ghosh
Pavel Hubáček
Kaave Hosseini
Hamed Hatami
Shuichi Hirahara
Sandy Irani
Akhil Jalan
Venkata Koppula
Swastik Kopparty
Esther Kelman
Zhenjian Lu
Jyun-Jie Liao
Jiabao Lin
Or Meir
Guy Moshkovitz
Björn Martinsson
Mikito Nanashima
Anand Kumar Narayanan
Igor Carboni Oliveira
Aditya Potukuchi
Carlos Palazuelos
Shravas Rao
Benjamin Rossman

Robert Andrews
Shyan Akmal
Guy Blanc
Vishwas Bhargava
Jop Briet
Ainesh Bakshi
Pranav Bisht
Suryajith Chillara
Bruno Cavalar
Chi-Ning Chou
Nai-Hui Chia
Prateek Dwivedi
Bill Fefferman
Joel Gärtner
Ankit Garg
Zeyu Guo
Pavel Hrubes
Max Hopkins
William Hoza
Pooya Hatami
Christian Ikenmeyer
Oded Nir Eliran Kachlon
Sajin Koroth
Gautam Kamath
Robert Kleinberg
James Lee
Jiatu Li
Bundit Laekhanukit
Dor Minzer
Nikhil Mande
Navid Nouri
Jakob Nordstrom
Izhar Oppenheim
Shir Peleg
Ján Pich
Youming Qiao
Artur Riazanov
Ramyaa Ramyaa

Kazuyuki Amano
Yoshinori Aono
Miriam Backens
Surender Baswana
Shalev Ben-David
Jonah Brown-Cohen
Matthew Coudron
Lijie Chen
Clément Canonne
Prerona Chatterjee
Anindya De
Noah Fleming
Mrinalkanti Ghosh
Spencer Gordon
Rohit Gurjar
Mika Göös
Nathaniel Harms
Prahladh Harsha
Sangxia Huang
Rahul Ilango
Stasys Jukna
Akinori Kawachi
Alexander Kelley
Akash Kumar
Yuhan Liu
Euiwoong Lee
Han-Hsuan Lin
Bingkai Lin
Noela Müller
Ian Mertz
Chinmay Nirkhe
Oded Nir
Rafael Mendes de Oliveira
Kevin Pratt
Aaron Potechin
Marc Roth
Noga Ron-Zewi
Ron Rothblum

Kilian Risse
Diptajit Roy
Amit Sinhababu
Aleksa Stankovic
Jayalal Sarma
Dmitry Sokolov
Rahul Santhanam
Makrand Sinha
Neil Thapen
Madhur Tulsiani
Ben Lee Volk
Thomas Vidick
Adam Bene Watts
Henry Yuen
C Ramya
Robert Robere
Karteek Sreenivasaiah
Srikanth Srinivasan
Luke Schaeffer
Kv Subrahmanyam
Suhail Sherif
Shashank Srivastava
Anamay Tengse
Avishay Tal
Danny Vagnozzi
Vinod Vaikuntanathan
Xinyu Wu
Jiapeng Zhang
Alexander Razborov
Sushant Sachdeva
Ramprasad Saptharishi
Ori Sberlo
Sunil Simon
Igor Shinkar
Tselil Schramm
Pascal Schweitzer
Suguru Tamaki
Justin Thaler
Ilya Volkovich
Pei Wu
Justin Yirka
Jeroen Zuiddam

The Approximate Degree of Bipartite Perfect Matching

Gal Beniamini ✉
The Hebrew University of Jerusalem, Israel

Abstract

The approximate degree of a Boolean function is the least degree of a real multilinear polynomial approximating it in the ℓ_∞-norm over the Boolean hypercube. We show that the approximate degree of the Bipartite Perfect Matching function, which is the indicator over all bipartite graphs having a perfect matching of order n, is $\widetilde{\Theta}(n^{3/2})$.

The upper bound is obtained by fully characterizing the unique multilinear polynomial representing the Boolean dual of the perfect matching function, over the reals. Crucially, we show that this polynomial has very small ℓ_1-norm – only exponential in $\Theta(n \log n)$. The lower bound follows by bounding the spectral sensitivity of the perfect matching function, which is the spectral radius of its cut-graph on the hypercube [1, 15]. We show that the spectral sensitivity of perfect matching is exactly $\Theta(n^{3/2})$.

2012 ACM Subject Classification Mathematics of computing → Matchings and factors; Theory of computation → Algebraic complexity theory; Theory of computation → Oracles and decision trees

Keywords and phrases Bipartite Perfect Matching, Boolean Functions, Approximate Degree

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.1

Related Version *Full Version*: `https://arxiv.org/abs/2004.14318`

Acknowledgements I would like to thank Noam Nisan and Nati Linial for helpful discussions. I would also like to thank Bruno Loff for pointing out typographical errors in an earlier version.

1 Introduction

The approximate degree of a Boolean function is the least degree of a real polynomial approximating the function in the ℓ_∞-norm over the Boolean cube, to within constant error. Approximate degree is an important complexity measure with applications throughout theoretical computer science. Lower bounds on the approximate degree of a Boolean function imply such bounds on the communication complexity (of a related composed problem) [28, 30], and for its quantum query complexity [2]. For families of Boolean functions, upper bounds on the approximate degree have algorithmic merit, for instance in learning theory [17, 16] and differential privacy [31, 11], and conversely lower bounds imply separations in circuit complexity [20, 27]. For a recent survey, we refer the reader to [10].

In this paper we study the approximate degree of the bipartite perfect matching function. This is the Boolean function representing the *decision problem* of perfect matching – determining whether a given balanced bipartite graph contains a subset of edges in which every vertex is incident to exactly one edge.

▶ **Definition.** *The bipartite perfect matching function* $\mathrm{BPM_n} : \{0,1\}^{n^2} \to \{0,1\}$ *is defined*

$$\mathrm{BPM_n}(x_{1,1}, \ldots, x_{n,n}) = \begin{cases} 1 & \{(i,j) : x_{i,j} = 1\} \text{ has a bipartite perfect matching} \\ 0 & \text{otherwise.} \end{cases}$$

The input bits of $\mathrm{BPM_n}$ select a subset of edges from the complete bipartite graph, and the output bit is set to 1 if and only if the chosen subgraph contains a bipartite perfect matching of order n.

It is well known that any Boolean function can be uniquely and *exactly* represented by a multilinear polynomial over the reals (see [25]). In [3], the unique polynomial representing $\mathrm{BPM_n}$ was characterized, and in particular was shown to have full degree, n^2. Conversely, it is not hard to construct low-degree polynomials *approximating* the perfect matching function, if one allows pointwise errors arbitrarily close to one half. Indeed, the $n \times n$ Permanent implies (by translation and scaling) such a polynomial of total degree n, and approximation error exponentially close to half. Approximate degree is an interpolation between these two settings, wherein we require the errors be bounded by an arbitrary constant *less than half*, say one third. The previous best-known upper bound on the approximate degree of perfect matching was $\mathcal{O}(n^{7/4})$, due to Lin and Lin [18], and no non-trivial lower bound was known.

Our main result is the following bound[1], which is tight up to low order terms.

▶ **Theorem 1** (The Approximate Degree Bipartite Perfect Matching)**.** *For every $n \in \mathbb{N}$, the approximate degree of the bipartite perfect matching function is:*

$$\widetilde{\deg}\left(\mathrm{BPM_n}\right) = \widetilde{\Theta}\left(n^{3/2}\right).$$

Most known techniques for bounding approximate degree are applicable only to functions which are either *symmetric* or *block-composed* (with some recent notable exceptions, e.g. [8, 9]). The perfect matching function falls into neither category, and is thus not amenable to standard techniques.

Our upper bound follows by investigating the "Boolean Dual" function of bipartite perfect matching: $\mathrm{BPM_n^\star}(x_{1,1}, \ldots, x_{n,n}) = 1 - \mathrm{BPM_n}(1 - x_{1,1}, \ldots, 1 - x_{n,n})$. In this representation, we reverse the roles of the symbols 0 and 1. Concretely, for any input graph, the dual function $\mathrm{BPM_n^\star}$ outputs 1 if and only if the *complement* of the graph *does not* contain a bipartite perfect matching. Equivalently, by Hall's Marriage Theorem, the output is 1 if and only if the input graph *contains* a biclique over $n + 1$ vertices.

To present our characterization of the dual, let us introduce some notation. A balanced bipartite graph is said to be *totally ordered*, if there exists an ordering of its left vertices such that their neighbour sets form a chain with respect to inclusion, i.e. $N(a_1) \subseteq N(a_2) \subseteq \cdots \subseteq N(a_n)$. We associate with every totally ordered graph a "representing sequence", which encodes its biadjacency matrix up to permutations over both bipartitions. To construct this sequence, consider the automorphism which sorts the left and right vertices in descending order of degree. This yields a graph whose biadjacency matrix consists of a monotonically increasing sequence of *blocks*, which we succinctly describe using a list of pairs of integers, describing the width and height of each such block. By way of example, the biclique $K_{s,t} \subseteq K_{n,n}$ is an ordered graph whose biadjacency matrix consists of two blocks; the first s left vertices are all adjacent to the first t vertices on the right, and the remainder are all isolated.

Our result is the following *complete characterization* of the unique polynomial representing $\mathrm{BPM_n^\star}$ over the reals, thereby resolving an open question of [3].

▶ **Theorem 2** (The Dual Polynomial of Bipartite Perfect Matching)**.**

$$\mathrm{BPM_n^\star}(x_{1,1}, \ldots, x_{n,n}) = \sum_{G \subseteq K_{n,n}} a_G^\star \prod_{(i,j) \in E(G)} x_{i,j}$$

[1] The same bound also holds even for approximations with *exponentially small* error, see Section 4.

- *If G is not totally ordered, then $a^\star_G = 0$.*
- *Otherwise:*

$$a^\star_G = \binom{n - k_{t-1} - 1}{n - d_t} \cdot \prod_{i=1}^{t-1} f\left(d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\right)$$

where $0 \leq d_1 < d_2 < \cdots < d_t \leq n$ and $0 = k_0 < k_1 < k_2 < \cdots < k_t = n$ form the representing sequence of G, and $f : \mathbb{Z}^3 \to \mathbb{Z}$ is defined:

$$f(n,d,k) = \begin{cases} \binom{n-1}{k}, & d \leq 0 \\ -\binom{n-d-1}{k-d}\binom{k-1}{d-1}, & d > 0. \end{cases}$$

This characterization allows us to deduce that the ℓ_1-norm of $\mathrm{BPM}^\star_\mathrm{n}$ (i.e., the sum of the magnitudes of its coefficients) is very small – only exponential in $\Theta(n \log n)$. The approximate degree upper bound then follows via two observations. Firstly, we relate the approximate degree of any Boolean function and its dual. Secondly, we show that any Boolean function whose representation over the $\{0,1\}$-basis has low ℓ_1-norm, can be efficiently approximated in the ℓ_∞-norm. The latter approach had also previously been employed by Sherstov in [29].

To obtain the lower bound on the approximate degree of matching, we consider a new complexity measure recently introduced by Aaronson, Ben-David, Kothari, Rao and Tal [1]. For any total Boolean function f, they define the *Spectral Sensitivity* to be the spectral radius of the bipartite graph defined by the f-bichromatic edges of the Hypercube (i.e., the f-cut of the cube). The notion of spectral sensitivity had notably (implicitly) also appeared at the heart of Huang's breakthrough proof of the Sensitivity Conjecture [15]. The main technical Theorem of [1] states that the spectral sensitivity of any total Boolean function lower bounds its approximate polynomial degree – and it is this relation that we leverage.

We prove the following tight bound on the spectral sensitivity of BPM_n.

▶ **Theorem 3** (The Spectral Sensitivity of Bipartite Perfect Matching)**.** *The Spectral Sensitivity of the bipartite perfect matching function is $\lambda(\mathrm{BPM}_\mathrm{n}) = \Theta(n^{3/2})$.*

One of our main motivations in studying the algebraic properties of BPM_n and its dual, is the following longstanding question: what is the least complexity of a *deterministic* algorithm for bipartite matching? Hopcroft and Karp's algorithm [14] from half a century ago attains a running time of $\mathcal{O}\left(n^{5/2}\right)$[2], and as of yet no known deterministic algorithm has been shown to break the "$n^{5/2}$-barrier". In the last section of this paper we explore the above barrier through the lens of the Demand Query Model [22], which is a concrete complexity model for matching due to Nisan. The demand model was shown in [22] to "capture" the complexity of a wide class of algorithms (i.e., *combinatorial* algorithms), therefore any non-trivial lower bound on algorithms within the model would have far reaching implications. To this end, we draw connections between the *algebraic quantities* explored throughout this work, including approximate degree and the ℓ_1-norm of the dual, and the *demand query complexity* of matching – see Figure 7. Furthermore, we exhibit an efficient quantum simulation for the demand model, showing that lower bounds in the quantum query model yield corresponding combinatorial bounds. The quantum query complexity of matching was shown by Zhang [32] to be at least $\Omega\left(n^{3/2}\right)$, and by Lin and Lin [18] to be at most $\mathcal{O}\left(n^{7/4}\right)$. Closing this gap is

[2] On *dense* graphs, wherein the number of edges is proportional to n^2.

left as an open question, and we remark that any polynomial improvement on the lower bound would yield a non-trivial bound in the demand model[3]. Finally, we note that all the bounds obtained in this paper are compatible with the existence of quasi-linear demand query algorithms for bipartite matching, and this might be seen as weak evidence pointing in this direction. Obtaining non-trivial bounds on the demand query complexity of matching is left as our main open problem.

2 Preliminaries and Notation

2.1 Boolean Functions and Polynomial Representation

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. A polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ is said to *represent* f, if for every $x \in \{0,1\}^n$, we have $p(x) = f(x)$. We recall that any Boolean function can be *uniquely* represented by a multilinear polynomial over the reals. Given the unique multilinear polynomial $p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \left(\prod_{i \in S} x_i\right)$ representing f, we denote by $\mathrm{mon}(f) = \{S \subseteq [n] : a_S \neq 0\}$ the set of all monomials appearing in its polynomial representation. Furthermore, we define the following two "norms", which are defined using the unique representations of Boolean functions, over the Boolean and Fourier bases.

▶ **Definition 4.** *Let $f : \{0,1\}^n \to \{0,1\}$, let $p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \Pi_{i \in S} x_i$ be the unique multilinear representation of f over the reals and let $\{\widehat{f}_S : S \subseteq [n]\}$ be the Fourier spectrum of f. The ℓ_1-norm and ℓ_1-Fourier-norm of f are defined:*

$$\|f\|_1 \stackrel{def}{=} \|p\|_1 \stackrel{def}{=} \sum_{S \subseteq [n]} |a_S|, \text{ and } \hat{\|}f\hat{\|}_1 \stackrel{def}{=} \sum_{S \subseteq [n]} |\widehat{f}_S|.$$

The ϵ-approximate degree of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is the least degree of a real multilinear polynomial *approximating* f in the ℓ_∞ norm, with error at most ϵ. Hereafter, we use the standard notation and write $\widetilde{\deg}_\epsilon(f)$ to denote the ϵ-approximate degree of f. In the case of $\epsilon = \frac{1}{3}$, we omit the ϵ and instead write $\widetilde{\deg}(f)$.

▶ **Definition 5.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $0 < \epsilon < \frac{1}{2}$. The ϵ-approximate degree of f, $\widetilde{\deg}_\epsilon(f)$, is the least degree of a real polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ satisfying $|f(x) - p(x)| \leq \epsilon$, for all $x \in \{0,1\}^n$.*

In the context of Boolean functions, it is sometimes useful to consider the transformation of a Boolean vector in which an arbitrary subset of bits have been flipped. Thus, if $x \in \{0,1\}^n$, and $S \subseteq [n]$, we use the notation x^S to indicate the vector in which the coordinates S have been flipped. For any $i \in [n]$, the notation x^i is shorthand for $x^{\{i\}}$. Using this notation, we define the following two complexity measures for Boolean functions.

▶ **Definition 6.** *Let f $\{0,1\}^n \to \{0,1\}$. The **sensitivity** of f at $x \in \{0,1\}^n$ is:*

$$\mathrm{sens}_f(x) = \left|\left\{i \in [n] : f(x) \neq f(x^i)\right\}\right|$$

*and similarly, the **block sensitivity** of f at x is:*

$$\mathrm{bs}_f(x) = \max\{s \in [n], \text{ such that } \exists B_1 \sqcup \cdots \sqcup B_s \subseteq [n] : \ \forall i \in [s] : \ f(x) \neq f(x^{B_i})\}.$$

The sensitivity and block sensitivity of f are then defined by their corresponding measures on the worst case input, namely $\mathrm{sens}(f) = \max_{x \in \{0,1\}^n} \mathrm{sens}_f(x)$ and $\mathrm{bs}(f) = \max_{x \in \{0,1\}^n} \mathrm{bs}_f(x)$.

[3] Theorem 1 implies that this lower bound *cannot be (polynomially) strengthened* by the "polynomial method". In fact, neither can the (nonnegative-weight) Ambainis' adversary technique, see [32].

2.2 Graph Theory

We use standard definitions and notation relating to graphs. If G is a graph, we denote its vertex set by $V(G)$, its edge set by $E(G)$, and its connected components by $C(G)$. The cardinalities of these sets are denoted $v(G)$, $e(G)$ and $c(G)$, respectively. For any vertex $v \in V(G)$, the neighbour set of v is denoted by $N(v)$, and its degree is denoted $\deg(v) = |N(v)|$. The set of all *perfect matchings* of G is denoted by $\mathrm{PM}(G)$. We also use the following slightly less common quantity:

▶ **Definition 7.** *The* ***cyclomatic number*** *of a graph is defined* $\chi(G) \stackrel{def}{=} e(G) - v(G) + c(G)$.

The graph $G - v$, where $v \in V(G)$, is the graph over the vertices $V(G) \setminus \{v\}$ in which all the edges incident to v are omitted. If $U \subseteq V(G)$ is a set of vertices, the notation $G[U]$ refers to the *induced graph* on the vertices U, whose vertices are U and whose edges are the edges of G which are incident only to vertices in U. If $G \subseteq K_{n,n}$ and $S \subseteq E(K_{n,n})$ the notation $G \cup S$ refers the a graph over the vertices of $K_{n,n}$, whose edge set is $E(G) \cup S$.

For any graph G, the *adjacency matrix* A_G is a symmetric matrix whose rows and columns are labeled by $V(G)$, and whose entries are given by $(A_G)_{u,v} = \mathbb{1}\{\{u,v\} \in E(G)\}$. The *spectral radius* of G is defined $\rho(G) \stackrel{\text{def}}{=} \max\{|\lambda_i| : \lambda_i \in Spec(A_G)\}$, i.e., the maximum magnitude of any eigenvalue in the spectrum of A_G. Since the spectrum of bipartite graphs is *symmetric*, it holds that for any bipartite graph $\rho(G) = \lambda_1$.

Throughout this paper, we restrict our attention to *balanced bipartite graphs over the vertices of the complete bipartite graph*, $K_{n,n}$. By convention, we label the left vertices of $K_{n,n}$ by $a_1, \ldots, a_n$, and the right vertices by $b_1, \ldots, b_n$. The notation $G \subseteq K_{n,n}$ is used to indicate that G is a balanced bipartite graph over the vertices of $K_{n,n}$. Similarly, the notation $G \subseteq H$ indicates that $V(G) = V(H)$ and $E(G) \subseteq E(H)$.

2.3 Quantum Query Complexity

We consider the standard quantum query model (see, e.g., [7]). For a recent textbook on the framework of quantum computing, we refer the reader to [21]. In this paper, we refer to the bounded-error quantum query complexity $Q_2(f)$, which is the smallest number d, such that there exists a quantum query algorithm $\mathcal{A}$ making at most d queries, and agreeing with the Boolean function f with probability at least two-thirds, on all Boolean inputs.

3 The Dual Polynomial of Bipartite Perfect Matching

This section centers around the proof of Theorem 2. To provide the proof, we must first familiarize ourselves with some useful definitions and notation. To this end, we begin by defining Boolean dual functions, and by recalling two relevant graph families: matching-covered graphs, and elementary graphs. Then, we introduce the notion of sorted and ordered graphs, which serve as the building blocks of our proof. Finally, we provide our proof of Theorem 2.

3.1 Boolean Dual Functions

▶ **Definition 8.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function. The* ***Boolean Dual*** *function of* f *is denoted* $f^\star : \{0,1\}^n \to \{0,1\}$ *and is defined* $f^\star(x_1, \ldots, x_n) = 1 - f(1 - x_1, \ldots, 1 - x_n)$.

Intuitively, in the Boolean dual, the symbols 0 and 1 switch roles. Geometrically, if we consider f to be a colouring of the vertices of the n-dimensional hypercube, the duality transformation simply mirrors all vertices and inverts their colours. Algebraically, when

representing the functions using multilinear polynomials over the reals, each monomial in the "primal" function corresponds to an AND function, whereas in the dual each monomial corresponds to an OR (over the original input bits). A Boolean function f and its dual $f^\star$ share many properties. For example, their Fourier spectra are identical (up to signs of Fourier coefficients, see [25]). Nevertheless, in the $\{0,1\}$ basis, the unique multilinear polynomials representing f and $f^\star$ can differ greatly. By way of example, the polynomial representing $\mathrm{AND_n}$ consists of a single monomial, whereas its dual ($\mathrm{AND}_n^\star = \mathrm{OR_n}$) has exactly $2^n - 1$ monomials.

3.2 Matching-Covered and Elementary Graphs

A graph $G \subseteq K_{n,n}$ is said to be *matching-covered* if every edge of G participates in some perfect matching, or equivalently if its edge set can be described as the union over a set of perfect matchings $S \subseteq PM(G)$.

▶ **Definition 9.** *Let $G \subseteq K_{n,n}$ be a graph. G is **matching-covered** if and only if:*

$$\forall e \in E(G) : \ \exists M \in PM(G) : \ e \in M.$$

Matching-covered graphs have many interesting combinatorial properties. The set of all such graphs, together with the subset relation over the edges, forms a lattice. A key result by Billera and Sarangarajan [5] showed that this lattice is, in fact, isomorphic to the face lattice of the Birkhoff polytope, B_n. This lattice was later shown by [3] to be intimately related to the multilinear polynomial representing the bipartite perfect matching function, $\mathrm{BPM_n}$. Namely, the monomials of the polynomial are the elements of the lattice, and their coefficients are the Möbius numbers of this lattice.

A closely related family of graphs are the *Elementary Graphs*.

▶ **Definition 10.** *Let $G \subseteq K_{n,n}$ be a graph. Then:*

*G is **elementary** $\iff$ G is a connected matching-covered graph.*

Hereafter, we denote by $\mathrm{MC_n} = \{G \subseteq K_{n,n} : G \text{ is matching-covered}\}$ the set of all matching-covered graphs, and similarly we denote $\mathrm{EL_n} = \{G \subseteq K_{n,n} : G \text{ is elementary}\}$ for all elementary graphs. Elementary graphs were studied at length, both by Lovász and Plummer [26], and earlier by Hetyei [13]. Through their works they formulated robust characterizations of elementary graphs. In particular, we require the following theorem, due mostly to Hetyei:

▶ **Theorem 11** ([13])**.** *Let $G = (A \sqcup B, E)$ be a bipartite graph. The following are equivalent:*

- *G is elementary.*
- *G has exactly two minimum vertex covers, A and B.*
- *$|A| = |B|$ and for every $\emptyset \neq X \subset A$, $|N(X)| > |X|$.*
- *$G = K_2$, or $v(G) \geq 4$ and for any $a \in A$, $b \in B$, $G - a - b$ has a perfect matching.*
- *G is connected and every edge is "allowed", i.e., appears in a perfect matching of G.*

3.3 Ordered Graphs

▶ **Definition 12.** *Let $G \subseteq K_{n,n}$. G is a **totally ordered graph**, if there exists an ordering $\pi \in S_n$ of its left vertices, such that $N(a_{\pi(1)}) \subseteq N(a_{\pi(2)}) \subseteq \cdots \subseteq N(a_{\pi(n)})$.*

Given a totally ordered graph G, we may permute the vertices in its left and right bipartitions (separately) so that both bipartitions are sorted in decreasing order of degree. This automorphism produces a graph $H \cong G$, which we refer to as a "sorted ordered graph". Our motivation in applying such a transformation is due to the fact that $\mathrm{BPM}_n^\star$ is invariant to permutations over its bipartitions. Thus, the dual coefficient of any ordered graph and its corresponding sorted ordered graph are identical.

▶ **Definition 13.** *Let $G \subseteq K_{n,n}$. G is a **sorted ordered graph** if:*

$$\deg(a_1) \leq \deg(a_2) \leq \cdots \leq \deg(a_n), \text{ and } \forall i \in [n] : \ N(a_i) = \{b_1, \ldots, b_{\deg(a_i)}\}.$$

The adjacency relation of a sorted ordered graph can be succinctly and uniquely described by a short sequence of integers, which we dub the "representing sequence" of the graph.

▶ **Definition 14.** *Let $G \subseteq K_{n,n}$ be a sorted ordered graph. The **representing sequence** of G is defined by $\mathcal{S}_G = \{(d_1, k_1), \ldots, (d_t, k_t)\}$, where:*

$$0 \leq d_1 < d_2 < \cdots < d_t \leq n, \text{ and } 0 < k_1 < k_2 < \cdots < k_t = n$$

and furthermore:

$$\forall i \in [n] : \ N(a_i) = \begin{cases} \{b_1, \ldots, b_{d_1}\}, & 0 < i \leq k_1 \\ \{b_1, \ldots, b_{d_2}\}, & k_1 < i \leq k_2 \\ \vdots & \vdots \\ \{b_1, \ldots, b_{d_t}\}, & k_{t-1} < i \leq k_t. \end{cases}$$

The *representing sequence* $\mathcal{S}_G$ of a sorted ordered graph $G \subseteq K_{n,n}$ is essentially a "compressed" form of its degree sequence; each pair (d_i, k_i) in the sequence indicates a run of $(k_i - k_{i-1})$ left vertices, all of whose neighbour sets are exactly $\{b_1, \ldots, b_{d_i}\}$. Thus, the biadjacency matrix of G is simply described in terms of $\mathcal{S}_G$, as shown in Figure 1.

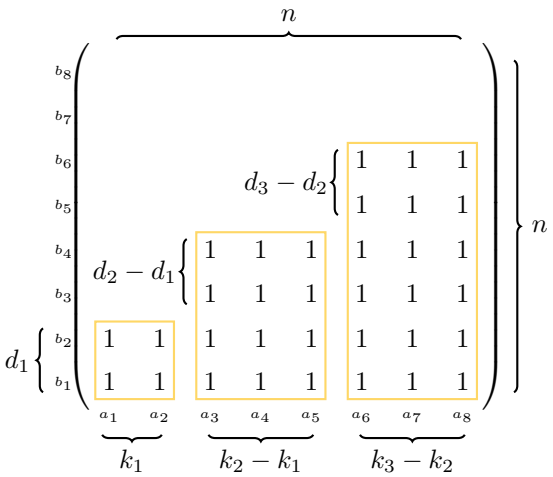


Figure 1 The biadjacency matrix of a sorted ordered graph G. The *representing sequence* of G is $\mathcal{S}_G = \{(d_1, k_1), (d_2, k_2), (d_3, k_3)\}$.

The building blocks in our proof of Theorem 2 consist of particular family of simple sorted ordered graphs – those whose representing sequence is of length exactly 2. In other words, these are the graphs whose left vertices can be partitioned into two sets, those having

full degree n, and those whose neighbour set is (the same) strict subset of the right vertices. This family also trivially includes all bicliques $K_{s,n}$. For this family of graphs, we introduce the following notation.

▶ **Notation 15.** *Let $0 \leq d \leq n$ and $0 < k < n$. The notation $\langle n, d, k \rangle$-block refers to the sorted ordered graph $G \subseteq K_{n,n}$, whose representing sequence is $\mathcal{S}_{\langle n,d,k \rangle} = \{(d, k), (n, n)\}$.*

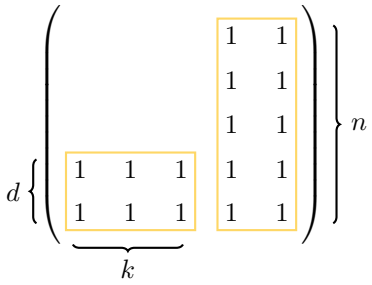


Figure 2 The biadjacency matrix of an $\langle n, d, k \rangle$-block.

3.4 Proof of Theorem 2

▶ **Definition 16.** *Let $\mathrm{BPM}_{\mathrm{n}}^{\star} : \{0,1\}^{n^2} \to \{0,1\}$ be the Boolean dual function of $\mathrm{BPM}_{\mathrm{n}}$:*

$$\mathrm{BPM}_{\mathrm{n}}^{\star}(x_{1,1}, \ldots, x_{n,n}) = \begin{cases} 1 & \{(i,j) : x_{i,j} = 0\} \text{ does } \underline{\text{not}} \text{ have a bipartite perfect matching} \\ 0 & \text{otherwise.} \end{cases}$$

In [3], a complete characterization of the multilinear polynomial representing $\mathrm{BPM}_{\mathrm{n}}$ over the reals was obtained, using a connection between the Möbius function of the Birkhoff polytope's face lattice, and the cyclomatic numbers of matching-covered graphs. The polynomial representing $\mathrm{BPM}_{\mathrm{n}}^{\star}$ may be similarly expressed through the Möbius function of some lattice (that of graphs covered by "Hall Violators", i.e., bicliques over $n+1$ vertices). These representations allowed for a *partial* description of the *support* of $\mathrm{BPM}_{\mathrm{n}}^{\star}$, which we require for our proof of Theorem 2 and will therefore now recall. The first two lemmas restrict the *support* of monomials in the dual polynomial to the set of *totally ordered graphs*, which are *not matching-covered*.

▶ **Lemma 17** ([3]). *Let $G \subseteq K_{n,n}$. If G is not totally ordered, then $a_G^{\star} = 0$.*

▶ **Lemma 18** ([3]). *Let $G \subseteq K_{n,n}$. If $G \in \mathrm{MC}_{\mathrm{n}}$, then $a_G^{\star} = 0$.*

The third lemma relates the Möbius numbers of the lattice of matching-covered graphs, with the dual coefficients of any graph $G \subseteq K_{n,n}$, thereby giving a closed-form expression for computing the dual coefficients (albeit by summing over possibly exponentially many summands).

▶ **Lemma 19** ([3]). *Let $G \subseteq K_{n,n}$. The dual coefficient of G is:*

$$a_G^{\star} = (-1)^{e(G)+1} \sum_{\substack{H \supseteq G \\ H \in \mathrm{MC}_{\mathrm{n}}}} (-1)^{\chi(H)}.$$

▶ **Corollary 20.** *Let $G \subseteq K_{n,n}$ be a graph. If all the left vertices or all the right vertices of G are in the same connected component, then:*

$$a_G^\star = \sum_{\substack{G \subseteq H \subseteq K_{n,n} \\ H \in \mathrm{EL_n}}} (-1)^{|E(H) \setminus E(G)|}.$$

Proof. Recall that every connected component of a matching-covered graph is elementary. Furthermore, elementary graphs are balanced. Thus, $G \subseteq H \in \mathrm{MC_n} \implies H$ is elementary, and we have:

$$\begin{aligned} a_G^\star &= (-1)^{e(G)+1} \sum_{\substack{H \supseteq G \\ H \in \mathrm{MC_n}}} (-1)^{\chi(H)} \\ &= (-1)^{e(G)+1} \sum_{\substack{H \supseteq G \\ H \in \mathrm{EL_n}}} (-1)^{e(H)-2n+1} = \sum_{\substack{H \supseteq G \\ H \in \mathrm{EL_n}}} (-1)^{|E(H) \setminus E(G)|}. \end{aligned}$$ ◀

3.4.1 Reducing to Permitted Edges

We now make the following observation: if G is a totally ordered graph whose coefficient we wish to compute using Lemma 19, then we may *restrict* our attention to a particular subset of edges. Whereas Lemma 19 mandates that we consider every possible "completion" of G to a matching-covered graph, the following lemma shows that we can instead only consider completions which are confined to the set of "permitted edges" for G.

▶ **Definition 21.** *Let $G \subseteq K_{n,n}$ be a sorted ordered graph, and let $\mathcal{S}_G = \{(d_1, k_1), \ldots, (d_t, k_t)\}$ be its representing sequence. The **permitted edges** for G, denoted $\mathcal{P}_G$, are defined as follows:*

$$(a_i, b_j) \in \mathcal{P}_G \iff \begin{cases} d_1 < j \le d_2, & 0 < i \le k_1 \\ d_2 < j \le d_3, & k_1 < i \le k_2 \\ \vdots & \vdots \\ d_t < j \le n, & k_{t-1} < i \le k_t. \end{cases}$$

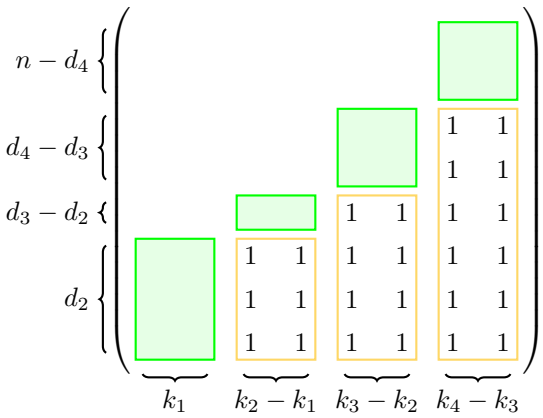


Figure 3 A sorted ordered graph G, with $\mathcal{S}_G = \{(d_1, k_1), (d_2, k_2), (d_3, k_3), (d_4, k_4)\}$. Orange blocks indicate the edges of G, and green blocks indicate the permitted edges, $\mathcal{P}_G$.

▶ **Lemma 22.** *Let $G \subseteq K_{n,n}$ be a sorted ordered graph. Then:*

$$a^\star_G = \sum_{\substack{G \subseteq H \in \mathrm{EL_n} \\ (E(H)\setminus E(G)) \subseteq \mathscr{P}_G}} (-1)^{|E(H)\setminus E(G)|}.$$

Proof. Let $S = E(K_{n,n}) \setminus (\mathscr{P}_G \sqcup E(G))$. By Lemma 19, Corollary 20, and using the inclusion-exclusion principle, we have:

$$\begin{aligned} a^\star_G &= (-1)^{e(G)+1} \sum_{\substack{H \supseteq G \\ H \in \overline{\mathrm{MC}}_\mathrm{n}}} (-1)^{\chi(H)} \\ &= \sum_{\substack{G \subseteq H \in \mathrm{EL_n} \\ (E(H)\setminus E(G)) \subseteq \mathscr{P}_G}} (-1)^{|E(H)\setminus E(G)|} + (-1)^{e(G)+1} \sum_{\substack{G \subseteq H \in \mathrm{MC_n} \\ E(H)\cap S \neq \emptyset}} (-1)^{\chi(H)} \\ &= \sum_{\substack{G \subseteq H \in \mathrm{EL_n} \\ (E(H)\setminus E(G)) \subseteq \mathscr{P}_G}} (-1)^{|E(H)\setminus E(G)|} + (-1)^{e(G)+1} \sum_{\emptyset \neq T \subseteq S} (-1)^{|T|} \sum_{(G \sqcup T) \subseteq H \in \mathrm{MC_n}} (-1)^{\chi(H)}. \end{aligned}$$

Observe that for every $\emptyset \neq T \subseteq S$, the graph $G \sqcup T$ is *not* totally ordered. Therefore, by Lemma 17, every summand $\sum_{(G\sqcup T)\subseteq H\in \mathrm{MC_n}} (-1)^{\chi(H)}$ in the above expression vanishes, thus concluding the proof. ◀

3.4.2 Factorizing into $\langle n, d, k\rangle$-blocks

Having shown that only "permitted edges" need be considered, our next step is to reduce the computation of the dual coefficient $a^\star_G$, to that of dual coefficients of *simpler* graphs. In order to do so, we must first handle the following "degenerate" case.

▶ **Lemma 23.** *Let $G \subseteq K_{n,n}$ be a sorted ordered graph and let $\mathcal{S}_G = \{(d_1,k_1),\ldots,(d_t,k_t)\}$ be the representing sequence of G. If $\exists i \in [t-1]$ such that $d_{i+1} \leq k_i$, then $a^\star_G = 0$.*

Proof. Let $i \in [t-1]$ such that $d_{i+1} \leq k_i$, and let $X = \{a_1,\ldots,a_{k_i}\} \subsetneq \{a_1,\ldots,a_n\}$. Then,

$$a^\star_G = \sum_{\substack{G \subseteq H \in \mathrm{EL_n} \\ (E(H)\setminus E(G)) \subseteq \mathscr{P}_G}} (-1)^{|E(H)\setminus E(G)|}$$

by Lemma 22, and therefore it suffices to show that any graph $H \supseteq G$ with $(E(H) \setminus E(G)) \subseteq \mathscr{P}_G$ is not elementary. Let H be such a graph. Then $|N_H(X)| \leq d_{i+1} \leq k_i = |X|$, and by Theorem 11, H is indeed not elementary. ◀

Any sorted ordered graph G whose representing sequence is not degenerate in the above sense, can be neatly factorized into a set of $\langle n,d,k\rangle$-blocks. In the following lemma we construct such a decomposition, and relate the dual coefficients of the each component with that of the original graph.

▶ **Lemma 24.** *Let $G \subseteq K_{n,n}$ be a sorted ordered graph. Let $\mathcal{S}_G = \{(d_1,k_1),\ldots,(d_t,k_t)\}$ be the representing sequence of G, where $\forall i \in [t-1]:\ d_{i+1} > k_i$. Denote $k_0 = 0$, $d_{t+1} = n$, and:*

$$\forall i \in [t]:\ A_i = \{a_{k_{i-1}+1},\ldots,a_{d_{i+1}}\}, \quad B_i = \{b_{k_{i-1}+1},\ldots,b_{d_{i+1}}\}$$

Furthermore, $\forall i \in [t]$ let $G_i = G[A_i \sqcup B_i]$ be the induced graph on the vertices $A_i \sqcup B_i$. Then:

$$a^\star_G = \prod_{i=1}^{t} a^\star_{G_i}.$$

Proof. For all $i \in [t]$, let $S_i = \{a_{k_{i-1}+1}, \ldots, a_{k_i}\}$ and $T_i = \{b_{d_i+1}, \ldots, b_{d_{i+1}}\}$. Observe that the permitted edges for G are partitioned by the sets S_i, T_i as follows: $\mathscr{P}_G = \bigsqcup_{i=1}^{t}(S_i \times T_i)$. Furthermore, since $\forall i \in [t] : \ d_i > k_{i-1}$, we have:

$$\forall i \in [t] : \ (A_i \times B_i) \cap \mathscr{P}_G = (S_i \times T_i).$$

Thus, each induced graph G_i "covers" the set $(S_i \times T_i)$, and the set of all induced graphs covers all the permitted edges $\mathscr{P}_G$. Since $d_i > k_{i-1}$, then $\forall i \in [t-1] : G_i$ has at least one left vertex, $a_{d_{i+1}}$, whose neighbour set in G_i is the entire right bipartition B_i. Similarly, in G_t the neighbour set of the right vertex $b_{k_{t-1}+1}$ is the entire left bipartition A_t. Thus, by Corollary 20:

$$\forall i \in [t] : \ a^{\star}_{G_i} = \sum_{\substack{G_i \subseteq H \\ H \text{ is elementary}}} (-1)^{|E(H) \setminus E(G)|}.$$

To complete the proof, it remains to show a bijection between elementary completions of G using the permitted edges $\mathscr{P}_G$, and elementary completions of each of the graphs G_i.

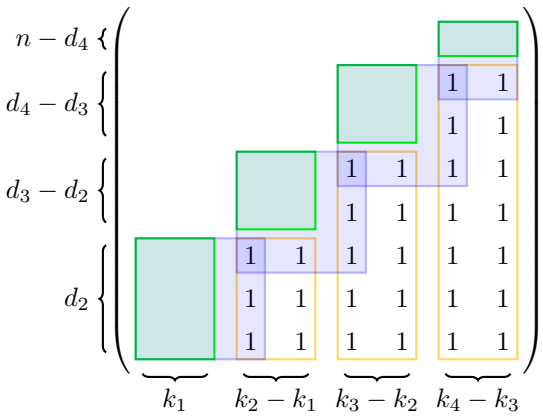


Figure 4 A sorted ordered graph G, with $\mathcal{S}_G = \{(d_1, k_1), (d_2, k_2), (d_3, k_3), (d_4, k_4)\}$. The permitted edges are covered. Orange blocks indicate the edges of G, green blocks indicate the permitted edges, and blue blocks are the induced graphs G_i.

Let $G \subseteq H \in \mathrm{EL_n}$ such that $(E(H) \setminus E(G)) \subseteq \mathscr{P}_G$. For all $i \in [t]$, let $H_i = H[A_i \sqcup B_i]$. Since $H \supseteq G$, clearly also $\forall i \in [t] : \ H_i \supseteq G_i$. It remains to show that every such H_i is elementary. To this end, we use Theorem 11: let $i \in [t]$ and let $\emptyset \neq X \subsetneq A_i$. If $X \cap A_{i+1} \neq \emptyset$ then H_i has a vertex of full degree, and so $|N_{H_i}(X)| = |B_i| = |A_i| > |X|$. Otherwise, if $X \cap A_{i+1} = \emptyset$ then let $X' = X \sqcup \{a_1, \ldots, a_{k_{i-1}}\}$. Observe that $N_H(X') = N_H(X) = N_{H_i}(X) \sqcup \{b_1, \ldots, b_{k_{i-1}}\}$. However, H is elementary, therefore $|N_H(X')| > |X'| = |X| + k_{i-1}$. In both cases we have $|N_{H_i}(X)| > |X|$ and H_i is elementary.

Conversely, let $H_1 \supseteq G_1, \ldots, H_t \supseteq G_t$ be elementary graphs. Then it suffices to show that $H \supseteq G$ whose edges are $E(H) = E(G) \cup E(H_1) \cup E(H_2) \cup \ldots\ldots E(H_t)$ is also elementary. Let $X \subsetneq A$, let i be the largest index such that $a_i \in X$, and let j be the index for which $k_{j-1} < i \leq k_j$. Thus:

$$N_H(X) = N_{H_j}(X \cap A_j) \sqcup \{b_1, \ldots, b_{k_{i-1}}\}.$$

If $X \cap A_j = A_j$, then $N_{H_j}(X \cap A_j) = B_j$ and thus $|N_H(X)| = k_{i-1} + |B_j| = d_{j+1} > k_j \geq |X|$, and indeed H is elementary. Otherwise, since H_j is elementary and $(X \cap A_j) \subsetneq A_j$, we have $|N_{H_j}(X \cap A_j)| > |X \cap A_j|$, and therefore:

$$|N_H(X)| = |N_{H_j}(X \cap A_j)| + k_{i-1} > |X \cap A_j| + k_{i-1} \geq |X|. \qquad \blacktriangleleft$$

3.4.3 The Dual Coefficients of $\langle n, d, k\rangle$-blocks

Finally, having reduced the computation of the dual coefficient of an arbitrary ordered graph G to that of simple "blocks", we are left with the task of directly computing the dual coefficient for any such block.

▶ **Lemma 25.** *Let $0 \le d \le n$, $0 < k < n$. Then, the coefficient of the $\langle n, d, k\rangle$-block is:*

$$a^\star_{\langle n,d,k\rangle} = \begin{cases} \binom{n-1}{k}, & d = 0 \\ -\binom{n-d-1}{k-d}\binom{k-1}{d-1}, & d > 0. \end{cases} \qquad (\star)$$

Proof. The proof is by induction on n, d and k. For the base case, let G be an $\langle 2, d, 1\rangle$-block, where $d \in \{0, 1, 2\}$. In all three cases, only $K_{2,2} \supseteq G$ is elementary, thus by Corollary 20 they all satisfy equation $(\star)$, as required. Next, we use complete induction. Let G be an $\langle n, d, k\rangle$-block, where $n > 2$, and assume equation $(\star)$ holds for all $\langle n', d', k'\rangle$-blocks, such that:

$$(n' < n) \vee (n' = n \ \wedge \ k' = k \ \wedge \ d' > d).$$

If $d > k$, then $\forall X \subsetneq \{a_1, \ldots, a_n\} : \ |N(X)| > |X|$, thus by Theorem 11, G is elementary and by Lemma 18, $a^\star_G = 0$. Otherwise, $d \le k$. In this case, denote $S = \{(a_1, b_n), \ldots, (a_k, b_n)\}$, and partition the set of all elementary graphs containing G into two disjoint sets,

$$\mathcal{H}_1 = \{G \subseteq H \in \mathrm{EL_n} : E(H) \cap S \neq \emptyset\}, \text{ and } \mathcal{H}_2 = \{G \subseteq H \in \mathrm{EL_n} : E(H) \cap S = \emptyset\},$$

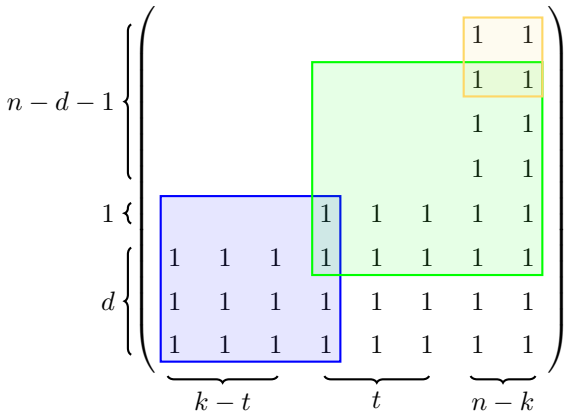
and by Corollary 20, the dual coefficient of G is given by the sum over these sets:

$$a^\star_G = \sum_{H \in \mathcal{H}_1} (-1)^{|E(H)\setminus E(G)|} + \sum_{H \in \mathcal{H}_2} (-1)^{|E(H)\setminus E(G)|}.$$

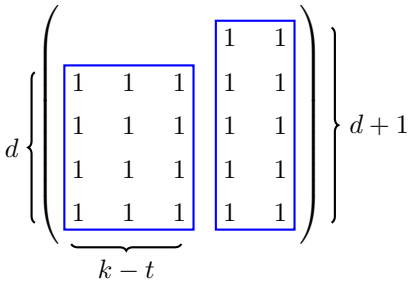
The contributions of $\mathcal{H}_1$. To sum the contributions of all graphs in $\mathcal{H}_1$, we use the inclusion-exclusion principle. First, note that $\mathrm{BPM}^\star_\mathrm{n}$ is invariant to permutations over each bipartition (that is, if $H \cong G$ then $a^\star_G = a^\star_H$). Therefore, for every subset $T \subseteq S$ of selected edges, we may, without loss of generality, "sort" the graph to obtain an isomorphic sorted ordered graph. Consequently, denote $\forall t \in [k] : \ G_t = G \cup \{(a_{k-t+1}, b_{d+1}), \ldots, (a_k, b_{d+1})\}$. By the inclusion-exclusion principle, we have

$$\sum_{H \in \mathcal{H}_1} (-1)^{|E(H)\setminus E(G)|} = \sum_{t=1}^{k} (-1)^{t+1} \binom{k}{t} \cdot (-1)^t \cdot a^\star_{G_t} = -\sum_{t=1}^{k} \binom{k}{t} \cdot a^\star_{G_t}.$$

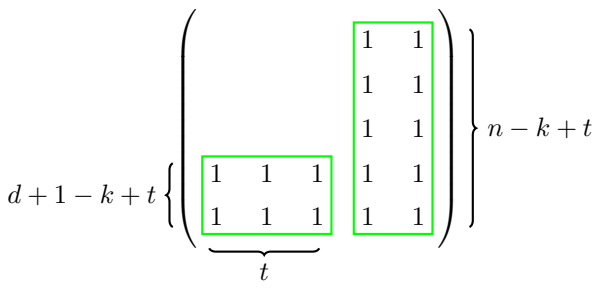
If $t = k$, then G_t is an $\langle n, d+1, k\rangle$-block, for which the induction hypothesis holds. Otherwise, for $t \in [k-1]$, the biadjacency matrix of each graph G_t can be partitioned into blocks, as follows:



If $t < k - d$, then by Definition 21 the permitted edges for the vertices $\{a_1, \ldots, a_{k-t}\}$ in G_t are only those connecting them to b_{d+1}. Therefore, G_t cannot be completed to an elementary graph using only permitted edges, and by Lemma 22, $a^\star_{G_t} = 0$. Otherwise, by Lemma 24, the coefficient $a^\star_{G_t}$ is the product of coefficients for each of the three blocks. The first two are an $\langle d+1, d, k-t\rangle$-block and a $\langle n-k+t, d+1-k+t, t\rangle$-block. The third is a complete bipartite graph over $n-k$ vertices, and thus does not affect the coefficient of G.



(a) $\langle d+1, d, k-t\rangle$-block.



(b) $\langle n-k+t, d+1-k+t, t\rangle$-block.

Observe that if $k - d < t < k$, then the $\langle d+1, d, k+1\rangle$-block is elementary, thus by Lemma 18, its dual coefficient is zero. Consequently, only two potentially non-zero cases remain: $t = k$ and $t = k - d$. For both cases, the induction hypothesis holds. Observe that if $d = 0$, both cases converge to a single case. Thus:

$d > 0$:

$$\begin{aligned}-\sum_{t=1}^{k}\binom{k}{t}\cdot a^\star_{G_t} &= -\binom{k}{k-d}\cdot a^\star_{\langle d+1,d,d\rangle} - \binom{k}{k}\cdot a^\star_{\langle n,d+1,k\rangle}\\ &= -\binom{n-d-2}{k-d-1}\cdot\left[\binom{k}{k-d}+\binom{k-1}{d}\right] = -\binom{n-d-2}{k-d-1}\binom{k-1}{d-1}.\end{aligned}$$

$d = 0$:

$$-\sum_{t=1}^{k}\binom{k}{t}\cdot a^\star_{G_t} = -\binom{k}{k}\cdot a^\star_{\langle n,1,k\rangle} = \binom{n-2}{k-1}.$$

The contributions of $\mathcal{H}_2$. If $k = n-1$, then $\mathcal{H}_2 = \emptyset$, thus there are no contributions from $\mathcal{H}_2$. This assertion follows since for any $H \supseteq G$ with $E(H) \cap S = \emptyset$, we have $|N(\{a_1, \ldots, a_{n-1}\})| \leq n-1 = |\{a_1, \ldots, a_{n-1}\}|$. Thus, by Theorem 11, H is not elementary. Otherwise, if $k < n-1$, we claim that:

$$\sum_{H \in \mathcal{H}_2} (-1)^{|E(H) \setminus E(G)|} = a^{\star}_{\langle n-1,d,k \rangle}.$$

Since the induction hypothesis holds for the $\langle n-1, d, k\rangle$-block, proving the above identity would yield an expression for the contributions of $\mathcal{H}_2$. Denote the $\langle n-1, d, k\rangle$-block by G'. To prove the aforementioned identity, it remains to show a bijection between elementary graphs $G' \subseteq H' \in EL_{n-1}$, and elementary graphs $G \subseteq H \in \mathrm{EL_n}$, where $E(H) \cap S = \emptyset$. Furthermore, we must also maintain $|E(H') \setminus E(G')| = |E(H) \setminus E(G)|$, for any two graphs H' and H which are mapped to one another by the bijection. The bijection is defined as follows:

$\underline{H \mapsto H'}$: Let $H \supseteq G$ be an elementary graph such that $E(H) \cap S = \emptyset$. We claim that $H' = H - a_n - b_n$ is also elementary. By Theorem 11, it suffices to show that $\forall X \subsetneq \{a_1, \ldots, a_{n-1}\} : |N_{H'}(X)| > |X|$. If $X \cap \{a_{k+1}, \ldots, a_{n-1}\} \neq \emptyset$, then $N_{H'}(X) = \{b_1, \ldots, b_{n-1}\}$. Thus $|N_{H'}(X)| = n-1 > |X|$. Otherwise, if $X \cap \{a_{k+1}, \ldots, a_{n-1}\} = \emptyset$, then $N_{H'}(X) = N_H(X)$ and therefore: $|N_{H'}(X)| = |N_H(X)| > |X|$, as required.

$\underline{H' \mapsto H}$: Let $H' \supseteq G'$ be an elementary graph. We claim that the graph $H \subseteq K_{n,n}$ where $E(H) = E(H') \sqcup \{(a_{k+1}, b_n), \ldots, (a_n, b_n)\}$, is also elementary. Once again, we use Theorem 11. Let $X \subsetneq \{a_1, \ldots, a_n\}$. If $X \cap \{a_{k+1}, \ldots, a_n\} \neq \emptyset$, then $N_H(X) = \{b_1, \ldots, b_n\}$. Thus $|N_H(X)| = n > |X|$. Otherwise, if $X \cap \{a_{k+1}, \ldots, a_n\} = \emptyset$, then $N_H(X) = N_{H'}(X)$ and therefore: $|N_H(X)| = |N_{H'}(X)| > |X|$, as required.

Summing up the contributions. Finally, we have reduced the computation of the coefficient $a^{\star}_{\langle n,d,k \rangle}$ to a sum of coefficients $a^{\star}_{\langle n',d',k' \rangle}$ for which the induction hypothesis holds. It now remains to sum up the contributions for each possible case. If $d > 0$ and $k = n-1$, then:

$$a^{\star}_G = \sum_{H \in \mathcal{H}_1} (-1)^{|E(H) \setminus E(G)|} = -\binom{n-d-2}{k-d-1}\binom{k-1}{d-1} = -\binom{n-d-1}{k-d}\binom{k-1}{d-1}.$$

If $d > 0$ and $k < n-1$, then:

$$a^{\star}_G = \sum_{H \in \mathcal{H}_1} (-1)^{|E(H) \setminus E(G)|} + \sum_{H \in \mathcal{H}_2} (-1)^{|E(H) \setminus E(G)|} = -\binom{n-d-1}{k-d}\binom{k-1}{d-1}.$$

If $d = 0$ and $k = n-1$, then:

$$a^{\star}_G = \sum_{H \in \mathcal{H}_1} (-1)^{|E(H) \setminus E(G)|} = \binom{n-2}{k-1} = \binom{n-1}{k}.$$

And lastly, if $d = 0$ and $k < n-1$, then:

$$a^{\star}_G = \sum_{H \in \mathcal{H}_1} (-1)^{|E(H) \setminus E(G)|} + \sum_{H \in \mathcal{H}_2} (-1)^{|E(H) \setminus E(G)|} = \binom{n-2}{k-1} + \binom{n-2}{k} = \binom{n-1}{k}. \blacktriangleleft$$

3.4.4 Putting It Together

We are now ready to prove Theorem 2.

Proof. Let $G \subseteq K_{n,n}$. If G is not totally ordered, then by Lemma 17, $a^\star_G = 0$. Otherwise, if G is totally ordered and there exists some $i \in [t-1]$ such that $d_{i+1} \leq k_i$, then by Lemma 23, $a^\star_G = 0$, and indeed:

$$f\left(d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\right) = \binom{d_{i+1} - k_{i-1} - 1}{k_i - k_{i-1}} = 0.$$

Finally, if G is totally ordered, and $\forall i \in [t-1] : \; d_{i+1} > k_i$, then let $A_i = \{a_{k_{i-1}+1}, \ldots, a_{d_{i+1}}\}$, $B_i = \{b_{k_{i-1}+1}, \ldots, b_{d_{i+1}}\}$, $\forall i \in [t]$, where $k_0 = 0$ and $d_{t+1} = n$. Furthermore $\forall i \in [t]$, let $G_i = G\left[A_i \sqcup B_i\right]$ be the induced graph on the vertices $A_i \sqcup B_i$. By Lemma 24, we have $a^\star_G = \prod_{i=1}^{t} a^\star_{G_i}$. Observe that $\forall i \in [t-1]$, the graph G_i is an $\langle d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\rangle$-block. Thus, its coefficient is given by the expression in Lemma 25. However, the last graph G_t may *not* be a "block". In fact, there are two possible cases: either $d_t = n$, in which case G_t is a complete bipartite graph, and thus $a^\star_G = 1$. Otherwise, $d_t < n$, and G_t is a biclique joining $n - k_{t-1}$ left vertices to $n - d_t$ right vertices. In this case, since $a^\star_G$ is invariant to swapping the two bipartitions, then without loss of generality we may do so, thus obtaining an isomorphic $\langle n - k_{t-1}, 0, n - d_t\rangle$-block. Thus, we have:

$$a^\star_{G_t} = \begin{cases} 1, & d_t = n \\ \binom{n - k_{t-1} - 1}{n - d_t}, & d_t < n \end{cases} = \binom{n - k_{t-1} - 1}{n - d_t}.$$

Putting it all together, we obtain:

$$\begin{aligned} a^\star_G = \prod_{i=1}^{t} a^\star_{G_i} &= \binom{n - k_{t-1} - 1}{n - d_t} \cdot \prod_{i=1}^{t-1} a^\star_{\langle d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\rangle} \\ &= \binom{n - k_{t-1} - 1}{n - d_t} \cdot \prod_{i=1}^{t-1} f\left(d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\right). \end{aligned}$$ ◀

3.5 Corollaries of Theorem 2: The ℓ_1-norms of $\mathrm{BPM_n}$

Theorem 2 allows us to compute the dual coefficient of any graph $G \subseteq K_{n,n}$. It is not hard to see that for some graphs $G \subseteq K_{n,n}$, the coefficient $a^\star_G$ may be *exponential* in n. For instance, the biclique $K_{n,n/2}$ is an ordered graph whose representing sequence is $\left(\frac{n}{2}, n\right)$. Therefore its dual coefficient is $\binom{n-1}{\frac{n}{2}} \sim \frac{2^n}{\mathrm{poly}(n)}$. We claim that the aforementioned bound is qualitatively tight. Namely, for *every* graph $G \subseteq K_{n,n}$, the coefficient $a^\star_G$ is *at most* exponential in $2n$.

▶ **Lemma 26.** *Let $G \subseteq K_{n,n}$. The dual coefficient of G is bounded by $|a^\star_G| \leq 2^{2n}$.*

Proof. If G is not totally ordered then by Lemma 17 $a^\star_G = 0$. Otherwise, let $\{(d_1, k_1), \ldots, (d_t, k_t)\}$ be the *representing sequence* of G, let $k_0 = 0$, and let:

$$f(n, d, k) = \begin{cases} \binom{n-1}{k}, & d \leq 0 \\ -\binom{n-d-1}{k-d}\binom{k-1}{d-1}, & d > 0. \end{cases}$$

Observe that both in the case $d \leq 0$ and in the case $d > 0$, we have $|f(n, d, k)| \leq 2^{n-d+k}$ (by bounding every binomial coefficient). Therefore, using Theorem 2 we obtain:

$$|a^\star_G| = \binom{n - k_{t-1} - 1}{n - d_t} \cdot \prod_{i=1}^{t-1} \left|f\left(d_{i+1} - k_{i-1}, d_i - k_{i-1}, k_i - k_{i-1}\right)\right| \leq 2^{n+d_t} \leq 2^{2n}.$$ ◀

Thus, the multilinear polynomial representing $\mathrm{BPM}_n^\star$ is "simple" in the following sense: its ℓ_1-norm (the sum of absolute values of its coefficients) is small, over both the $\{0,1\}$ and Fourier basis.

▶ **Corollary 27.** *Let $n > 2$. Then:*

$$\|\mathrm{BPM}_n^\star\|_1 = 2^{\Theta(n\log n)}, \text{ and furthermore } \hat{\|}\,\mathrm{BPM}_n\,\hat{\|}_1 = \hat{\|}\,\mathrm{BPM}_n^\star\,\hat{\|}_1 = 2^{\mathcal{O}(n\log n)}.$$

Proof. In [3], the number of monomials in $\mathrm{BPM}_n^\star$ was bounded by:

$$(n!)^2 \leq |\mathrm{mon}\,(\mathrm{BPM}_n^\star)| \leq (n+2)^{2n+2}$$

thus, using Lemma 26, we deduce:

$$(n!)^2 \leq \|\mathrm{BPM}_n^\star\|_1 \leq (n+2)^{2n+2}\cdot 2^{2n} \;\Rightarrow\; \|\mathrm{BPM}_n^\star\|_1 = 2^{\Theta(n\log n)}.$$

As for the Fourier ℓ_1-norm, we note that the magnitudes of the Fourier coefficients of any Boolean function and its dual are identical (see e.g., [25]), therefore $\hat{\|}\,\mathrm{BPM}_n\,\hat{\|}_1 = \hat{\|}\,\mathrm{BPM}_n^\star\,\hat{\|}_1$. Furthermore, recall that $\forall S \subseteq [n]$, $\hat{\|}\,\mathrm{AND}_S\,\hat{\|} = \hat{\|}\Pi_{i\in S}x_i\hat{\|} = 1$. Thus, by subadditivity and homogeneity:

$$\hat{\|}\,\mathrm{BPM}_n\,\hat{\|}_1 = \hat{\|}\,\mathrm{BPM}_n^\star\,\hat{\|}_1 \leq \sum_{G\subseteq K_{n,n}} |a_G^\star|\cdot\hat{\|}\,\mathrm{AND}_G\,\hat{\|} = \|\mathrm{BPM}_n^\star\|_1 = 2^{\mathcal{O}(n\log n)}. \qquad ◀$$

4 The Upper Bound on $\widetilde{\deg}_\epsilon(\mathrm{BPM}_n)$

In this section we obtain an upper bound on the approximate degree of the bipartite perfect matching function, which holds even for exponentially small values of ϵ.

▶ **Theorem 28.** *Let $2^{-n\log n} \leq \epsilon \leq \frac{1}{3}$. The ϵ-approximate degree of* BPM_n *is bounded by:*

$$\widetilde{\deg}_\epsilon(\mathrm{BPM}_n) = \mathcal{O}(n^{3/2}\sqrt{\log n}).$$

This bound is essentially a corollary Theorem 2, alongside two further observations. First, we show that the approximate degree of any Boolean function and its dual are identical (for all $\epsilon > 0$). We then prove that Boolean functions whose representing polynomials have small ℓ_1-norm over the $\{0,1\}$ basis, can be efficiently approximated by low degree polynomials. The latter approach was also employed by Sherstov in [29]. Let us remark that, to obtain the upper bound on the approximate degree of BPM_n it would have sufficed to merely show that the magnitudes of all dual coefficients are, at most, exponential in $\Theta(n\log n)$. However, we do not know of a simpler proof of this fact, other than leveraging the complete characterization of $\mathrm{BPM}_n^\star$ given by Theorem 2.

▶ **Lemma 29.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $f^\star$ be its dual. Then:*

$$\forall 0 < \epsilon < \frac{1}{2} : \quad \widetilde{\deg}_\epsilon(f) = \widetilde{\deg}_\epsilon(f^\star).$$

Proof. Let $\epsilon > 0$, and let $p \in \mathbb{R}[x_1,\ldots,x_n]$ be a real polynomial that ϵ-approximates f pointwise. Let $p^\star \in \mathbb{R}[x_1,\ldots,x_n]$ be the real polynomial defined by: $p^\star(x_1,\ldots,x_n) = 1 - p(1-x_1,\ldots,1-x_n)$ (i.e., replace each variable x_i with $(1-x_i)$, negate all coefficients, and add 1). Observe that $\deg(p^\star) \leq \deg(p)$, since $p^\star$ is obtained by a linear transformation on p, thus the degree cannot increase. Furthermore, $\forall x_1,\ldots,x_n \in \{0,1\}$, we have:

$$\begin{aligned}|f^\star(x_1,\ldots,x_n) - p^\star(x_1,\ldots,x_n)| &= |1 - f(1-x_1,\ldots,1-x_n) - (1 - p(1-x_1,\ldots,1-x_n))| \\ &= |f(1-x_1,\ldots,1-x_n) - p(1-x_1,\ldots,1-x_n)| \leq \epsilon.\end{aligned}$$

The converse similarly follows, since $(f^\star)^\star = f$. ◀

For the second lemma, we require a well known Theorem regarding the approximate degree of the AND_n function. Nisan and Szegedy [23] first showed that for the regime of $\epsilon = \Theta(1)$, we have $\widetilde{\deg}_{1/3}(\mathrm{AND}_n) = \Theta(\sqrt{n})$. Their result was extended by Buhrman, Cleve, De Wolf and Zalka [6], who determined the approximate degree of AND for any $\epsilon > 0$.

▶ **Theorem 30** ([6]). *Let $n \in \mathbb{N}$ and let $2^{-n} \leq \epsilon \leq \frac{1}{3}$. Then:*

$$\widetilde{\deg}_{\epsilon}(\mathrm{AND}_n) = \Theta\left(\sqrt{n \cdot \log(1/\epsilon)}\right).$$

Consider a Boolean function f. If the representing polynomial of f has small ℓ_1-norm, then one may use the following straightforward approach for constructing a low-degree approximating polynomial for f: approximate (with sufficiently small ϵ) every *monomial* of the representing polynomial. Since each monomial is an AND function, we may appeal to Theorem 30 to obtain a low-degree approximation. Thus, the polynomial approximating f is given by summing the approximating polynomials for each of its monomials. This scheme is implemented in the following lemma, whose proof appears in the full version of this paper.

▶ **Lemma 31.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $p \in \mathbb{R}[x_1, \ldots, x_n]$ be the unique multilinear polynomial representing f. If $3 < \|p\|_1 < 2^n$, then:*

$$\forall \epsilon \in \left(\frac{1}{\|p\|_1}, \frac{1}{3}\right) : \quad \widetilde{\deg}_{\epsilon}(f) = \mathcal{O}\left(\sqrt{n \cdot \log \|p\|_1}\right).$$

The proof of Theorem 28 now follows.

Proof. Let $2^{-n \log n} \leq \epsilon \leq \frac{1}{3}$. Then, using Corollary 27 and Lemmas 29 and 31, we have:

$$\widetilde{\deg}_{\epsilon}(\mathrm{BPM}_n) = \widetilde{\deg}_{\epsilon}(\mathrm{BPM}_n^{\star}) = \mathcal{O}(n^{3/2}\sqrt{\log n}).$$ ◀

5 The Lower Bound $\widetilde{\deg}(\mathrm{BPM}_n) = \Omega(n^{3/2})$

In this section we obtain a lower bound on the approximate degree of perfect matching, which matches the upper bound of Theorem 28, up to the low order term $\sqrt{\log n}$.

▶ **Theorem 32.** *The approximate degree of BPM_n is bounded by $\widetilde{\deg}(\mathrm{BPM}_n) = \Omega(n^{3/2})$.*

Aaronson, Ben-David, Kothari, Rao and Tal [1] recently proved that for any *total* Boolean function f, $\deg(f) = \mathcal{O}\left(\widetilde{\deg}(f)^2\right)$ (which is optimal, as exemplified by the OR_n function). Their proof is composed of two primary steps. First, they make the key observation that, at the heart of Huang's proof for the sensitivity conjecture [15], there (implicitly) lies a new complexity measure: **Spectral Sensitivity**. Their main technical Theorem is to then show that this aforementioned quantity lower-bounds approximate degree. It is this relation that we wish to leverage. [4]

[4] From the quadratic relation between degree and approximate degree [1] and using the fact that BPM_n has full degree [3], the *weaker* lower bound of $\widetilde{\deg}(\mathrm{BPM}_n) = \Omega(n)$ also immediately follows.

5.1 Spectral Sensitivity and the Sensitivity Graph

▶ **Definition 33** (Sensitivity Graph [1]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. The **Sensitivity Graph** of f is the graph G_f over the vertices $\{0,1\}^n$ whose edges are:*

$$\forall x, y \in \{0,1\}^n : \ \{x,y\} \in E(G_f) \iff |x \oplus y| = 1 \wedge f(x) \neq f(y).$$

Thus, G_f is the subgraph containing all the bi-chromatic edges of the n-dimensional Hypercube whose vertices are labeled by f (the "f-cut" of the Hypercube).

▶ **Definition 34** (Spectral Sensitivity [1]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let G_f be its sensitivity graph. The **Spectral Sensitivity** of f is defined by $\lambda(f) \stackrel{def}{=} \rho(G_f)$.*

Observe that the sensitivity graph of *any* Boolean function f is a bipartite graph whose bipartitions are given by $f^{-1}(0)$ (hereafter, the "left" vertices) and $f^{-1}(1)$ (the "right" vertices). Clearly as all the edges of the sensitivity graph are bi-chromatic, these two sets form a valid bipartition. In the case of $\mathrm{BPM_n}$, we note that (perhaps rather confusingly) the sensitivity graph is a bipartite graph in which each vertex $x \in \{0,1\}^{n^2}$ is, itself, associated with a bipartite graph (corresponding to the input x).

Under this notation, the main Theorem of [1] states the following.

▶ **Theorem 35** ([1]). *For any total Boolean function $f : \{0,1\}^n \to \{0,1\}$, we have:*

$$\lambda(f) = \mathcal{O}(\widetilde{\deg}(f)).$$

5.2 A Tight Bound on the Spectral Sensitivity of $\mathrm{BPM_n}$

In what follows, we obtain tight bounds on the Spectral Sensitivity of $\mathrm{BPM_n}$.

▶ **Theorem 36.** *The spectral sensitivity of matching is bounded by $\lambda(\mathrm{BPM_n}) = \Theta(n^{3/2})$.*

This tight bound on $\lambda(\mathrm{BPM_n})$ yields our approximate degree lower bound, and also shows that this is the *best bound attainable* by the method of Spectral Sensitivity for the perfect matching function.

▶ **Corollary 37.** *The approximate degree of matching is bounded by $\widetilde{\deg}(\mathrm{BPM_n}) = \Omega(n^{3/2})$.*

Proof. Follows from Theorem 36 and Theorem 35. ◀

5.2.1 The Upper Bound $\lambda(\mathrm{BPM_n}) = \mathcal{O}(n^{3/2})$

The spectrum of bipartite graphs has several nice properties. Their eigenfunctions come in pairs with negated eigenvalues (thus their spectrum is symmetric). Another well-known result regarding the spectrum of bipartite graphs is Hölder's inequality for matrix norms:

▶ **Proposition 38.** *Let G be a bipartite graph and let Δ_L and Δ_R be the maximal left and right degrees, correspondingly. Then, $\rho(G) \leq \sqrt{\Delta_L \Delta_R}$.*

Recall that the degree of any vertex in the sensitivity graph is equal, by definition, to the number of bi-chromatic edges incident to it – which is its sensitivity. Thus, by Proposition 38, for any Boolean function f we have $\lambda(f) \leq \sqrt{s_0(f) \cdot s_1(f)}$, where $s_b(f) \stackrel{\text{def}}{=} \max_{x \in f^{-1}(b)} \mathrm{sens}_f(x)$, $\forall b \in \{0,1\}$. This simple observation suffices to obtain the upper bound:

▶ **Proposition 39.** *For any $n \geq 1$, we have $\lambda(\mathrm{BPM_n}) \leq n^{3/2}$.*

Proof. Recall that $\lambda(\mathrm{BPM_n}) \leq \sqrt{s_0(\mathrm{BPM_n}) \cdot s_1(\mathrm{BPM_n})}$. Since the sensitivity of *any* input is at most n^2 (the number of input bits), we immediately have $s_0(\mathrm{BPM_n}) \leq n^2$ (and in fact, $s_0(\mathrm{BPM_n}) = \Theta(n^2)$). As for the 1-sensitivity, clearly for every $G \in \mathrm{BPM}_n^{-1}(1)$, the only sensitive edges are those present in some matching. The union of all matchings is matching-covered, and every edge in an elementary component is sensitive if and only if the component is K_2 (see Theorem 11), thus $s_1(\mathrm{BPM_n}) \leq n$. ◀

5.2.2 The Lower Bound $\lambda(\mathrm{BPM_n}) = \Omega(n^{3/2})$

Let us now consider the lower bound. By Cauchy's Interlace Theorem (and using the fact that for bipartite graphs, $\rho(G) = \lambda_1$), it holds that for any *bipartite* graph G, the spectral radius of G is no smaller than that of any *induced subgraph* of G. Thus, it suffices to exhibit an induced subgraph of the sensitivity graph of $\mathrm{BPM_n}$, whose spectral radius is large. In the following Theorem, we construct such a connected bi-regular induced subgraph, and bound its spectral radius.

▶ **Theorem 40.** *For any $n > 2$ we have* $\lambda(\mathrm{BPM_n}) \geq \dfrac{n^{3/2}}{3\sqrt{3}} - \mathcal{O}(n) = \Omega(n^{3/2})$.

Proof. Let $n > 2$ and let $\frac{n+1}{3} < k < n$ be a natural number. Let $A = A_1 \sqcup A_2$ and $B = B_1 \sqcup B_2$ be disjoint sets such that $|A_1| = |B_1| = k$ and $|A_2| = |B_2| = n - k$. For every $t \in \mathbb{N}^+$, denote the set of all matchings joining t vertices of A_2 with t vertices of B_2, by $M_t(A_2, B_2)$. Consider the following two sets of graphs:

$$\mathcal{R} = \{G = (A \sqcup B, (A_1 \times B_2) \sqcup (A_2 \times B_1) \sqcup E(M)) \ : \ M \in M_{n-2k}(A_2, B_2)\}$$
$$\mathcal{L} = \{G = (A \sqcup B, (A_1 \times B_2) \sqcup (A_2 \times B_1) \sqcup E(M)) \ : \ M \in M_{n-2k-1}(A_2, B_2)\}.$$

Let $G_{\mathrm{BPM_n}}$ be the sensitivity graph of $\mathrm{BPM_n}$, and let $H = G_{\mathrm{BPM_n}}[\mathcal{L} \sqcup \mathcal{R}]$ be its induced subgraph over the aforementioned set of graphs (vertices). By Cauchy's Interlace Theorem the spectral radius of H is at most that of G, therefore $\lambda(\mathrm{BPM_n}) = \rho(G_{\mathrm{BPM_n}}) \geq \rho(H)$.

Observe that every graph $G \in \mathcal{R}$ has a perfect matching, which can be constructed by taking the $(n-2k)$-size matching between A_2 and B_2 and matching the remaining k vertices of A_2 with B_1, and similarly the remaining k vertices of B_2 with A_1 (this can always be done, since the bicliques K_{A_2,B_1}, K_{A_1,B_2} are subgraphs of G). Conversely, every graph $G \in \mathcal{L}$ does not have a perfect matching. For example, the set A_2 violates Hall's condition, since: $|N(A_2)| = n - 2k - 1 + k = n - k - 1 < n - k = |A_2|$. Thus, $\mathcal{R}$ is the right bipartition of H, and $\mathcal{L}$ is its left bipartition.

Let us characterize the edges of H. Let $G \in \mathcal{R}$ and let M be its corresponding $(n-2k)$-size matching between A_2 and B_2. For every edge $e \in M$, we have by construction $(G \setminus \{e\}) \in \mathcal{L}$. Furthermore, for any edge $e \in (E(G) \setminus M)$, the graph $(G \setminus \{e\})$ does not contain one of the bicliques K_{A_2,B_1}, K_{A_1,B_2}, and is therefore not in $\mathcal{R}$. Consequently the degree of each right vertex of H is $d_R = \deg_H(G) = |M| = n - 2k$.

Similarly, let $G \in \mathcal{L}$ and let M be its $(n-2k-1)$-size matching between A_2 and B_2. Denote by S, T the left and right vertices of M, correspondingly. Then, for any $u \in (A_2 \setminus S)$, $v \in (B_2 \setminus T)$, the graph $G \sqcup \{(u, v)\}$ has a $(n-2k)$-size matching, and is thus in $\mathcal{R}$. Adding any other edge e to G would either join a vertex from A_1 to a vertex from B_1, or e would be incident to a vertex in M. In both cases, $G \sqcup \{e\}$ is not in $\mathcal{L}$. Thus the degree of each left vertex of H is $d_L = \deg_H(G) = (|A_2| - |S|) \cdot (|B_2| - |T|) = (k+1)^2$.

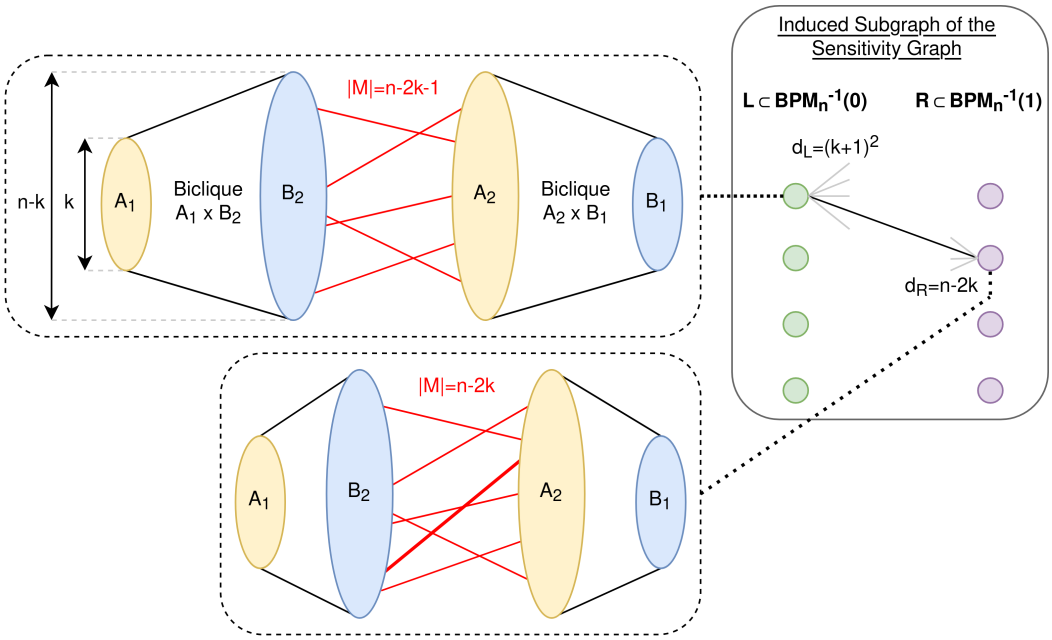


Figure 6 The induced subgraph $H = G_{\mathrm{BPM_n}}[\mathcal{L} \sqcup \mathcal{R}]$ of the sensitivity graph for $\mathrm{BPM_n}$.

Finally, observe that any bi-regular bipartite graph, and in particular H, satisfies $\rho(H) \geq \sqrt{d_L \cdot d_R}$ (this follows, for example, by considering the eigenfunction which places weight $\sqrt{d_L}$ on each left vertex, and $\sqrt{d_R}$ on each right vertex)[5]. To conclude the proof, fix $k = \lfloor \frac{n}{3} \rfloor - 1$. Thus:

$$\lambda(\mathrm{BPM_n}) = \rho(G) \geq \rho(H) \geq \sqrt{\left(\left\lfloor \frac{n}{3} \right\rfloor\right)^2 \cdot \left(n - 2\left(\left\lfloor \frac{n}{3} \right\rfloor - 1\right)\right)} = \frac{n^{3/2}}{3\sqrt{3}} - \mathcal{O}(n). \qquad \blacktriangleleft$$

References

1 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of Huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

2 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald De Wolf. Quantum lower bounds by polynomials. *Journal of the ACM (JACM)*, 48(4):778–797, 2001.

3 Gal Beniamini and Noam Nisan. Bipartite perfect matching as a real polynomial. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

4 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.

5 Louis J Billera and Aravamuthan Sarangarajan. The combinatorics of permutation polytopes. In *Formal power series and algebraic combinatorics*, volume 24, pages 1–23, 1994.

6 Harry Buhrman, Richard Cleve, Ronald De Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 358–368. IEEE, 1999.

7 Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[5] We remark that it is not hard to see that H is connected for any $n > 2$. Thus the top eigenvalue of H is *simple*, and consequently our bound does not freely extend, by interlacing, to eigenvalues other than the spectral radius of $G_{\mathrm{BPM_n}}$.

8 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*, pages 297–310, 2018.

9 Mark Bun and Justin Thaler. A nearly optimal lower bound on the approximate degree of AC^0. *SIAM Journal on Computing*, 49(4):FOCS17–59, 2019.

10 Mark Bun and Justin Thaler. Guest column: Approximate degree in classical and quantum computing. *ACM SIGACT News*, 51(4):48–72, 2021.

11 Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 387–402, 2014.

12 Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

13 Gábor Hetyei. Rectangular configurations which can be covered by 2×1 rectangles. *Pécsi Tan. Foisk. Közl*, 8:351–367, 1964.

14 John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

15 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

16 Adam Tauman Kalai, Adam R Klivans, Yishay Mansour, and Rocco A Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.

17 Adam R Klivans and Rocco A Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004.

18 Cedric Yen-Yu Lin and Han-Hsuan Lin. Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester. In *Proceedings of the 30th Conference on Computational Complexity*, CCC '15, 2015.

19 Kurt Mehlhorn and Erik M Schmidt. Las vegas is better than determinism in VLSI and distributed computing. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 330–337, 1982.

20 Marvin L Minsky and Seymour A Papert. Perceptrons: expanded edition, 1988.

21 Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

22 Noam Nisan. The demand query model for bipartite matching. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 592–599. SIAM, 2021.

23 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994.

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

25 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

26 M.D. Plummer and L. Lovász. *Matching Theory*. North-Holland Mathematics Studies. Elsevier Science, 1986.

27 Alexander A Sherstov. Separating AC^0 from depth-2 majority circuits. *SIAM Journal on Computing*, 38(6):2113–2129, 2009.

28 Alexander A Sherstov. The pattern matrix method. *SIAM Journal on Computing*, 40(6):1969–2000, 2011.

29 Alexander A Sherstov. Algorithmic polynomials. *SIAM Journal on Computing*, 49(6):1173–1231, 2020.

30 Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *arXiv preprint*, 2007. `arXiv:0710.0095`.

31 Justin Thaler, Jonathan Ullman, and Salil Vadhan. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming*, pages 810–821. Springer, 2012.

32 Shengyu Zhang. On the power of Ambainis's lower bounds. In *International Colloquium on Automata, Languages, and Programming*, pages 1238–1250. Springer, 2004.

A Towards Fine Grained Bounds for Bipartite Perfect Matching

The main thrust of this section, and indeed one of the motivating factors for the work in this paper, revolves around the following longstanding open question:

▶ **Open Problem 41** (The $n^{5/2}$-Barrier for Bipartite Matching[6]). *Is there a* ***deterministic*** *algorithm for bipartite perfect matching running in time $o(n^{5/2})$?*

Hopcroft and Karp's [14] algorithm, designed half a century ago, attains a runtime of $\mathcal{O}\left(n^{5/2}\right)$ when applied to *dense* graphs (i.e., when the number of edges is $\Theta(n^2)$). Since then, no known deterministic algorithm has been able to break this barrier, in the dense regime. To make matters concrete, in what follows let us consider the *decision variant* of the problem, as represented by BPM_n; we are given a balanced bipartite graph with n vertices in each bipartition, and wish to determine whether a perfect matching exists. Secondly, let us fix the following computational model.

The Demand Query Model

In recent work, Nisan [22] introduced a new *concrete complexity model* for bipartite matching, known as the "Demand Query Model". This model appears to be particularly well-suited for the matching problem, for two primary reasons. Firstly, Nisan showed that *combinatorial* matching algorithms can be efficiently simulated within the model (in fact, this holds even for parallel, online, approximate and other classes of algorithms, see [22]). For instance, Hopcroft and Karp's algorithm, whose running time is $\mathcal{O}(n^{5/2})$, can be "translated" into demand query algorithm making $\mathcal{O}(n^{3/2})$ queries. Since each query can be trivially simulated in $\mathcal{O}(n)$ time, this appears to capture the complexity of the aforementioned algorithm in a fine-grained manner. Secondly, the queries in this model are *simple enough* that *we could hope to prove lower bounds against them.*

In this framework, algorithms are modeled by decision trees. Each internal node corresponds to a demand query, and each leaf is labeled by an output, either 0 or 1. A demand query consists of a left vertex u and an ordering $\pi \in S_n$, induced on the right vertices. The result of such a query is the first right vertex v, according to the ordering π, for which the edge (u, v) exists in the graph (or $\perp$ if no such edge exists). A root-to-leaf path in the tree corresponds to a particular set of answers to the queries made along the path. Thus, the set of all such paths partitions the set of all graphs $G \subseteq K_{n,n}$, whereby each graph G is associated with a single leaf. Any graph $G \subseteq K_{n,n}$ which is "consistent" with the answers made along a root-to-leaf path, must also be consistent with the labeling of that leaf. The "cost" of an algorithm in this model is measured by the depth of the tree (i.e., the worst-case amount of queries made on any particular input). As this is an information-theoretic model, we disregard the amount of computation necessary to construct (or deduce the existence of) a perfect matching, and instead only measure the minimal amount of *information* required to do so.

A.1 The Demand Query Complexity of Matching

Open Problem 41 remains as of yet unsettled. In light of the efficient simulation of combinatorial algorithms by the demand model, one could formulate the following closely related question: "can one construct quasi-linear demand-query algorithms for matching?", or in the contrapositive:

[6] In fact we are only interested in *polynomial* improvements to this running time, i.e., bounds of the form $n^{5/2-\varepsilon}$ for some constant $\varepsilon > 0$.

▶ **Open Problem 42** (The Demand Query Complexity of Matching). *Is there some constant* $\varepsilon > 0$ *such that* $\mathrm{Demand}(\mathrm{BPM_n}) = \Omega(n^{1+\varepsilon})$*?*

To better understand $\mathrm{Demand}(\mathrm{BPM_n})$, we have drawn connections between the demand query complexity of $\mathrm{BPM_n}$ and other, mostly algebraic, complexity measures relating to $\mathrm{BPM_n}$ and its dual – a representative collection of which are detailed in Figure 7.

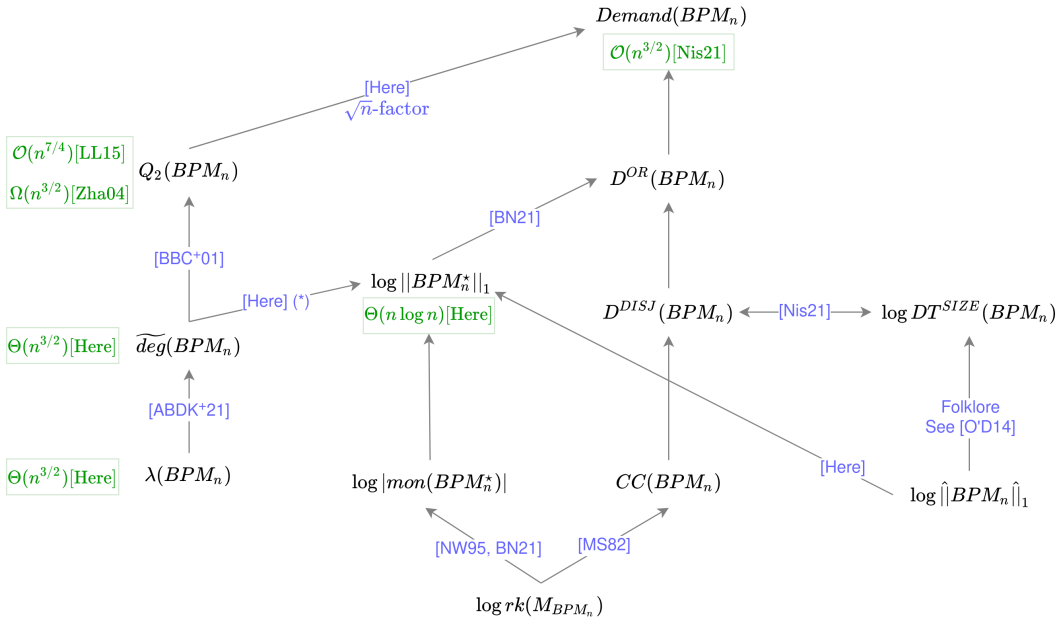


Figure 7 Relations between complexity measures of $\mathrm{BPM_n}$. An arrow $f(n) \to g(n)$ indicates that $f = \widetilde{\mathcal{O}}(g)$ (excluding logs) – with two exceptions. The arrow $Q_2(\mathrm{BPM_n}) \to \mathrm{Demand}(\mathrm{BPM_n})$ incurs a $\sqrt{n}$-factor loss, see Subsection A.3, and the arrow $\widetilde{\deg}(\mathrm{BPM_n}) \to \log\|\mathrm{BPM_n^\star}\|_1$ represents the bound provided by Lemma 31. Green blocks correspond to bounds on their adjacent quantity. Every arrow $f \to g$ is accompanied by the corresponding citation in blue, apart from trivial relations wherein they are omitted. Bounds and relations marked [Here] denote results shown in this paper.

The following table details the complexity measures appearing in Figure 7.

Query Complexity Measures	
Measure	**Definition**
$\mathrm{Demand}(\mathrm{BPM_n})$	The least depth of a decision tree computing $\mathrm{BPM_n}$, whose internal nodes are labeled by demand queries.
$D^{\mathrm{OR}}(\mathrm{BPM_n})$	The least depth of a decision tree computing $\mathrm{BPM_n}$, whose internal nodes are labeled by ORs over arbitrary subsets of the input bits.
$D^{\mathrm{DISJ}}(\mathrm{BPM_n})$	The least depth of a decision tree computing $\mathrm{BPM_n}$, whose internal nodes are labeled by *disjunctions* over *literals*, e.g. $(x_1 \vee \bar{x}_3 \vee x_7)$.
$\mathrm{DT}^{\mathrm{SIZE}}(\mathrm{BPM_n})$	The least amount of leaves in a *classical* decision tree computing $\mathrm{BPM_n}$.
$Q_2(\mathrm{BPM_n})$	The bounded-error quantum query complexity of $\mathrm{BPM_n}$.

Communication Complexity Measures	
Measure	**Definition**
$CC(\mathrm{BPM_n})$	The two-party deterministic communication complexity of $\mathrm{BPM_n}$, where we fix an *arbitrary* partition over the input bits.
$rk(M_{\mathrm{BPM_n}})$	The real rank of the communication matrix corresponding to the above communication problem.

Algebraic Complexity Measures	
Measure	**Definition**
$\lvert mon(\mathrm{BPM_n^\star})\rvert$	Number of non-zero coefficients in the unique representing polynomial.
$\lVert \mathrm{BPM_n^\star} \rVert_1$	Sum of magnitudes of coefficients in the unique representing polynomial.
$\lambda(\mathrm{BPM_n})$	The spectral sensitivity of $\mathrm{BPM_n}$ (see Definition 34).
$\widetilde{\deg}(\mathrm{BPM_n})$	The approximate degree of $\mathrm{BPM_n}$.

A.2 Drawing the Connections

Decision Tree Measures

It is not hard to see that every demand query can be simulated by at most logarithmically many OR-queries, by performing binary search on the right vertices. Similarly trivially, every OR query can be seen as a disjunction wherein no literal is negated, thus we also have $D^{\mathrm{DISJ}}(\mathrm{BPM_n}) \leq D^{\mathrm{OR}}(\mathrm{BPM_n})$. The latter quantity, $D^{\mathrm{DISJ}}(\mathrm{BPM_n})$, is of particular interest – Nisan observed [22] that for any Boolean function the least *depth* of a *disjunction decision tree* computing the function is equivalent, up to a $\log n$-factor, to the minimum *size* (i.e., number of leaves) of a *classical decision tree* computing it. The minimum decision tree size computing a Boolean function is known to be related to Fourier-analytic properties of the function. For example, a folklore result states that it is lower bounded by the Fourier ℓ_1-norm of the function (see e.g. [25]).

Communication Complexity Measures

Given a disjunction decision tree computing a Boolean function, one naturally obtains a corresponding 2-party deterministic communication protocol. The protocol simply simulates the tree by "solving", at every step, the current disjunction. This simulation can be done efficiently, since any disjunction requires only 2-bits of communication (Alice and Bob compute their parts of the disjunction separately, and communicate the answer bits to one another). In the argument above, the actual partition determining Alice and Bob's shares of the input bits is inconsequential. For any such fixed partition, one can consider the *communication matrix*, which is the Boolean matrix whose rows are indexed by Alice's inputs, and columns by Bob's inputs. It is well known (by a result of [19]), that the log of the real rank of this matrix yields a lower bound on the deterministic communication complexity of its corresponding problem.

The ℓ_1-norm of $\mathrm{BPM_n^\star}$

A surprisingly pivotal complexity measure arising in Figure 7 is the ℓ_1-norm of the dual function of matching. Firstly, this measure trivially bounds the number of *monomials* appearing in its representing polynomial (since all coefficients are integers), which in turn

bounds the rank of the communication matrix, by a classical result of [24] (every monomial corresponds to a rank-1 matrix). The same quantity, $\|\mathrm{BPM}_n^\star\|_1$, also bounds the Fourier ℓ_1-norm of BPM_n (equivalently $\mathrm{BPM}_n^\star$), as we observe in Corollary 27, as well as yielding bounds on the approximate degree, via the scheme detailed in Lemma 31. With regards to lower bounds, in [3] it was shown that for any Boolean function f it holds that $\log \|f^\star\|_1$ lower bounds the least depth of an OR decision tree computing f.

Through our complete characterization of the dual polynomial given in Theorem 2, we were able to deduce the tight bound $\log \|\mathrm{BPM}_n^\star\|_1 = \Theta(n \log n)$ (see Corollary 27), thereby implying all of the aforementioned bounds. We conjecture that this low-norm representation of $\mathrm{BPM}_n^\star$ has more far-reaching consequences – in particular, that it can be used to construct a quasi-linear deterministic communication protocol for the bipartite matching problem.[7]

Approximate Degree and Quantum Query Complexity

The "polynomial method" in quantum computation [2] states that the acceptance probability of any d query quantum algorithm can be written as a degree $2d$ polynomial. Thus, the approximate degree of any Boolean function serves as a lower bound on its Quantum query complexity. In this paper we have obtained tight upper and lower bounds on this quantity, showing that $\widetilde{\deg}(\mathrm{BPM}_n) = \widetilde{\Theta}(n^{3/2})$. To complete the connections specified in Figure 7, it remains to relate the quantum query complexity to our main object of study; $\mathrm{Demand}(\mathrm{BPM}_n)$.

A.3 Quantum Bounds Imply Combinatorial Bounds

In this section, we make one final simple observation regarding the demand query model: *demand query algorithms can be efficiently simulated by quantum queries*. Recall that every demand query can be simulated by logarithmically many OR-queries, each over at most n bits (corresponding to the right vertices). In his seminal paper, Grover [12] showed that the OR_n function can be computed, to constant error, using $\mathcal{O}(\sqrt{n})$ quantum queries (which is tight, see [4]). Thus, replacing each demand query by the majority over several invocations of Grover's algorithm, and using Chernoff's bound to suitably reduce the error, we obtain:[8]

▶ **Proposition 43.** *If there exists a demand query algorithm for* BPM_n *making at most* d *queries, then:*

$$Q_2(\mathrm{BPM}_n) = \mathcal{O}(\sqrt{n} \cdot d \cdot \mathrm{polylog}(d)).$$

Consequently, any lower bound of the form $Q_2(\mathrm{BPM}_n) = \Omega(n^{3/2+\varepsilon})$, for some constant $\varepsilon > 0$, would imply a (polynomially) super-linear lower bound on the demand query complexity of BPM_n, thereby resolving Open Question 42. Such a result might suggest that quasi-linear *combinatorial algorithms* for bipartite perfect matching are improbable, which we consider a very interesting prospect. Nevertheless, at present the quantum query complexity of BPM_n remains undetermined. Lin and Lin [18] constructed an efficient quantum algorithm, yielding

[7] Indeed, it is not hard to show that for any *monotone* Boolean function $f : \{0,1\}^n \to \{0,1\}$ and any partition over its inputs, we have $CC(f) \leq \min\{|mon(f)|, |mon(f^\star)|\}^2$, which can be seen as a single step towards this direction.

[8] In fact, by a similar approach we can also show that for any Boolean function $f : \{0,1\}^n \to \{0,1\}$, the quantum query complexity is bounded by $Q_2(f) = \mathcal{O}\left(\sqrt{n} \cdot \log \mathrm{DT}^{\mathrm{SIZE}}(f) \cdot \log\log \mathrm{DT}^{\mathrm{SIZE}}(f)\right)$, where $\mathrm{DT}^{\mathrm{SIZE}}(f)$ is the minimal *size* of a classical decision tree computing f. This observation might be useful in cases where there exist relatively "unbalanced" decision trees computing f.

an upper bound of $\mathcal{O}\left(n^{7/4}\right)$. Conversely, through Ambainis' adversary technique, Zhang [32] has obtained an upper bound of $\Omega\left(n^{3/2}\right)$. Our main theorem (Theorem 1) implies that this lower bound *cannot be (polynomially) strengthened by the "Polynomial Method"*. In fact, neither can Ambainis' adversary bounds be used to this end, since it is known (see e.g. [32]) that the best bound attainable by this method cannot exceed $\sqrt{C_0(f)C_1(f)}$. Closing this gap is left as an open problem.

▶ **Open Problem 44** (Quantum Query Complexity of Matching)**.** *Close the gap on* $Q_2(\mathrm{BPM_n})$.

On the Satisfaction Probability of k-CNF Formulas

Till Tantau ✉
Institute for Theoretical Computer Science, Universität zu Lübeck, Germany

Abstract

The *satisfaction probability* $\sigma(\phi) := \Pr_{\beta:\mathrm{vars}(\phi)\to\{0,1\}}[\beta \models \phi]$ of a propositional formula ϕ is the likelihood that a random assignment β makes the formula true. We study the complexity of the problem kSAT-PROB$_{>\delta} = \{\phi$ is a kCNF formula $\mid \sigma(\phi) > \delta\}$ for fixed k and δ. While 3SAT-PROB$_{>0}$ = 3SAT is NP-complete and SAT-PROB$_{>1/2}$ is PP-complete, Akmal and Williams recently showed 3SAT-PROB$_{>1/2} \in$ P and 4SAT-PROB$_{>1/2} \in$ NP-complete; but the methods used to prove these striking results stay silent about, say, 4SAT-PROB$_{>3/4}$, leaving the computational complexity of kSAT-PROB$_{>\delta}$ open for most k and δ. In the present paper we give a complete characterization in the form of a trichotomy: kSAT-PROB$_{>\delta}$ lies in AC^0, is NL-complete, or is NP-complete; and given k and δ we can decide which of the three applies. The proof of the trichotomy hinges on a new order-theoretic insight: Every set of kCNF formulas contains a formula of maximal satisfaction probability. This deceptively simple result allows us to (1) kernelize kSAT-PROB$_{\geq\delta}$, (2) show that the variables of the kernel form a strong backdoor set when the trichotomy states membership in AC^0 or NL, and (3) prove a locality property by which for every kCNF formula ϕ we have $\sigma(\phi) \geq \delta$ iff $\sigma(\psi) \geq \delta$ for every fixed-size subset ψ of ϕ's clauses. The locality property will allow us to prove a conjecture of Akmal and Williams: The majority-of-majority satisfaction problem for kCNFS lies in P for all k.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Satisfaction probability, majority {k}-sat, kernelization, well orderings, locality

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.2

Related Version *Full Version*: https://arxiv.org/abs/2201.08895

1 Introduction

For a propositional formula ϕ like $(x \vee y \vee z) \wedge (\neg x \vee \neg y) \wedge (y \vee z)$ it is, in general, a very hard problem to obtain much information about the number $\#(\phi)$ of satisfying assignments or, equivalently, about the *satisfaction probability* $\sigma(\phi)$ defined as

$$\sigma(\phi) := \Pr_{\beta:\mathrm{vars}(\phi)\to\{0,1\}}\big[\beta \models \phi\big] = \frac{\#(\phi)}{2^n}$$

where $n = |\mathrm{vars}(\phi)|$ is the number of variables in ϕ. By Cook's Theorem [7] it is already NP-complete to determine whether $\sigma(\phi) > 0$ holds; and to determine whether $\sigma(\phi) > 1/2$ holds is complete for PP. Indeed, the function $\#(\cdot)$ itself is complete for #P, a counting class high up in the complexity hierarchies. Writing SAT-PROB$_{>\delta}$ for $\{\phi \mid \sigma(\phi) > \delta\}$, we can rephrase these results as "SAT-PROB$_{>0}$ is NP-complete" (Cook's Theorem) and "SAT-PROB$_{>1/2}$ is PP-complete" (and so is SAT-PROB$_{\geq 1/2}$, see for instance [13, Theorem 4.1]).

Cook's result on the complexity of SAT-PROB$_{>0}$ = SAT is quite robust regarding the kinds of formulas one considers: The problem stays NP-complete for formulas in CNFS, the set of formulas in conjunctive normal form, so CNF-SAT-PROB$_{>0}$ = CNF-SAT $\in$ NP-complete, and even for formulas $\phi \in$ 3CNFS, that is, when all clauses of ϕ have at most three literals, so 3SAT-PROB$_{>0}$ = 3SAT $\in$ NP-complete. Similarly, CNF-SAT-PROB$_{>1/2}$ has the same complexity as SAT-PROB$_{>1/2}$, see [13, lemma on page 80], and $\#(\cdot)$ is still #P-hard for formulas in 3CNFS and even in 2CNFS, see [15].

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 2; pp. 2:1–2:27

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

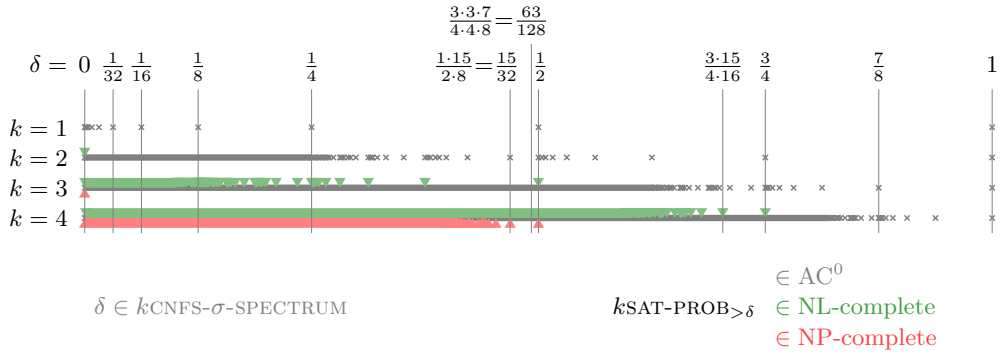



Figure 1 Visualization of the complexity of kSAT-PROB$_{>\delta}$ for $k = 1$, $k = 2$, $k = 3$, and $k = 4$. Each green triangle represents a value of δ for which the problem is NL-complete, while each red triangle represents a value for which it is NP-complete (so there is actually no green triangle directly above a red triangle, but this is not possible to visualize as these are highly intertwined). Each gray cross is an element of kCNFS-σ-SPECTRUM, for which kSAT-PROB$_{>\delta}$ lies in AC^0 according to Theorem 1.11. For "white" values of δ, which lie outside the spectra, kSAT-PROB$_{>\delta}$ also lies in AC^0. Note that while the visualization may suggest that the spectra become dense close to 0, they are in fact nowhere dense (being well-ordered by $>$). For a discussion of the marked specific values like $\delta = 63/128$, please see the conclusion.

In sharp contrast to these well-established hardness results, Akmal and Williams [2] recently showed that 3SAT-PROB$_{>1/2}$ can be solved in polynomial time – in fact, they show this time bound for 3SAT-PROB$_{>\delta} = \{\phi \in 3\text{CNFS} \mid \sigma(\phi) > \delta\}$ for all rational $\delta > 0$ (intriguingly, their argument does not apply to non-rational δ and leaves open the complexity of, say, 3SAT-PROB$_{>1/\pi}$). Yet again in contrast, they also show that kSAT-PROB$_{>1/2}$ is NP-complete for every $k \geq 4$. To complicate things even further, the NP-completeness result for 4SAT-PROB$_{>1/2}$ can easily be extended to 4SAT-PROB$_{>1/4}$, to 4SAT-PROB$_{>1/8}$, to 4SAT-PROB$_{>1/16}$ and so on, and with very little extra work to more exotic values of δ like $\delta = 15/32$ – but apparently *not* to certain other values like $\delta = 3/4$ or $\delta = 63/128$. Indeed, 4SAT-PROB$_{>15/16}$ is a *trivial* problem (lies in AC^0) as *every* nonempty formula in 4CNFS has a satisfaction probability of at most 15/16 (a single nontrivial clause already rules out 1/16th of all assignments). In other words, even for a fixed k the complexity of kSAT-PROB$_{>\delta}$ might fluctuate wildly for changing δ (does, in fact, as Figure 1 makes quite clear) and it is unclear how the methods introduced in [2] could be used to show that, say, 4SAT-PROB$_{>63/128}$ is NL-complete while 3SAT-PROB$_{>1/\pi}$ lies in AC^0.

1.1 Contributions of This Paper

In the present paper we continue the investigation of the complexity of the satisfaction probability function for kCNF formulas initiated by Akmal and Williams. We will look at this complexity from three different angles – order-theoretic, algorithmic, and complexity-theoretic – and now sketch the main results we obtain for each of these aspects. (A small remark on notation first, however: For convenience, in this paper we make no difference between a CNF formula and its set of clauses. Each $\phi \in$ CNFS is a finite *set* of clauses, which are sets of literals – so the formula from the paper's first line is actually $\phi = \big\{\{x, y, z\}, \{\neg x, \neg y\}, \{y, z\}\big\} \in 3\text{CNFS}$ – and we forbid already syntactically that clauses contain both a variable and its negation, so $\big\{\{x, \neg x\}\big\} \notin$ CNFS, while $\big\{\{x\}, \{\neg x\}\big\} \in 1\text{CNFS}$.)

First, we address the *order-theoretic* properties of the number set kCNFS-σ-SPECTRUM $:= \{\sigma(\phi) \mid \phi \in k\text{CNFS}\} \subseteq [0,1]$ for different k and show that for all k the following holds:

▶ **Theorem 1.1** (Spectral Well-Ordering Theorem). *kCNFS-σ-SPECTRUM is well-ordered by $>$.*

Here, a set $X \subseteq [0,1]$ is called *well-ordered by* $>$ if there is no infinite strictly *increasing* sequence of elements of X or, equivalently, if every subset of X contains a maximal element.

An equivalent way of stating the Spectral Well-Ordering Theorem is as follows (and observe that the corollary certainly does not hold when we replace "maximal" by "minimal" as the set $\Phi = \big\{\big\{\{x_1\},\dots,\{x_n\}\big\} \mid n \in \mathbb{N}\big\}$ shows):

▶ **Corollary 1.2.** *Every $\Phi \subseteq k$CNFS contains a formula of maximal satisfaction probability.*

Yet another equivalent way of stating the Spectral Well-Ordering Theorem is terms of the *spectral gaps below* $\delta \in [0,1]$*:*

▶ **Definition 1.3.** *Let* spectral-gap$_{k\text{CNFS}}(\delta) := \sup\{\epsilon \mid (\delta-\epsilon,\delta) \cap k\text{CNFS-}\sigma\text{-SPECTRUM} = \emptyset\}$.

In Figure 1, the spectral gaps can be "seen" directly: For any position δ (not necessarily a member of the spectrum), the spectral gap "stretches left till the next cross (member of the spectrum)." The key observation is that there always *is* a stretch:

▶ **Corollary 1.4.** *For all k and $\delta \in [0,1]$, we have* spectral-gap$_{k\text{CNFS}}(\delta) > 0$.

As we will see, this deceptively simple statement has far-reaching consequences.

Second, we use the existence of spectral gaps below all $\delta \in [0,1]$ to develop three new *algorithmic* methods for showing kSAT-PROB$_{\geq\delta} \in \text{AC}^0$ (note the "$_{\geq\delta}$" rather than "$_{>\delta}$" subscript). They are simpler than the algorithm of Akmal and Williams and make the tools of fixed-parameter tractability (FPT) theory accessible for the analysis. One of them, while impractical and having by far the worst resource bounds of the three algorithms (but still in P), has an underlying idea that is of independent interest (recall that we consider CNF formulas to be sets of clauses):

▶ **Lemma 1.5** (Threshold Locality Lemma). *For each k and each $\delta \in [0,1]$ there is a $C \in \mathbb{N}$ so that for all $\phi \in k$CNFS we have $\sigma(\phi) \geq \delta$, iff $\sigma(\psi) \geq \delta$ holds for all $\psi \subseteq \phi$ with $|\psi| \leq C$.*

While the above lemma is foremost a mathematical statement about properties of kCNF formulas, we can trivially derive a decision algorithm for kSAT-PROB$_{\geq\delta}$ from it (Algorithm 2 later on): Iterate over all $\psi \subseteq \phi$ with $|\psi| \leq C$ and compute $\sigma(\psi)$ by brute force ($|\psi|$ is constant) and accept if all computed values are at least δ.

The Threshold Locality Lemma also lies at the heart of a proof of MAJ-MAJ-kSAT $\in \text{AC}^0$ for all k. The problem is defined as follows: Given a formula $\phi \in k$CNFS with vars$(\phi) \subseteq X \cup Y$, determine whether for a majority of the assignments $\beta\colon X \to \{0,1\}$ the majority of extensions of β to $\beta'\colon X \cup Y \to \{0,1\}$ makes ϕ true. Akmal and Williams [2] conjectured MAJ-MAJ-kSAT $\in$ P. Corollary 3.10 shows that this is, indeed, the case.

Third, we apply the developed theory to kSAT-PROB$_{>\delta}$; a problem whose complexity is somewhat more, well, complex than that of kSAT-PROB$_{\geq\delta}$. It turns out that a complete classification of the complexity for all values of k and δ is possible in the form of a trichotomy:

▶ **Theorem 1.6** (Spectral Trichotomy Theorem). *Let $k \geq 1$ and $\delta \in [0,1]$ be a real number. Then kSAT-PROB$_{>\delta}$ is* NP*-complete or* NL*-complete or lies in* AC^0.

In the following, we explore each of the above "points of view" in a bit more detail and have a brief look at the main ideas behind the proofs of the main results. Later, each angle will be addressed in detail in a dedicated main section of this paper.

Order-Theoretic Results. In a sense, the "reason" why the functions $\sigma(\cdot)$ and $\#(\cdot)$ are so hard to compute in general, lies in the fact that the spectrum CNFS-σ-SPECTRUM $= \{\sigma(\phi) \mid \phi \in \text{CNFS}\} = \bigcup_k k$CNFS-$\sigma$-SPECTRUM is just the set $\mathbb{D}$ of dyadic rationals (numbers of the form $m/2^e$ for integers m and e) between 0 and 1 (see Lemma 2.1). In particular, it is a dense subset of $[0,1]$ and in order to conclusively decide whether, say, $\sigma(\phi) \geq 1/3$ holds for an arbitrary $\phi \in$ CNFS, we may need to determine all of the first n bits of $\sigma(\phi)$.

A key insight of Akmal and Williams is that for fixed k, the spectra kCNFS-σ-SPECTRUM behave differently, at least near to 1: There are "holes" like 3CNFS-σ-SPECTRUM$\cap(7/8,1) = \emptyset$ since for a 3CNF formula ϕ we cannot have $7/8 < \sigma(\phi) < 1$ (a single size-3 clause already lowers the satisfaction probability to at most 7/8). This implies immediately that, say, 3SAT-PROB$_{>9/10}$ is actually a quite trivial problem: The *only* formula in 3CNFS having a satisfaction probability larger than 9/10 has probability 1 and is the trivial-to-detect tautology $\phi = \emptyset$. In general, for all k we have kCNFS-σ-SPECTRUM $\cap\, (1-2^{-k},1) = \emptyset$.

Of course, 3CNFS-σ-SPECTRUM does not have "holes above every number δ" as we can get arbitrarily close to, say, $\delta = 3/4$: Just consider the sequence of 3CNF formulas $\phi_1 = \big\{\{a,b,x_1\}\big\}$, $\phi_2 = \big\{\{a,b,x_1\},\{a,b,x_2\}\big\}$, $\phi_3 = \big\{\{a,b,x_1\},\{a,b,x_2\},\{a,b,x_3\}\big\}$, and so on with $\sigma(\phi_i) = 3/4 + 2^{-i-2}$ and $\lim_{i\to\infty}\sigma(\phi_i) = 3/4$. Nevertheless, Akmal and Williams point out that their algorithm is in some sense based on the intuition that there are "lots of holes" in kCNFS-σ-SPECTRUM. The new Spectral Well-Ordering Theorem, Theorem 1.1 above, turns this intuition into a formal statement.

Well-orderings are a standard notion of order theory; we will just need the special case that we are given a set X of non-negative reals and consider the total order $>$ on it. Then X *is well-ordered (by $>$)* if there is no infinite strictly *increasing* sequence $x_0 < x_1 < x_2 < \cdots$ of numbers $x_i \in X$ or, equivalently, if X is bounded and for every $x \in \mathbb{R}^{\geq 0}$ there is an $\epsilon > 0$ such that $(x-\epsilon, x) \cap X = \emptyset$ or, again equivalently, if every subset of X contains a maximal element. In particular, Theorem 1.1 tells us that every $\Phi \subseteq k$CNFS contains a formula $\phi \in \Phi$ of maximal satisfaction probability, that is, $\sigma(\phi) \geq \sigma(\phi')$ for all $\phi' \in \Phi$.

Observe that kCNFS-σ-SPECTRUM is certainly not well-ordered by $<$, only by $>$. However, with a small amount of additional work we will be able to show that it is at least *topologically closed,* see Lemma 3.5. A succinct way of stating both this and Theorem 1.1 is that for every set $X \subseteq k$CNFS-σ-SPECTRUM of numbers we have $\sup X \in X$ and $\inf X \in k$CNFS-σ-SPECTRUM. An interesting corollary of this is that non-rational, non-dyadic thresholds can always be replaced by dyadic rationals (recall $\mathbb{D} = \{m/2^e \mid m, e \in \mathbb{Z}\}$):

▶ **Corollary 1.7** (of Lemma 3.5). *For every non-dyadic $\delta \in [0,1] \setminus \mathbb{D}$, for $\delta' = \inf\{\sigma(\phi) \mid \phi \in k\text{CNFS}, \sigma(\phi) > \delta\} \in \mathbb{D}$ we have kSAT-PROB$_{>\delta} = k$SAT-PROB$_{\geq\delta} = k$SAT-PROB$_{\geq\delta'}$.*

The proof of Theorem 1.1 will need only basic properties of well-ordered sets of reals, like their being closed under finite sums and unions, and the following simple relationship (which also underlies the analysis of Akmal and Williams [2]) between the satisfaction probability of a formula ϕ and the size of *packings* $\pi \subseteq \phi$, which are just sets of pairwise variable-disjoint clauses:

▶ **Lemma 1.8** (Packing Probability Lemma). *Let $\phi \in k$CNFS and let $\pi \subseteq \phi$ be a packing. Then $\sigma(\phi) \leq (1-2^{-k})^{|\pi|}$ and, equivalently, $\log_{1-2^{-k}}(\sigma(\phi)) \geq |\pi|$.*

Proof. We have $\sigma(\phi) \leq \sigma(\pi) = \prod_{c\in\pi}(1-2^{-|c|}) \leq (1-2^{-k})^{|\pi|}$ as all clauses of π are variable-disjoint and, hence, their satisfaction probabilities are pairwise independent. ◀

A simple consequence of the Packing Probability Lemma will be that for every $\phi \in k$CNFS, we can write $\sigma(\phi)$ as a sum $\sum_{i=1}^{s}\sigma(\phi_i)$ with $\phi_i \in (k-1)$CNFS in such a way that s depends only on $\sigma(\phi)$, which will almost immediately yield Theorem 1.1.

Algorithmic Results. The algorithm for deciding kSAT-PROB$_{\geq\delta}$ in [2] is complex (in the words of the authors from the technical report version [1]: "[it] depends on quite a few parameters, so the analysis becomes rather technical" and, indeed, the description of these parameters and constants alone takes one and a half pages, followed by eleven pages of analysis). Surprisingly, the purely order-theoretic Theorem 1.1 allows us to derive three simple algorithms for kSAT-PROB$_{\geq\delta}$, Algorithms 1, 2 and 3, whose underlying ideas are briefly described in the following.

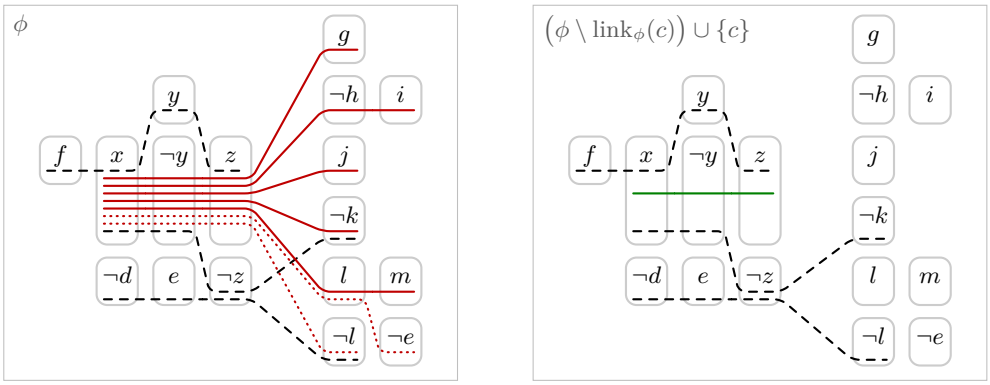


Figure 2 On the left, a formula $\phi \in 5$CNFS is visualized by drawing, for each clause in ϕ, a line that "touches" exactly the clause's literals; so the upper dashed line represents the clause $\{f, x, y, z\}$. The "solid clauses" (meaning "clauses represented by a solid line") form a sunflower $\psi \subseteq \phi$ with core $c = \{x, \neg y, z\}$. All red clauses, including the dotted clauses, are part of the link of this core, but the dotted clauses are not part of the sunflower: The upper dotted clause $\{x, \neg y, z, l, \neg e\}$ shares the literal "l" with the petal $\{x, \neg y, z, l, m\}$ of the sunflower, while the second dotted clause shares the variable "l" (though not the literal) with this petal. The dashed clauses are not part of the link (let alone the sunflower) as they do not contain all of the literals of the core (containing the variables is not enough). A key property of a sunflower is that it is "unlikely that an assignment makes the sunflower true, but not its core": For ϕ, this happens only when g, j, and $\neg k$ are all set to true as well as at least one of $\neg h$ or i, and one of l or m. The probability that all of this happens is just $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{3}{4} = \frac{9}{128}$, meaning that this is the maximal difference in the satisfaction probabilities of ϕ and the formula shown right, the result of "collapsing the link to the core c".

The first algorithm is based on *sunflowers,* just like that of Akmal and Williams, but uses them in (far) less complex ways. They can be thought of as generalizations of packings, which are, indeed, the special case of a sunflower with an empty core.

▶ **Definition 1.9.** *A* sunflower with core c *is a formula* $\psi \in$ CNFS *such that* $c \subseteq e$ *holds for all* $e \in \psi$ *and such that for any two different* $e, e' \in \psi$ *we have* $\text{vars}(e) \cap \text{vars}(e') = \text{vars}(c)$.

In other words, the clauses of a sunflower "agree on the literals in c, but are variable-disjoint otherwise", see the clauses represented by solid lines in Figure 2 for an example.

The importance of sunflowers lies in an easy observation: If ψ is a sunflower with core c, then $\sigma(\psi) = \sigma(\{c\}) + \sigma(\{e \setminus c \mid e \in \psi\})$. As $\{e \setminus c \mid e \in \psi\}$ is clearly a packing, the Packing Probability Lemma implies that the probability that an assignment β satisfies a sunflower ψ, but not its core, is at most $(1 - 2^{-k})^{|\psi|}$, which is a value that decreases exponentially as the size $|\psi|$ of the sunflower increases. Now for fixed k and $\delta \in [0, 1]$ consider Algorithm 1. Function KERNELIZE(ϕ) will be familiar to readers interested in FPT theory: This is exactly the standard kernel algorithm for the hitting set problem with bounded hyperedge size based on "collapsing sunflowers" or, more precisely, on "collapsing the links $\text{link}_\phi(c) = \{e \in \phi \mid c \subseteq e\}$ for cores c of large sunflowers." In particular, standard results from

Algorithm 1 The SUNFLOWER-COLLAPSING algorithm decides whether $\sigma(\phi) \geq \delta$ holds for $\phi \in k\text{CNFS}$ and fixed $\delta \in [0,1]$. The "spectral gap" in the algorithm is the size of the "hole" below δ in $k\text{CNFS-}\sigma\text{-SPECTRUM}$, see Definition 1.3. The $\text{link}_\phi(c)$ is simply the set $\{e \in \phi \mid c \subseteq e\}$. The test $\sigma(\kappa) \geq \delta$ can be performed by "brute force" as the returned kernel κ will have fixed size. The algorithm's correctness follows from the fact that in a collapsing step (the assignment in line 3), the satisfaction probability cannot "tunnel through" the spectral gap below δ, see Figure 3.

```
1  algorithm KERNELIZE(φ, h)
2      while φ contains a sunflower of size at least h + 1 with some core c do
3          φ ← (φ \ link_φ(c)) ∪ {c}
4      return φ
5
6  algorithm SUNFLOWER-COLLAPSING(φ) // φ ∈ kCNFS must hold
7      κ ← KERNELIZE(φ, log_{1−2^{−k}}(spectral-gap_{kCNFS}(δ))) // See Definition 1.3
8      if σ(κ) ≥ δ then return "σ(φ) ≥ δ" else return "σ(φ) < δ"
```

FPT theory (in this case, Erdős' Sunflower Lemma) tell us that κ will have a size depending only on h and k and, thus, κ *has constant size* (that is, a size depending only on h) and we can clearly compute $\sigma(\kappa)$ in constant time when κ has only a constant number of clauses and hence also a constant number of variables.

The crucial question is, of course, why this algorithm works, that is, why should $\sigma(\kappa) \geq \delta$ iff $\sigma(\phi) \geq \delta$ hold? The deeper reason is Corollary 1.4 to the Spectral Well-Ordering Theorem, by which there is a gap of size $\epsilon := \text{spectral-gap}_{k\text{CNFS}}(\delta) > 0$ below δ in $k\text{CNFS-}\sigma\text{-SPECTRUM}$ and *we cannot tunnel through this gap* by the following lemma (see also Figure 3):

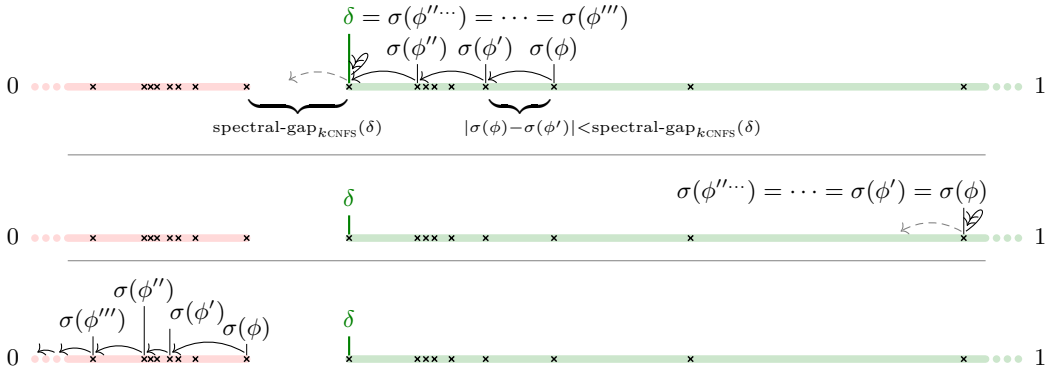


Figure 3 Three ways how $\sigma(\phi)$ can change due to the assignment $\phi' \leftarrow (\phi \setminus \text{link}_\phi(c)) \cup \{c\}$ in Algorithm 1: In each row, the crosses in the red lines are elements of $k\text{CNFS-}\sigma\text{-SPECTRUM}$ smaller than δ, while the crosses on the green lines are at least δ. In the first line, $\sigma(\phi) \geq \delta$ holds and each assignment yields a new ϕ' such that $\sigma(\phi')$ is not larger – but $|\sigma(\phi) - \sigma(\phi')| < \text{spectral-gap}_{k\text{CNFS}}(\delta)$ ensures that $\sigma(\phi''^{\cdots})$ gets "stuck" at δ as it cannot "tunnel through" the spectral gap by Lemma 1.10 and the dashed arrow is an impossible change in the satisfaction probability. In the second line, $\sigma(\phi)$ is stuck at a much larger value than δ. In the third line, $\sigma(\phi)$ is below δ and can get smaller and smaller (unless there is another gap of size $\text{spectral-gap}_{k\text{CNFS}}(\delta)$ further down).

▶ **Lemma 1.10** (No Tunneling Lemma). *For $\delta \in [0,1]$, let any two $\phi, \phi' \in k\text{CNFS}$ be given with $|\sigma(\phi) - \sigma(\phi')| < \text{spectral-gap}_{k\text{CNFS}}(\delta)$. Then $\sigma(\phi) \geq \delta$ iff $\sigma(\phi') \geq \delta$.*

Proof. W. l. o. g. $\sigma(\phi) \geq \sigma(\phi')$. Clearly $\sigma(\phi') \geq \delta$ implies $\sigma(\phi) \geq \delta$. If $\sigma(\phi) \geq \delta$, then $\sigma(\phi') > \delta - \text{spectral-gap}_{k\text{CNFS}}(\delta)$. As $\sigma(\phi')$ cannot lie in the spectral gap, $\sigma(\phi') \geq \delta$. ◀

The crucial assignment $\phi' = (\phi \setminus \text{link}_\phi(c)) \cup \{c\}$ in the kernelization removes the link of c in ϕ, which is just the set of all superclauses of c (which includes all clauses of the sunflower), but adds the core. *This can only reduce the satisfaction probability of ϕ by at most $\sigma(\{e \setminus c \mid e \in \psi\})$.* As this probability is at most $(1-2^{-k})^{|\psi|}$ by the Packing Probability Lemma, if the sunflower is larger than $h = \log_{1-2^{-k}}\big(\text{spectral-gap}_{k\text{CNFS}}(\delta)\big)$, then $\sigma(\phi) - \sigma(\phi') < \text{spectral-gap}_{k\text{CNFS}}(\delta)$. Thus, by the No Tunneling Lemma, $\sigma(\phi) \geq \delta$ iff $\sigma(\phi') \geq \delta$ as it is impossible that the satisfaction probability of ϕ "tunnels through the interval $(\delta - \text{spectral-gap}_{k\text{CNFS}}(\delta), \delta)$" in any step of the while loop.

Algorithm 2 A simple algorithm for deciding kSAT-PROB$_{\geq\delta}$. Note that while the algorithm runs in polynomial time (in fact, in AC^0), on input ϕ the runtime is of the form $(2|\phi|)^{C+1}$ for a (possibly huge) constant C. Phrased in terms of FPT theory, unlike Algorithm 1, the algorithm has an "XP-runtime" rather than an "FPT-runtime".

```
1  algorithm LOCALITY-BASED(φ) // φ ∈ kCNFS must hold
2      C ← (2(1 + log_{1−2^{−k}}(spectral-gap_{kCNFS}(δ))))^k · k!
3      foreach ψ ⊆ φ of size at most C do // at most |φ|^{C+1} subsets
4          if σ(ψ) < δ then // check by brute force
5              return "σ(φ) < δ"
6      return "σ(φ) ≥ δ"
```

Algorithm 2 is even simpler than the just-presented kernel algorithm. For its correctness, first note that the output in line 5 is certainly correct as $\psi \subseteq \phi$ implies $\sigma(\phi) \leq \sigma(\psi)$. To to see that the output in line 6 is correct (which is essentially the claim of Lemma 1.5, the Threshold Locality Lemma), consider a ϕ for which $\sigma(\phi) < \delta$ holds. Starting with $\psi = \phi$, as long as possible, pick a clause $e \in \psi$ such that $\sigma(\psi \setminus \{e\}) < \delta$ still holds and set $\psi \leftarrow \psi \setminus \{e\}$. For the ψ obtained in this way, $\sigma(\psi) < \delta$, $\psi \subseteq \phi$, and for all clauses $e \in \psi$ we have $\sigma(\psi \setminus \{e\}) \geq \delta$. Now, *$\psi$ cannot contain a large sunflower* since, otherwise, we could remove any petal e of the sunflower and this *would raise the satisfaction probability by at most the spectral gap.* In particular, by the No Tunneling Lemma, $\sigma(\psi \setminus \{e\}) < \delta$ would still hold, contradicting that we can no longer remove clauses from ψ. By Erdős' Sunflower Lemma we conclude that $|\psi| \leq C$ for a constant C depending only on k and δ.

The Threshold Locality Lemma will also play a key role in the proof of Corollary 3.10, which states that MAJ-MAJ-kSAT $\in \text{AC}^0$ holds for all k. In the proof, we use the lemma to turn certain satisfaction probability threshold problems (questions of the form "$\sigma(\rho) \geq \delta$?") into model checking problems (questions of the form "$\beta \models \omega$?"). This will allow us to present a reduction of MAJ-MAJ-kSAT to lSAT-PROB$_{\geq 1/2}$ for some (large) l and we know already that the latter problem lies in AC^0.

The third algorithm addresses the well-established observation from FPT theory that while a kernel algorithm for a parameterized problem is in some sense the best one can hope for from a theoretical point of view, from a practical point of view it is also of high interest how we can actually decide whether $\sigma(\kappa) \geq \delta$ holds for a kernel: Of course, as the kernel size is fixed, this can be decided in constant time by brute-forcing all assignments – but a practical algorithm will need to use different ideas, such as those used in Algorithm 3. This algorithm uses the fact that if on input ϕ we can compute an interval $I \subseteq [0,1]$ with $\sigma(\phi) \in I$ and $|I| < \epsilon = \text{spectral-gap}_{k\text{CNFS}}(\delta)$, then we will have $\sigma(\phi) < \delta$ iff $\max I < \delta$. The key insight is that we can compute such an interval recursively: If the interval returned by INTERVAL(ϕ) is not yet small enough, by the Packing Probability Lemma, ϕ must contain a small maximal packing, which we call PACK(ϕ) and which can be obtained greedily. We can

Algorithm 3 A recursive algorithm for deciding $\sigma(\phi) \geq \delta$. The recursion computes an interval I with $\sigma(\phi) \in I$ and $|I| \leq \epsilon$, by first obtaining a candidate interval using $\text{INTERVAL}(\phi)$. If this is too large, by the Packing Probability Lemma $\text{PACK}(\phi)$ (a maximal packing obtained greedily) is small and we can write $\sigma(\phi)$ as the sum $\sum_\beta \sigma(\phi|_\beta)$, where $\phi|_\beta$ has as its models (= satisfying assignments) exactly all models β' of ϕ that extend (or "agree with") β. Crucially, $\phi|_\beta$ will be a $(k-1)$CNF formula, if ϕ was a kCNF formula, meaning that the recursion stops after at most k steps. The addition $I + J$ of two intervals is defined as $\{i + j \mid i \in I, j \in J\}$.

```
1  algorithm PACK(φ) // computes a maximal packing π ⊆ φ
2      π ← ∅; foreach c ∈ φ do if vars(c) ∩ vars(π) = ∅ then π ← π ∪ {c}
3      return π
4
5  algorithm INTERVAL(φ) // computes a possibly large interval I with σ(φ) ∈ I
6      if PACK(φ) = φ then
7          return [∏_{c∈φ}(1 − 2^{−|c|}), ∏_{c∈φ}(1 − 2^{−|c|})] // = [σ(φ), σ(φ)]
8      else
9          return [0, (1 − 2^{−k})^{|PACK(φ)|}]
10
11 algorithm BOUNDED-INTERVAL(φ, ε) // computes an interval I with σ(φ) ∈ I and |I| < ε
12     if |INTERVAL(φ)| < ε then
13         return INTERVAL(φ)
14     else // |vars(PACK(φ))| ≤ k · |PACK(φ)| ≤ k log_{1−2^{−k}} ε
15         return ∑_{β:vars(PACK(φ))→{0,1}} BOUNDED-INTERVAL(φ|_β, ε/2^{|vars(PACK(φ))|})
16
17 algorithm INTERVAL-BOUNDING(φ) // φ ∈ kCNFS must hold
18     I ← BOUNDED-INTERVAL(φ, spectral-gap_{kCNFS}(δ))
19     if max I < δ then return "σ(φ) < δ" else return "σ(φ) ≥ δ"
```

then *expand* ϕ using this packing (also yet another key insight of Akmal and Williams [2]): For a formula ϕ and an assignment $\beta\colon V \to \{0,1\}$, where $V \neq \text{vars}(\phi)$ is permissible, we write $\phi|_\beta$ for the formula where we remove all clauses from ϕ that contain literals set to true by β, remove all literals from the remaining clauses that are set to false by β, and add one singleton clause for each literal set to true by β. For instance, for $\phi = \big\{\{a,b\},\{\neg b,c,\neg f\},\{d,e,g\}\big\}$ and $\beta\colon \{b,d\} \to \{0,1\}$ with $\beta(b) = 1$ and $\beta(d) = 0$ we have $\phi|_\beta = \big\{\{\cancel{a},\cancel{b}\},\{\cancel{\neg b},c,\neg f\},\{\cancel{d},e,g\}\big\} \cup \big\{\{b\},\{\neg d\}\big\} = \big\{\{c,\neg f\},\{e,g\},\{b\},\{\neg d\}\big\}$. For a maximal packing $\pi \subseteq \phi$ we then have $\sigma(\phi) = \sum_{\beta:\text{vars}(\pi)\to\{0,1\}} \sigma(\phi|_\beta)$ and, importantly, all $\phi|_\beta$ have a *smaller maximum clause size*. This will imply that Algorithm 3 correctly solves kSAT-PROB$_{\geq\delta}$.

Complexity-Theoretic Results. Our proof of Theorem 1.6, which states that kSAT-PROB$_{>\delta}$ is always NP-complete, NL-complete, or lies in AC^0, will be constructive as we can explicitly state for which values of k and δ which case applies. For this statement, let us call $\delta \in [0,1]$ a *k-target for l*CNFS if there is a formula $\omega \in k$CNFS with

$$\sigma(\omega) = \delta \text{ and } \omega_{>k-l} \not\models \omega_{\leq k-l},$$

meaning that there is an assignment $\beta\colon \text{vars}(\omega) \to \{0,1\}$ that is a model (= satisfying assignment) of $\omega_{>k-l} := \{c \in \omega \mid |c| > k-l\}$, but not of $\omega_{\leq k-l} := \{c \in \omega \mid |c| \leq k-l\}$ ("the large clauses do not imply the small clauses"). For instance, $\delta = 7/32$ is a 3-target for 2CNFS as demonstrated by $\omega = \big\{\{a\},\{b\},\{c_1,c_2,c_3\}\big\} \in 3$CNFS since

$$\sigma(\omega) = \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{7}{8} = \frac{7}{32} \text{ and } \big\{\{c_1,c_2,c_3\}\big\} \not\models \big\{\{a\},\{b\}\big\}.$$

In contrast, no $\delta > 1/2$ can be a 3-target for 2CNFS as $\omega_{>1} \not\models \omega_{\leq 1}$ implies that $\omega_{\leq 1}$ is not empty and, thus, $\omega \supseteq \omega_{\leq 1}$ contains at least one singleton clause, which implies $\sigma(\omega) \leq 1/2$.

▶ **Theorem 1.11** (Spectral Trichotomy Theorem, Constructive Version)**.** *For each k and δ:*

1. *If δ is a k-target for* 3CNFS*, then* kSAT-PROB$_{>\delta}$ *is* NP*-complete.*
2. *If δ is a k-target for* 2CNFS*, but not for* 3CNFS*, then* kSAT-PROB$_{>\delta}$ *is* NL*-complete.*
3. *In all other cases,* kSAT-PROB$_{>\delta}$ *lies in (*DLOGTIME*-uniform)* AC^0*.*

The hardness results implicit in the claim of the theorem turn out to be easy to prove, mainly due to the fact that the definition of "δ is a k-target for 2CNFS (or 3CNFS)" is tailored exactly towards making these results easy-to-prove: Take the above example $\delta = 7/32$ via the above $\omega = \big\{\{a\},\{b\},\{c_1,c_2,c_3\}\big\}$. To reduce 2SAT to 3SAT-PROB$_{>\delta}$, on input of $\psi \in$ 2CNFS (with fresh variables, that is, $\mathrm{vars}(\psi) \cap \mathrm{vars}(\omega) = \emptyset$) output

$$\rho = \big\{\{a\} \cup d \mid d \in \psi\big\} \cup \big\{\{b\} \cup d \mid d \in \psi\big\} \cup \big\{\{c_1,c_2,c_3\}\big\},$$

that is, "add the clauses of ψ to each 'small' clause, that is, to each clause in $\omega_{\leq k-l}$." The important observation is that all satisfying assignments of ω are also satisfying assignments of ρ (so $\sigma(\rho) \geq \sigma(\omega)$), but there will be *additional* satisfying assignments of ρ when ψ is satisfiable via some $\alpha\colon \mathrm{vars}(\psi) \to \{0,1\}$: This additional satisfying assignment is obtained by merging α with any assignment β that witnesses $\big\{\{c_1,c_2,c_3\}\big\} \not\models \big\{\{a\},\{b\}\big\}$. The same works for the NP-hardness, now for "δ is a k-target for 3CNFS."

The tricky part are the upper bounds for the last two items. For the last item (the other one is very similar), we wish to show kSAT-PROB$_{>\delta} \in \mathrm{AC}^0$ when δ is *not* a k-target for 2CNFS. The initial idea is easy: On input ϕ, compute the kernel κ using Algorithm 1. If $\sigma(\kappa) > \delta$ or $\sigma(\kappa) < \delta$, the same inequality will hold for $\sigma(\phi)$; but in case $\sigma(\kappa) = \delta$, we need to answer the question whether there is a satisfying assignment of ϕ that does not also satisfy the kernel. (An NP-machine could easily answer this and this already proves the upper bound of the first statement – but we seek an AC^0 upper bound.)

At this point, we use another tool from classical FPT theory: *backdoor sets.* A (strong) backdoor set for a formula ω is a set V of variables such that if $\omega \in$ SAT, then for all assignments $\beta\colon V \to \{0,1\}$ the formula $\omega|_\beta$ is (highly) tractable, meaning for instance that it is a Horn formula or lies in 2CNFS or lies even in 1CNFS.

The central intuition for the AC^0 upper bound is that *the variables in the kernel κ should always form a backdoor set into* 1CNFS *for ϕ* when δ is not a k-target for 2CNFS: Suppose there is a $\beta\colon \mathrm{vars}(\kappa) \to \{0,1\}$ for which $\phi|_\beta$ contains a clause d with $|d| \geq 2$, meaning that $\mathrm{vars}(\kappa)$ is not a backdoor into 1CNFS. Then *κ must contain a clause c^* of size at most $k-2$:* Each clause $d \in \phi|_\beta$ results from some clause $e \in \phi \in k$CNFS from which we remove all variables in the kernel, so $|c^*| = |e^*| - |d| \leq k-2$. This shows that $\kappa_{\leq k-2}$ is not empty – meaning that we are in the situation from the above *lower* bound for $\omega = \kappa$: There is at least one "target clause c^*" to which we could add the clauses of 2CNF formulas during a reduction – exactly what we ruled out by assuming that δ is *not* a k-target for 2CNFS. Figure 4 on page 21 visualizes this intuition (for 2CNFS rather than 1CNFS, though).

There is a catch, however: To serve as target for a reduction, it does not suffice that a clause of size $k-2$ or less exists in ω. We also need that $\omega_{>k-2} \not\models \omega_{\leq k-2}$, that is, that the small clause that we wish to use as target for the reduction is not already implied by the large clauses (and, thus, adding literals to the small clause does not actually give new models and does not raise the satisfaction probability). Fixing this problem is the last step in the algorithm: Instead of just setting $\omega = \kappa$, we let ω be *the small clauses of κ plus the links of the large clauses of κ.* Two simple lemmas will show that *this* ω has the desired properties: The variables in κ form a backdoor into 1CNFS for ω; and $\sigma(\phi) > \delta$ (which is what we wish to decide) iff $\delta(\omega) > \delta(\kappa) = \delta$ (which we can decide in AC^0 by checking whether $\omega|_\beta \in$ SAT for some $\beta\colon \mathrm{vars}(\kappa) \to \{0,1\}$ with $\beta \not\models \kappa$; and $\mathrm{vars}(\kappa)$ is a strong backdoor into 1CNFS).

1.2 Related Work

The history of determining the complexity of the many different variants of the satisfiability problem for propositional formulas dates back all the way to Cook's original NP-completeness proof [7]. Since then, it has become textbook knowledge that kSAT is in AC^0 for $k = 1$, is NL-complete for $k = 2$, and is NP-complete for $k \geq 3$.

Determining whether the number of satisfying assignments of a formula is not just positive, but whether "a lot" of assignments are satisfying, is a quite different problem. Determining whether a majority of assignments are satisfying is a canonical PP-complete problem [10, 13]; and it does not matter whether one considers "strictly more than 1/2" ($\text{SAT-PROB}_{>1/2}$) or "more than or equal to 1/2" or ($\text{SAT-PROB}_{\geq 1/2}$). Indeed, any fixed value different from 1/2 can also be used and it does not matter whether "$>$" or "$\geq$" is used [13, Theorem 4.1]. Because of the indifference of the complexity to the exact problem definition, it is often a bit vague how the problem "MAJORITY-SAT" is defined, exactly, in a paper (indeed, the common meaning of "majority" in voting suggests that "strictly more than one half" is perhaps the natural interpretation).

Given that the tipping point between "easy" and "hard" satisfaction problems is exactly from $k = 2$ to $k = 3$, it seemed natural to assume that $k\text{SAT-PROB}_{\geq 1/2}$ and $k\text{SAT-PROB}_{>1/2}$ are also both PP-complete for $k \geq 3$. Indeed, given that computing $\#(\phi)$ for $\phi \in 2\text{CNFS}$ is known to be #P-complete [15], even $2\text{SAT-PROB}_{\geq 1/2}$ being PP-complete seemed possible and even natural. It was thus surprising that Akmal and Williams [2] were recently able to show that $k\text{SAT-PROB}_{\geq \delta} \in \text{LINTIME}$ holds for *all* k and $\delta \in \mathbb{Q}$. As pointed out by Akmal and Williams, not only has the opposite generally been believed to hold, this has also been claimed repeatedly (page 1 of [1] lists no less than 15 different papers from the last 20 years that conjecture or even claim hardness of $3\text{SAT-PROB}_{\geq 1/2}$). Just as surprising was the result of Akmal and Williams that while $4\text{SAT-PROB}_{\geq 1/2}$ lies in P, the seemingly almost identical problem $4\text{SAT-PROB}_{>1/2}$ is NP-complete. This has lead Akmal and Williams to insist on a precise notation in [2]: They differentiate clearly between MAJORITY-SAT and GT-MAJORITY-SAT and consider these to be special cases of the threshold problems THR_δ-SAT and GT-THR_δ-SAT – and all of these problems can arise in a "-kSAT" version. The notations $k\text{SAT-PROB}_{>\delta}$ and $k\text{SAT-PROB}_{\geq \delta}$ from the present paper are a proposal to further simplify, unify, and clarify the notation, no new problems are introduced.

We will use tools from FPT theory, namely kernels in Algorithm 1 and backdoor sets in Section 4. As computing (especially hitting set) kernels is very well-understood from a complexity-theoretic point of view (see [16] for the algorithmic state of the art and [5] for the upper bounds on the parallel parameterized complexity), we can base proofs on this for $k\text{SAT-PROB}_{\geq \delta} \in AC^0$ for all k and δ. Of course, different parameterized versions of SAT are studied a lot in FPT theory, see [9] for a starting point, but considering the satisfaction probability as a parameter (as we do in the present paper) is presumably new.

The algorithm of Akmal and Williams in [2] was the main inspiration for the results of the present paper and it shares a number of characteristics with Algorithm 1 (less with Algorithm 3 and none with Algorithm 2, though): Both algorithms search for and then collapse sunflowers. However, without the Spectral Well-Ordering Theorem, one faces the problem that collapsing large sunflowers *repeatedly* could conceivably lower $\sigma(\phi)$ past δ. To show that this does not happen (without using the Spectral Well-Ordering Theorem) means that one has to redo all the arguments used in the proof of the Spectral Well-Ordering Theorem, but now with explicit parameters and constants and one has to intertwine the algorithmic and the underlying order-theoretic arguments in rather complex ways (just the analysis of the algorithm in [1] takes eleven pages in the main text plus two pages in the appendix).

The majority-of-majority problem for arbitrary CNF formulas, called MAJ-MAJ-SAT in [2], is known to be complete for $\mathrm{PP}^{\mathrm{PP}}$ and of importance in "robust" satisfaction probability estimations [6, 12]. The arguments from both the present paper and [2] on kSAT-PROB$_{\geq 1/2}$ do not generalize in any obvious way to MAJ-MAJ-kSAT: The difficulty lies in the "mixed" clauses that contain both X- and non-X-variables. In a clever argument, Akmal and Williams were able to show that for $k = 2$ one can "separate" the necessary satisfaction probability estimations for the X- and non-X-variables in polynomial time; a feat facilitated by the fact that a mixed size-2 clause must contain exactly one X-literal and one non-X-literal. They conjectured that MAJ-MAJ-kSAT $\in$ P holds for all k (which is indeed the case by Corollary 3.10), but point out that it is unclear how (or whether) their algorithm can be extended to larger k. The approach taken in the present paper (via locality arguments) seems quite different and not directly comparable.

2 Order-Theoretic Results

There is a sharp contrast between the structural properties of the "full" spectrum of satisfaction probabilities of arbitrary propositional formulas and the spectrum of values kCNF formulas can have. The full spectrum CNFS-σ-SPECTRUM $= \{\sigma(\phi) \mid \phi \in \text{CNFS}\}$ is, well, "full" as it is the set of all dyadic rationals between 0 and 1 (recall $\mathbb{D} = \{m/2^e \mid m, e \in \mathbb{Z}\}$):

▶ **Lemma 2.1.** CNFS-σ-SPECTRUM $= \mathbb{D} \cap [0, 1]$.

Proof. For any propositional formula ϕ we have, by definition, $\sigma(\phi) = m/2^e$ for $m = \#(\phi)$ and $e = |\mathrm{vars}(\phi)|$. For the other direction, let $e \in \mathbb{N}$ and $m \in \{0, \dots, 2^e\}$. Consider the truth table over the variables $X = \{x_1, \dots, x_e\}$ in which the first m lines are set to 1 ("models") and the rest are set to 0 ("not models"). Then each CNF formula ϕ with $\mathrm{vars}(\phi) = X$ having this truth table has exactly m models and, hence, $\sigma(\phi) = m/2^e$. ◀

By the lemma, CNFS-σ-SPECTRUM is a dense subset of the real interval $[0, 1]$, it is not closed topologically, and it is order-isomorphic to $\mathbb{Q} \cup \{-\infty, \infty\}$ with respect to both $<$ and $>$. We will soon see that the properties of each kCNFS-σ-SPECTRUM could hardly be more different: They are nowhere-dense, they are closed, and they are well-ordered. Of these properties, the well-orderedness is the most important one both for algorithms in later sections and because the other properties follow from the well-orderedness rather easily.

To get a better intuition about the spectra, let us have a closer look at the first two, 1CNFS-σ-SPECTRUM and 2CNFS-σ-SPECTRUM. The first is simple:

$$1\text{CNFS-}\sigma\text{-SPECTRUM} = \{1, \tfrac{1}{2}, \tfrac{1}{4}, \tfrac{1}{8}, \tfrac{1}{16}, \dots\} \cup \{0\} \tag{1}$$

as a 1CNF formula (a conjunction of literals) has a satisfaction probability of the form 2^{-e} or is 0. Readers familiar with order theory will notice immediately that 1CNFS-σ-SPECTRUM is order-isomorphic to the ordinal $\omega + 1$ with respect to $>$. The spectrum 2CNFS-σ-SPECTRUM is already much more complex:

$$2\text{CNFS-}\sigma\text{-SPECTRUM} = \{1, \tfrac{3}{4}, \tfrac{5}{8}, \tfrac{9}{16}, \tfrac{17}{32}, \tfrac{33}{64}, \dots\} \cup \{\tfrac{1}{2}, \tfrac{15}{32}\} \cup \{\dots\} \tag{2}$$

where $\{\dots\}$ contains only numbers less than $15/32$. To see that this is, indeed, the case, observe that the formulas $\{\{a, x_1\}\}$, $\{\{a, x_1\}, \{a, x_2\}\}$, $\{\{a, x_1\}, \{a, x_2\}, \{a, x_3\}\}$, … show that every number of the form $1/2 + 2^{-e}$ is in the spectrum. Furthermore, there are no other numbers larger than $1/2$ in the spectrum as the first formula with two variable-disjoint clauses has a satisfaction probability of $\sigma(\{\{a, b\}, \{c, d\}\}) = 9/16$ which we happen

to have already had; and adding any additional clause makes the probability drop to at most $\sigma\big(\{\{a,b\},\{c,d\},\{c,e\}\}\big) = 3/4 \cdot 5/8 = 15/32$. Below this, the exact structure of 2CNFS-σ-SPECTRUM becomes ever more complex as we get nearer to 0 and it is unclear what the order-type of 2CNFS-σ-SPECTRUM with respect to $>$ actually is (an educated guess is $\omega^\omega + 1$).

Our aim in the rest of this section is to prove the Spectral Well-Ordering Theorem, Theorem 1.1, by which all kCNFS-σ-SPECTRUM are well-ordered by $>$. The surprisingly short proof, presented in Section 2.2, will combine results from the following Section 2.1 on some simple properties of well-orderings with some simple properties of $\sigma(\phi)$ for kCNF formulas ϕ. The theorem implies the *existence* of spectral gaps below each δ in the spectra, but the proof does not provide us with any quantitative information about the *sizes* of these gaps. It is possible, but challenging to obtain bounds on the sizes of spectral gaps; please see the technical report version [14] of this paper for the (very) technical details.

2.1 Well-Orderings and Their Properties

Well-orderings are a basic tool of set theory, but for our purposes only a very specific type of orderings will be of interest (namely only sets of non-negative reals with the strictly-greater-than relation as the only ordering relation). For this reason, we reserve the term "well-ordering" only for the following kind of orderings:

▶ **Definition 2.2.** *A set $X \subseteq \mathbb{R}^{\geq 0}$ is* well-ordered (by $>$) *if there is* no *sequence $(x_i)_{i\in\mathbb{N}}$ with $x_i \in X$ for all i and $x_0 < x_1 < \cdots$. Let* WO *denote the set of all (such) well-ordered sets.*

There is extensive literature on the properties of well-orderings in the context of classical set theory, see for instance [11] as a starting point. We will need only those stated in the following lemma, where the first items are standard, while the last are specific to the present paper. For $X, Y \subseteq \mathbb{R}^{\geq 0}$ let $X + Y$ denote $\{x + y \mid x \in X, y \in Y\}$ and $n \cdot X = X + \cdots + X$ (of course, n times). Note that "sequence" always means "infinite sequence" in the following and that "strictly increasing" means $x_i < x_j$ for $i < j$ while just "decreasing" (without the "strictly") means $x_i \geq x_j$ for $i < j$.

▶ **Lemma 2.3.**

1. *Let $X \in$ WO. Then X contains a largest element.*
2. *Let $Y \subseteq X \in$ WO. Then $Y \in$ WO.*
3. *Let $X, Y \in$ WO. Then $X \cup Y \in$ WO. Thus,* WO *is closed under finite unions.*
4. *Let $(X_p)_{p\in\mathbb{N}}$ with $X_p \in$ WO for all p and $\lim_{p\to\infty} \max X_p = 0$. Then $\bigcup_{p\in\mathbb{N}} X_p \in$ WO.*
5. *Let $X \in$ WO and let $(x_i)_{i\in\mathbb{N}}$ be an arbitrary sequence of $x_i \in X$. Then there is an infinite $I \subseteq \mathbb{N}$ such that $(x_i)_{i\in I}$ is decreasing (that is, $x_i \geq x_j$ for $i < j$ and $i, j \in I$).*
6. *Let $X, Y \in$ WO. Then $X + Y \in$ WO. Thus,* WO *is closed under finite sums.*
7. *Let $X \in$ WO and $n \in \mathbb{N}$. Then $n \cdot X \in$ WO.*

Proof.

1. There would otherwise be an (infinite) strictly increasing sequence in X.
2. Any strictly increasing sequence in Y would be a strictly increasing sequence in X.
3. Any strictly increasing sequence in $X \cup Y$ would contain a strictly subsequence in X or Y.
4. Assume there is a sequence $0 < x_0 < x_1 < x_2 < \cdots$ where $x_i \in \bigcup_{p\in\mathbb{N}} X_p$ holds for all i. As the maxima of the X_p tend towards 0, there is some q such that $x_0 > \max X_p$ holds for all $p \geq q$. In particular, the whole sequence $(x_i)_{i\in\mathbb{N}}$ contains only elements of $\bigcup_{p<q} X_p$. By the previous item, this is well-ordered, contradicting that it contains an infinite strictly increasing sequence.

5. The set $\{x_i \mid i > 0\}$ is a subset of X and must hence contain a maximal element x_{i_0} by the first item. Then $\{x_i \mid i > i_0\} \subseteq X$ must contain a maximal element x_{i_1} for some $i_1 > i_0$. Next, consider $\{x_i \mid i > i_1\} \subseteq X$ and let x_{i_2} for some $i_2 > i_1$ be a maximal element. In this way, for $I = \{i_0, i_1, \dots\}$ we get an infinite subsequence $(x_i)_{i\in I}$ that is clearly (not necessarily strictly) decreasing as each chosen element was the maximum of all following elements.
6. Suppose there is a sequence $z_0 < z_1 < z_2 < \dots$ of numbers $z_i \in X + Y$. Then for each i there must exist $x_i \in X$ and $y_i \in Y$ with $z_i = x_i + y_i$. By the previous item there is a decreasing subsequence $(x_i)_{i\in I}$ for some infinite I. Then $(y_i)_{i\in I}$ is an infinite strictly increasing sequence in Y as for any $i, j \in I$ with $i < j$ we have $y_i = z_i - x_i \leq z_i - x_j < z_j - x_j = y_j$. This contradicts $Y \in \mathrm{WO}$.
7. This follows immediately from the previous item. ◀

2.2 Proof of the Spectral Well-Ordering Theorem

For the proof of the Spectral Well-Ordering Theorem, besides the Packing Probability Lemma (Lemma 1.8 from the introduction) we will need the notion of *expansions,* which allow us to express the satisfaction probability of a formula as a sum of satisfaction probabilities of *restricted* formulas. The definition is based on the well-known *unit rule:* For a formula $\phi \in \mathrm{CNFS}$, let $\mathrm{unitrule}(\phi)$ be obtained by removing, for each clause $\{l\} \in \phi$ containing a single literal l, all *other* clauses from ϕ containing l (so, perhaps a bit non-standard, we leave the singleton clause $\{l\}$, which "triggered" the unit rule, in the formula) and removing all occurrences of the negated literal from all remaining clauses.

▶ **Definition 2.4.** *For a set V of variables, $\beta\colon V \to \{0,1\}$ and $\phi \in \mathrm{CNFS}$, the* restriction $\phi|_\beta$ of ϕ to β *is* $\mathrm{unitrule}\big(\phi \cup \big\{\{v\} \mid v \in V, \beta(v) = 1\big\} \cup \big\{\{\neg v\} \mid v \in V, \beta(v) = 0\big\}\big)$.

As an example, for $\phi = \{\{a,b\}, \{\neg b, c, \neg f\}, \{d,e,f,g\}\}$ and $\beta\colon \{b,c,h\} \to \{0,1\}$ with $\beta(b) = 1$ and $\beta(c) = \beta(h) = 0$ we have $\phi|_\beta = \big\{\{\cancel{a,b}\}, \{\cancel{\neg b}, \cancel{c}, \neg f\}, \{d,e,f,g\}\big\} \cup \big\{\{b\}, \{\neg c\}, \{\neg h\}\big\} = \big\{\{\neg f\}, \{d,e,f,g\}, \{b\}, \{\neg c\}, \{\neg h\}\big\}$. Note that we apply the unit rule only once to the initial singleton clauses, we do not do unit *propagation,* which is P-hard and thus computationally too expensive in later contexts. The importance of restrictions for our purposes lies in two simple lemmas:

▶ **Lemma 2.5.** *Let $\phi \in \mathrm{CNFS}$ and V be some variables. Then the set of models of ϕ over the variables $V \cup \mathrm{vars}(\phi)$ is exactly the disjoint union of the models of $\phi|_\beta$ for $\beta\colon V \to \{0,1\}$.*

Proof. Each satisfying assignment $\alpha\colon V \cup \mathrm{vars}(\phi) \to \{0,1\}$ of ϕ satisfies $\phi|_\beta$ for the assignment β that agrees with α on the variables in V, but α satisfies no $\phi|_{\beta'}$ for $\beta' \neq \beta$. ◀

▶ **Corollary 2.6.** *Let $\phi \in \mathrm{CNFS}$ and V be some variables. Then $\sigma(\phi) = \sum_{\beta: V\to\{0,1\}} \sigma(\phi|_\beta)$.*

▶ **Lemma 2.7.** *Let $\phi \in k\mathrm{CNFS}$ for $k \geq 2$ and let $\pi \subseteq \phi$ be a maximal packing. Then $\phi|_\beta \in (k-1)\mathrm{CNFS}$ for all $\beta\colon \mathrm{vars}(\pi) \to \{0,1\}$.*

Proof. In $\phi|_\beta$, we either remove a clause or remove at least one literal from it as $\mathrm{vars}(\pi)$ intersects $\mathrm{vars}(c)$ for all $c \in \phi$ (as π would not be maximal, otherwise). ◀

Proof of the Spectral Well-Ordering Theorem, Theorem 1.1. By induction on k. The base case is $k = 1$ where $1\mathrm{CNFS}\text{-}\sigma\text{-}\mathrm{SPECTRUM} = \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\} \cup \{0\} = \{2^{-i} \mid i \in \mathbb{N} \cup \{\infty\}\}$, which is clearly well-ordered (with order type $\omega + 1$). For the inductive step from $k-1$ to k, we show that

$$k\mathrm{CNFS}\text{-}\sigma\text{-}\mathrm{SPECTRUM} \subseteq \bigcup_{p\in\mathbb{N}} \big(2^{kp} \cdot (k-1)\mathrm{CNFS}\text{-}\sigma\text{-}\mathrm{SPECTRUM}\big) \cap \big[0, (1-2^{-k})^p\big].$$

This will prove kCNFS-σ-SPECTRUM $\in$ WO as $(k-1)$CNFS-σ-SPECTRUM $\in$ WO by the induction hypothesis, and thus $2^{kp} \cdot (k-1)$CNFS-σ-SPECTRUM $\in$ WO as a finite sum of well-orderings. By intersecting this with the ever-smaller intervals $\left[0, (1-2^{-k})^p\right]$, we still get elements of WO by item 2 of Lemma 2.3 and can then apply item 4 to get that the infinite union lies in WO.

It remains to prove the inclusion. Let $x \in k$CNFS-σ-SPECTRUM be witnessed by $\phi \in k$CNFS, that is, $x = \sigma(\phi)$. Let π be a maximal packing $\pi \subseteq \phi$ and let $p = |\pi|$. By the Packing Probability Lemma, $\sigma(\phi) \leq (1-2^{-k})^p$. By Corollary 2.6, we have $\sigma(\phi) = \sum_{\beta:\mathrm{vars}(\pi)\to\{0,1\}} \sigma(\phi|_\beta)$ and by Lemma 2.7, each $\phi|_\beta$ is a $(k-1)$CNF formula. Thus, $\sigma(\phi)$ is the sum of at most 2^{kp} values from $(k-1)$CNFS-σ-SPECTRUM, proving that we have $x \in \left(2^{kp} \cdot (k-1)\text{CNFS-}\sigma\text{-SPECTRUM}\right) \cap \left[0, (1-2^{-k})^p\right]$. ◀

3 Algorithmic Results

Having established that the spectra kCNFS-σ-SPECTRUM are well-ordered by $>$ in the previous section, we now turn our attention to the algorithmic aspects of deciding whether $\sigma(\phi) \geq \delta$ holds (the question of whether $\sigma(\phi) > \delta$ holds will be discussed in the next section). The focus will be less on the exact complexity of these algorithm (they can all be implemented by AC^0 circuits, sometimes trivially, sometimes with a bit of effort), but more on how the algorithms work. We will touch on "practical" considerations only very briefly in the following, as they do not lie at the heart of this paper; some ideas towards optimizations, implementations, and practical heuristics can be found in the technical report version [14].

This section includes two "excursions" beyond the three main algorithms. First, an interesting corollary of the algorithmic analysis of the first algorithm will be another structural result concerning the spectra kCNFS-σ-SPECTRUM, though not an order-theoretic one, but a topological one: The spectra are *closed,* meaning that for every converging sequence $\sigma(\phi_1), \sigma(\phi_2), \ldots$ with $\phi_i \in k$CNF there is some $\phi \in k$CNF with $\lim_{i\to\infty} \sigma(\phi_i) = \sigma(\phi)$. Another way of saying this is that for every number $\delta \notin k$CNFS-σ-SPECTRUM there is an $\epsilon > 0$ with $(\delta-\epsilon, \delta+\epsilon) \cap k$CNFS-$\sigma$-SPECTRUM $= \emptyset$. Second, we will show that the Threshold Locality Lemma, which underlies the second algorithm, can also be used to reduce certain satisfaction probability threshold problems to model checking problems. This will allow us to "chip away" one "majority-of-" in any majority-of-majority-of-...-majority problem. In particular, we will get a proof of the Akmal–Williams conjecture MAJ-MAJ-kSAT $\in$ P.

3.1 The Sunflower-Collapsing Algorithm

Our first new algorithm for deciding whether $\sigma(\phi) \geq \delta$ holds for a given $\phi \in k$CNFS and fixed δ is based on computing "sunflower kernels." The name "sunflower" comes from classical combinatorics, the name "kernel" comes from fixed-parameter tractability (FPT) theory. Note that we just use kernels as a "tool" without making formal statements in the sense of FPT theory, but see the technical report version [14] for some first formal statements.

Recall from Definition 1.9 that a *sunflower with core* c is a formula $\psi \in$ CNFS such that $c \subseteq e$ holds for all clauses $e \in \psi$ and such that any two different $e, e' \in \psi$ we have $\mathrm{vars}(e) \cap \mathrm{vars}(e') = \mathrm{vars}(c)$, that is, the clauses "agree on the literals in c, but are variable-disjoint otherwise". An example of a size-3 sunflower with core $\{a, \neg b\}$ is $\big\{\{a, \neg b, c\}, \{a, \neg b, \neg d, \neg e\}, \{a, \neg b, f\}\big\}$.

Sunflowers play a key role in FPT theory, especially in the computation of kernels for the hitting set problem [9] and related problems: Suppose that for a given formula ϕ we want to find a size-k set V of variables such that for each clause $e \in \phi$ we have $\mathrm{vars}(e) \cap V \neq \emptyset$

(each clause is "hit" by V). Then if there is a sunflower $\psi \subseteq \phi$ of size $h+1$ in ϕ with some core c, any size-h hitting set V must hit c since, otherwise, we would need $h+1$ variables to hit the "petals" of the sunflower (we would need to have $V \cap (\mathrm{vars}(e) \setminus \mathrm{vars}(c)) \neq \emptyset$ for $h+1$ pairwise disjoint sets $\mathrm{vars}(e) \setminus \mathrm{vars}(c)$). This means that ϕ has a size-h hitting set iff $(\phi \setminus \psi) \cup \{c\}$ has (indeed, iff $(\phi \setminus \mathrm{link}_\phi(c)) \cup \{c\}$ has, where $\mathrm{link}_\phi(c) = \{e \in \phi \mid c \subseteq e\}$). Most importantly, applying this reduction rule "as often as possible" leads to a formula whose size is bounded by a constant depending only on h, not on the original formula (this is known as a "kernelization" in fixed-parameter theory). The reason for this size bound is the following Sunflower Lemma (rephrased in terms of positive formulas rather than hypergraphs, as would be standard, where a *positive* formula is a formula without negations):

▶ **Fact 3.1** (Sunflower Lemma, [8]). *Every positive $\phi \in k\mathrm{CNFS}$ with more than $h^k \cdot k!$ clauses contains a sunflower of size $h+1$.*

The "positive" in the statement is due to the fact that in combinatorics sunflowers usually do not care about the "sign" of the variables (whether or not it is negated). In particular, for a formula ϕ let ϕ' be the formula where all negations are simply removed. Then the Sunflower Lemma tells us that if ϕ' is sufficiently large, then it has a large sunflower $\psi \subseteq \phi'$ with core c. This large sunflower does not necessarily become a large sunflower of the original ϕ if we just reinsert the negations: While this makes no difference for the petals outside the core, there may now suddenly be up to $2^{|c|}$ different versions of the core. However, for the version of this core that is present in the maximum number of petals, the number of these petals is at least a fraction of $1/2^{|c|} \geq 1/2^k$ of the size of the "unsigned" sunflower. This yields the following corollary:

▶ **Corollary 3.2.** *Every $\phi \in k\mathrm{CNFS}$ with more than $(2h)^k \cdot k!$ clauses contains a sunflower of size $h+1$.*

The just-described kernel algorithm for the hitting set problem ("as long as there is a sufficiently large sunflower, remove it and add the core instead") also turns out to allow us to decide whether $\sigma(\phi) \geq \delta$ holds. The detailed algorithm is Algorithm 1 from page 6. Its important properties are summarized in the following lemma:

▶ **Lemma 3.3.** *For each fixed k and δ, in Algorithm 1 from page 6*

1. *the while loop will terminate after a number of iterations that is linear in the size of ϕ,*
2. *the size $|\kappa|$ will be bounded by a fixed number that depends only on k and δ, and*
3. *$\sigma(\phi) \geq \delta$ iff $\sigma(\kappa) \geq \delta$ will hold (the "kernel property") and, thus, the output of the algorithm* SUNFLOWER-COLLAPSING(ϕ) *will be correct.*

Proof. For item 1, each iteration of the while loop reduces the size of ϕ, so after a linear number of iterations the loop must stop. (Note that finding large sunflowers is a bit of an art and there is extensive literature on how to do this efficiently, see [5, 9, 16] for starting points. But as the size h is fixed in our case, we could even brute-force the search here.)

For item 2, we use Corollary 3.2, which states that as long as there are more than $(2h)^k k!$ clauses in ϕ, there is still a sunflower and, hence, the while loop will not have ended. Thus, $|\kappa| \leq (2h)^k k!$ and this is clearly a constant as h is a constant depending only on δ.

For item 3, we need to show that for $\phi' = (\phi \setminus \mathrm{link}_\phi(c)) \cup \{c\}$, where c is the core of a size-$(h+1)$ sunflower ψ in ϕ, we have $\sigma(\phi) \geq \delta$ iff $\sigma(\phi') \geq \delta$. As already pointed out in the introduction, the probability $\sigma(\phi) - \sigma(\phi')$ that an assignment β satisfies ϕ, but not ϕ', is at most $\sigma(\{e \setminus c \mid e \in \mathrm{link}_\phi(c)\})$. As the link contains a size-$(h+1)$ sunflower,

$\{e \setminus c \mid e \in \text{link}_\phi(c)\}$ contains a size-$(h+1)$ packing and by the Packing Probability Lemma we get $\sigma(\phi) - \sigma(\phi') \leq (1-2^{-k})^{h+1} < (1-2^{-k})^h = \text{spectral-gap}_{k\text{CNFS}}(\delta)$. By the No Tunneling Lemma, Lemma 1.10, we get the claim. ◀

Excursion 1: The Spectra Are Topologically Closed. While the sunflower collapsing algorithm provides a surprisingly simple method of deciding whether $\sigma(\phi) \geq \delta$ holds, it can also be seen as a purely combinatorial statement:

▶ **Lemma 3.4.** *For every $\delta \in [0,1]$ and k there is a size s such that for every formula $\phi \in k\text{CNFS}$ with $\sigma(\phi) \geq \delta$ there is a formula $\kappa \in k\text{CNFS}$ of size $|\kappa| \leq s$ with $\sigma(\phi) \geq \sigma(\kappa) \geq \delta$.*

Proof. For fixed δ and k, let s be the maximum size of the output κ of KERNELIZE(ϕ) in Algorithm 1. The kernel always has the property $\sigma(\phi) \geq \sigma(\kappa)$. As shown in the proof of the second item of Lemma 3.3, $s \leq \left(2\log_{1-2^{-k}}(\text{spectral-gap}_{k\text{CNFS}}(\delta))\right)^k k!$. Finally, by the third item, $\sigma(\phi) \geq \delta$ implies $\sigma(\kappa) \geq \delta$. ◀

This lemma provides us with an easy way of showing that kCNFS-σ-SPECTRUM is topologically *closed,* which means that its complement is an open set. Note that this does not follow from the fact that the spectra are well-ordered as the set $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots\}$ is well-ordered, but not closed (it misses 0).

▶ **Lemma 3.5.** *Let $\Phi \subseteq k\text{CNFS}$. Then $\inf\{\sigma(\phi) \mid \phi \in \Phi\} \in k\text{CNFS-}\sigma\text{-SPECTRUM}$.*

Proof. Let $\delta = \inf\{\sigma(\phi) \mid \phi \in \Phi\}$. Then there must be a sequence $(\phi_0, \phi_1, \phi_2, \dots)$ with $\phi_i \in \Phi$ with $\lim_{i\to\infty} \sigma(\phi_i) = \delta$. Consider the sequence $(\kappa_0, \kappa_1, \kappa_2, \dots)$ where each κ_i is the formula from Lemma 3.4 for ϕ_i. Then, clearly, $\lim_{i\to\infty} \sigma(\kappa_i) = \delta$. If necessary, rename the variables in each κ_i to that they are $\{v_1, \dots, v_q\}$ for $q = k \cdot s$, where s is the constant form Lemma 3.4, and note that this is always possible. Then $K := \{\kappa_i \mid i \in \mathbb{N}\}$ is a *finite* set as there are only finitely many different CNF formulas over the variables $\{v_1, \dots, v_q\}$. This means that there is some $\kappa^* \in K$ with $\sigma(\kappa^*) = \min\{\sigma(\kappa) \mid \kappa \in K\}$. Then $\sigma(\kappa^*) = \delta$ must hold and $\kappa^* \in k\text{CNFS}$ witnesses $\delta \in k\text{CNFS-}\sigma\text{-SPECTRUM}$. ◀

▶ **Corollary 3.6.** *kCNFS-σ-SPECTRUM is topologically closed for each k.*

3.2 The Threshold Locality Lemma and Algorithm

A second algorithm for showing $k\text{SAT-PROB}_{\geq\delta} \in \text{AC}^0$ is based on Lemma 1.5 from the introduction, which states: *For each k and each $\delta \in [0,1]$ there is a $C \in \mathbb{N}$ so that for all $\phi \in k\text{CNFS}$ we have $\sigma(\phi) \geq \delta$, iff $\sigma(\psi) \geq \delta$ holds for all $\psi \subseteq \phi$ with $|\psi| \leq C$.* The proof, which was already sketched in the introduction, is surprisingly easy:

Proof of the Threshold Locality Lemma (Lemma 1.5). First note that $\psi \subseteq \phi$ clearly implies $\sigma(\psi) \geq \sigma(\phi)$ (as ϕ has "just more clauses" it is "harder to satisfy"). Thus, $\sigma(\phi) \geq \delta$ immediately implies $\sigma(\psi) \geq \delta$ for all $\psi \subseteq \phi$, regardless of their size.

For the other direction, suppose $\sigma(\phi) < \delta$ holds. We must show that there is some $\psi \subseteq \phi$ with $\sigma(\psi) < \delta$ and $|\psi| \leq C$ for a constant C. We claim that we can always find such a ψ when we set

$$C := \Big(2(1 + \underbrace{\log_{1-2^{-k}}(\text{spectral-gap}_{k\text{CNFS}}(\delta))}_{=:g})\Big)^k \cdot k!.$$

To find such a ψ, starting with $\psi_1 = \phi$, as long as possible, pick a clause $e \in \psi_i$ such that $\sigma(\psi_i \setminus \{e\}) < \delta$ still holds and set $\psi_{i+1} = \psi_i \setminus \{e\}$. As we have $\sigma(\psi_1) < \delta$ and as $\sigma(\emptyset) = 1 \geq \delta$, the process must end after $s \leq |\phi|$ steps with some formula $\psi := \psi_s$. Then $\sigma(\psi) < \delta$, $\psi \subseteq \phi$, and for all clauses $e \in \psi$ we have $\sigma(\psi \setminus \{e\}) \geq \delta$.

Suppose we had $|\psi| > C$. By Corollary 3.2 to the Sunflower Lemma, ψ must then contain a sunflower $\rho \subseteq \psi$ of size $|\rho| \geq g + 2$. Let c be the core of ρ and let $a \in \rho$ be arbitrary. Then $\psi \setminus \{a\}$ still contains a sunflower (namely $\rho \setminus \{a\}$) of size at least $g + 1$. Now consider an assignment β with $\beta \models \psi \setminus \{a\}$, but $\beta \not\models \psi$. Then $\beta \not\models c \subseteq a$ as, otherwise, $\beta \models \psi$ would hold. Thus, β must be a model of the packing $\{e \setminus c \mid e \in \rho \setminus \{a\}\}$ and hence, by the Packing Probability Lemma, $\Pr_\beta[\beta \models \psi \setminus \{a\}, \beta \not\models \psi] \leq (1 - 2^{-k})^{|\rho \setminus \{a\}|} \leq (1 - 2^{-k})^{g+1} < (1 - 2^{-k})^g = \text{spectral-gap}_{k\text{CNFS}}(\delta)$. This implies that $\sigma(\psi \setminus \{a\}) - \sigma(\psi)$ is smaller than the spectral gap of δ and by the No Tunneling Lemma we get $\sigma(\psi \setminus \{a\}) < \delta$; contradicting that we could no longer remove clauses from ψ without raising the satisfaction probability to at least δ. ◀

As pointed out in the introduction, Lemma 1.5 clearly implies that Algorithm 2 is correct; we just state this once more for reference:

▶ **Lemma 3.7.** *For each fixed k and δ, on every input $\phi \in k\text{CNFS}$ the output of Algorithm 2 from page 7 is correct.*

Excursion 2: Solving the Majority-of-Majority Problem. Besides the simple just-sketched algorithm for solving $k\text{SAT-PROB}_{\geq\delta}$ efficiently, the Threshold Locality Lemma has another surprising and highly nontrivial consequence: The lemmas lies at the heart of an algorithm for solving the majority-of-majority problem efficiently for $k\text{CNFS}$. Recall that for this problem we are given a formula $\phi \in k\text{CNFS}$ with $\text{vars}(\phi) \subseteq X \cup Y$ for two disjoint, infinite sets ("sorts") X and Y of variables. The question is whether for at least half of all possible assignments $\beta\colon X \to \{0,1\}$ at least half of all possible extensions $\beta'\colon X \cup Y \to \{0,1\}$ (meaning $\beta'(x) = \beta(x)$ for all $x \in X$) make ϕ true, that is, $\beta' \models \phi$.

While it is tempting to try to solve this problem by focusing on the statement "at least half of all possible assignments $\beta\colon X \to \{0,1\}$" initially, it turns out that it is the statement "at least half of all possible extensions $\beta'\colon X \cup Y \to \{0,1\}$" that we must address first. Our objective will be to replace any formula $\phi \in k\text{CNFS}$ with $\text{vars}(\phi) \subseteq X \cup Y$ by a formula $\omega \in l\text{CNFS}$ with $\text{vars}(\omega) \subseteq X$ such that ω "encodes the validity of the second statement for every β." The following definition and lemma make these ideas precise (and generalize them to values of δ other than $1/2$):

▶ **Definition 3.8.** *Let $\phi \in \text{CNFS}$ with $\text{vars}(\phi) \subseteq X \cup Y$ for disjoint sets X and Y. Let $\beta\colon X \to \{0,1\}$. We write ϕ/β for the formula resulting from ϕ when we remove all clauses that contain an X-literal l (so $l = x$ or $l = \neg x$ for some $x \in X$) with $\beta(l) = 1$ and where we remove all remaining X-literals from all remaining clauses.*

Note that $\text{vars}(\phi/\beta) \subseteq Y$ and that we "almost" have $\phi|_\beta = \phi/\beta$, but the difference is that in $\phi|_\beta$ we have an additional singleton clause for each $x \in X$, whereas ϕ/β contains none. As an example, for $\phi = \{\{x_1, \neg x_2, y_1\}, \{\neg x_1, y_2, \neg y_3\}, \{\neg x_2\}, \{y_4\}\}$ and $\beta\colon \{x_1, x_2\} \to \{0,1\}$ with $\beta(x_1) = \beta(x_2) = 1$, we have $\phi/\beta = \{\cancel{\{x_1, \neg x_2, y_1\}}, \cancel{\{\neg x_1}, y_2, \neg y_3\}, \cancel{\{\neg x_2\}}, \{y_4\}\} = \{\{\neg y_3\}, \emptyset, \{y_4\}\}$ while $\phi|_\beta = \{\{\neg y_3\}, \emptyset, \{y_4\}, \{x_1\}, \{x_2\}\}$. The notation makes handling the satisfaction probabilities of extensions easier due to the following simple connection:

$$\sigma(\phi/\beta) = \Pr_{\beta'\colon X \cup Y \to \{0,1\},\, \beta' \text{ extends } \beta}[\beta' \models \phi].$$

Note that $\text{MAJ-MAJ-}k\text{SAT} = \{\phi \in k\text{CNFS} \mid \Pr_{\beta\colon X \to \{0,1\}}[\sigma(\phi/\beta) \geq 1/2] \geq 1/2\}$.

▶ **Lemma 3.9** (Threshold Encoding Lemma). *For each k and $\delta \in [0,1]$ there are a number l and an AC^0 circuit family that maps every formula $\phi \in k\mathrm{CNFS}$ with $\mathrm{vars}(\phi) \subseteq X \cup Y$ to a formula $\omega \in l\mathrm{CNFS}$ with $\mathrm{vars}(\omega) \subseteq X$ such that for all $\beta\colon X \to \{0,1\}$ we have*

$$\sigma(\phi/\beta) \geq \delta \iff \beta \models \omega.$$

Proof. Consider the formula ρ that results from ϕ if we simply remove all X-variables from all clauses (so $\mathrm{vars}(\rho) \subseteq Y$). This formula is, in a sense, the "worst case" of what ϕ/β could look like regarding the satisfaction probability: ρ is the formula where *no* clause is already satisfied by the assignment β, leaving a maximal number of clauses that need to be satisfied. Note that $\phi/\beta \subseteq \rho$ and, thus, if $\sigma(\rho) \geq \delta$ happens to hold, we have $\sigma(\phi/\beta) \geq \delta$ for all β and could set ω to an arbitrary tautology. The interesting question is, thus, what happens when $\sigma(\rho) < \delta$: Which β will remove enough clauses from ρ to raise the probability above δ?

To answer this question (and to turn it into a formula ω), we use the Threshold Locality Lemma. By this lemma, there is a constant C such that we have $\sigma(\phi/\beta) \geq \delta$ iff for every $\psi \subseteq \phi/\beta$ with $|\psi| \leq C$ we have $\sigma(\psi) \geq \delta$. In particular, for a given $\psi \subseteq \rho$ with $\sigma(\psi) < \delta$ we must have $\psi \not\subseteq \phi/\beta$ to have a chance that $\sigma(\phi/\beta) \geq \delta$ holds. Now, $\psi \not\subseteq \phi/\beta$ means that there must be at least one clause $c \in \psi$ that is not contained in ϕ/β. By Definition 3.8 this is the case iff there is a clause $c \in \psi$ such that for all $d \in \phi$ from which c resulted (recall that each clause in ρ is obtained from some clause of ϕ by removing all X-variables) at least one X-literal in d is set to 1 by β (because, then, c is not added to ϕ/β).

To summarize, we have two conditions to check:

1. For every $\psi \subseteq \rho$ with $|\psi| \leq C$ and $\sigma(\psi) < \delta$ we must have $\psi \not\subseteq \phi/\beta$, which is the case iff
2. there is a clause $c \in \psi$ such that for all $d \in \phi$ from which c resulted, at least one X-literal in d is set to 1 by β.

It turns out, we can express these conditions using a single formula $\omega \in l\mathrm{CNFS}$. Let us start with the second condition for a fixed $\psi \subseteq \rho$ and let us try to find a single formula $\omega_\psi \in l\mathrm{CNFS}$ for some l expressing it. The condition is clearly a *disjunction* ("there is a clause") over all $c \in \psi$ of the following $k\mathrm{CNF}$ formulas $\omega_{\psi,c}$ ("such that for all" is expressed by the union):

$$\omega_{\psi,c} = \bigcup_{d \in \phi, c \subseteq d, \mathrm{vars}(d) \setminus X = \mathrm{vars}(c)} \{d \setminus c\}.$$

We can turn the disjunction of the at most C many $\omega_{\psi,c} \in k\mathrm{CNFS}$ into a single conjunction $\omega_\psi \in l\mathrm{CNF}$ using the distributive law if we set $l := k \cdot C$. To sum up: For the resulting formula ω_ψ we have $\psi \not\subseteq \phi/\beta$ iff $\beta \models \omega_\psi$.

It is now easy to express the first condition:

$$\omega = \bigcup_{\psi \subseteq \rho, |\psi| \leq C, \sigma(\psi) < \delta} \omega_\psi$$

and note that $\omega \in l\mathrm{CNFS}$ holds. Note furthermore that an AC^0 circuit can construct this ω as C is a constant and, hence, we can consider all size-C subsets of ρ in parallel and can hardwire the tests $\sigma(\psi) < \delta$.

For the correctness of the construction, let us briefly reiterate: First, if $\beta \models \omega$, consider any subset $\psi \subseteq \phi/\beta$ of size at most C. Then $\sigma(\psi) < \delta$ is impossible as ω_ψ would now require that some clause $c \in \psi$ is missing from ϕ/β. Thus, $\sigma(\psi) \geq \delta$ always holds and, by the Threshold Locality Lemma, we have $\sigma(\phi/\beta) \geq \delta$. Second, suppose $\sigma(\phi/\beta) \geq \delta$. Then $\sigma(\psi) \geq \delta$ trivially holds for every subset $\psi \subseteq \phi/\beta$. Thus, for every ω_ψ considered in ω, we have $\psi \not\subseteq \phi/\beta$. But then, $\beta \models \omega_\psi$. Thus, $\beta \models \omega$. ◀

▶ **Corollary 3.10.** MAJ-MAJ-kSAT $\in$ AC^0 *for all* k.

Proof. Consider the number l from the Threshold Encoding Lemma for k and $\delta = 1/2$. We can reduce MAJ-MAJ-kSAT to lSAT-PROB$_{\geq 1/2}$ by simply mapping an input formula ϕ to the formula ω from the lemma. Since lSAT-PROB$_{\geq 1/2} \in$ AC^0 (this is implicit in the main algorithmic results of the present section, but see Theorem 4.1 for a detailed discussion), we get the claim. ◀

Writing MAJ^i-kSAT for MAJ-MAJ-$\cdots$MAJ-kSAT with of course exactly i repetitions (so MAJ^1-kSAT $=$ kSAT-PROB$_{\geq 1/2}$ and MAJ^2-kSAT $=$ MAJ-MAJ-kSAT) and with the obvious semantics for $i > 2$, we get:

▶ **Corollary 3.11.** MAJ^i-kSAT $\in$ AC^0 *for all* k *and* i.

Proof. Just repeat the reduction from the previous corollary $i - 1$ times. ◀

As a final remark, note that the Threshold Encoding Lemma actually provides us with yet another direct method for showing kSAT-PROB$_{\geq \delta} \in$ AC^0: Just set $X = \emptyset$. This means that in the proof of the above corollary, we can actually apply the reduction i times, not $i - 1$ times, and get a "perfectly uniform" argument.

3.3 The Recursive Interval-Bounding Algorithm

While the sunflower-collapsing algorithm for deciding $\sigma(\phi) \geq \delta$ gives insight into the structure of the problem and lies at the heart of the complexity-theoretic results in the next section, the starting point for a presumably more practical algorithm can be derived from the expansion operation (which we already used in the proof the Spectral Well-Ordering Theorem). The idea behind Algorithm 3 is simple: Starting with a formula ϕ, we wish to compute an interval $I \subseteq [0, 1]$ with $\sigma(\phi) \in I$ and $|I| := \max I - \min I < \epsilon$. If we achieve this and if ϵ is at most the spectral gap of δ, then $\max I < \delta$ will hold iff $\sigma(\phi) < \delta$. Depending on the structure of ϕ, we may be able to easily obtain an interval with the desired properties (using the method INTERVAL(ϕ)). If, however, the initially obtained interval is too large, by the Packing Probability Lemma it will contain a (relatively) small packing, allowing us to expand the formula and to *recurse to* $(k-1)$CNF *formulas.* The following lemma makes these claims precise.

▶ **Lemma 3.12.** *For each fixed* k *and* δ*, in Algorithm 3 from page 8*

1. *for every* ϕ *the algorithm* INTERVAL(ϕ) *returns an interval* I *with* $\sigma(\phi) \in I$ *and with* $|I| = 0$ *when* $\phi \in$ 1CNFS*,*
2. *for every* ϕ *and* $\epsilon > 0$ *the algorithm* BOUNDED-INTERVAL(ϕ, ϵ) *will*
   a. *call itself recursively only for formulas with a strictly smaller clause size,*
   b. *call itself at most* $2^{k \log_{1-2^{-k}} \epsilon}$ *times,*
   c. *return an interval* I *with* $\sigma(\phi) \in I$ *and* $|I| < \epsilon$*, and*
3. *the output of* INTERVAL-BOUNDING(ϕ) *will be correct.*

Proof. For item 1, we clearly have $\sigma(\phi) \in I$ as the interval we output is either exactly $[\sigma(\phi), \sigma(\phi)]$ (namely, when ϕ is a packing and $\sigma(\phi)$ is exactly equal to $\prod_{c \in \phi}(1 - 2^{-|c|})$) or is $[0, (1 - 2^{-k})^{|\text{PACK}(\phi)|}]$ and the Packing Probability Lemma tells us that $\sigma(\phi)$ lies in this interval. When $\phi \in$ 1CNFS, then ϕ is always a packing (unless it is contradictory as it contains both $\{v\}$ and $\{\neg v\}$, but this can be filtered easily) and, thus, $|I| = 0$.

For item 2a, observe that $\textsc{pack}(\phi)$ has the property that $\text{vars}(\textsc{pack}(\phi))$ intersects the variables in each clause of ϕ (otherwise, $\textsc{pack}(\phi)$ would not be maximal). Thus, in all calls $\phi|_\beta$ has a strictly smaller maximal clause size (unless $\phi \in 1\textsc{cnfs}$ did already hold, but then there will be no recursive calls at all).

Item 2b follows as a recursive call is only made when $|I| \geq \epsilon$, which implies that we have $I = [0, (1-2^{-k})^{|\textsc{pack}(\phi)|}]$ and thus $(1-2^{-k})^{|\textsc{pack}(\phi)|} \geq \epsilon$. This clearly shows $|\textsc{pack}(\phi)| \leq \log_{1-2^{-k}} \epsilon$ and thus $|\text{vars}(\textsc{pack}(\phi))| \leq k \log_{1-2^{-k}} \epsilon$ which in turn shows that the number of different β used for the recursive calls is the claimed number.

Item 2c follows by induction on k: By the first item, if $\phi \in 1\textsc{cnfs}$, then $|I| = 0 < \epsilon$. For the inductive step, for any $\phi \in k\textsc{cnfs}$ we have $\sigma(\phi) = \sum_\beta \sigma(\phi|_\beta)$. By the induction hypothesis, our algorithm will compute for each $\phi|_\beta$ an interval I_β containing $\sigma(\phi|_\beta)$. Then the sum of these intervals will contain $\sigma(\phi)$. Furthermore, the size of the sum of these intervals will be the sum of the sizes of the intervals. As the size of each I_β at most ϵ divided by the number of calls made, the total interval size it at most ϵ.

The last item now follows as an interval I with $|I| < \text{spectral-gap}_{k\textsc{cnfs}}(\delta)$ and $\sigma(\phi) \in I$ has the following property: If $\sigma(\phi) < \delta$, then $\sigma(\phi) \leq \delta - \text{spectral-gap}_{k\textsc{cnfs}}(\delta)$ and thus $\sigma(\phi) < \delta - |I|$ or, equivalently, $\sigma(\phi) + |I| < \delta$. In particular, $\max I < \delta$, which is exactly what we test. In contrast, if $\sigma(\phi) \geq \delta$ we immediately get $\max I \geq \delta$. Thus, the output is always correct. ◀

4 Complexity-Theoretic Results

We prove Theorem 1.11, which tells us precisely for each $\delta \in [0,1]$ whether $k\text{SAT-PROB}_{>\delta}$ is NP-complete, NL-complete, or in AC^0, in three steps: First, we show that $k\text{SAT-PROB}_{\geq\delta}$ lies in AC^0 for all k and δ. This means that from a complexity-theoretic point of view it will always be "trivial" to check whether $\sigma(\phi) \geq \delta$ holds – the tricky part is showing that $\sigma(\phi) \neq \delta$ then holds. Second, we introduce a technical definition of being a *k-target for lCNFS*, which is set up in such a way that if a number δ has this property, it will be possible to reduce $l\text{SAT}$ to $k\text{SAT-PROB}_{>\delta}$. This will prove the NL- and NP-hardness results for those δ where Theorem 1.11 claims that $k\text{SAT-PROB}_{>\delta}$ is NL- or NP-compete. Third, we address the upper bounds, meaning that we present NP-, NL-, and AC^0-algorithms that match the lower bounds. Here, we show that the variables in the kernel form backdoor sets for the "error the kernelization makes" into $2\textsc{cnfs}$ or $1\textsc{cnfs}$, when δ is *not* a k-target for $3\textsc{cnfs}$ or $2\textsc{cnfs}$, respectively. Figure 4 is an attempt to visualize the underlying ideas in a simplified way.

4.1 Upper Complexity Bounds for the Greater-or-Equal Problem

Although our ultimate objective in this section is the complexity of the problem of telling whether $\sigma(\phi) > \delta$ holds, we begin with the following:

▶ **Theorem 4.1.** *For all k and δ the problem $k\text{SAT-PROB}_{\geq\delta}$ lies in* DLOGTIME*-uniform* AC^0.

Proof. We claim that Algorithm 1 can be implemented by AC^0 circuits. It is well-established that kernels for hitting sets can be computed by (DLOGTIME-uniform) AC^0-circuits parameterized by the size of the hitting set and the size of the hyperedges, see [3, 4]. Translated to our setting, these algorithms compute the core κ from Algorithm 1 for positive ϕ in AC^0 when parameterized by the maximum clause size k and the size h of the to-be-collapsed sunflowers. As the algorithm can easily be adapted to cope with the fact that sunflowers for formulas must take the "signs" of the literals in the cores into account, see the discussion prior to Corollary 3.2, we get the claim as both k and h are fixed. ◀

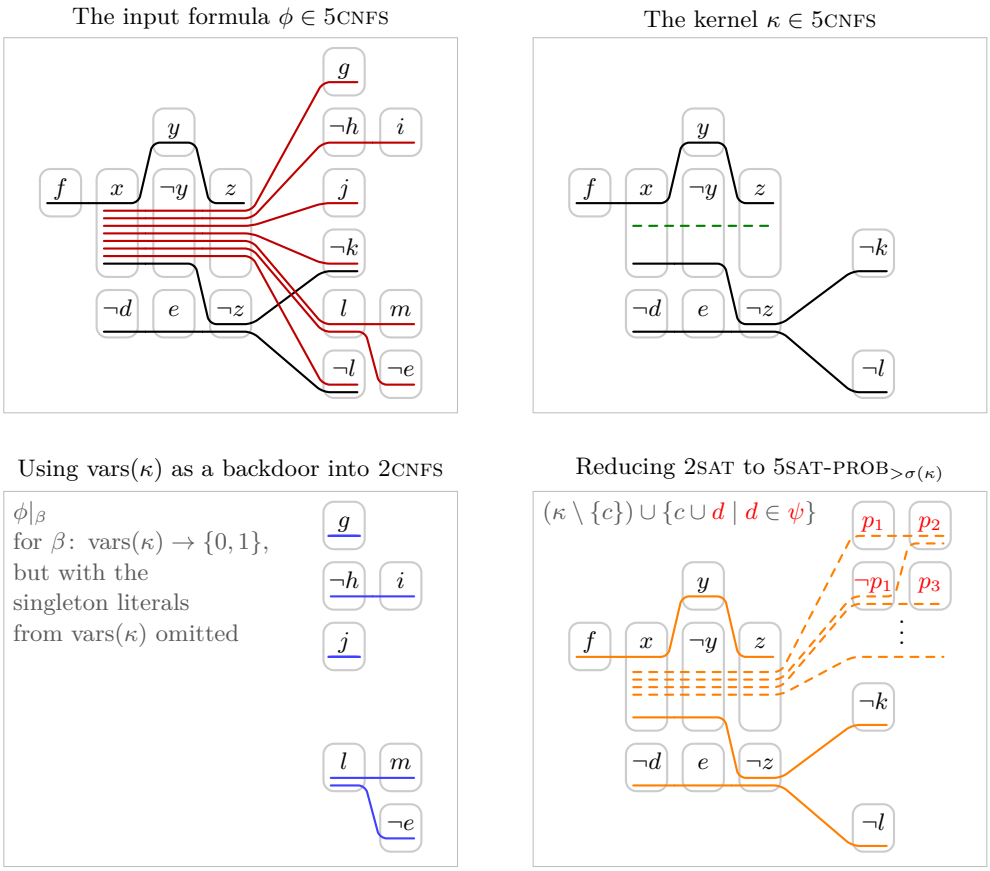


Figure 4 Continuing the example from Figure 2 from page 5: At the top, a formula $\phi \in$ 5CNFS is shown together with a kernel κ obtained by collapsing the red link to the dashed green clause $c = \{x, \neg y, z\}$. The size 3 of c is important: First, it is the minimum size of clauses in the kernel, which implies that if we *remove all variables in the kernel from the clauses of* ϕ, we get the lower left figure in which the blue clauses all have size at most $2 = 5 - 3$. In particular, $\phi|_\beta$ for β: vars(κ) $\to \{0,1\}$ lies in 2CNFS and vars(κ) is a backdoor into 2CNFS. Second, the clause c with $|c| \leq 3$ has "room left for two literals," meaning that for $\psi \in$ 2CNFS we have $\{c \cup d \mid d \in \psi\} \in$ 5CNFS. The lower right figure shows what happens when we add the clauses of ψ (assuming vars(ψ) $= \{p_1, \ldots, p_n\}$, these new variables are colored red) to the kernel by "adding them to c" (the dashed clauses). The resulting orange formula will have a satisfaction probability strictly larger than that of κ iff ψ is satisfiable. Note, however, that instead of κ we may need to use a related formula ω instead, see equation (3) on page 24 for details, namely when the clauses in κ other than c already imply c (which they do not in the example).

This theorem can be thought of as a strengthened version of the result by Akmal and Williams [2] that kSAT-PROB$_{\geq\delta} \in$ LINTIME holds for all k and δ, but two remarks are in order: First, the algorithm of Akmal and Williams can actually also be turned into an AC^0 algorithm [17], meaning that the result is already implicit in their work. Second, the existence of an AC^0-circuit family for a problem does not necessarily imply that the problem can be solved in linear time – one needs an AC^0 circuit family of linear size for this. It is, however, not clear whether such a family exists as the natural way of searching for packings or sunflowers involves color coding, which seems to need a *quadratic* size for derandomization. While these considerations will not be important for the complexity-theoretic statements of the main theorem, further research on the complexity of kSAT-PROB$_{\geq\delta}$ could address these questions.

4.2 Lower Complexity Bounds for the Strictly-Greater-Than Problem

We now show that kSAT-PROB$_{>\delta}$ is NL- or NP-hard for certain values of k and δ. For this, we need the already mentioned notion of "k-targets for lCNFS" whose definition was as follows (recall $\omega_{>s} = \{c \in \omega \mid |c| > s\}$ and $\omega_{\leq s} = \{c \in \omega \mid |c| \leq s\}$):

▶ **Definition 4.2.** *For numbers k and l, we say that a number $\delta \in [0,1]$ is a* k-target for lCNFS *if $\delta = \sigma(\omega)$ for some $\omega \in k$CNFS with $\omega_{>k-l} \not\models \omega_{\leq k-l}$.*

An example was $\delta = 7/32$, which is a 3-target for 2CNFS as demonstrated by $\omega = \big\{\{a\}, \{b\}, \{c_1, c_2, c_3\}\big\}$ since $\omega_{>3-2} = \big\{\{c_1, c_2, c_3\}\big\} \not\models \big\{\{a\}, \{b\}\big\} = \omega_{\leq 3-2}$. The term "target" is motivated by the following lemma: The formulas ω can serve as "targets for a reduction from lSAT", which proves the lower complexity bounds for kSAT-PROB$_{>\delta}$ from Theorem 1.11.

▶ **Lemma 4.3.** *If δ is a k-target for* 3CNFS*, then k*SAT-PROB$_{>\delta}$ *is* NP*-hard.*

Proof. We reduce 3SAT to kSAT-PROB$_{>\delta}$: Let ω witness that δ is a k-target for 3CNFS and let $\beta\colon \mathrm{vars}(\omega) \to \{0,1\}$ witness $\omega_{>k-3} \not\models \omega_{\leq k-3}$ (that is, β is a model of $\omega_{>k-3}$ but not of $\omega_{\leq k-3}$). Let $\psi \in$ 3CNFS be an input for the reduction, that is, we wish to reduce the question of whether $\sigma(\psi) > 0$ holds to the question of whether $\sigma(\rho) > \delta$ holds for some $\rho \in k$CNFS. If necessary, rename the variables in ψ to ensure $\mathrm{vars}(\psi) \cap \mathrm{vars}(\omega) = \emptyset$, and map ψ to $\rho = \omega_{>k-3} \cup \{c \cup d \mid c \in \omega_{\leq k-3}, d \in \psi\}$.

We claim $\sigma(\rho) > \delta$ iff $\psi \in$ 3SAT. For the first direction, assume $\sigma(\rho) > \delta = \sigma(\omega)$. As $\omega \models \rho$ (every clause of ρ is a superclause of some clause of ω), there must exist an assignment γ with $\gamma \models \rho \supseteq \omega_{>k-3}$ and $\gamma \not\models \omega \supseteq \omega_{\leq k-3}$. This means that there must be a clause $c^* \in \omega_{\leq k-3}$ with $\gamma \not\models \{c^*\}$. However, $\gamma \models \rho \supseteq \{c^* \cup d \mid d \in \psi\}$, which implies $\gamma \models \psi$ and $\psi \in$ 3SAT.

Second, assume $\psi \in$ 3SAT and let $\alpha\colon \mathrm{vars}(\psi) \to \{0,1\}$ witness $\psi \in$ 3SAT. Consider the assignment $\gamma\colon \mathrm{vars}(\psi) \cup \mathrm{vars}(\phi) \to \{0,1\}$ defined by $\gamma(v) = \alpha(v)$ for $v \in \mathrm{vars}(\psi)$ and $\gamma(v) = \beta(v)$ for $v \in \mathrm{vars}(\phi)$. Trivially, $\gamma \not\models \omega$ as $\beta \not\models \omega_{\leq k-3} \subseteq \omega$. However, $\gamma \models \rho$: Each clause $c \in \omega_{>k-3}$ is satisfied as $\beta \models \omega_{>k-3}$. Each clause $c \cup d \in \rho$ with $d \in \psi$ is satisfied as $\alpha \models \psi$ and $d \in \psi$. In total, γ is a model of all clauses of ρ and thus $\sigma(\rho) > \sigma(\omega) = \delta$. ◀

▶ **Lemma 4.4.** *If δ is a k-target for* 2CNFS*, then k*SAT-PROB$_{>\delta}$ *is* NL*-hard.*

Proof. The proof is identical to that of the previous lemma, only we reduce from 2SAT. ◀

In Figure 4 in the lower right subfigure the effect of adding the clauses of a formula $\psi \in$ 2CNFS to the small clauses of an example formula $\kappa = \omega$ is shown (although there is only a single small clause c in the example).

4.3 Upper Complexity Bounds for the Strictly-Greater-Than Problem

To complete the proof of Theorem 1.11, we need to prove the upper bounds. To get some intuition, let us start with the easiest case, the upper bound of NP, that is, the claim that kSAT-PROB$_{>\delta} \in$ NP always holds. Recall that in Theorem 4.1 we showed kSAT-PROB$_{\geq\delta} \in \mathrm{AC}^0$ by kernelizing input formulas ϕ: We replaced the question of whether $\sigma(\phi) \geq \delta$ holds by the question of whether $\sigma(\kappa) \geq \delta$ holds – and $\sigma(\kappa)$ can be computed by brute force. However, this is not directly helpful for deciding whether $\sigma(\phi) > \delta$ holds: If $\sigma(\kappa) > \delta$, we also know that $\sigma(\phi) > \delta$ holds; if $\sigma(\kappa) < \delta$, we also know that $\sigma(\phi) < \delta$ holds (because of the spectral gap); but if $\sigma(\kappa) = \delta$, both $\sigma(\phi) = \delta$ and $\sigma(\phi) > \delta$ are still possible. The "critical" case $\sigma(\kappa) = \delta$ forces us to investigate further: We must find out whether the sunflower collapsing process "destroyed solutions," that is, whether there is an assignment $\beta\colon \mathrm{var}(\phi) \to \{0,1\}$ with $\beta \models \phi$, but $\beta \not\models \kappa$. Fortunately, this is easy to do using an NP-machine: We can just guess such an assignment. Let us state this observation as a lemma for future reference:

▶ **Lemma 4.5.** *For all k and δ, we have k*SAT-PROB$_{>\delta} \in$ NP*.*

For the NL upper bound, we can basically proceed the same way – we just need a way to determine whether there exists a β with $\beta \models \phi$, but $\beta \not\models \kappa$ using an NL-machine. An idea towards achieving this is to simply iterate over all possible $\beta\colon \mathrm{vars}(\kappa) \to \{0,1\}$ with $\beta \not\models \kappa$ and then test whether $\phi|_\beta$ is satisfiable. Of course, we still have to answer (many) questions of the form "$\phi|_\beta \in$ SAT?" – but we may now hope that $\phi|_\beta \in$ 2CNFS might hold: After all, clauses in $\phi|_\beta$ result from removing all variables in the core from the clauses of ϕ and a large core thus means small clauses in $\phi|_\beta$, see the lower left part of Figure 4 for a concrete example. If we always had $\phi|_\beta \in$ 2CNFS, then $\mathrm{vars}(\kappa)$ would be called a *backdoor set into* 2CNFS and this would suffice to show kSAT-PROB$_{>\delta} \in$ NL.

Now, the variables in the kernel do *not* always form a backdoor set into 2CNFS, *but* we will be able to construct a formula ω for which they do and this formula will serve as a "replacement" for ϕ: If it is not equivalent to ϕ, then either $\sigma(\phi) > \sigma(\omega) > \delta$ will follow immediately or δ is a k-target for 3CNFS. As the latter is ruled out by the assumption of the theorem, we will have $\sigma(\phi) > \delta$ iff $\sigma(\omega) > \delta$. Thus, having a backdoor set in 2CNFS for ω will suffice to show kSAT-PROB$_{>\delta} \in$ NL. The details follow.

Backdoor Sets. Backdoor sets are a powerful tool from FPT theory [18] with a rich theory around them; but for our purposes only the following kind will be important:

▶ **Definition 4.6.** *Let $\Delta \subseteq$ CNFS and let $\psi \in$ CNFS. A (strong)* backdoor set for ψ into Δ *is a set V of variables such that for all $\beta\colon V \to \{0,1\}$ we have $\psi|_\beta \in \Delta$.*

The importance of backdoor sets in FPT theory lies, of course, in the fact that in order to determine whether ψ is satisfiable, it suffices to find a $\beta\colon V \to \{0,1\}$ such that $\psi|_\beta$ is satisfiable. If Δ is a tractable set (like $\Delta =$ 2CNFS), this allows us to efficiently decide $\psi \in$ SAT as long as the backdoor set is not too large. In our context, the variables in the kernel will form a backdoor set into 2CNFS or 1CNFS, not for the original formula ϕ, but only for an "intermediate" formula ω, defined as follows.

The Replacement Formula. Fix k and l and fix a formula $\phi \in k$CNFS. Let κ denote the kernel computed in Algorithm 1 on input ϕ (that is, the result of collapsing the links of cores of sunflowers large enough to ensure that the collapse step reduces the satisfaction probability by less than the spectral gap of δ, which ensures by the No Tunneling Lemma that $\sigma(\phi)$ "stays above δ, if it was above δ") and assume that we are in the critical case $\delta = \sigma(\kappa)$ where

"there is still work to be done." Define $\omega \in k\text{CNFS}$ as follows (with $\kappa_{<s} = \{c \in \kappa \mid |c| < s\}$ and $\kappa_{\geq s} = \{c \in \kappa \mid |c| \geq s\}$; and note that $\kappa_{<s} = \kappa_{\leq s-1}$ and $\kappa_{\geq s} = \kappa_{>s-1}$, so the subscripts of κ are "shifted by -1" relative to the subscripts of ω in Definition 4.2):

$$\omega := \kappa_{<k-l} \cup \bigcup_{c \in \kappa_{\geq k-l}} \text{link}_\phi(c). \tag{3}$$

The importance of ω lies in the following two lemmas:

▶ **Lemma 4.7** (Backdoor Lemma). *The set* $\text{vars}(\kappa)$ *is a backdoor set for* ω *into* $l\text{CNFS}$.

Proof. For any $\beta\colon \text{vars}(\kappa) \to \{0,1\}$, consider any clause $e \in \omega|_\beta$. We wish to show $|e| \leq l$. By definition of $\omega|_\beta$, e resulted from taking some clause in $d \in \omega$ and stripping away all occurrences of variables in $\text{vars}(\kappa)$. If $d \in \kappa_{<k-l} \subseteq \kappa$, we would have $e = \emptyset$ and $|e| = 0$. If $d \in \text{link}_\phi(c)$ for some $c \in \kappa_{\geq k-l}$, then $|e| = |d \setminus c| = |d| - |c| \leq k - (k-l) = l$. ◀

The Backdoor Lemma tells us that we can "handle" ω well for $l = 2$ and $l = 1$. The question is, of course, how, exactly, ω is related to ϕ and κ. The next lemma tells the story:

▶ **Lemma 4.8.** *One of the following always holds:*

1. $\delta = \sigma(\kappa) = \sigma(\omega) = \sigma(\phi)$ *or*
2. $\delta = \sigma(\kappa) < \sigma(\omega) = \sigma(\phi)$ *or*
3. $\delta = \sigma(\kappa) < \sigma(\omega) < \sigma(\phi)$ *or*
4. δ *is a* k*-target for* $(l+1)\text{CNFS}$.

Proof. As we clearly have $\kappa \models \omega \models \phi$ (that is, every model of κ is also a model of ω, and ω's models are models of ϕ), we have $\delta = \sigma(\kappa) \leq \sigma(\omega) \leq \sigma(\phi)$. This means that the only possibility not covered by the first three items is $\delta = \sigma(\kappa) = \sigma(\omega) < \sigma(\phi)$ and we must show that this implies that δ is a k-target for $(l+1)\text{CNFS}$. For this we must show that there is an assignment β with $\beta \not\models \omega_{\leq k-l-1}$ and $\beta \models \omega_{>k-l-1}$.

If $\sigma(\omega) < \sigma(\phi)$, there must be a model $\beta \models \phi$ with $\beta \not\models \omega$ and we claim that this model is a witness for δ being a k-target. To see this, first note that $\beta \models \phi$ and $\beta \not\models \omega$ implies $\beta \not\models \omega \setminus \phi$ and $\omega \setminus \phi \subseteq \omega_{<k-l}$ by (3). Thus, $\beta \not\models \omega_{\leq k-l-1} = \omega_{<k-l}$. Second, $\beta \models \omega_{>k-l-1} = \omega_{\geq k-l}$ is implied by $\beta \models \phi$ together with $\phi \supseteq \omega_{\geq k-l}$ as every clause of size $k-l$ or larger in ω comes from a link and links are subsets of ϕ. ◀

The Algorithm and Its Correctness. We are now ready to assemble the ideas into an algorithm, Algorithm 4, and prove its correctness.

▶ **Lemma 4.9.** *Let* δ *not be a* k*-target for* $(l+1)\text{CNFS}$. *Then for each* $\phi \in k\text{CNFS}$ *the output of* REDUCE-TO-SAT(ϕ) *from Algorithm 4 will be correct.*

Proof. The output of the then-clause of the first if-statement in line 3 is correct as we trivially have $\sigma(\kappa) \leq \sigma(\phi)$ (replacing a sunflower by its core can only reduce the satisfaction probability). The output of the then-clause of the second if in line 4 is also correct as $\sigma(\kappa) \geq \delta$ iff $\sigma(\phi) \geq \delta$ holds by Lemma 3.3.

By Lemma 4.8, for ω computed in line 6 one of the following holds (the fourth possibility is ruled out by the assumption):

1. $\delta = \sigma(\kappa) = \sigma(\omega) = \sigma(\phi)$ or
2. $\delta = \sigma(\kappa) < \sigma(\omega) = \sigma(\phi)$ or
3. $\delta = \sigma(\kappa) < \sigma(\omega) < \sigma(\phi)$.

Algorithm 4 An algorithm for deciding kSAT-PROB$_{>\delta}$ that is correct if δ is *not* a k-target for $(l+1)$CNFS: In this case, Lemma 4.8 states that ω is a suitable replacement for ϕ and Lemma 4.7 states that the variables in the kernel form a backdoor set into lCNFS for ω, allowing us to perform the tests in line 8 by an NL machine for $l = 2$, and even by AC^0-circuits for $l = 1$.

1 ***algorithm*** REDUCE-TO-SAT(ϕ) // $\phi \in k$CNFS must hold, l is a number
2 $\quad\kappa \leftarrow$ KERNELIZE$\big(\phi, \log_{1-2^{-k}}(\text{spectral-gap}_{k\text{CNFS}}(\delta))\big)$
3 $\quad$***if*** $\sigma(\kappa) > \delta$ ***then return*** "$\sigma(\phi) > \delta$"
4 $\quad$***if*** $\sigma(\kappa) < \delta$ ***then return*** "$\sigma(\phi) < \delta$"
5 $\quad$// Critical case: $\sigma(\kappa) = \delta$
6 $\quad\omega \leftarrow \kappa_{<k-l} \cup \bigcup_{c \in \kappa_{\geq k-l}} \text{link}_\phi(c)$
7 $\quad$***foreach*** $\beta\colon \text{vars}(\kappa) \to \{0,1\}$ *with* $\beta \not\models \kappa$ ***do***
8 $\quad\quad$***if*** $\omega|_\beta \in l$SAT ***then***
9 $\quad\quad\quad$// we now know $\sigma(\omega) > \sigma(\kappa) = \delta$
10 $\quad\quad\quad$***return*** "$\sigma(\phi) > \delta$"
11 $\quad$// we now know $\sigma(\omega) = \sigma(\kappa) = \delta$
12 $\quad$***return*** "$\sigma(\phi) = \delta$"

In particular, if $\delta = \sigma(\kappa) = \sigma(\omega)$, we know that $\delta = \sigma(\phi)$ must also hold; and if $\sigma(\kappa) < \sigma(\omega)$ we trivially have $\delta < \sigma(\phi)$. In other words:

$$\sigma(\omega) > \delta \iff \sigma(\phi) > \delta. \tag{4}$$

In the main for-loop of the algorithm, we check whether there is a model of ω that is not a model of κ. Clearly, this is the case iff there is some $\beta\colon \text{vars}(\kappa) \to \{0,1\}$ with $\beta \not\models \kappa$ and $\omega|_\beta \in$ SAT. However, since by Lemma 4.7 we know that vars(κ) is a backdoor for ω into lCNFS, it is correct to test only $\omega|_\beta \in l$SAT inside the loop.

All told, when the comment lines 9 or 11 are reached, the comments' statements are correct. By (4) this means that the two outputs in the subsequent lines are correct. ◀

The Upper Bounds. We can now prove the three upper bounds from Theorem 1.11 and thus prove the Spectral Trichotomy Theorem, Theorem 1.6 from the introduction. The NP upper bound has already been stated in Lemma 4.5. The argument for NL is as follows:

▶ **Lemma 4.10.** *If δ is not a k-target for* 3CNFS*, then* kSAT-PROB$_{>\delta} \in$ NL*.*

Proof. By Lemma 4.9, Algorithm 4 will correctly decide kSAT-PROB$_{>\delta}$ in this case. To see that the algorithm can be implemented by an NL-machine, observe that the for-loop iterates only over a constant number of β (the kernel size is fixed) and can thus be hardwired into the machine. The central test $\omega|_\beta \in 2$SAT clearly only requires an NL machine. ◀

▶ **Lemma 4.11.** *If δ is not a k-target for* 2CNFS*, then* kSAT-PROB$_{>\delta} \in$ AC^0*.*

Proof. Just as in the previous corollary, we invoke Lemma 4.9 and the for-loop still iterates only over a constant number of β. The central test is now $\phi|_\beta \in 1$SAT, which is possible to perform using AC^0 circuits. ◀

5 Conclusion

The results of the present paper settle the complexity of kSAT-PROB$_{>\delta}$ from a complexity-theoretic point of view: The problem is either NP-complete or NL-complete or lies in AC^0 – and which of these is the case depends on whether or not $\delta = \sigma(\omega)$ holds for some

formula ω with certain syntactic properties. The proof is based on the insight that the spectra kCNFS-σ-SPECTRUM are well-ordered with respect to $>$, as this implies (1) that the standard sunflower-based kernel algorithm for hitting sets allows us to compute kernels for kSAT-PROB$_{\geq\delta}$ and (2) that the variables in the kernel form strong backdoor sets into 2CNFS or 1CNFS for formulas whose satisfaction probabilities "behave the same way" as those of the input formula.

An attempt to visualize the "landscape" of the complexity of kSAT-PROB$_{>\delta}$ for $k \leq 4$ can be found in Figure 1 on page 2. For $k = 4$, two values of special interest are $\delta_1 = 15/32 = \frac{1}{2} - \frac{1}{32}$ and $\delta_2 = 63/128 = \frac{1}{2} - \frac{1}{128}$. There is a "red triangle" in the figure at δ_1, meaning that 4SAT-PROB$_{>15/32}$ is NP-complete by Theorem 1.11 as $15/32 = \sigma\big(\big\{\{a\}, \{x, y, z, w\}\big\}\big)$. There is a "green triangle" at δ_2 (as well as at many, many other positions in $(15/32, 1/2)$, but still only at a nowhere dense subset despite the "solid line" in the visualization) as 4SAT-PROB$_{>63/128}$ is NL-complete since $63/128 = \sigma(\{\{a, b\}, \{c, d\}, \{e, f, g\}\})$, but no 4CNF formula ϕ containing a singleton clause has $\sigma(\phi) = 63/128$.

For larger values of k, observe that, on the one hand, kSAT-PROB$_{>1-2^{-(k-2)}}$ is NL-complete for all k (since $\sigma\big(\big\{\{a_1, \ldots, a_{k-2}\}\big\}\big) = 1 - 2^{-(k-2)}$, but $\sigma(\omega) \neq 1 - 2^{-(k-2)}$ for all $\omega \in k$CNFS containing a clause of size $k-3$ as this clause already lowers the satisfaction probability to at most $1 - 2^{-(k-3)} < 1 - 2^{-(k-2)}$); while on the other hand, kSAT-PROB$_{>2^{-i}}$ is NP-complete for all $k \geq 4$ and $i \geq 1$.

In addition to the "strictly greater than" problem kSAT-PROB$_{>\delta}$ and the "boring" problem kSAT-PROB$_{\geq\delta}$ (which is always in AC^0), one can also consider the "equal to" version. Combining the results from this paper immediately yields: *For the same k and δ as in Theorem 1.11, the problem k*SAT-PROB$_{=\delta}$ *is* coNP*-complete,* NL*-complete, or lies in* AC^0. Spelled out, we get results like the following: "it is NL-complete to decide on input of a 3CNF formula whether exactly half of the assignments are satisfying" and "it is coNP-complete to decide on input of a 4CNF formula whether exactly half of the assignments are satisfying," but also stranger ones like "it is NL-complete to decide on input of a 4CNF formula whether the fraction of satisfying assignments is exactly $\frac{1}{2} - \frac{1}{128}$" while "it is coNP-complete to decide on input of a 4CNF formula whether the fraction of satisfying assignments is exactly $\frac{1}{2} - \frac{1}{32}$".

Since the algorithms presented in this paper depend so heavily on the size of spectral gaps, it is of interest to determine these sizes precisely. While explicit bounds can be shown [14], it is very much unclear whether these hyperexponentiation bounds are even remotely tight. It would also be of interest to determine explicit values: A close look at Figure 1 reveals spectral-gap$_{2\text{CNFS}}(1/2) = 1/32$, but what is the value of spectral-gap$_{3\text{CNFS}}(1/2)$?

On the one hand, the results of the present paper "settle" the complexity of many versions of satisfaction probability threshold problems for kCNFS, including the majority-of-majority version; on the other hand, many new problems arise. These include questions concerning satisfaction probability problems for constraint satisfaction problems, questions concerning the fixed-parameter tractability of satisfaction probability problems, questions surrounding the complexity of algebraic representations of formulas, a whole bunch of questions arising from logical and descriptive reformulations, and finally – of course – questions concerning practical implementation. In the technical report version [14] some first ideas and partial answers to these questions are presented, but there is certainly still much to do.

References

1 Shyan Akmal and Ryan Williams. MAJORITY-3SAT (and related problems) in polynomial time. Technical Report abs/2107.02748, Cornell University, 2021. arXiv:2107.02748.

2 Shyan Akmal and Ryan Williams. MAJORITY-3SAT (and related problems) in polynomial time. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1033–1043, 2022. Citations and page references in the main text refer to the technical report version [1] of this paper. doi:10.1109/FOCS52979.2021.00103.

3 Max Bannach, Christoph Stockhusen, and Till Tantau. Fast parallel fixed-parameter algorithms via color coding. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015*, volume 43 of *LIPIcs*, pages 224–235. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015.

4 Max Bannach and Till Tantau. Computing kernels in parallel: Lower and upper bounds. In *13th International Symposium on Parameterized and Exact Computation, IPEC 2018*, volume 115 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.

5 Max Bannach and Till Tantau. Computing hitting set kernels by AC^0-circuits. *Theory Comput. Syst.*, 64(3):374–399, 2020.

6 Arthur Choi, Yexiang Xue, and Adnan Darwiche. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning*, 53(9):1415–1428, 2012. doi:10.1016/j.ijar.2012.04.005.

7 Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971.

8 P. Erdős and R. Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960.

9 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006. doi:10.1007/3-540-29953-X.

10 John T. Gill. Computational complexity of probabilistic turing machines. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pages 91–95, New York, NY, USA, 1974. Association for Computing Machinery. doi:10.1145/800119.803889.

11 Thomas Jech. *Set Theory: The Third Millennium Edition*. Springer, 2003.

12 Umut Oztok, Arthur Choi, and Adnan Darwiche. Solving pppp-complete problems using knowledge compilation. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'16, pages 94–103. AAAI Press, 2016.

13 Janos Simon. *On Some Central Problems in Computational Complexity*. PhD thesis, Cornell University, January 1975.

14 Till Tantau. On the satisfaction probability of kCNF formulas. Technical Report abs/2201.08895, Cornell University, 2022. arXiv:2201.08895.

15 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi:10.1137/0208032.

16 René van Bevern. Towards optimal and expressive kernelization for d-hitting set. *Algorithmica*, 70(1):129–147, September 2014. doi:10.1007/s00453-013-9774-3.

17 Ryan Williams. Personal communication, 2021.

18 Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1173–1178, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

Hitting Sets for Regular Branching Programs

Andrej Bogdanov
The Chinese University of Hong Kong, China

William M. Hoza
Simons Institute for the Theory of Computing, Berkeley, CA, USA

Gautam Prakriya
The Chinese University of Hong Kong, China

Edward Pyne
Harvard University, Cambridge, MA, USA

Abstract

We construct improved hitting set generators (HSGs) for ordered (read-once) regular branching programs in two parameter regimes. First, we construct an explicit ε-HSG for *unbounded-width* regular branching programs with a single accept state with seed length

$$\widetilde{O}(\log n \cdot \log(1/\varepsilon)),$$

where n is the length of the program. Second, we construct an explicit ε-HSG for width-w length-n regular branching programs with seed length

$$\widetilde{O}\left(\log n \cdot \left(\sqrt{\log(1/\varepsilon)} + \log w\right) + \log(1/\varepsilon)\right).$$

For context, the "baseline" in this area is the pseudorandom generator (PRG) by Nisan (Combinatorica 1992), which fools ordered (possibly non-regular) branching programs with seed length $O(\log(wn/\varepsilon) \cdot \log n)$. For regular programs, the state-of-the-art PRG, by Braverman, Rao, Raz, and Yehudayoff (FOCS 2010, SICOMP 2014), has seed length $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$, which beats Nisan's seed length when $\log(w/\varepsilon) = o(\log n)$. Taken together, our two new constructions beat Nisan's seed length in all parameter regimes except when $\log w$ and $\log(1/\varepsilon)$ are *both* $\Omega(\log n)$ (for the construction of HSGs for regular branching programs with a single accept vertex).

Extending work by Reingold, Trevisan, and Vadhan (STOC 2006), we furthermore show that an explicit HSG for regular branching programs with a single accept vertex with seed length $o(\log^2 n)$ in the regime $\log w = \Theta(\log(1/\varepsilon)) = \Theta(\log n)$ would imply improved HSGs for *general* ordered branching programs, which would be a major breakthrough in derandomization. Pyne and Vadhan (CCC 2021) recently obtained such parameters for the special case of permutation branching programs.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Pseudorandomness, hitting set generators, space-bounded computation

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.3

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2021/143/`

Funding *Andrej Bogdanov*: Supported by RGC GRF CUHK 14209419 and CUHK 14209920. Part of the work was completed at the Simons Institute for the Theory of Computing, UC Berkeley.
William M. Hoza: This work was done while the author was visiting the Simons Institute for the Theory of Computing.
Edward Pyne: Supported by NSF grant CCF-1763299.

Acknowledgements We thank Sumegha Garg and Salil Vadhan for many helpful discussions, and Salil Vadhan and Chin Ho Lee for comments on a draft of this paper.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 3; pp. 3:1–3:22

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

1 Introduction

Random choices can make computing easier sometimes, but random bits are not always available. We therefore want to understand when randomized algorithms have an inherent advantage and when randomness is unnecessary. In this work, we focus on the interplay between randomness and *space complexity*. Starting with the work of Ajtai, Komlós, and Szemerédi [2], there have been three decades of work on the derandomization of space-bounded computation, with the goal of eventually proving that every halting decision algorithm can be derandomized with only a constant factor space blowup ($\mathbf{L} = \mathbf{BPL}$). As in previous work, we will use the following nonuniform model of space-bounded computation, which captures how a randomized small-space algorithm uses its random bits.

▶ **Definition 1.** *An **(ordered) branching program** B of **length** n and **width** w computes a function $B : \{0,1\}^n \to \{0,1\}$. On an input $x \in \{0,1\}^n$, the branching program computes as follows. It starts at a fixed start state $v_0 \in [w]$. Then for $t = 1, \ldots, n$, it reads the next input bit x_t and updates its state according to a transition function $B_t : [w] \times \{0,1\} \to [w]$ by taking $v_t = B_t(v_{t-1}, x_t)$. Note that the transition function B_t can differ at each time step.*

Moreover, there is a set $V_{acc} \subseteq [w]$ of accept states. Let v_n be the final state reached by the branching program on input x. If $v_n \in V_{acc}$ the branching program accepts, denoted $B(x) = 1$, and otherwise the program rejects, denoted $B(x) = 0$. We also consider branching programs restricted to having a single accept state, which is always denoted v_{acc}.

Arguably the most natural approach to derandomizing space-bounded computation is to design a *pseudorandom generator*, defined next. We let U_i denote the uniform distribution over $\{0,1\}^i$.

▶ **Definition 2.** *Let $\mathcal{F}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$. An **$\varepsilon$-pseudorandom generator (ε-PRG)** for $\mathcal{F}$ is a function $G : \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,*

$$\left| \Pr_{x \leftarrow U_n}[f(x) = 1] - \Pr_{x \leftarrow U_s}[f(G(x)) = 1] \right| \leq \varepsilon.$$

*We say that G **ε-fools** $\mathcal{F}$ if it is an ε-PRG for $\mathcal{F}$. The input length s is the **seed length** of the generator.*

It can be shown via the probabilistic method that there is a (non-explicit) ε-PRG for ordered branching programs of length n and width w that has seed length $O(\log(nw/\varepsilon))$, and moreover this seed length is optimal. We say a generator G is **explicit** if the output is computable in space $O(s)$. Decades of work has focused on constructing explicit pseudorandom generators with parameters matching the probabilistic method. All results we subsequently discuss are explicit constructions. In 1990, Nisan [23] constructed an ε-PRG for general ordered branching programs of length n and width w with seed length

$$O(\log n \cdot (\log n + \log w + \log(1/\varepsilon))).$$

Nisan's PRG is a factor of $O(\log n)$ from optimal, and achieves seed length $O(\log^2 n)$ when $w \leq \text{poly}(n)$ and $\varepsilon \geq 1/\text{poly}(n)$, rather than the optimal $O(\log n)$.

There has been extensive work analyzing restricted classes of branching programs with additional structure. We focus on the well-studied class of *regular* programs:

▶ **Definition 3.** *An **(ordered) regular branching program** of length n and width w is an ordered branching program where for every $t = 1, \ldots, n$ and every $v \in [w]$, there are exactly 2 pairs $(u, b) \in [w] \times \{0,1\}$ such that $B_t(u, b) = v$.*

In 2010, Braverman, Rao, Raz and Yehudayoff [6] constructed a PRG for regular branching programs with near-optimal dependence on n, achieving seed length:[1]

$$O(\log n \cdot (\log\log n + \log w + \log(1/\varepsilon))).$$

(Subsequently, De also presented a PRG for regular programs, albeit with a somewhat inferior seed length [13].) Given Braverman et al.'s result [6], there are two natural challenges regarding regular programs. The first challenge is to improve the $\log n \cdot \log w$ term in the seed length; this is necessary to beat Nisan's generator in the polynomial-width regime (e.g., $w = n$). The second challenge is to improve the $\log n \cdot \log(1/\varepsilon)$ term; this is necessary to beat Nisan's generator in the small-error regime (e.g., $\varepsilon = 1/n$).

Designing PRGs that meet these challenges seems to be quite difficult, but fortunately PRGs are not the only approach to derandomization. To address the challenges we will instead aim to construct *hitting set generators* (HSGs). An HSG (defined next) is a "one-sided" generalization of a PRG that is still valuable for derandomization.

▶ **Definition 4.** *Let $\mathcal{F}$ be a class of functions $f : \{0,1\}^n \to \{0,1\}$. An **$\varepsilon$-hitting set generator ($\varepsilon$-HSG)** for $\mathcal{F}$ is a function $H : \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$ where $\Pr_{x \leftarrow U_n}[f(x) = 1] > \varepsilon$, there exists $x \in \{0,1\}^s$ such that $f(H(x)) = 1$.*

Note that with our definitions, an ε-PRG for a class $\mathcal{F}$ is an ε-HSG for $\mathcal{F}$. In many cases, there has been more success at developing HSGs than at developing PRGs. Indeed, in the context of regular branching programs, in addition to their PRG construction, Braverman, Rao, Raz, and Yehudayoff constructed an HSG with seed length $O(w \log n)$ (the set of all strings of Hamming weight at most w), achieving optimal seed length for constant width [6].[2]

1.1 Our Contributions

In this work, we present improved HSGs for regular branching programs. For our first result, we focus on improving the dependence on w, the width of the program. In fact, we study the intriguing setting of *unbounded-width* programs [22, 18, 24, 25]. We design an HSG for unbounded-width regular branching programs with a single accept vertex with a near-optimal dependence on the length of the program n.

▶ **Theorem 5.** *Given $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there is an explicit ε-HSG for regular branching programs of length n and unbounded width with a single accept state that has seed length*

$$O(\log n \cdot (\log\log n + \log(1/\varepsilon))).$$

This result eliminates all dependence on w from the seed length of Braverman, Rao, Raz, and Yehudayoff's PRG [6], with the caveats that we only obtain an HSG and we assume that there is only one accept state. For regular branching programs of width $w = \mathrm{poly}(n)$ (the regime most relevant for the derandomization of space-bounded computation), Theorem 5 is the first explicit construction with seed length $o(\log^2 n)$. In the superpolynomial-width

[1] They consider regular branching programs with a single accept state, but dividing ε by w to allow an arbitrary set of accept states does not change the seed length.

[2] The lack of dependence on ε can be explained by Braverman et al.'s observation that every width-w regular branching program that has nonzero acceptance probability has acceptance probability at least $1/2^{w-1}$ [6], so without loss of generality $\varepsilon > 1/2^w$, i.e., $w > \log(1/\varepsilon)$.

regime, the state of the art prior to our work was the analysis of Hoza, Pyne, and Vadhan [18], which implies that Nisan's generator is an HSG for unbounded-width regular branching programs with a single accept vertex with seed length $O(\log n \cdot \log(n/\varepsilon))$.

Even under the assumption that there is only one accepting vertex, the unbounded-width regular branching program model is highly nontrivial. For example, it can compute doubly exponentially many distinct functions [18]. Admittedly, there are also some very simple functions that it cannot compute, such as the majority function on three bits. However, Theorem 5 extends to the setting of programs with a accept states where a is small. Indeed, if such a program has acceptance probability at least ε, then there must be an accept state that is reached with probability at least ε/a, so we obtain the following corollary:[3]

▶ **Corollary 6.** *Given $n, a \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$, there is an explicit ε-HSG for regular branching programs of length n and unbounded width with a accept states that has seed length*

$$O(\log n \cdot (\log\log n + \log(a/\varepsilon))).$$

For our second result, we focus on improving the dependence on ε, the threshold of the HSG.

▶ **Theorem 7.** *For every $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there exists an explicit ε-HSG for width-w length-n regular branching programs with seed length*

$$O\left(\log n \cdot \left(\sqrt{\log(1/\varepsilon)} + \log w + \log\log n\right) + \log(1/\varepsilon)\right).$$

Comparing to the seed length of Braverman, Rao, Raz, and Yehudayoff's PRG [6], we improve the $\log n \cdot \log(1/\varepsilon)$ term to $\log n \cdot \sqrt{\log(1/\varepsilon)}$. Recall that Braverman et al. also constructed an HSG with seed length $O(w \log n)$, independent of ε. Our seed length has a much better dependence on w, so for example, when $w = 2^{O(\sqrt{\log n})}$ and $\varepsilon = 1/n$, our seed length is $\widetilde{O}(\log^{3/2} n)$, whereas prior work could not beat Nisan's $O(\log^2 n)$ seed length for that regime. Furthermore, since every nonzero width-w regular program has acceptance probability at least $2^{-(w-1)}$ [6], Theorem 7 implies that we can achieve seed length $\widetilde{O}\left(\sqrt{w}\log n + w\right)$, independent of ε. Theorem 7 makes progress on a problem posed by Hoza and Zuckerman [19]: they asked for an HSG with seed length $\widetilde{O}(\log(n/\varepsilon))$ for regular branching programs of width $\operatorname{polylog} n$.

Taken together, Theorems 5 and 7 improve the seed length of Nisan's construction for hitting regular branching programs with a single accept state when $\log(1/\varepsilon) = o(\log n)$ *or* $\log w = o(\log n)$, thereby identifying the regime $\log(1/\varepsilon) \sim \log w \sim \log n$ as the remaining target. An explicit HSG of seed length $o(\log^2 n)$ in that regime would also be an explicit HSG of seed length $o(\log^2 n)$ for polynomial-width regular branching programs with an *arbitrary* set of accept states. In turn, we show that such an HSG would imply a major advance in derandomizing space-bounded computation:

▶ **Theorem 8.** *For every $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there are values $w' = \operatorname{poly}(nw/\varepsilon)$ and $n' = O(n \log(nw/\varepsilon))$ such that if there is an explicit ε-HSG (resp. PRG) $G : \{0,1\}^s \to \{0,1\}^{n'}$ for width-w' length-n' regular branching programs, then there is an explicit $O(\varepsilon)$-HSG (resp. PRG) $G' : \{0,1\}^s \to \{0,1\}^n$ for width-w length-n ordered branching programs with the same seed length.*

[3] We remark that it is not possible to improve this dependence on a by any analysis of the INW generator that only uses the expansion properties of the auxiliary expanders [26], even for HSGs.

The conclusion of Theorem 8 yields a derandomization of decision problems solvable in randomized logspace with one-sided error (the class **RL**). Cheng and Hoza [10] show more generally that a two-sided error derandomization (the class **BPL**) follows. Thus we obtain the following corollary:

▶ **Corollary 9.** *Suppose there is an explicit ε-HSG of seed length $O(\log(nw/\varepsilon))$ for regular branching programs of length n and width w. Then* $\mathbf{BPL} = \mathbf{L}$.

1.2 Related work

Theorem 8 extends a result of Reingold, Trevisan, and Vadhan [28]. They show an analogous transformation for PRGs over alphabet size $\mathrm{poly}(nw/\varepsilon)$. In contrast, Theorem 8 also applies to HSGs and works over the binary alphabet.

A number of results improve the seed length of Braverman et al. [6] for the restricted class of "permutation" branching programs. A *permutation branching program* is a regular branching program with the further restriction that the transitions $B_t(\cdot, 1)$ and $B_t(\cdot, 0)$ are permutations of $[w]$ for every t. While most of these results are tailored to the constant-width regime [7, 20, 13, 31, 27], two exceptions construct generators for unbounded-width permutation branching programs with a single accept vertex:

- Hoza, Pyne, and Vadhan [18] (building on work by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [1]) construct a PRG with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$, and
- Pyne and Vadhan [24] construct an HSG[4] with seed length $\widetilde{O}\left(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon)\right)$.

Our Theorem 5 matches the seed length of Hoza, Pyne, and Vadhan [18] for the more general setting of regular programs. Our Theorem 7 can be viewed as an analogue of the result of Pyne and Vadhan [24]. Unfortunately, our seed length includes an additional $O(\log n \cdot \log w)$ term. If this term were at all improved, we would obtain $o(\log^2 n)$ seed length HSGs for general ordered branching programs via Theorem 8.

The arguments used for unbounded-width permutation branching programs [18, 24] rely heavily on the permutation condition to leverage powerful results in spectral graph theory [30, 12, 1] that are not applicable in the regular setting.[5] In contrast, our proofs are combinatorial. In particular, our Theorem 7 is proved via combinatorial rather than spectral error-reduction methods, providing some hope that the aforementioned improvement in the $O(\log n \cdot \log w)$ term might not be out of reach.

1.3 Overview of Proofs

1.3.1 The Unanimity Program Model

The proofs of Theorem 5 and Theorem 7 both rely on a new generalization of ordered branching programs that we call *unanimity programs*. A unanimity program is defined like an ordered branching program, except that every vertex (not just those in the last layer) is labeled either "accept" or "reject." The program accepts if *every* vertex it visits is an accepting vertex; otherwise it rejects. More precisely:

[4] Their result gives a more general object called "weighted PRG" or "pseudorandom pseudodistribution" [5].
[5] Specifically, permutation branching programs have the property that the underlying graph remains regular – and hence the uniform distribution remains stationary – even when we restrict to a pseudorandom sequence of paths.

▶ **Definition 10.** *An **(ordered) unanimity program** B of length n and width w starts at a fixed start state $v_0 \in [w]$. In each step $t \in [n]$, the program reads the next input symbol x_t and updates its state according to a transition function $B_t \colon [w] \times \{0,1\} \to [w]$ by taking $v_t = B_t(v_{t-1}, x_t)$. For every $t \in \{0, 1, \ldots, n\}$, there is a set of accept states $V_{acc}^{(t)} \subseteq [w]$. The program accepts, denoted $B(x) = 1$, if for every t, we have $v_t \in V_{acc}^{(t)}$. Otherwise the program rejects, denoted $B(x) = 0$.*

*A unanimity program is **regular** if for every $t = 1, \ldots, n$ and every $v \in [w]$, there are exactly two pairs $(u, b) \in [w] \times \{0, 1\}$ such that $B_t(u, b) = v$. The program is a **permutation unanimity program** if for every $t = 1, \ldots, n$ and every $b \in \{0, 1\}$, the function $B_t(\cdot, b)$ is a permutation on $[w]$.*

The standard definition of an ordered branching program is the special case that for $t < n$, we have $V_{\text{acc}}^{(t)} = [w]$. Throughout this paper, the phrase "branching program" will always refer to the standard model, whereas "unanimity program" will refer to the more general model.

A width-w unanimity program can trivially be simulated by a width-$(w + 1)$ branching program. However, this simulation does not preserve regularity, and in fact it is not possible in general to simulate a regular unanimity program by a regular branching program of a similar width.[6]

Despite the fact that the unanimity model is strictly more powerful, we show (Lemma 20) that designing PRGs for regular unanimity programs is essentially equivalent to designing PRGs for regular branching programs (with an arbitrary set of accept states in the final layer). Therefore, known PRGs for regular branching programs [6, 13] automatically also fool this broader class of statistical tests. We use this lemma in two different ways to prove our two main results (Theorems 5 and 7).

1.3.2 The Large-Width Case

To prove Theorem 5, for every unbounded-width regular branching program B with a single accept state and every $\varepsilon > 0$, we construct an ε-"lower-approximator" for B. The lower approximator is a regular unanimity program B_L of width $O(1/\varepsilon)$ that accepts on a subset of the strings accepted by B, and has acceptance probability (under the uniform distribution) within ε of that of B.

▶ **Lemma 11.** *Let $\varepsilon > 0$. Every regular (respectively permutation) branching program of length n and unbounded width, with a single accept state, is ε-lower approximated by a regular (respectively permutation) unanimity program of length n and width $2 \cdot \lfloor 1/\varepsilon \rfloor$.*

A standard argument shows that an HSG for a lower approximator of B is also an HSG for B, so given Lemma 11, it follows that the BRRY PRG for bounded-width regular branching programs [6] is our desired HSG for unbounded-width regular branching programs.

We prove Lemma 11 by a more careful analysis of a result of Hoza, Pyne, and Vadhan [18], who prove that B is ε-lower approximated by an ordered branching program of width $O(n/\varepsilon)$. The key observation behind our improvement is that regular branching programs cannot concentrate low probability events. More precisely, say that a vertex v of B is "negligible" if the probability of visiting v is at most ε when B reads a uniform random input. We show that the probability of visiting some negligible vertex and then accepting is at most ε,

[6] For example, the AND function on n bits can be computed by a width-2 permutation unanimity program, but it cannot be computed by a width-w regular branching program unless $w \geq n + 1$.

with no dependence on n. Thus, if we reject all inputs that visit negligible vertices (i.e., we impose a unanimity condition), then we get an ε-lower approximator. Furthermore, the "effective width" of the approximator (namely, the maximum number of *accepting* vertices in any individual layer) is at most $1/\varepsilon$. A regular unanimity program of effective width w_{eff} can be simulated by one of actual width $2w_{\text{eff}}$ (see Lemma 23), completing the proof.

1.3.3 The Low-Threshold Case

To prove Theorem 7, we follow the approach of Hoza and Zuckerman [19]. They designed a method to convert any "moderate-error" PRG for ordered branching programs into an ε-HSG, where ε is potentially very small [19]. (See also related work on error reduction for "weighted PRGs" [5, 9, 11, 24, 17].) We develop a modified version of their framework that is suitable for the setting of regular programs.

Let B be an ordered branching program that accepts with probability p. Let $K > 1$, and let S be the set of vertices from which the acceptance probability is at least Kp. The starting point of the construction is our Lemma 25, which states that on a uniformly random input, B has a moderately large $\Omega(1/K)$ chance of visiting some vertex in S. If B is regular, then the predicate of failing to visit S can be computed by a regular unanimity program of the same width. Therefore, when B reads a pseudorandom string produced by a moderate-error PRG for regular branching programs, there is still an $\Omega(1/K)$ chance of visiting S. The HSG guesses where to truncate the pseudorandom string to land in S and then repeats the process, increasing the acceptance probability to K^2p, then K^3p, etc., until eventually an accepting vertex is reached. To keep the overall seed length low, we recycle the PRG's seed from one iteration to the next using a hitter (also known as a disperser).

The upshot is that for any $\varepsilon < \varepsilon_0 < 0.1$, we can convert an ε_0-PRG for width-$(2w)$ regular branching programs into an ε-HSG for width-w regular branching programs. If the ε_0-PRG has seed length s, then our ε-HSG has seed length

$$O\left(s + \frac{\log(1/\varepsilon) \cdot \log n}{\log(1/\varepsilon_0)} + \log(wn/\varepsilon)\right).$$

We plug in the PRG construction by Braverman, Rao, Raz, and Yehudayoff [6] with error $\varepsilon_0 = 2^{-\sqrt{\log(1/\varepsilon)}}$ to complete the proof of Theorem 7.

Hoza and Zuckerman's original version of the reduction [19] is similar. The key difference is that in each iteration of their setup, the probability of visiting their "target set" S of vertices is only $1/\operatorname{poly}(n)$. As a result, their version of the reduction requires the "moderate-error" PRG to have error $\varepsilon_0 < 1/\operatorname{poly}(n)$, which would be too small for our purposes. Our refined lemma allows us to greatly increase the probability of visiting S. Unlike in Hoza and Zuckerman's setting, however, the set S may now be spread across multiple layers of the branching program and thus has to be analyzed in the unanimity program model.

1.4 Other Results

Our approach for constructing HSGs for unbounded-width regular branching programs also works, mutatis mutandis, for some other unbounded-width models. For unbounded-width *permutation* branching programs with a single accept state, by plugging in the best PRGs for constant-width permutation branching programs [13, 31], we achieve seed length $\log n \cdot \operatorname{poly}(1/\varepsilon)$, which is optimal when the threshold ε is constant.

▶ **Proposition 12.** *Given $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-HSG for permutation branching programs of length n and unbounded width with a single accept state that has seed length*

$$O(\log n \cdot \log(1/\varepsilon) \cdot (1/\varepsilon^4)).$$

The approach also works in the more challenging "unordered" model. For unordered permutation branching programs, we get near-optimal seed length for constant threshold ε.

▶ **Proposition 13.** *Given $n \in \mathbb{N}$ and $\varepsilon > 0$, there exists an explicit ε-HSG for unbounded-width unordered permutation branching programs of length n with a single accept state that has seed length*

$$O(\log(n/\varepsilon) \cdot \log\log n \cdot (1/\varepsilon^4)).$$

We also get an improvement for unordered *regular* branching programs. In the unordered setting, it tends to be difficult to take advantage of regularity, because the regularity condition is not preserved under restrictions. This issue has forced some prior works to settle for fooling permutation programs [27, 8]. However, our reduction does not involve any restrictions, so we are not affected by the issue. For unbounded-width unordered regular branching programs, we get seed length $\widetilde{O}((\log^2 n)/\varepsilon)$, which admittedly is still far from optimal, but keep in mind that the state-of-the-art PRG for general polynomial-width unordered branching programs has seed length $O(\log^3 n)$ [15].

▶ **Proposition 14.** *Given $n \in \mathbb{N}$ and $\varepsilon > 0$, there exists an explicit ε-HSG for unbounded-width unordered regular branching programs of length n with a single accept state that has seed length*

$$O(\log(n/\varepsilon) \cdot \log n \cdot \log\log n \cdot (1/\varepsilon)).$$

Our results for unordered branching programs (Propositions 13 and 14) rely on PRGs developed by prior work of Chattopadhyay, Hatami, Hosseini and Lovett [8] and Forbes and Kelley [15] respectively.

We observe that the BRRY [6] HSG for constant-width regular branching programs can be viewed more generally as a *co-HSG* for unbounded-width regular branching programs with a constant number of *accept* states.

▶ **Proposition 15.** *Given $n, a \in \mathbb{N}$, the set $H = \{x \in \{0,1\}^n : \mathrm{wt}(x) \leq a\}$ where $\mathrm{wt}(x)$ denotes the Hamming weight of x is a co-hitting set for regular branching programs of length n and unbounded width with a accept states. That is, for every regular branching program B with at most a accept states that is not the constant function $B(x) = 1$, there is $x \in H$ such that $B(x) = 0$.*

In contrast, [18] show that a random function is not a co-HSG for regular branching programs of unbounded width and a single accept state unless the seed length is $\Omega(n)$.[7] Thus, we obtain a very simple explicit construction with exponentially shorter seed length than that obtained via the probabilistic method.

[7] Their result is stated as showing a random function is not a PRG, but the argument also rules out a co-HSG.

1.5 Preliminaries

First, we define notation relating to states and transitions in a branching program. Here, we adopt the perspective of a branching program as a directed graph, with edges from layer i to layer $i+1$ corresponding to the ith transition function.

▶ **Definition 16.** *For a branching program B of length n, let $V = V_0 \cup V_1 \cup \ldots \cup V_n$ be the vertex set of the branching program, where V_i holds the vertices corresponding to states in layer i. We will overload notation and consider the transition function as a map $B_t \colon V_{t-1} \times \{0,1\} \to V_t$ in addition to thinking of it as a map $B_t \colon [w] \times \{0,1\} \to [w]$. Similarly, we will often think of the start state v_0 as being an element of V_0 instead of an element of $[w]$, and similarly $V_{acc} \subseteq V_n$ instead of $V_{acc} \subseteq [w]$, etc. For $v \in V_i$ and $u \in V_j$ for $j > i$, we write $B[v,x] = u$ if the program transitions to state u starting from state v on input $x \in \{0,1\}^{j-i}$.*

Next, we define notation for the probability of reaching a state from the start state, and notation for the probability of accepting from that state.

▶ **Definition 17.** *Let B be a branching program, let $v_0 \in V_0$ be the start state, and let $V_{acc} \subseteq V_n$ be the set of accept states. For every state $v \in V_i$, let $p_{\to v} = \Pr[B[v_0, U_i] = v]$ be the probability v is reached from the start state over U_i, and let $p_{v\to} = \Pr[B[v, U_{n-i}] \in V_{acc}]$ be the probability the program accepts over U_{n-i} starting from v, where we define $p_{\to v_0} = 1$ and $p_{\to v} = 0$ for all $v \in V_0 \setminus \{v_0\}$ and likewise $p_{v\to} = 1$ for $v \in V_{acc}$ and $p_{v\to} = 0$ for $v \in V_n \setminus V_{acc}$.*

For a state v with transitions to u_1, u_2 (which are not necessarily distinct), the accept probability $p_{v\to}$ is exactly equal to $(p_{u_1\to} + p_{u_2\to})/2$. Next, we formally define the concept of a lower approximator.

▶ **Definition 18.** *Given a branching program B of length n and $\varepsilon > 0$, a length-n branching program B_L is an **ε-lower approximator of B** if $B_L^{-1}(1) \subseteq B^{-1}(1)$ and*

$$|\Pr[B(U_n) = 1] - \Pr[B_L(U_n) = 1]| \leq \varepsilon.$$

Finally, we define the unordered branching program model (although it is not the focus of this paper).

▶ **Definition 19.** *An **unordered (oblivious read-once) branching program** B consists of an ordered branching program B' and a permutation $\pi \colon [n] \to [n]$. It computes the function*

$$B(x) = B'(x_{\pi(1)}, \ldots, x_{\pi(n)}).$$

*We similarly define **unordered regular branching programs**, etc.*

1.6 Organization

In Section 2 we prove that fooling regular unanimity programs is essentially equivalent to fooling regular branching programs. In Section 3, we show that regular branching programs with a single accept state can be ε-lower-approximated by regular unanimity programs of width $O(1/\varepsilon)$. In Section 4 we combine these two results and conclude Theorem 5 (our HSG for unbounded-width regular branching programs); we also prove our other results for unbounded-width programs in this section. In Section 5 we construct our low-threshold HSG. In Section 6 we prove that pseudorandom objects for regular programs over a binary alphabet imply pseudorandom objects for regular programs over an arbitrary alphabet. In

Section 7 we state a formulation of Reingold, Trevisan, and Vadhan's reduction [28] that is convenient for us (the proof is in the full version of this paper [4]) and we combine it with the analysis in Section 6, thereby proving Theorem 8 (our reduction from the general case to the regular case).

2 PRGs for Unanimity Programs

In this section, as outlined in Section 1.3.1, we prove that PRGs for regular branching programs also fool the more general model of regular *unanimity* programs.

▶ **Lemma 20.** *Let $w, n \in \mathbb{N}$ and let $G\colon \{0,1\}^s \to \{0,1\}^n$. If G is an ε-PRG for width-$(2w)$ regular (respectively permutation) branching programs, then G is a (2ε)-PRG for width-w regular (respectively permutation) unanimity programs.*

Proof. Let B be a width-w length-n unanimity program. Let the layers of the program be $V_0, \ldots, V_n$ and let $v_0 \in V_0$ be the start state. For $t \in \{0, \ldots, n\}$, let $V_{\text{acc}}^{(t)} \subseteq V_t$ be the set of accepting vertices in layer t, and define a function $R_t\colon \{0,1\}^n \to \{0,1\}$ by

$$R_t(x) = 1 \iff B[v_0, x_{1..t}] \notin V_{\text{acc}}^{(t)}.$$

That is, $R_t(x)$ indicates whether $B(x)$ visits a reject state in layer t. Then

$$1 - B(x) = \bigvee_{t=0}^{n} R_t(x) = 2^{-n} \cdot \sum_{T \subseteq \{0,\ldots,n\}} \bigoplus_{t \in T} R_t(x), \tag{1}$$

because by the Fourier expansion of the OR function we have we have $\mathsf{OR}(y_0, \ldots, y_n) = 2 \cdot \mathbb{E}_{T \subseteq \{0,\ldots,n\}} \left[\bigoplus_{t \in T} y_t\right]$. For each $T \subseteq \{0, \ldots, n\}$, define $B^{(T)}(x) = \bigoplus_{t \in T} R_t(x)$. Let us design a width-$(2w)$ branching program to compute $B^{(T)}$. The vertex set in layer $t \in \{0, \ldots, n\}$ is given by $V_t^{(T)} = V_t \times \{0,1\}$. The start state is (v_0, a) where

$$a = (0 \in T \wedge v_0 \notin V_{\text{acc}}^{(0)}).$$

For $t > 0$, the transition function $B_t^{(T)}\colon V_{t-1}^{(T)} \times \{0,1\} \to V_t^{(T)}$ is given by $B_t^{(T)}((v_{t-1}, a_{t-1}), b) = (v_t, a_t)$, where

$$v_t = B_t(v_{t-1}, b)$$
$$a_t = a_{t-1} \oplus (t \in T \wedge v_t \notin V_{\text{acc}}^{(t)}).$$

In the final layer, the set of accept states is $V_{\text{acc}} = V_n \times \{1\}$. This program indeed computes $B^{(T)}(x)$, because in layer t, the program reaches the state (v_t, a_t), where $v_t = B[v_0, x_{1..t}]$ and $a_t = \bigoplus_{t \in T \cap \{0,\ldots,t\}} R_t(x)$.

We claim that if B is a regular program, then so is $B^{(T)}$, and furthermore if B is a permutation program, then so is $B^{(T)}$. To prove it, fix some $t > 0$ and some $(v, a) \in V_t^{(T)}$. If B is regular, then $|B_t^{-1}(v)| = 2$, say $B_t^{-1}(v) = \{(u_0, b_0), (u_1, b_1)\}$. Define

$$a' = a \oplus (t \in T \wedge v \notin V_{\text{acc}}^{(t)}).$$

From the definition of $B^{(T)}$, we have

$$(B_t^{(T)})^{-1}((v, a)) = \{((u_0, a'), b_0), ((u_1, a'), b_1)\}.$$

In particular, $|(B_t^{(T)})^{-1}((v,a))| = 2$, showing that $B^{(T)}$ is regular. If furthermore B is a permutation program, then $b_0 \neq b_1$, which immediately implies that $B^{(T)}$ is a permutation program.

Consequently, G fools each function $B^{(T)}$ with error ε. By Equation 1, it follows that G fools $1 - B$ (and therefore B) with error $2^{-n} \cdot \sum_{T \subseteq \{0,\ldots,n\}} \varepsilon = 2\varepsilon$. ◀

The idea of reducing disjunctions to parity functions (like what we do in Equation 1) is not new. Prior works have used a similar technique in other settings (e.g. [32, 21, 14]).

3 Lower Approximators for Regular Branching Programs

In this section, we show that unbounded-width regular branching programs can be lower approximated by bounded-width regular unanimity programs. Hoza, Pyne, and Vadhan showed [18, Theorem 4.1] that unbounded-width regular branching programs are ε-lower approximated by regular *branching* programs of width $O(n^2/\varepsilon)$, which is too large for our application.[8] We obtain a lower approximator of width $O(1/\varepsilon)$:

▶ **Lemma 11.** *Let $\varepsilon > 0$. Every regular (respectively permutation) branching program of length n and unbounded width, with a single accept state, is ε-lower approximated by a regular (respectively permutation) unanimity program of length n and width $2 \cdot \lfloor 1/\varepsilon \rfloor$.*

To get the width down to $O(1/\varepsilon)$, we make two changes to the proof by Hoza, Pyne, and Vadhan [18]. First, rather than retaining the n/ε most important states at each layer, we prove that retaining only the $1/\varepsilon$ most important states suffices. Second, we show that the unanimity program model allows us to rewire edges that previously pointed to deleted vertices (in such a way that the approximator rejects whenever such edges are crossed) while only increasing the width by a constant factor, whereas Hoza, Pyne, and Vadhan paid another factor of n at this stage to obtain a regular branching program [18].

We begin with the following claim, which shows that a regular branching program cannot concentrate low-probability events.

▷ Claim 21. Let $\varepsilon > 0$. Let B be a regular branching program, and let $V^\varepsilon = \{v : p_{\to v} \leq \varepsilon\}$ be the vertices of B that have at most ε probability of being reached over a uniformly random string. Then, for every $i \in \{0, 1, \ldots, n\}$ and $v \in V_i$, the probability of reaching v and visiting at least one vertex from V^ε along the way is at most ε. That is,

$$\Pr_{x \leftarrow U_i}\left[(B[v_0, x] = v) \bigwedge \bigvee_{j=1}^{i} (B[v_0, x_{1..j}] \in V^\varepsilon)\right] \leq \varepsilon.$$

Proof. The proof is by induction on i. The property is trivially true for states in the 0th layer. Assuming it holds for layer i, consider $v \in V_{i+1}$. If $v \in V^\varepsilon$ the property holds since

$$\Pr_{x \leftarrow U_{i+1}}\left[(B[v_0, x] = v) \bigwedge \bigvee_{j=1}^{i+1} (B[v_0, x_{1..j}] \in V^\varepsilon)\right] = \Pr_{x \leftarrow U_{i+1}}[B[v_0, x] = v] = p_{\to v} \leq \varepsilon.$$

[8] They also constructed lower approximators of width $O(n/\varepsilon)$. Those programs are not quite regular, but even setting aside issues of regularity, they are still too wide for our application.

Otherwise,

$$\begin{aligned}
&\Pr_{x\leftarrow U_{i+1}}\left[(B[v_0,x]=v)\bigwedge\bigvee_{j=1}^{i+1}(B[v_0,x_{1..j}]\in V^\varepsilon)\right]\\
&\quad=\Pr_{x\leftarrow U_{i+1}}\left[\left(\bigvee_{(u,b)\in B_{i+1}^{-1}(v)}(B[v_0,x_{1..i}]=u\wedge x_{i+1}=b)\right)\bigwedge\bigvee_{j=1}^{i}(B[v_0,x_{1..j}]\in V^\varepsilon)\right]\\
&\quad=\sum_{(u,b)\in B_{i+1}^{-1}(v)}\frac{1}{2}\cdot\Pr_{x\leftarrow U_i}\left[(B[v_0,x]=u)\bigwedge\bigvee_{j=1}^{i}(B[v_0,x_{1..j}]\in V^\varepsilon)\right]\\
&\quad\le\sum_{(u,b)\in B_{i+1}^{-1}(v)}\frac{\varepsilon}{2}\\
&\quad=\varepsilon,
\end{aligned}$$

where the last step uses regularity. ◁

Using Claim 21, we now show that a regular branching program with a single accept state is ε-lower approximated by a regular unanimity program with at most $O(1/\varepsilon)$ accept states in each layer. The maximum number of accept states in an individual layer of the program can be considered a measure of the "effective width" of the program.

▶ **Lemma 22.** *Let $\varepsilon > 0$. Every regular (respectively permutation) branching program of length n and width w, with a single accept state, is ε-lower approximated by a regular (respectively permutation) unanimity program of length n and width w, with at most $\lfloor 1/\varepsilon \rfloor$ accept states in each layer.*

Proof. Let B be a regular (permutation) branching program of length n and width w, with a single accept state v_{acc}, and let $V^\varepsilon = \{v : p_{\to v} \le \varepsilon\}$. We construct a unanimity program B_L that ε-lower approximates B. We take B_L to have the same set of states, transitions, and start state as B. For $i \in \{0,\dots,n-1\}$, the set of accept states in layer i of B_L is $V_i \setminus V^\varepsilon$, i.e., the set of states reached with probability greater than ε. In layer n, the set of accept states is $\{v_{\text{acc}}\} \setminus V^\varepsilon$, i.e., the unique accept state of B (or the empty set if $\Pr[B(U_n)=1] \le \varepsilon$). By construction, each layer of B_L has at most $\lfloor 1/\varepsilon \rfloor$ accept states. Moreover, B_L is a regular (permutation) unanimity program if B is a regular (permutation) branching program. Finally, note that $B_L(x) = 1$ if and only if the path in B corresponding to the string x reaches v_{acc} without visiting any state in V^ε. Therefore, by Claim 21, B_L is an ε-lower approximator of B. ◀

Finally, we show that regular unanimity programs of "effective width" w_{eff} can be simulated by regular unanimity programs of actual width $2 \cdot w_{\text{eff}}$ (Lemma 23). Lemma 11 follows from Lemmas 22 and 23.

▶ **Lemma 23.** *Let $w_{\text{eff}} \in \mathbb{N}$. Every regular (respectively permutation) unanimity program with at most w_{eff} accept states in each layer has an equivalent regular (respectively permutation) unanimity program of width $2 \cdot w_{\text{eff}}$.*

Proof. Let B be a regular (permutation) unanimity program of length n, with at most w_{eff} accept states in each layer. We may assume without loss of generality that B has exactly w_{eff} accept states in each layer[9] and that the set of accept states in each layer is $[w_{\text{eff}}]$. Assume also that the start state of B, v_0, is an accept state, otherwise the claim is trivial.

We claim that there exists a regular (permutation) program A of length n and width w_{eff} that includes every edge from an accept state of B to another accept state of B. That is, for $t > 0$, $B_t(u, b) = A_t(u, b)$ whenever $u \in [w_{\text{eff}}]$ and $B_t(u, b) \in [w_{\text{eff}}]$. Indeed, such a program A can be constructed greedily.

Now, for an input $x \in \{0,1\}^n$ and $t > 0$, let $v_t = B[v_0, x_{1..t}]$, i.e., v_t is the vertex that B reaches in layer t. Observe that $B(x) = 1$ if and only if $B_t(v_{t-1}, x_t) = A_t(v_{t-1}, x_t)$, for all $t > 0$. We construct a regular (permutation) unanimity program B' of length n and width $2 \cdot w_{\text{eff}}$ that accepts x if and only if $B_t(v_{t-1}, x_t) = A_t(v_{t-1}, x_t)$, for all $t > 0$.

Identify the state space $[2 \cdot w_{\text{eff}}]$ with $[w_{\text{eff}}] \times \{0,1\}$. The start state of B' is $(v_0, 0)$, and the set of accept states in each layer is $[w_{\text{eff}}] \times \{0\}$. For $t > 0$, the transition function B'_t is given by $B'_t((u_{t-1}, a_{t-1}), b) = (u_t, a_t)$, where

$$u_t = A_t(u_{t-1}, b)$$
$$a_t = a_{t-1} \oplus \mathbf{1}[A_t(u_{t-1}, b) \neq B_t(u_{t-1}, b)].$$

One can verify that B' is a regular (permutation) unanimity program by an argument similar to the proof of Lemma 20. ◀

4 Hitting Sets for Unbounded-Width Regular Branching Programs

Combining Lemmas 11 and 20, we get a general transfer theorem, which says that any PRG for width-$O(1/\varepsilon)$ regular programs is also an HSG for unbounded-width regular programs with a single accept vertex.

▶ **Theorem 24.** *Let $n \in \mathbb{N}$ and $\varepsilon > 0$, and let $G\colon \{0,1\}^s \to \{0,1\}^n$. If G is an ε-PRG for width-$(4 \cdot \lfloor 1/\varepsilon \rfloor)$ regular (respectively permutation) branching programs, then G is a (3ε)-HSG for unbounded-width regular (respectively permutation) branching programs with a single accept vertex.*

Proof. Fix an arbitrary regular (resp. permutation) branching program B of length n and unbounded width with a single accept state where $\Pr[B(U_n) = 1] > 3\varepsilon$. Applying Lemma 11, there is a regular (resp. permutation) unanimity program B_L of length n and width $2 \cdot \lfloor 1/\varepsilon \rfloor$ such that B_L is an ε-lower approximator of B, i.e. $\Pr[B_L(U_n) = 1] > 2\varepsilon$ and $B_L^{-1}(1) \subseteq B^{-1}(1)$. By Lemma 20, G fools B_L with error 2ε. In particular, this implies that G hits B_L and thus B. ◀

Then Theorem 5 follows from the BRRY PRG [6], Proposition 12 follows from the PRG of Steinke [31], and Proposition 14 and Proposition 13 follow from the PRGs of Forbes and Kelley [15] and Chattopadhyay, Hatami, Hosseini, and Lovett [8] respectively.[10]

[9] This is because we can add w_{eff} dummy states (unreachable from the start state) to each layer, along with transitions between dummy states to maintain the regularity/permutation condition. We can then assign some of the dummy states to be accept states to ensure that each layer has exactly w_{eff} accept states.

[10] Theorem 24 focuses on ordered programs, but the analogous theorem for the unordered programs follows. To see why, fix some ordered program B and some permutation $\pi\colon [n] \to [n]$. A generator G fools/hits $B(x_{\pi(1)}, \ldots, x_{\pi(n)})$ if and only if the generator $G'(x) \stackrel{\text{def}}{=} (G(x)_{\pi(1)}, \ldots, G(x)_{\pi(n)})$ fools/hits B, so we can apply the theorem to G' and draw conclusions about G.

Finally, we give a direct proof of Proposition 15, which we recall. The set is identical, and the proof of correctness is nearly identical, to the hitting set for width-w regular branching programs of Braverman, Rao, Raz, and Yehudayoff [6].

▶ **Proposition 15.** *Given $n, a \in \mathbb{N}$, the set $H = \{x \in \{0,1\}^n : \mathrm{wt}(x) \leq a\}$ where $\mathrm{wt}(x)$ denotes the Hamming weight of x is a co-hitting set for regular branching programs of length n and unbounded width with a accept states. That is, for every regular branching program B with at most a accept states that is not the constant function $B(x) = 1$, there is $x \in H$ such that $B(x) = 0$.*

Proof. Let B be an arbitrary regular branching program of length n and unbounded width with at most a accept states such that $B(x)$ is not the constant 1 function.

We say a state v is *doomed* if $p_{v\to} = 1$. We say that v is *important* if $B[v,0]$ is doomed and $B[v,1]$ is not, or vice versa. We claim that B has at most a layers with at least one important state. To prove this, first note that if there are k doomed states in V_i, there are at least k doomed states in V_{i+1}. This is because for doomed $v \in V_i$, by definition $B[v,1]$ and $B[v,0]$ are doomed, and states in V_{i+1} have in-degree at most 2. Furthermore, note that if there are k doomed states in V_i and a non-doomed $v \in V_i$ is important, the number of doomed states in V_{i+1} is at least $k+1$, because there are at least $2k+1$ transitions that must end at doomed states in V_{i+1}. We conclude by noting that there at at most a doomed states in V_n, so the claim follows.

Finally, we show that the hitting set has a string that reaches a reject state. Consider an algorithm starting at $u = v_0 \in V_0$. At each step, if u is an important state, take the transition that leads to a non-doomed state, and otherwise take the 0 transition. Since B is not the constant function $B(x) = 1$, this procedure reaches a reject state, and by the claim we take at most a 1 transitions, so there is $x \in H$ such that $B(x) = 0$. Since B was arbitrary, we conclude. ◀

5 Error Reduction for Regular Branching Programs

In this section, as outlined in Section 1.3.3, we use error reduction methods to construct an HSG for regular branching programs with seed length

$$\widetilde{O}\left(\log n \cdot \left(\sqrt{\log(1/\varepsilon)} + \log w\right) + \log(1/\varepsilon)\right).$$

The first step is to show that for any ordered branching program, there is a noticeable chance of visiting a vertex from which the acceptance probability has gone up significantly, refining a lemma of Hoza and Zuckerman [19]. We reiterate that in the following lemma, the set S is not guaranteed to be contained within a single layer.

▶ **Lemma 25.** *Let $K > 1$ be a real number, and let B be a (possibly non-regular) branching program with $\mathbb{E}[B] = p \leq 1/K$. Let V be the set of vertices in B, and let*

$$S = \{v \in V : p_{v\to} \geq Kp\}.$$

Then when B reads a uniform random input, the probability that it visits S is at least $\frac{1}{2K}$.

Proof. Let $S' = \{v \in V : Kp \leq p_{v\to} < 2Kp\}$. Because B has degree 2, the acceptance probability $p_{v\to}$ can at most double when we move from a vertex to one of its outneighbors. Therefore, every accepting path from the start vertex v_0 must visit S'.

For each vertex $v \in S'$, define $g_v \colon \{0,1\}^n \to \{0,1\}$ by letting $g_v(x) = 1$ if and only if $B(x)$ visits v and v is the *first* vertex in S' that B visits. Then

$$\begin{aligned} p = \mathbb{E}[B] = \Pr_{x \leftarrow U_n}\left[\bigvee_{v \in S'} (B(x) = g_v(x) = 1)\right] &= \sum_{v \in S'} \Pr_{x \leftarrow U_n}[B(x) = g_v(x) = 1] \\ &= \sum_{v \in S'} \mathbb{E}[g_v] \cdot p_{v\to} \\ &< 2Kp \cdot \sum_{v \in S'} \mathbb{E}[g_v] \\ &= 2Kp \cdot \Pr_{x \leftarrow U_n}[B(x) \text{ visits } S']. \end{aligned}$$

Therefore, when B reads a uniform random input, there is at least a $1/(2K)$ chance that it visits $S' \subseteq S$. ◀

Now we present our HSG. The construction and analysis closely follow those of Hoza and Zuckerman [19]; the main difference is that we need to invoke Lemma 20 (the equivalence between unanimity programs and branching programs) to argue that when B reads a pseudorandom input generated by the ε_0-PRG, there is still a noticeable chance of visiting the set S of Lemma 25.

Let $w, n \in \mathbb{N}$, let $\varepsilon_0 < 0.1$, let $G \colon \{0,1\}^s \to \{0,1\}^n$ be an ε_0-PRG for width-$(2w)$ regular branching programs, and let $K = \frac{1}{6\varepsilon_0} > 1$. Let $0 < \varepsilon < \varepsilon_0$; we will construct an ε-HSG for width-w length-n regular branching programs.

The construction uses a tool called a "hitter" [16]. A (θ, δ)-hitter is a function $\mathsf{Hit} \colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ such that for every set $E \subseteq \{0,1\}^s$, if $|E| \geq \theta \cdot 2^s$, then

$$\Pr_{x \leftarrow U_\ell}[\exists y, \mathsf{Hit}(x,y) \in E] \geq 1 - \delta.$$

(A hitter is a one-sided version of a "sampler," and one can show that it is equivalent to the concept of a "disperser.") Let Hit be a (θ, δ)-hitter with threshold $\theta = \varepsilon_0$ and failure probability $\delta = \frac{1}{2wn}$. Our HSG G' is given by

$$G'(x, t, y_1, \ldots, y_t, n_1, \ldots, n_t) = G(\mathsf{Hit}(x, y_1))_{1..n_1} \circ \cdots \circ G(\mathsf{Hit}(x, y_t))_{1..n_t},$$

where $x \in \{0,1\}^\ell$, t is a positive integer with $t \leq \left\lceil \frac{\log(1/\varepsilon)}{\log K} \right\rceil$, $y_1, \ldots, y_t \in \{0,1\}^q$, and $n_1, \ldots, n_t$ are positive integers with $n_1 + \cdots + n_t = n$. Here $\circ$ denotes string concatenation.

▷ Claim 26. G' is an ε-HSG for width-w length-n regular branching programs.

Proof. Let B be a regular branching program with $\mathbb{E}[B] > \varepsilon$, and let V be the set of vertices in B. For each vertex $u \in V$, we define a "target set" $S_u \subseteq V$ by the rule

$$S_u = \begin{cases} \{v \in V : p_{v\to} \geq K p_{u\to}\} & \text{if } p_{u\to} \leq 1/K \\ V_{\text{acc}} & \text{if } p_{u\to} > 1/K. \end{cases}$$

Say u is in layer i of the program. We define a function $g_u \colon \{0,1\}^n \to \{0,1\}$ where $g_u(x)$ indicates whether B ever visits the target set S_u when we start at u and read x, i.e.,

$$g_u(x) = \bigvee_{j=0}^{n-i} (B[u, x_{1..j}] \in S_u).$$

By Lemma 25, $\mathbb{E}[g_u] \geq 1/(2K) = 3\varepsilon_0$. Furthermore, $1 - g_u$ can be computed by a width-w regular unanimity program. Therefore, by Lemma 20, G fools g_u with error $2\varepsilon_0$, so $\mathbb{E}[g_u(G(U_s))] \geq \varepsilon_0$. Let $E_u = \{z \in \{0,1\}^s : g_u(G(z)) = 1\}$. Then by the hitter condition,

$$\Pr_{x \leftarrow U_\ell}[\exists y, \mathsf{Hit}(x,y) \in E_u] \geq 1 - \frac{1}{2wn}.$$

By the union bound, therefore, there exists some $x_* \in \{0,1\}^\ell$ such that for every vertex $u \in V$, there exists a $y \in \{0,1\}^q$ such that $\mathsf{Hit}(x_*, y) \in E_u$.

Now we inductively define a sequence of vertices $u_0, u_1, \ldots$, a sequence of strings $y_1, y_2, \ldots$, and a sequence of positive integers $n_1, n_2, \ldots$ as follows. We begin with $u_0 = v_0$ (the start vertex of B). Assume that we have defined $u_0, u_1, \ldots, u_{i-1}$. Let y_i be such that $\mathsf{Hit}(x_*, y) \in E_{u_{i-1}}$. Recalling the definition of $E_{u_{i-1}}$, this means that if we start at u_{i-1} and read the string $G(\mathsf{Hit}(x_*, y))$, we visit the target set $S_{u_{i-1}}$. Let u_i be the first vertex in the target set $S_{u_{i-1}}$ that we visit, and let n_i be the number of steps from u_{i-1} to u_i. We terminate the process when we reach some vertex $u_i \in V_{\mathrm{acc}}$ in the final layer.

Let t be the number of iterations. In every iteration except possibly the last, the acceptance probability goes up by at least a factor of K, by the definition of the target set S_u. Therefore, $\varepsilon \cdot K^{t-1} < 1$, so $t < 1 + \frac{\log(1/\varepsilon)}{\log K}$. By construction,

$$B(G'(x_*, t, y_1, \ldots, y_t, n_1, \ldots, n_t)) = 1. \qquad \blacktriangleleft$$

Proof of Theorem 7. The sampling algorithm by Bellare, Goldreich, and Goldwasser [3] implies that for every $s \in \mathbb{N}$ and every $\theta, \delta > 0$, there is an explicit (θ, δ)-hitter $\mathsf{Hit}\colon \{0,1\}^\ell \times \{0,1\}^q \to \{0,1\}^s$ with $\ell = O(s + \log(1/\delta))$ and $q = O(\log(1/\theta) + \log\log(1/\delta))$. In our case, we get $\ell = O(s + \log(wn))$ and $q = O(\log(1/\varepsilon_0) + \log\log(wn))$. Therefore, the seed length of G' is bounded by

$$\begin{aligned}
&\ell + O(\log\log(1/\varepsilon)) + \left(1 + \frac{\log(1/\varepsilon)}{\log K}\right) \cdot (q + \log n) \\
&\quad \leq O\left(s + \frac{\log(1/\varepsilon) \cdot (\log n + \log\log w)}{\log(1/\varepsilon_0)} + \log(wn/\varepsilon)\right) \\
&\quad \leq O\left(s + \frac{\log(1/\varepsilon) \cdot \log n}{\log(1/\varepsilon_0)} + \log(wn/\varepsilon)\right),
\end{aligned}$$

where the last step holds without loss of generality because if $\log\log w > \log n$ then the claimed seed length is greater than n, which is trivial. Finally, we choose $\epsilon_0 = 2^{-\sqrt{\log(1/\varepsilon)}}$ and take G to be the BRRY PRG [6], which has seed length $s = O(\log n \cdot (\log(w/\varepsilon_0) + \log\log n))$. $\blacktriangleleft$

6 Reductions From Large Alphabets

In this section, we prove that PRGs and HSGs for regular branching programs over a binary alphabet imply PRGs and HSGs for regular branching programs over larger alphabets, with mild degradation in parameters. Together with the modified proof of Reingold, Trevisan, and Vadhan in Appendix 7, this suffices to establish Theorem 8 (the reduction from general ordered branching programs to the regular case).

To do so, we first define branching programs of higher degree.

▶ **Definition 27.** *An **(ordered) branching program** B of length n, width w, and **degree / alphabet size** D computes a function $B : [D]^n \to \{0,1\}$. On an input $x \in [D]^n$, the branching program computes as follows. It starts at a fixed start state $v_0 \in [w]$. Then for*

$t = 1, \ldots, n$, it reads the next input symbol x_t and updates its state according to a transition function $B_t : [w] \times [D] \to [w]$ by taking $v_t = B_t(v_{t-1}, x_t)$. As in the $D = 2$ case, there is a set V_{acc} of accept states. Let v_n be the final state reached by the branching program on input x. If $v_n \in V_{acc}$ the branching program accepts, denoted $B(x) = 1$, and otherwise the program rejects, denoted $B(x) = 0$. The program B is ***regular*** *if for every $t \in 1, \ldots, n$ and $v \in [w]$ there are exactly D pairs $(u, \sigma) \in [w] \times [D]$ such that $B_t(u, \sigma) = v$.*

We define notation for states and transitions in branching programs analogously to the $D = 2$ case. In particular, for a degree D branching program B we write $B[v, x] = u$ if B reaches state $u \in V_j$ from state $v \in V_i$ over input $x \in [D]^{j-i}$.

Finally, we formally define HSGs and PRGs over larger alphabets. We use U_S to denote the uniform distribution over the set S.

▶ **Definition 28.** *Let $\mathcal{F}$ be a class of functions $f : [D]^n \to \{0, 1\}$. An* ***ε-hitting set generator*** *(****ε-HSG****) for $\mathcal{F}$ is a function $H : \{0, 1\}^s \to [D]^n$ such that for every $f \in \mathcal{F}$ where $\Pr_{x \leftarrow U_{[D]^n}}[f(x) = 1] > \varepsilon$, there exists $x \in \{0, 1\}^s$ such that $f(H(x)) = 1$.*

▶ **Definition 29.** *Let $\mathcal{F}$ be a class of functions $f : [D]^n \to \{0, 1\}$. An* ***ε-pseudorandom generator*** *(****ε-PRG****) for $\mathcal{F}$ is a function $G : \{0, 1\}^s \to [D]^n$ such that for every $f \in \mathcal{F}$,*

$$\left| \Pr_{x \leftarrow U_{[D]^n}}[f(x) = 1] - \Pr_{x \leftarrow U_s}[f(G(x)) = 1] \right| \leq \varepsilon.$$

We can now state our main theorem for transferring pseudorandom objects over a binary alphabet into pseudorandom objects over larger alphabets:

▶ **Theorem 30.** *Given $n, w, D \in \mathbb{N}$ and $\varepsilon > 0$, there exist values $w' = O(wn^2 D/\varepsilon)$ and $n' = O(n \log(nD/\varepsilon))$ and an explicit function $p \colon \{0, 1\}^{n'} \to [D]^n$ such that if $G : \{0, 1\}^s \to \{0, 1\}^{n'}$ is an ε-PRG (resp. HSG) for regular branching programs of length n', width w', and degree 2, then $p \circ G$ is a (3ε)-PRG (resp. HSG) for regular branching programs of length n, width w and degree D.*

6.1 Overview of Proof of Theorem 30

Theorem 30 follows from a pair of reductions (Lemmas 31 and 32). Here we focus on the case where the initial object is a PRG for simplicity.

First, we show how to convert a PRG over the binary alphabet into a PRG over the alphabet $[R]$, where R is an arbitrary power of two, say $R = 2^r$ where $r \in \mathbb{N}$. To establish this, we take an arbitrary regular branching program B of length n and width w over the alphabet $[R]$. We define a new program $B' \colon \{0, 1\}^{nr} \to \{0, 1\}$ that simulates B as follows. The state space is $[w] \times \{0, 1\}^r$. Each input in $\{0, 1\}^{nr}$ is divided into n blocks of r bits. When B' is in state (u, x) and it reads bit i of block t, it replaces the ith bit of x with the input bit. When it finishes reading a block, it furthermore updates u according to the transition function of the original function $u \leftarrow B_t(u, x)$, and updates x in such a way that regularity is maintained. In effect, the new program stores every block of r bits into an auxiliary component of the state and then uses this register (viewed as a number in R) to make the appropriate transition in the original program. This increases the width by a factor of R but exactly preserves the computed function, so G ε-fooling B' implies $p \circ G$ ε-fools B, where p simply maps each block of r bits to a number in $[R]$.

Second, we show how to convert a PRG for regular branching programs over the alphabet $[R]$ into a PRG over the alphabet $[D]$, where D is arbitrary (not necessarily a power of two) and R is a sufficiently large power of two. We let $p(x) = x \mod D$ where we apply the

mod function entrywise. To show $p \circ G$ fools regular branching programs over the alphabet $[D]$, we let m be the largest multiple of D less than R. Given a regular branching program $B : [R]^n \to \{0,1\}$, we can compute $B'(x) = (B \circ p)(x) \wedge \{x \leq m\}$ by a regular branching program that G is required to fool. Furthermore, the condition $x \leq m$ is satisfied with probability at least $1 - \varepsilon$ over uniformly random input, and there is a regular branching program that tests if its input satisfies $x \leq m$ (that G is required to fool).

6.2 Proof of Theorem 30

We now precisely state the pair of reductions outlined in the preceding section. The first reduction transforms a binary PRG into a PRG for alphabets of size arbitrary powers of two.

▶ **Lemma 31** (Generator for degree two $\implies$ generator for degree any power of two)**.** *Given $n, w, R \in \mathbb{N}$ and $\varepsilon > 0$ where $R = 2^r$, there is an explicit map $p : \{0,1\}^{nr} \to [R]^n$ such that if $G : \{0,1\}^s \to \{0,1\}^{nr}$ is an ε-PRG (resp. HSG) for regular branching programs of length nr, degree 2 and width wR, then $p \circ G$ is an ε-PRG (resp. HSG) for regular branching programs of length n, degree R and width w.*

Proof. Let $p_e : \{0,1\}^r \to [R]$ be an explicit bijection and define

$$p(x) = (p_e(x_1, \ldots, x_r), \ldots, p_e(x_{(n-1)r}, \ldots, x_n)).$$

Note that p maps uniformly random input to uniformly random output.

Now fix an arbitrary regular branching program B of length n, width w, and degree R. For every transition function $B_t : [w] \times [R] \to [w]$, define $\mathrm{Rot}_t : [w] \times \{0,1\}^r \to [w] \times \{0,1\}^r$ such that Rot_t is injective, and $\mathrm{Rot}_t(u, x) = (v, y) \implies B_t(u, p_e(x)) = v$. Such a function exists since $|B_t^{-1}(v)| = R$ for every v. (We use the notation "Rot" because the function is closely connected to the concept of the "rotation map" of a regular digraph [29, 30].) Then define a new branching program $B' : \{0,1\}^{nr} \to \{0,1\}$ where the states in each layer are $[w] \times \{0,1\}^r$. For $t \in [n]$ and $i \in [r]$ we define the transition function as

$$B'_{r(t-1)+i}((u, x), b) = \begin{cases} (u, x^{i \leftarrow b}) & i < r \\ \mathrm{Rot}_t(u, x^{i \leftarrow b}) & i = r, \end{cases}$$

where $x^{i \leftarrow b}$ denotes replacing the ith bit of x with b. Then B' is regular, because the operation of replacing the ith bit is regular. By choosing the start state to be $(v_0, 0^r)$ and marking as accept all states (u, x) where u is an accept state in B, we obtain that $B \circ p = B'$.

To conclude, we break into cases depending on the base pseudorandom object:

1. **(G is an ε-HSG):** Assuming that $\Pr[B(U_{[R]^n}) = 1] > \varepsilon$ then

   $$\Pr[B'(U_{nr}) = 1] = \Pr[B(U_{[R]^n}) = 1] > \varepsilon$$

   and so there is some x such that $B'(G(x)) = 1$ and thus $B((p \circ G)(x)) = 1$.
2. **(G is an ε-PRG):** We have

   $$\left| \Pr_{x \leftarrow U_s}[B((p \circ G)(x)) = 1] - \Pr_{x \leftarrow U_{[R]^n}}[B(x) = 1] \right|$$
   $$= \left| \Pr_{x \leftarrow U_s}[B'(G(x)) = 1] - \Pr_{x \leftarrow U_{nr}}[B'(x) = 1] \right| \leq \varepsilon.$$

In both cases since B was arbitrary we obtain the desired result. ◀

We remark that the preceding component of the reduction does not preserve the property of B being a *permutation* branching program, since the "overwriting the ith bit" operation does not produce a permutation branching program. We next show that PRGs and HSGs over large alphabets imply PRGs and HSGs over smaller alphabets, including alphabet sizes that are not powers of two.

▶ **Lemma 32** (Generator for large degree $\implies$ generator for small degree)**.** *Given $n, w, d \in \mathbb{N}$ and $\varepsilon > 0$, there is $R_0 = O(nD/\varepsilon)$ such that for every $R \geq R_0$, there is an explicit function $p : [R]^n \to [D]^n$ such that if $G : \{0,1\}^s \to [R]^n$ is an ε-PRG (resp. HSG) for regular branching programs of length n, degree R and width $w \cdot (n+1)$, then $p \circ G$ is a (3ε)-PRG (resp. HSG) for regular branching programs of length n, degree D and width w.*

Proof. Let R be large enough that $\frac{D}{R}n < \varepsilon$. Then let $p_e : [R] \to [D]$ be defined as $p_e(x_i) = x_i \mod D$ and define

$$p(x) = (p_e(x_1), \ldots, p_e(x_n)).$$

Let $m \leq R$ be the largest multiple of D not greater than R. For $x \in [R]^n$, we write $x \leq m$ if for all i, $x_i \leq m$. Let $\rho = \Pr_{x \leftarrow U_{[R]^n}}[x \not\leq m]$, and observe $\rho \leq n \cdot D/R < \varepsilon$, and furthermore there exists a length n, width $n \leq w$, degree R regular branching program Q where $Q(x) = \mathbf{1}[x \not\leq m]$.

Now fix an arbitrary regular branching program $B : [D]^n \to \{0,1\}$ of width w with states $V_0, \ldots, V_n$. Let $B' : [R]^n \to \{0,1\}$ be a branching program of length n, degree R and width $w(n+1)$. We identify the states of B' with $V_i \cup ([n] \times V_i)$. The states in V_i simulate B, while the other states are dummy rejection states. We now define the transition function B'_i. For $v \in V_i$ define

$$B'_i(v, \sigma) = \begin{cases} B_i(v, \sigma \mod D) & \sigma \leq m \\ (v, i) & \text{otherwise.} \end{cases}$$

Next, for $(v, j) \in (V_i \times [n])$ define

$$B'_i((v, j), \sigma) = \begin{cases} (v, j) & \sigma \leq m \text{ or } j \neq i \\ v & \text{otherwise.} \end{cases}$$

The accept states of B' are the accept states of B. It can be seen that B' is regular. Note that $B[v, \sigma \mod D] = B[v, p_e(\sigma)]$, so B' computes the function

$$B'(x) = B(p(x)) \cdot \mathbf{1}[x \leq m]. \tag{2}$$

Furthermore, p maps the conditional distribution $(x \leftarrow U_{[R]^n} | x \leq m)$ to the uniform distribution $U_{[D]^n}$. Therefore,

$$\Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1 \mid x \leq m] = \Pr_{x \leftarrow U_{[D]^n}}[B(x) = 1],$$

so

$$\Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1] = \Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1 | x \not\leq m] \cdot \rho + \Pr_{x \leftarrow U_{[D]^n}}[B(x) = 1] \cdot (1 - \rho),$$

and hence

$$\left| \Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1] - \Pr_{x \leftarrow U_{[D]^n}}[B(x) = 1] \right|$$
$$= \rho \cdot \left| \Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1 | x \not\leq m] - \Pr_{x \leftarrow U_{[D]^n}}[B(x) = 1] \right| \leq \rho \leq \varepsilon. \tag{3}$$

To conclude, we break into cases depending on the base pseudorandom object:

1. **(G is an ε-HSG):** If $\Pr[B(U_{[D]^n}) = 1] > 2\varepsilon$, then $\Pr[B'(U_{[R]^n}) = 1] > 2\varepsilon - \varepsilon$ by Equation 3. Thus by assumption on G there is some x where $B'(G(x)) = 1$. Since $B'(x) = 0$ on all $x \not\leq m$, we have $1 = B'(G(x)) = B((p \circ G)(x))$, i.e. $p \circ G$ hits B.
2. **(G is an ε-PRG):** We have that G ε-fools Q by assumption, so $\Pr_{x \leftarrow U_s}[G(x) \not\leq m] \leq \varepsilon + \varepsilon$. Then by Equation 2, we obtain:

$$\left| \Pr_{x \leftarrow U_s}[B'(G(x)) = 1] - \Pr_{x \leftarrow U_s}[B((p \circ G)(x)) = 1] \right| \leq \Pr_{x \leftarrow U_s}[G(x) \not\leq m] \leq 2\varepsilon. \tag{4}$$

We finish by repeated application of the triangle inequality:

$$\begin{aligned}
&\left| \Pr_{x \leftarrow U_s}[B(p \circ G(x)) = 1] - \Pr_{x \leftarrow U_{[D]^n}}[B(x) = 1] \right| \\
&\quad \leq \left| \Pr_{x \leftarrow U_s}[B(p \circ G(x)) = 1] - \Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1] \right| + \varepsilon && \text{(Equation 3)} \\
&\quad \leq \left| \Pr_{x \leftarrow U_s}[B'(G(x)) = 1] - \Pr_{x \leftarrow U_{[R]^n}}[B'(x) = 1] \right| + 2\varepsilon && \text{(Equation 4)} \\
&\quad \leq 3\varepsilon && \text{(Assumption).}
\end{aligned}$$

◀

7 Transfer to General Branching Programs

The original formulation of the result of Reingold, Trevisan and Vadhan stated that a "pseudoconverging walk generator" (an object implied by a PRG) with sufficiently short seed implies $\mathbf{BPL} = \mathbf{L}$. We extend their results to HSGs, and derive the degradation in parameters in the notation of branching programs.

▶ **Theorem 33** (Variant of [28]). *Given $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there are values $D' = O(n^3 w/\varepsilon^3)$ and $w' = O(n^6 \cdot w^2/\varepsilon^5)$ and an explicit map $p : [D']^n \to \{0,1\}^n$ such that if $G : \{0,1\}^s \to [D']^n$ is an ε-PRG (resp. HSG) for regular branching programs of length n, width w' and degree D', then $p \circ G$ is a (16ε)-PRG (resp. HSG) for (possibly non-regular) branching programs of length n and width w.*

The proof of Theorem 33 can be found in the full version of this paper [4]. Theorem 8 follows from Theorem 30 and Theorem 33.

References

1 AmirMahdi Ahmadinejad, Jonathan A. Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil P. Vadhan. High-precision estimation of random walks in small space. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306, 2020.

2 Miklós Ajtai, János Komlós, and E. Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 132–140, 1987.

3 Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

4 Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. Technical Report TR21-143, Electronic Colloquium on Computational Complexity (ECCC), 2021. URL: `https://eccc.weizmann.ac.il/report/2021/143/`.

5 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM J. Comput.*, 49(5), 2020. doi:10.1137/18M1197734.

6 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.

7 Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 30–39, 2010. doi:10.1109/FOCS.2010.10.

8 Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. *Theory Comput.*, 15:1–26, 2019. doi:10.4086/toc.2019.v015a010.

9 Eshan Chattopadhyay and Jyun-Jie Liao. Optimal error pseudodistributions for read-once branching programs. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 25:1–25:27, 2020.

10 Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. In *Proceedings of the 35th Computational Complexity Conference (CCC)*, pages 10:1–10:25, 2020. doi:10.4230/LIPIcs.CCC.2020.10.

11 Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error Reduction for Weighted PRGs Against Read Once Branching Programs. In *Proceedings of the 36th Computational Complexity Conference (CCC)*, pages 22:1–22:17, 2021.

12 Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. Solving directed Laplacian systems in nearly-linear time through sparse LU factorizations. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 898–909, 2018.

13 Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC)*, pages 221–231, 2011. doi:10.1109/CCC.2011.23.

14 Dean Doron, Pooya Hatami, and William M. Hoza. Log-Seed Pseudorandom Generators via Iterated Restrictions. In *Proceedings of the 35th Annual Computational Complexity Conference (CCC)*, pages 6:1–6:36, 2020.

15 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)*, pages 946–955, 2018. doi:10.1109/FOCS.2018.00093.

16 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*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011. doi:10.1007/978-3-642-22670-0_24.

17 William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Proceedings of the 25th International Conference on Randomization and Computation (RANDOM)*, pages 28:1–28:23, 2021.

18 William M. Hoza, Edward Pyne, and Salil P. Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In James R. Lee, editor, *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 7:1–7:20, 2021.

19 William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success RL. *SIAM J. Comput.*, 49(4):811–820, 2020. doi:10.1137/19M1268707.

20 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 263–272, 2011. doi:10.1145/1993636.1993672.

21 Chin Ho Lee. Fourier Bounds and Pseudorandom Generators for Product Tests. In *Proceedings of the 34th Annual Computational Complexity Conference (CCC)*, pages 7:1–7:25, 2019.

22 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM J. Comput.*, 42(3):1275–1301, 2013.

23 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

24 Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract). In *Proceedings of the 36th Annual Computational Complexity Conference (CCC)*, pages 33:1–33:15, 2021.

25 Edward Pyne and Salil Vadhan. Deterministic approximation of random walks via queries in graphs of unbounded size. In *Proceedings of the 5th Symposium on Simplicity in Algorithms (SOSA)*, pages 57–67, 2022.

26 Edward Pyne and Salil P. Vadhan. Limitations of the Impagliazzo-Nisan-Wigderson pseudorandom generator against permutation branching programs. In *Proceedings of the 27th International Computing and Combinatorics Conference (COCOON)*, pages 3–12, 2021. doi:10.1007/978-3-030-89543-3_1.

27 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM)*, pages 655–670, 2013.

28 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks in regular digraphs and the RL vs. L problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 457–466, 2006.

29 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1), January 2002.

30 Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM)*, pages 436–447, 2005.

31 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. Technical Report TR12-083, Electronic Colloquium on Computational Complexity (ECCC), July 2012. URL: http://eccc.hpi-web.de/report/2012/083/.

32 R. Ryan Williams. Counting Solutions to Polynomial Systems via Reductions. In *Proceedings of the 1st Symposium on Simplicity in Algorithms (SOSA)*, pages 6:1–6:15, Dagstuhl, Germany, 2018.

Linear Branching Programs and Directional Affine Extractors

Svyatoslav Gryaznov
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Pavel Pudlák
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Navid Talebanfard
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

A natural model of *read-once linear branching programs* is a branching program where queries are $\mathbb{F}_2$ linear forms, and along each path, the queries are linearly independent. We consider two restrictions of this model, which we call *weakly* and *strongly* read-once, both generalizing standard read-once branching programs and parity decision trees. Our main results are as follows.

- **Average-case complexity.** We define a pseudo-random class of functions which we call *directional affine extractors*, and show that these functions are hard on average for the strongly read-once model. We then present an explicit construction of such function with good parameters. This strengthens the result of Cohen and Shinkar (ITCS'16) who gave such average-case hardness for parity decision trees. Directional affine extractors are stronger than the more familiar class of affine extractors. Given the significance of these functions, we expect that our new class of functions might be of independent interest.
- **Proof complexity.** We also consider the proof system Res[$\oplus$], which is an extension of resolution with linear queries, and define the regular variant of Res[$\oplus$]. A refutation of a CNF in this proof system naturally defines a linear branching program solving the corresponding search problem. If a refutation is regular, we prove that the resulting program is read-once. Conversely, we show that a weakly read-once linear BP solving the search problem can be converted to a regular Res[$\oplus$] refutation with constant blow up, where the regularity condition comes from the definition of weakly read-once BPs, thus obtaining the equivalence between these proof systems.

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

Keywords and phrases Boolean Functions, Average-Case Lower Bounds, AC0[2], Affine Dispersers, Affine Extractors

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.4

Related Version *Full Version*: https://arxiv.org/abs/2201.10997

Funding *Svyatoslav Gryaznov*: Supported by GAČR grant 19-05497S.
Pavel Pudlák: Supported by GAČR grant 19-27871X.
Navid Talebanfard: Supported by GAČR grant 19-27871X.

Acknowledgements We wish to thank Swastik Kopparty for useful correspondence regarding affine extractors.

1 Introduction

Circuit complexity and proof complexity are two major lines of inquiry in complexity theory (see [13, 15] for extensive introductions). The former theme attempts to identify explicit Boolean functions which are not computable by small circuits from a certain restricted class, and the latter aims to find tautologies which are not provable by short proofs in

a given restricted proof system. These seemingly unrelated topics are bound together in at least two different ways: via feasible interpolation where a circuit lower bound for a concrete computational problem implies proof size lower bounds (see, e.g., [11]), and more fundamentally many proof systems have an underlying circuit class where proof lines come from. Notable examples are Frege, bounded depth Frege, and extended Frege systems where proof lines are De Morgan formulas, AC^0 circuits, and general Boolean circuits, respectively. Intuitively we expect that understanding a circuit class in terms of lower bounds and techniques should yield results in the proof complexity counterpart. This intuition has been supported by bounded depth Frege lower bounds using specialized Switching Lemmas (see, e.g., [10]), the essential ingredient of AC^0 lower bounds.

$\mathsf{AC}^0[2]$ circuits and $\mathsf{Res}[\oplus]$ proof system. It is not clear if this intuition should always hold. Lower bounds for $\mathsf{AC}^0[p]$ circuits (AC^0 circuits with Mod_p gates) have been known for a long time [19, 23] yet lower bounds for bounded depth Frege systems with modular gates still elude us. Perhaps this failure is not too surprising since our understanding of $\mathsf{AC}^0[p]$ circuits is not of the same status as our understanding of AC^0. For example, even for $\mathsf{AC}^0[2]$, that is AC^0 with parity gates, no strong average-case lower bound is known. Settling such bounds is an important challenge, since Shaltiel and Viola [22] showed that for standard worst-case to average-case hardness amplification techniques to work, the circuit class is required to compute the majority function, which is not the case for $\mathsf{AC}^0[2]$. Several works have highlighted the special case of $\mathsf{AC}^0 \circ \mathsf{Mod}_2$, where the parity gates are next to the input [21, 2, 8]. Among these works we pay special attention to the result of Cohen and Shinkar [8] who considered the depth-3 case of this problem and proved a strong average-case hardness for the special case of parity decision trees. The more general case of $\mathsf{DNF} \circ \mathsf{Mod}_2$ remains open.

In the proof complexity parallel, a special case of $\mathsf{AC}^0[2]$-Frege was suggested by Itsykson and Sokolov [12]. They considered the system $\mathsf{Res}[\oplus]$ that is an extension of resolution which reasons about disjunctions of linear equations over $\mathbb{F}_2$, which we call linear clauses. The rules of this system are:

- *the weakening rule*: from a linear clause we can derive any other linear clause which is semantically implied,
- *the resolution rule*: for every two linear clauses C and D and linear form f, we can derive $C \vee D$ from $(f = 0) \vee C$ and $(f = 1) \vee D$.

They proved exponential lower bounds for the tree-like restriction of this system. These lower bounds were later extended in [9, 18]. For DAG-like proofs, the only known results are due to Khaniki [14] who proved almost quadratic lower bounds, and to Lauria [16] for a restriction of the system when parities are on a bounded number of variables. Super-polynomial lower bounds for unrestricted DAG-like $\mathsf{Res}[\oplus]$ are widely open.

Parity decision trees and tree-like $\mathsf{Res}[\oplus]$. Given an unsatisfiable CNF $F = C_1 \wedge \ldots \wedge C_m$, the *search problem for F* is the computational problem of finding a clause C_i falsified by a given assignment to the variables. A tree-like $\mathsf{Res}[\oplus]$ refutation of F can be viewed as a parity decision tree solving the search problem for an unsatisfiable CNF [9]. Recall that the strongest average-case lower bounds for $\mathsf{AC}^0[2]$ are in fact for parity decision trees. Thus it seems that parity decision trees are at the frontier of our understanding in these two areas. Therefore a natural approach to make progress towards both general $\mathsf{Res}[\oplus]$ lower bounds and average-case hardness for $\mathsf{AC}^0[2]$ is to consider DAG-like structures more general than decision trees.

1.1 Our contributions

Motivated by strengthening tree-like $\mathsf{Res}[\oplus]$ lower bounds as well as average-case lower bounds for parity decision trees to more general models, we consider a model of read-once branching programs (BPs) with linear queries. The most natural way to interpret the property of being read-once in BPs with linear queries, is to impose that along every path, the queries are linearly independent. We consider two restrictions of this model which we call weakly read-once and strongly read-once, both of which extend parity decision trees and standard read-once branching programs.

For strongly read-once BPs, we prove average-case hardness for a new class of pseudo-random functions, and we give an explicit construction of such a function, thus strengthening the result of Cohen and Shinkar [8] and making progress towards average-case hardness for $\mathsf{DNF} \circ \mathsf{Mod}_2$. Our pseudo-random functions are defined below and might be of independent interest.

Directional affine extractors. The average-case hardness result of Cohen and Shinkar [8] is for affine extractors. An affine extractor for dimension d and bias ϵ is a function such that restricted to any affine subspace of dimension at least d it has bias at most ϵ. Explicit constructions for such functions are known (e.g., [5, 24, 4]). For our purposes it is not clear if affine extractors are sufficient. Therefore we consider a more robust concept. We say that a function $f\colon \{0,1\}^n \to \{0,1\}^n$ is a *directional affine extractor* for dimension d with bias ϵ, if for every non-zero $a \in \{0,1\}^n$, the derivative of f in the direction a, $D_a f(x) = f(x+a)+f(x)$, is an affine extractor for dimension d with bias ϵ. We give an explicit construction of a good directional affine extractor for dimension larger than $2n/3$.

For weakly read-once BPs we show a correspondence with $\mathsf{Res}[\oplus]$. More precisely, we show that a weakly read-once BP solving the search problem for a CNF F, can be converted to a $\mathsf{Res}[\oplus]$ refutation of F while preserving the proof structure. This also justifies defining a $\mathsf{Res}[\oplus]$ counterpart to regular resolution similarly to weakly read-once BPs. Recall that in a regular resolution proof, no variable is resolved more than once along any path. It is well-known that a read-once BP solving the search problem for an unsatisfiable CNF can be converted to a regular resolution refutation of the formula. Our result should be interpreted as an extension of this result to $\mathsf{Res}[\oplus]$.

1.2 Read-once linear branching programs

The model of read-once branching programs is a natural and extensively studied model of computation for which strong lower bounds are known [20, 3]. Here we consider an extension of this model where queries are linear forms. A linear branching program[1] $\mathcal{P}$ in the variables x is a DAG with the following properties:

- it has exactly one source;
- it has two sinks labeled with 0 and 1 representing the values of the function that $\mathcal{P}$ computes;
- every inner node is labeled by a linear form q over $\mathbb{F}_2$ in x which we call *queries*;
- every inner node with a label q has two outgoing edges labeled with 0 and 1 representing the value of q.

[1] This term has already been used before in [1] with a different meaning in the context of quantum computation.

Any assignment to the input variables naturally defines a path in the program. We say that $\mathcal{P}$ computes a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^n$, the path in $\mathcal{P}$ defined by x ends in the sink labeled with $f(x)$.

We now define read-once linear BPs. Given an inner node v of a linear branching program $\mathcal{P}$, we define $\mathrm{Pre}(v)$ as the span of all queries that appear on any path from the source of $\mathcal{P}$ to v, excluding the query at v. We define $\mathrm{Post}(v)$ as the span of all queries in the subprogram starting at v.

▶ **Definition 1** (Weakly and strongly read-once linear branching programs). *We say that a linear branching program $\mathcal{P}$ is* weakly read-once *if for every inner node v of $\mathcal{P}$ which queries q, it holds that $q \notin \mathrm{Pre}(v)$.*

We say that a linear branching program $\mathcal{P}$ is strongly read-once *if for every inner node v of $\mathcal{P}$, it holds that $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$.*

It follows from both definitions that queries alongside any path in weakly or strongly read-once BP are linearly independent. Furthermore, both of these models generalize standard read-once BPs and parity decision trees. When the distinction between weakly and strongly read-once is not important, we simply write "read-once".

1.3 Regular Res[⊕]

We also define a regular variant of Res[⊕] using similar conditions that we require from weakly read-once BPs.

▶ **Definition 2** (Regular Res[⊕]). *A* Res[⊕] *refutation is* (weakly) regular *if for every clause C obtained by the application of the resolution rule on a literal q, the span of all literals appearing in any application of the resolution rule to C or a clause derived from C, does not contain q.*

Intuitively, the read-once (or regular) nature of this definition can be described as following: after we resolve on q, we restrict the query space U by its complimentary to q, i.e., the space W with $q \notin W$ and $\mathrm{span}(q) + W = U$. If a clause C was derived using q, there exists a linear isomorphism L that maps q to a variable y, and for every application of the resolution rule to C of a clause derived from C on a linear literal q', Lq' does not contain y.

In Section 6 we establish the connection between weakly read-once BPs and regular Res[⊕] refutations.

2 Notation and basic facts

Each path in a read-once program defines an affine subspace given by the set of solutions of the system corresponding to the queries on the path. Any affine subspace can be represented by a vector space shifted by a vector from the affine space. For our purposes, we need to choose this shift carefully.

Let p be a path in a read-once linear BP $\mathcal{P}$ leading to a node v with queries $q_1, \ldots, q_k$ and answers $a_1, \ldots, a_k$ to these queries which define the affine subspace $S_p = \{x : \bigwedge_{i=1}^{k} q_i(x) = a_i\}$. Let V_p be the supporting vector space of S_p, i.e., $V_p = \{x : \bigwedge_{i=1}^{k} q_i(x) = 0\}$. Then clearly $S_p = V_p + b$ for any $b \in S_p$. Choose an arbitrary basis $q'_1, \ldots, q'_t$ for $\mathrm{Post}(v)$. Since $q'_1, \ldots, q'_t$ are independent of $q_1, \ldots, q_k$, there exists b such that $\bigwedge_{i=1}^{k} q_i(b) = a_i$ and $\bigwedge_{i=1}^{t} q'_i(b) = 0$. Then $S_p = V_p + b$ and for every $q \in \mathrm{Post}(v)$, we have $q(b) = 0$.

▶ **Definition 3** (Canonical affine subspace). *Given a path p which ends at a node v, we call S_p the* canonical affine subspace *for p. Furthermore a* canonical representation *of S_p is any $V_p + b = S_p$ where every $q \in \mathrm{Post}(v)$ vanishes on b.*

Throughout the paper we drop the word *representation* and simply say $V_p + b$ is the canonical affine subspace of p to mean that it is a canonical representation of S_p.

Since we will often use canonical affine subspaces to represent paths in BPs, we adopt the following algebraic notation. Let us denote the space of all linear forms on $\mathbb{F}_2^n$ (the dual space) as $(\mathbb{F}_2^n)^*$. Given a subspace V of $\mathbb{F}_2^n$, we define $V^\perp$ as the space of all linear forms from $(\mathbb{F}_2^n)^*$ that vanish on V (this space is sometimes called the annihilator of V), i.e.,

$$V^\perp = \{\ell \in (\mathbb{F}_2^n)^* : \forall v \in V,\ \ell(v) = 0\}.$$

Given a path p with queries $q_1, \ldots, q_k$ and its canonical affine subspace $V + b$, the space $V^\perp$ is the query space of p, i.e., $V^\perp = \mathrm{span}(q_1, \ldots, q_k)$.

It is clear that every read-once BP is a strongly read-once linear BP. We also show that every parity decision trees can be transformed into an equivalent tree that is also a strongly read-once linear BP, without increasing its size. We need the following notation. Let V and W be two subspaces. Then the sum of V and W is the subspace

$$V + W \coloneqq \{v + w : v \in V,\ w \in W\}.$$

Note that $V + W = \mathrm{span}(V \cup W)$.

▶ **Lemma 4.** *Let T be a parity decision tree. Then there exists a parity decision tree T' computing the same function, which is a strongly read-once linear BP.*

Proof. Without loss of generality, we may assume that for every node v of T, the query at v is linearly independent of the queries leading to v, since otherwise we can replace v with one of its children. To prove this lemma, we will use the fact that exactly one path passes through any node of the tree. We construct T' inductively on the depth of a tree node starting from the root. Let v be a node of T labeled with the query q. Let $B = \{\beta_1, \ldots, \beta_t\}$ be a basis of $\mathrm{Pre}(v) + \mathrm{span}(q)$ and B' its extension to a basis of the whole query space $(\mathbb{F}_2^n)^*$. We can rewrite every query in the subtree rooted at v in the new basis $B \cup B'$. Since the program is a tree, every query from B has the unique value. Thus, we can safely substitute them in every query in the v-subtree, possible changing the labels of the outgoing edges. After this transformation, every query in $\mathrm{Post}(v)$, except for the query at v, will be expressed in terms of B'. By construction, $B \cup B'$ is a basis and $\mathrm{Pre}(v)$ and $\mathrm{Post}(v)$ are expressed using different basis vectors, which implies $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$. ◀

Throughout the paper we adopt the following notation.

- Given a vector $c \in \{0,1\}^n$ the *support of c* is defined as

  $$\mathrm{supp}(c) \coloneqq \{i : c_i \neq 0\}.$$

- Let σ be a partial assignment to the variables $x_1, \ldots, x_n$. Then

  $$\mathrm{dom}(\sigma) \coloneqq \{i : \sigma(x_i) \text{ is defined}\}.$$

- We say that $a \in \{0,1\}^n$ is *consistent* with a partial assignment σ to $x_1, \ldots, x_n$ if for every $i \in \mathrm{dom}(\sigma)$, it holds that $\sigma(x_i) = a_i$.
- We write $a + b$ without specifying the underlying field, if it is clear from the context and often intended to be $\mathbb{F}_2$.

2.1 The trace map

The trace map $\mathrm{Tr}\colon \mathbb{F}_{p^n} \to \mathbb{F}_p$ is defined as

$$\mathrm{Tr}(x) \coloneqq \sum_{i=0}^{n-1} x^{p^i}.$$

One important property that we need is that Tr is an $\mathbb{F}_p$-linear map. We also use the following fact about the trace.

▶ **Proposition 5** (cf. [17]). *For every $\mathbb{F}_p$-linear map $\pi\colon \mathbb{F}_{p^n} \to \mathbb{F}_p$ there exists $\mu \in \mathbb{F}_{p^n}$ such that for all $x \in \mathbb{F}_{p^n}$ we have*

$$\pi(x) = \mathrm{Tr}(\mu \cdot x).$$

Furthermore, π is trivial if and only if $\mu = 0$.

Since, we are interested in Boolean functions, we will only consider the case $p = 2$. Let $\phi\colon \mathbb{F}_2^n \to \mathbb{F}_{2^n}$ be any $\mathbb{F}_2$-linear isomorphism. Then $\mathrm{Tr}(\mu \cdot \phi(x))$ is a linear Boolean function of x and we have the following:

▶ **Proposition 6.** *The set of all linear Boolean functions coincides with the set of functions $\ell_\mu(x) = \mathrm{Tr}(\mu \cdot \phi(x))$, where $\mu \in \mathbb{F}_{2^n}$.*

In the rest of the paper we fix ϕ. To make the proofs more readable we use bold font to denote the corresponding elements of $\mathbb{F}_{2^n}$, e.g., $\mathbf{x}$ for $\phi(x)$.

2.2 Affine extractors and dispersers

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is an *affine disperser* for dimension d if f is not constant on any affine subspace of dimension d. Let us also recall affine extractors, which are generalizations of affine dispersers.

The *bias* of f is defined as

$$\mathrm{bias}(f) \coloneqq \left| \mathop{\mathbb{E}}_{x \in U_n} [(-1)^{f(x)}] \right|,$$

where U_n is a uniform distribution on $\{0,1\}^n$. Given an affine subspace f, *the bias of f restricted to $S \subseteq \{0,1\}^n$* is defined as

$$\mathrm{bias}(f|_S) \coloneqq \left| \mathop{\mathbb{E}}_{x \in U(S)} [(-1)^{f(x)}] \right|,$$

where $U(S)$ is a uniform distribution on S.

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is an *affine extractor* for dimension d with bias ϵ if for every affine subspace S of dimension d, the bias of f restricted to S, $\mathrm{bias}(f|_S)$, is at most ϵ.

3 Affine mixedness

In this section we give a criterion for functions to be worst-case hard for read-once linear BPs. Let us first recall *mixedness* from standard read-once BPs.

▶ **Definition 7.** *A Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$ *is* d-mixed[2] *if for every* $I \subseteq [n]$ *of size at most* $n-d$ *and every two distinct partial assignments* σ *and* τ *with* $\mathrm{dom}(\sigma) = \mathrm{dom}(\tau) = I$*, it holds that* $f|_\sigma \neq f|_\tau$*.*

▶ **Theorem 8** (Folklore; see [13] for a proof)**.** *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a* d*-mixed Boolean function. Then any read-once branching program computing* f *has size at least* $2^{n-d}-1$*.*

Explicit constructions of d-mixed functions with $d = o(n)$ and thus $2^{n-o(n)}$ size lower bounds for read-once BPs were given in [20, 3]. We generalize this notion for linear branching programs. We need the following equivalent definition of d-mixedness.

▶ **Lemma 9.** *A Boolean function* f *is* d*-mixed if and only if for every partial assignments* σ *of size at most* $n-d$ *and every* $c \neq 0$ *with* $\mathrm{supp}(c) \subseteq \mathrm{dom}(\sigma)$*, there exists* x *consistent with* σ *such that* $f(x) \neq f(x+c)$*.*

Proof. ($\Leftarrow$) Let σ and τ be two distinct partial assignments with domain I of size at most $n-d$. Define $c_i = \tau(x_i) + \sigma(x_i)$ for $i \in I$ and $c_i = 0$ otherwise. By assumption there exists x consistent with σ such that $f(x) \neq f(x+c)$. It follows from the definition of c that $x+c$ is consistent with τ. Define $J = [n] \setminus I$ and $z = x_J = (x+c)_J$. Then $f|_\sigma(z) = f(x) \neq f(x+c) = f|_\tau(z)$.

($\Rightarrow$) Let σ be a partial assignment with a domain of size at most $n-d$ and let c be given such that $\mathrm{supp}(c) \subseteq \mathrm{dom}(\sigma)$. Define $\tau(x_i) = \sigma(x_i) + c_i$ for $i \in \mathrm{dom}(\sigma)$. By assumption $f|_\sigma \neq f|_\tau$, hence there exists z such that $f|_\sigma(z) \neq f|_\tau(z)$. Define x to take the same value as σ on $\mathrm{dom}(\sigma)$ and equal to z otherwise. Then $f(x) = f|_\sigma(z) \neq f|_\tau(z) = f(x+c)$. ◀

▶ **Definition 10.** *A Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$ *is* d-affine mixed *if for every affine subspace* S *of dimension at least* d *and every vector* $c \notin V$*, where* V *is the supporting vector space of* S*, there exists* $x \in S$ *such that* $f(x) \neq f(x+c)$*.*

It follows from Lemma 9 that d-affine mixedness implies d-mixedness since a partial assignment is a special case of an affine subspace.

Now we are ready to prove a generalization of Theorem 8.

▶ **Theorem 11.** *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a* d*-affine mixed Boolean function. Then any strongly read-once linear branching program computing* f *has size at least* $2^{n-d}-1$*.*

Proof. We prove that any such program $\mathcal{P}$ computing f starts with a complete binary tree of depth $n-d-1$. Assume for the sake of contradiction that there are two paths p and q of length at most $n-d-1$, which meet for the first time at a node v. Let $V+a$ and $W+b$ be their corresponding canonical affine subspaces. Both of them have dimension at least $d+1$.

We start by proving $V^\perp = W^\perp$ which implies $V = W$. Suppose that it is not the case. Without loss of generality, there exists $\ell \in W^\perp \setminus V^\perp$. By the read-once property $\ell \notin \mathrm{Post}(v)$.

Consider two affine subspaces $V' + a_1$ and $V' + a_2$ obtained by intersecting $V+a$ with $\ell(x) = 0$ and $\ell(x) = 1$ such that for every $\ell' \in \mathrm{Post}(v)$, $\ell'(a_1) = 0$ and $\ell'(a_2) = 0$ (recall that we can choose such a_1 and a_2 since $\mathrm{Pre}(v) \cap \mathrm{Post}(v) = \{0\}$). By construction, they have dimension at least d. Since f is d-affine mixed, there exists $z \in V'$ such that $f(z+a_1) \neq f(z+a_2)$. Consider any query ℓ' in the subprogram starting at v. The fact that

[2] This definition is commonly given for sets of size d instead of $n-d$. We deviate from this since for our generalization to affine spaces, it corresponds to dimension which is more natural.

$\ell' \in \mathrm{Post}(v)$ implies $\ell'(a_1) = \ell'(a_2) = 0$. Thus, we have $\ell'(z + a_1) = \ell'(z) = \ell'(z + a_2)$. It implies that in the subprogram starting at v both $z + a_1$ and $z + a_2$ must follow the same path contradicting $f(z + a_1) \neq f(z + a_2)$.

Now, since $V = W$, $V + b$ is the canonical affine subspace for q, and $a \neq b$ since p and q are different paths. Again, since f is d-affine mixed, there exists $z \in V$ such that $f(z + a) \neq f(z + b)$. Analogously to the previous case, for every $\ell' \in \mathrm{Post}(v)$ we have $\ell'(a) = \ell'(b) = 0$, and thus $\ell'(z + a) = \ell'(z + b)$ contradicting $f(z + a) \neq f(z + b)$. ◀

4 Affine dispersers for directional derivatives

In this section we give an explicit construction of an affine mixed function for linear dimension. In fact, we give an even more powerful construction, which allows us to get an average-case lower bound for strongly read-once linear branching programs.

For a Boolean function f its directional derivative with respect to a non-zero vector a is defined as

$$D_a f(x) \coloneqq f(x + a) + f(x).$$

▶ **Definition 12.** *A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is a* directional affine extractor *for dimension d with bias ϵ if for every non-zero a, the derivative $D_a f$ is an affine extractor for dimension d with bias ϵ.*

Similarly, f is a directional affine disperser *for dimension d if for every non-zero a, $D_a f$ is an affine disperser for dimension d.*

Observe that this notion is stronger than the one defined in the previous section: if f is a directional affine disperser for dimension d, then it is d-affine mixed.

In what follows we construct a Boolean function f in n variables that is a good directional affine extractor for dimensions bigger than $\frac{2}{3}n$.

It is a well-known fact that the inner product function is an affine extractor. IP is a member of the class of bent functions, which are all affine extractors. A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is called a *bent function* if all Fourier coefficients of its ± 1 representation $f_{\pm}(x) \coloneqq (-1)^{f(x)}$ have the same absolute value.

▶ **Lemma 13** (Folklore; for a proof see, e.g., [8, 7])**.** *Let f be a bent function on n variables and $c \geq 1$ be an integer. Then, f is an affine extractor for dimension $k = n/2 + c$ with bias at most 2^{-c}. In particular, f is an affine disperser for dimension $n/2 + 1$.*

We apply this result to prove that the following function is an affine extractor.

▶ **Lemma 14.** *Let $a_0, a_1, a_2, a_3 \in \mathbb{F}_{2^k}$ with $a_0 \neq 0$. Let $g\colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ be the function defined as*

$$g(x, y) = \mathrm{Tr}(a_0 \cdot \phi(x) \cdot \phi(y) + a_1 \cdot \phi(x) + a_2 \cdot \phi(y) + a_3).$$

Then g is an affine extractor for dimension $k + c$ with bias at most 2^{-c}. In particular, g is an affine disperser for dimension $k + 1$.

Intuitively, the functions $\mathrm{Tr}(x \cdot y)$ and $\mathrm{IP}(x, y)$ behave similarly: the Fourier transform of functions with domain $\mathbb{F}_{2^n}$ is sometimes defined in terms of the trace map. Here we prove the statement directly using the usual definition of the Fourier transform for Boolean functions.

Proof. Let $g_\pm$ be the ± 1 representation of g. By Lemma 13, it is enough to prove that all Fourier coefficients of $g_\pm$ have the same absolute value. Recall that given $\alpha \in \{0,1\}^{2k}$ the α-character χ_α is defined as $\chi_\alpha(x,y) = (-1)^{\alpha\cdot(x,y)}$, where $\alpha \cdot (x,y)$ is the inner product. Fourier coefficient $\widehat{g_\pm}(\alpha)$ can be computed as follows.

$$\widehat{g_\pm}(\alpha) = \sum_{x,y\in\{0,1\}^k} g_\pm(x,y)\cdot\chi_\alpha(x,y) = \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3)}\cdot\chi_\alpha(x,y).$$

Split α into two equal parts: $\alpha = (\alpha_1,\alpha_2)$. Then $\alpha\cdot(x,y) = \alpha_1\cdot x+\alpha_2\cdot y$. By Proposition 6, there exist $\mu_1,\mu_2 \in \mathbb{F}_{2^k}$ such that $\alpha_1\cdot x = \mathrm{Tr}(\mu_1\cdot\mathbf{x})$ and $\alpha_2\cdot y = \mathrm{Tr}(\mu_2\cdot\mathbf{y})$. Also define

$$\begin{aligned}
b_1 &\coloneqq a_0^{-1}\cdot(a_1+\mu_1),\\
b_2 &\coloneqq a_0^{-1}\cdot(a_2+\mu_2),\\
b_3 &\coloneqq a_3 + a_0^{-1}\cdot(a_1+\mu_1)\cdot(a_2+\mu_2).
\end{aligned}$$

Then we can express $\widehat{g_\pm}(\alpha)$ in terms of b_i:

$$\begin{aligned}
\widehat{g_\pm}(\alpha) &= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3)}\cdot(-1)^{\mathrm{Tr}(\mu_1\cdot\mathbf{x})+\mathrm{Tr}(\mu_2\cdot\mathbf{y})}\\
&= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot\mathbf{x}\cdot\mathbf{y}+a_1\cdot\mathbf{x}+a_2\cdot\mathbf{y}+a_3+\mu_1\cdot\mathbf{x}+\mu_2\cdot\mathbf{y})}\\
&= \sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot(\mathbf{x}+b_2)\cdot(\mathbf{y}+b_1)+b_3)}\\
&= (-1)^{\mathrm{Tr}(b_3)}\cdot\sum_{x,y\in\{0,1\}^k} (-1)^{\mathrm{Tr}(a_0\cdot(\mathbf{x}+b_2)\cdot(\mathbf{y}+b_1))}.
\end{aligned}$$

Since x and y iterate through all vectors from $\{0,1\}^k$, $a_0\cdot(\mathbf{x}+b_2)$ and $\mathbf{y}+b_1$ take all possible values from $\mathbb{F}_{2^k}$. It follows that

$$\widehat{g_\pm}(\alpha) = (-1)^{\mathrm{Tr}(b_3)}\widehat{g_\pm}(0).$$ ◀

We are now ready to present our directional affine extractor.

▶ **Theorem 15.** *Let $f\colon \{0,1\}^k\times\{0,1\}^k\times\{0,1\}^k \to \{0,1\}$ be the function defined by*

$$f(x,y,z) = \mathrm{Tr}(\phi(x)\cdot\phi(y)\cdot\phi(z)).$$

Then f is a directional affine extractor for dimension $2k+c$ with bias $\epsilon \le 2^{-c}$. In particular, f is a directional affine disperser for dimension $2k+1$.

Proof. Consider the directional derivative of f in the non-zero direction $a = (a_1,a_2,a_3)$:

$$\begin{aligned}
D_a f(x,y,z) &= f(x+a_1,y+a_2,z+a_3)+f(x,y,z)\\
&= \mathrm{Tr}(\phi(x+a_1)\cdot\phi(y+a_2)\cdot\phi(z+a_3))+\mathrm{Tr}(\mathbf{x}\cdot\mathbf{y}\cdot\mathbf{z}).
\end{aligned}$$

By linearity of Tr and ϕ we have

$$\begin{aligned}
D_a f(x,y,z) &= \mathrm{Tr}((\mathbf{x}+\mathbf{a_1})\cdot(\mathbf{y}+\mathbf{a_2})\cdot(\mathbf{z}+\mathbf{a_3})+\mathbf{x}\cdot\mathbf{y}\cdot\mathbf{z})\\
&= \mathrm{Tr}(\mathbf{a_1}\cdot\mathbf{y}\cdot\mathbf{z}+\mathbf{a_2}\cdot\mathbf{x}\cdot\mathbf{z}+\mathbf{a_3}\cdot\mathbf{x}\cdot\mathbf{y}+\ell(\mathbf{x},\mathbf{y},\mathbf{z})),
\end{aligned}$$

where ℓ is an affine function.

Without loss of generality we may assume that $a_3 \neq 0$. Let S be an affine subspace with dimension at least $2k + c$. We need to show that the bias of f restricted to S is at most ϵ. Given $z_0 \in \{0,1\}^k$ define $S_{z_0} \coloneqq \{(x,y) : (x,y,z_0) \in S\}$. For every z_0 the affine subspace S_{z_0} is either empty or has dimension at least $k + c$. Consider the restriction of $D_a f$ to $z = z_0$.

$$h_{z_0}(x,y) \coloneqq D_a f(x,y,z_0) = \mathrm{Tr}(\mathbf{a_3} \cdot \mathbf{x} \cdot \mathbf{y} + \ell'_{z_0}(\mathbf{x},\mathbf{y})),$$

where ℓ'_{z_0} is an affine function. By Lemma 14, h_{z_0} is an affine extractor for dimension $k + c$ with bias $\epsilon \leq 2^{-c}$. In particular, if S_{z_0} is non-empty, then $\mathrm{bias}(h_{z_0}|_{S_{z_0}}) \leq \epsilon$.

Thus, the bias of $D_a f$ restricted to S can easily be bounded as follows:

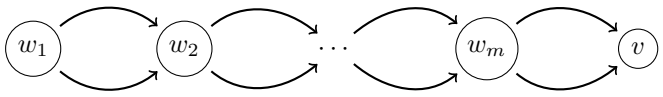
$$\begin{aligned}
\mathrm{bias}(D_a f|_S) &= \left| \frac{1}{|S|} \sum_{(x,y,z)\in S} (-1)^{D_a f(x,y,z)} \right| \\
&= \left| \frac{1}{|S|} \sum_{z_0\in\{0,1\}^n} \sum_{(x,y,z_0)\in S} (-1)^{D_a f(x,y,z_0)} \right| \\
&= \left| \frac{1}{|S|} \sum_{z_0\in\{0,1\}^n} \sum_{(x,y)\in S_{z_0}} (-1)^{h_{z_0}(x,y)} \right| \\
&\leq \frac{1}{|S|} \sum_{z_0\in\{0,1\}^n} \left| \sum_{(x,y)\in S_{z_0}} (-1)^{h_{z_0}(x,y)} \right| \\
&\leq \frac{1}{|S|} \sum_{z_0\in\{0,1\}^n} \epsilon \cdot |S_{z_0}| = \epsilon.
\end{aligned}$$

◀

5 Average-case lower bound

We consider a canonical form of strongly read-once linear branching programs. We adopt the terminology of [6] and say that a read-once linear branching program is *full* if for every inner node v of the program, all the paths leading to v have the same query space.

A *multipath* $(w_1, \ldots, w_m, v)$ is a linear branching program of the form



That is, the program ignores the answers to the queries at w_i for every i. Given a program $\mathcal{P}$, we say that a subset of nodes is an *antichain* if none of its nodes is a descendant of another. For example, the set of nodes at a fixed depth and the set of leaves form an antichain. The following lemma and its proof are easy extensions of Lemma 3.7 in [6].

▶ **Lemma 16.** *Every weakly read-once or strongly read-once linear branching program $\mathcal{P}$ of size s in n variables has an equivalent full weakly read-once or strongly read-once linear branching program $\mathcal{P}'$, respectively, of size at most $3n \cdot s$. Furthermore, the size of every antichain in $\mathcal{P}'$ is at most $2s$.*

Proof. We construct $\mathcal{P}'$ inductively. Consider the nodes of $\mathcal{P}$ in topological order. It is clear that the start node satisfies the fullness property. Let v be a node of $\mathcal{P}$ and $p_1, \ldots, p_k$ the paths that meet at v, and $V_1 + a_1, \ldots, V_k + a_k$ their canonical affine subspaces. For every $i \in [k]$ choose a set of linearly independent queries Q_i such that $V_i^{\perp} + \mathrm{span}(Q_i) = \mathrm{Pre}(v)$.

For every $i \in [k]$ do the following. Let $Q_i = \{q_1, \dots, q_m\}$. Replace the edge $u_i \to v$ with a multipath $(w_1, \dots, w_m, v)$ and an edge $u_i \to w_1$, where w_i are labeled with q_i. After this transformation, every path to v will have the query space $\mathrm{Pre}(v)$.

Since a branching program of size s has at most $2s$ edges and we replaced every edge with a multipath of length at most n, the size of the constructed full read-once linear branching program $\mathcal{P}'$ is at most $s + 2s \cdot n \le 3n \cdot s$.

Consider an antichain A in $\mathcal{P}'$. We map every node in A to nodes in $\mathcal{P}$. Each node in A is either originally in $\mathcal{P}$ or it was created by a multipath. In the former case we map it to itself, and in the latter case we map it to the parent node from which it was created. Since the out-degree in $\mathcal{P}$ is 2 and A is an antichain, at most 2 nodes are mapped to the same node. This proves the result. ◀

Denote by $\mathrm{dist}(f, g)$ the relative distance between Boolean function f and g.

▶ **Theorem 17.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a directional affine extractor for dimension d with bias $\epsilon < \frac{1}{2}$. Then for every $g\colon \{0,1\}^n \to \{0,1\}$ computed by a strongly read-once linear branching program $\mathcal{P}$ of size at most $\epsilon \cdot 2^{n-d-1}$, it holds that $\mathrm{dist}(f, g) \ge \frac{1-\sqrt{2\epsilon}}{2}$.*

Proof. Let s denote the size of $\mathcal{P}$. We first convert $\mathcal{P}$ into a full program. By Lemma 16, the size of every antichain is at most $2s$. We then construct an equivalent program $\mathcal{P}'$ in which every path has length at least $n - d$. We can achieve this by extending every leaf of low depth by a multipath of an appropriate length.

Consider the set A of nodes in $\mathcal{P}'$ at depth exactly $n - d$. Note that every $v \in A$ is either a node at depth $n - d$ in $\mathcal{P}$, or it is uniquely defined by a leaf of $\mathcal{P}$ by a multipath. Thus A is identified by an antichain in $\mathcal{P}$ and thus $|A| \le 2s$.

We call an input x *wrong* if $f(x) \ne g(x)$. The distance $\mathrm{dist}(f, g)$ between f and g is the fraction of wrong inputs.

▷ Claim 18. Let $v \in A$ and k the numbers of paths that meet at v. Then the number of wrong inputs that pass through v is at least

$$\frac{k \cdot 2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right).$$

Proof. Since the program is full, the corresponding canonical affine subspaces for the paths that meet at v are $V + a_1, \dots, V + a_k$, for some d-dimensional vector space V, and distinct $a_1, \dots, a_k \in \{0,1\}^n$. Recall that f is a directional affine extractor with bias ϵ. Then for every $i \ne j$, it holds that $D_{a_i + a_j} f = f(x + (a_i + a_j)) + f(x)$ is an affine extractor with bias ϵ, thus

$$\begin{aligned}\left|\sum_{x \in V} (-1)^{f(x+a_i)} \cdot (-1)^{f(x+a_j)}\right| &= \left|\sum_{x \in V + a_j} (-1)^{f(x+a_i+a_j)+f(x)}\right| \\ &= \mathrm{bias}\Big(D_{a_i+a_j} f\big|_{V+a_j}\Big) \cdot |V| \le \epsilon |V|.\end{aligned} \tag{1}$$

Every $x \in V$ produces a partition of $[k]$ into two parts $(J, [k] \setminus J)$ such that $f(x + a_i) = 0$ for $i \in J$ and $f(x + a_i) = 1$ for $i \notin J$. Let m_x be the size of the *smallest* part. By definition of canonical affine subspace and the choice of a_i, for any linear query $q \in \mathrm{Post}(v)$ we have $q(a_i) = 0$ for all $i \in [k]$. Then $x + a_1, \dots, x + a_k$ will follow the same path in the subprogram

starting at v. Hence, for every $x \in V$ it holds that $f(x + a_1) = \cdots = f(x + a_k)$. It implies that at least m_x inputs of the form $x + a_i$ are wrong and the total number of wrong inputs passing through v is at least

$$m \coloneqq \sum_{x \in V} m_x.$$

Now consider the following sum

$$E \coloneqq \sum_{\substack{x \in V \\ 1 \le i < j \le k}} |f(x + a_i) - f(x + a_j)|.$$

We apply double counting to this quantity to obtain the result. On the one hand, by definitions of m_x and m, we have

$$E = \sum_{x \in V} m_x \cdot (k - m_x) = km - \sum_{x \in V} m_x^2.$$

By the Cauchy–Schwarz inequality, $\sum_{x \in V} m_x^2 \ge \left(\sum_{x \in V} m_x\right)^2 / |V| = m^2/|V|$. Thus,

$$E \le km - m^2/|V|. \tag{2}$$

On the other hand, E can be rewritten as follows.

$$\begin{aligned} E &= \sum_{\substack{x \in V \\ 1 \le i < j \le k}} \frac{1}{4}\left((-1)^{f(x+a_i)} - (-1)^{f(x+a_j)}\right)^2 \\ &= \frac{1}{4} \sum_{1 \le i < j \le k} \left(2|V| - 2 \sum_{x \in V} (-1)^{f(x+a_i)} \cdot (-1)^{f(x+a_j)}\right). \end{aligned}$$

Applying (1), we obtain the following lower bound on E.

$$E \ge \frac{1}{2}\binom{k}{2}|V|(1 - \epsilon). \tag{3}$$

Combining (2) and (3), we get

$$km - m^2/|V| \ge \frac{1}{2}\binom{k}{2}|V|(1 - \epsilon).$$

This can be written as

$$\begin{aligned} \left(m - \frac{k|V|}{2}\right)^2 &\le \frac{1}{4}k^2|V|^2 - \frac{1-\epsilon}{2}\binom{k}{2}|V|^2 \\ &= \frac{k^2|V|^2}{4}\left(1 - (1-\epsilon)\left(1 - \frac{1}{k}\right)\right) \\ &\le \frac{k^2|V|^2}{4}\left(\epsilon + \frac{1}{k}\right). \end{aligned}$$

Thus,

$$m \ge \frac{k|V|}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right) = \frac{k \cdot 2^d}{2}\left(1 - \sqrt{\epsilon + \frac{1}{k}}\right). \qquad \lhd$$

Let $k(v)$ denote the number of paths that meet at v and define $w(k)$ as

$$w(k) \coloneqq \frac{k2^d}{2}\left(1-\sqrt{\epsilon+\frac{1}{k}}\right).$$

Then by Claim 18 the total number of bad inputs that pass through A is at least

$$\sum_{v\in A} w(k(v)) = \sum_{v\in A} \frac{k(v)2^d}{2}\left(1-\sqrt{\epsilon+\frac{1}{k(v)}}\right).$$

Since all paths in $\mathcal{P}'$ has length at least $n-d$, $\sum_{v\in A} k(v) = 2^{n-d}$.

The function w is convex, hence by Jensen's inequality, the total number of bad inputs passing through A is at least

$$\sum_{v\in A} w(k(v)) \geq |A| \cdot w\left(\frac{\sum_{v\in A} k(v)}{|A|}\right) = \frac{1}{2}2^n\left(1-\sqrt{\epsilon+\frac{|A|}{2^{n-d}}}\right).$$

Since $|A| \leq 2s \leq \epsilon 2^{n-d}$, this expression is at least $\frac{1-\sqrt{2\epsilon}}{2}2^n$. ◀

Plugging in the function of Theorem 15 we get the following corollary.

▶ **Corollary 19.** *Let $f : \{0,1\}^{\frac{n}{3}} \times \{0,1\}^{\frac{n}{3}} \times \{0,1\}^{\frac{n}{3}} \to \{0,1\}$ be defined by $f(x,y,z) = \mathrm{Tr}(\phi(x)\cdot\phi(y)\cdot\phi(z))$. Then for every $g : \{0,1\}^n \to \{0,1\}$ computed by a strongly read-once linear BP of size at most $2^{\frac{n}{3}-o(n)}$, $\mathrm{dist}(f,g) \geq \frac{1}{2} - 2^{-o(n)}$.*

6 Weakly read-once BPs and Res[⊕]

In this section we prove an analogue of the correspondence between read-once BPs and regular resolution for Res[⊕] and weakly read-once BPs. The first part of the proof is an extension of a standard argument; the second part, while also easy, is more subtle.

▶ **Theorem 20.**

1. *Every* Res[⊕] *refutation of an unsatisfiable CNF F can be translated into a linear BP solving the corresponding search problem without increasing its size. If the refutation is regular, then the resulting program is weakly read-once.*
2. *Every weakly read-once BP of size s solving the search problem for CNF $F = C_1 \wedge \ldots \wedge C_m$ in n variables can be translated into a regular* Res[⊕] *refutation of F of size $O(ns)$.*

Proof.

1. Consider an application of the resolution rule in the proof DAG G. Suppose that it is applied to clauses $C_0 \vee (f=0)$ and $C_1 \vee (f=1)$. Then we label the outgoing edges with $f=1$ and $f=0$ respectively. We leave the edges corresponding to the weakening rule unlabeled.

Let u be a vertex in G and C_u the clause it is labeled with. It can be shown by induction on the depth of u that for every path to u, the linear system obtained from the equations written on the edges on this path implies $\neg C_u$. The source contains the empty clause, hence the base case holds. For the inductive step, consider any path leading to u and let v be the parent of u on this path. Consider the case when v corresponds to an application of the resolution rule and w be its other child. Let $C_0 \vee (f=b)$, $C_1 \vee (f=b+1)$, and $C_0 \vee C_1$ be the labels of u, w, and v respectively, where $b \in \{0,1\}$. By the induction hypothesis, every path to v implies

$\neg(C_0 \vee C_1)$. In particular, it implies $\neg C_0$. By construction, the edge (v, u) is labeled with $f = b+1$. Then every path to u going through v implies $\neg C_0 \wedge (f = b+1) = \neg(C_0 \vee (f = b))$. Now consider the case when u corresponds to an application of the weakening rule and let v be it parent on this path. Let C and D be the labels of u and v. Every path to v implies $\neg D$ by the induction hypothesis and $\neg D \vDash \neg C$. Thus, every path to u through v implies $\neg C$.

In particular, every path to the sinks of G falsifies some clause of F. To obtain a linear BP, we remove labels at the inner nodes and contract all unlabeled edges. If the refutation is regular, the resulting linear BP is indeed read-once since we did not essentially change the structure of the underlying DAG.

2. A linear clause $C = \bigvee_{i=1}^{k}(f_i = a_i)$ can be viewed as a negation of a linear system $\neg C = \bigwedge_{i=1}^{k}(f_i = a_i + 1)$. We first convert P into a full BP of size $O(ns)$ using Lemma 16. Inductively, to every node v we associate a linear clause C_v such that:

1. Every assignment reaching v falsifies C_v.

2. If $\neg C_v$ represents a linear system $Bx = b$, then the row space of B is $\mathrm{Pre}(v)$.

For the base case, with each leaf v we associate the clause C_v it is labeled with. The first condition holds since P solves the search problem. To see the second property, note that any path reaching v can be expressed as a linear system on a basis for $\mathrm{Pre}(v)$ which forces every literal in C_v. This implies that single variables in C_v are in $\mathrm{Pre}(v)$.

For the inductive step, consider a node v, which queries q with outgoing neighbors u and w, in the directions $q = 0$ and $q = 1$ respectively. Observe that $\neg C_u \not\models q(x) = 1$ and $\neg C_w \not\models q(x) = 0$. Thus, there are only two cases to consider:

1. $\neg C_u \not\models q(x) = 0$ or $\neg C_w \not\models q(x) = 1$,

2. $\neg C_u \models q(x) = 0$ and $\neg C_w \models q(x) = 1$.

In the first case, we simply let C_v be C_u or C_w, depending on which condition holds. For the second case, let $B = \{\beta_1, \ldots, \beta_t\}$ be a basis of $\mathrm{Pre}(v)$. Fullness implies $\mathrm{Pre}(u) = \mathrm{Pre}(w) = \mathrm{Pre}(v) + \mathrm{span}(q)$. Applying the inductive hypothesis, we can write $\neg C_u = (q(x) = 0) \wedge (B_u x = b_u)$ and $\neg C_w = (q(x) = 1) \wedge (B_w x = b_w)$, where B_u and B_w are matrices with rows in $\beta_1, \ldots, \beta_t$ and b_u and b_w are some vectors. To write C_u and C_w in these forms, we might need to change the basis, which we can do by applying the weakening rule. We claim that setting C_v so that $\neg C_v$ can be written as $B_u x = b_u \wedge B_w x = b_w$ satisfies the requirements.

Consider any path to v. Such a path can be described by a system $Rx = b$ where rows in R are from B. Since every such path can be extended to both u and w, it follows that $B_u x = b_u \vDash Rx = b$ and $B_w x = b_w \vDash Rx = b$. This means that $B_u x = b_u \wedge B_w x = b_w$ is consistent and thus the derivation of C_v from C_u and C_w (possibly changing the basis) is a valid $\mathsf{Res}[\oplus]$ step. It is easy to see that conditions 1 and 2 hold for C_v.

Since for every v we create at most 2 extra clauses, the total size of the proof is at most $O(ns)$. ◀

Note that we can also define a "strongly regular" variant of $\mathsf{Res}[\oplus]$ that would be equivalent to strongly read-once programs. Since we have obtained average case lower bounds on strongly read-once BPs, it is plausible that it might be easier to prove lower bounds for this restriction of $\mathsf{Res}[\oplus]$.

7 Conclusion

Several problems are immediately suggested by our work:

- *Explicit constructions.* Give an explicit construction of directional affine extractors (or dispersers) for smaller dimension d, ideally $d = o(n)$.
- *BP lower bounds.* Prove worst-case and average-case hardness results for the weakly read-once BPs.
- *Proof complexity.* Prove a read-once linear BP lower bound for a search problem, that is for some unsatisfiable CNF $F = C_1 \wedge \ldots \wedge C_m$, show that a read-once linear BP with leaves labeled by C_is solving the corresponding search problem has to be large.

References

1 Farid M. Ablayev, Aida Gainutdinova, Marek Karpinski, Cristopher Moore, and Chris Pollett. On the computational power of probabilistic and quantum branching program. *Inf. Comput.*, 203(2):145–162, 2005. doi:10.1016/j.ic.2005.04.003.

2 Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC0-MOD2. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260. ACM, 2014. doi:10.1145/2554797.2554821.

3 Alexander E. Andreev, Juri L. Baskakov, Andrea E. F. Clementi, and José D. P. Rolim. Small pseudo-random sets yield hard functions: New tight explicit lower bounds for branching programs. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 179–189. Springer, 1999. doi:10.1007/3-540-48523-6_15.

4 Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012. doi:10.1137/110826254.

5 Jean Bourgain. On the construction of affine extractors. *Geom. Funct. Anal.*, 17(1):33–57, 2007. doi:10.1007/s00039-007-0593-z.

6 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Comput. Complex.*, 24(2):333–392, 2015. doi:10.1007/s00037-015-0100-0.

7 Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $AC^0 \circ MOD_2$ lower bounds for the boolean inner product. *J. Comput. Syst. Sci.*, 97:45–59, 2018. doi:10.1016/j.jcss.2018.04.006.

8 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58. ACM, 2016. doi:10.1145/2840728.2840734.

9 Svyatoslav Gryaznov. Notes on resolution over linear equations. In *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 168–179. Springer, 2019. doi:10.1007/978-3-030-19955-5_15.

10 Johan Håstad. On small-depth Frege proofs for Tseitin for grids. *J. ACM*, 68(1):1:1–1:31, 2021. doi:10.1145/3425606.

11 Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 121–131. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.20.

12 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Ann. Pure Appl. Log.*, 171(1), 2020. doi:10.1016/j.apal.2019.102722.

13 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-24508-4.

14 Erfan Khaniki. On proof complexity of resolution over polynomial calculus. *Electron. Colloquium Comput. Complex.*, page 34, 2020. URL: https://eccc.weizmann.ac.il/report/2020/034.

15 Jan Krajíček. *Proof Complexity*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. doi:10.1017/9781108242066.

16 Massimo Lauria. A note about k-DNF resolution. *Inf. Process. Lett.*, 137:33–39, 2018. doi:10.1016/j.ipl.2018.04.014.

17 Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1996. doi:10.1017/CBO9780511525926.

18 Fedor Part and Iddo Tzameret. Resolution with counting: Dag-like lower bounds and different moduli. *Comput. Complex.*, 30(1):2, 2021. doi:10.1007/s00037-020-00202-x.

19 A. A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

20 Petr Savický and Stanislav Žák. A read-once lower bound and a (1, +k)-hierarchy for branching programs. *Theor. Comput. Sci.*, 238(1-2):347–362, 2000. doi:10.1016/S0304-3975(98)00219-9.

21 Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. *Electron. Colloquium Comput. Complex.*, page 144, 2012. URL: https://eccc.weizmann.ac.il/report/2012/144.

22 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.

23 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

24 Amir Yehudayoff. Affine extractors over prime fields. *Comb.*, 31(2):245–256, 2011. doi:10.1007/s00493-011-2604-9.

Quantum Search-To-Decision Reductions and the State Synthesis Problem

Sandy Irani
Department of Computer Science, University of California, Irvine, CA, USA

Anand Natarajan
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Chinmay Nirkhe
Department of Computer Science, University of California, Berkeley, CA, USA
Challenge Institute for Quantum Computation, University of California, Berkeley, CA, USA

Sujit Rao
CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA

Henry Yuen
Department of Computer Science, Columbia University, New York, NY, USA

Abstract

It is a useful fact in classical computer science that many search problems are reducible to decision problems; this has led to decision problems being regarded as the *de facto* computational task to study in complexity theory. In this work, we explore search-to-decision reductions for quantum search problems, wherein a quantum algorithm makes queries to a classical decision oracle to output a desired quantum state. In particular, we focus on search-to-decision reductions for QMA, and show that there exists a quantum polynomial-time algorithm that can generate a witness for a QMA problem up to inverse polynomial precision by making one query to a PP decision oracle. We complement this result by showing that QMA-search does *not* reduce to QMA-decision in polynomial-time, relative to a quantum oracle.

We also explore the more general *state synthesis problem*, in which the goal is to efficiently synthesize a target state by making queries to a classical oracle encoding the state. We prove that there exists a classical oracle with which any quantum state can be synthesized to inverse polynomial precision using only one oracle query and to inverse exponential precision using two oracle queries. This answers an open question of Aaronson [1], who presented a state synthesis algorithm that makes $O(n)$ queries to a classical oracle to prepare an n-qubit state, and asked if the query complexity could be made sublinear.

2012 ACM Subject Classification Theory of computation → Quantum computation theory

Keywords and phrases Search-to-decision, state synthesis, quantum computing

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.5

Related Version *Full Version*: https://arxiv.org/abs/2111.02999 [8]

Funding *Chinmay Nirkhe*: Supported by NSF Quantum Leap Challenges Institute Grant number OMA2016245 and an IBM Quantum PhD internship.
Henry Yuen: Supported by AFOSR award FA9550-21-1-0040 and NSF CAREER award CCF-2144219.

Acknowledgements SI, CN, and HY were participants in the Simons Institute for the Theory of Computing *Summer Cluster on Quantum Compuatation*. Additionally, we thank Aram Harrow and Zeph Landau for insightful discussions.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 5; pp. 5:1–5:19

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

1 Introduction

It is a useful fact in classical computer science that *search* problems are often efficiently reducible to *decision* problems. For example, the canonical way of constructing a satisfying assignment of a given 3SAT formula φ (if there exists one) using an oracle for the decision version of 3SAT is to adaptively query the oracle for the satisfiability of φ conditioned on some partial assignment to the variables of the formula. Based on the oracle answers, the partial assignment can be extended bit-by-bit to a full assignment. Each oracle query reveals an additional bit of the assignment. This strategy generally works for any problem in NP. Likewise, the optimal value of an optimization problem can be calculated to exponential accuracy using binary search. The main consequence of this is that complexity theory often focuses on decision problems (without losing generality) and less on the complexity of search problems.

Quantum information and computation has shifted our perspective on these traditional notions of classical complexity theory. In this paper we consider *quantum* search problems, where the goal is to output a quantum state (as opposed to a classical bit string) satisfying some condition. In the quantum setting, it is no longer apparent that search-to-decision reductions still hold, and thus it is unclear whether the complexity of quantum search problems can be directly related to the complexity of corresponding quantum decision problems.

To illustrate this, we consider the analogues of P and NP in quantum computing, which are the complexity classes BQP and QMA, respectively[1]. The analogue of the NP-complete problem 3SAT for QMA is the *Local Hamiltonian* problem, in which one has to decide whether the lowest energy state of a local Hamiltonian $H = H_1 + \cdots + H_m$ acting on n qubits has energy greater than a or less than b for $a - b = 1/\mathsf{poly}(n)$, where each term H_i acts non-trivially on only a constant number of qubits. This problem was proven to be QMA-complete by Kitaev [10]. Is there an efficient search-to-decision reduction for the Local Hamiltonians problem, or more generally for the class QMA? In other words, given quantum query access to an oracle deciding the Local Hamiltonians problem, can a polynomial-time quantum algorithm (i.e. BQP machine) efficiently *prepare* a low-energy state $|\psi\rangle$ of a given local Hamiltonian?

The classical strategy of incrementally building a partial assignment does not appear to work in the QMA setting. First, there does not appear to be a natural way of "conditioning" a quantum state on a partial assignment. Second, quantum states are exponentially complex: the description size (complexity) of a general quantum state on n qubits is exponential in n, and this is suspected to remain true even when considering ground states of local Hamiltonians[2]. This complexity of quantum states poses a significant challenge to finding a search-to-decision reduction for QMA; it is not clear how yes/no answers to QMA decision problems (even when obtained in superposition) can be used to construct exponentially-complex QMA witnesses.

On the other hand, there *is* a natural quantum analogue of the bit-by-bit search-to-decision algorithm for NP that works for constructing *general* quantum states. This is due to a general algorithm for *state synthesis* described by Aaronson in [1] (for which we give

[1] Technically speaking, BQP and QMA are better thought of as the quantum analogues of BPP and MA, respectively. However, even in this randomized setting, there are efficient search-to-decision randomized reductions.

[2] Due to the QMA $\neq$ QCMA conjecture [3]. Formally, there is no known poly-sized description of a witness (proof) for every local Hamiltonian problem.

an overview of in Section 1.1): there exists a polynomial-time quantum algorithm A such that every n-qubit state $|\psi\rangle$ can be encoded into a classical oracle f where, by making $O(n)$ superposition queries to the oracle f, the algorithm A will output a state that is exponentially close to $|\psi\rangle$. One can observe that for states $|\psi\rangle$ that QMA witnesses (such as ground states of local Hamiltonians), the oracle f corresponds to a PP function (which is at least as powerful as a QMA oracle). This yields a search-to-decision reduction for QMA, albeit with a decision oracle of higher complexity.

In this work, we explore the complexity of search-to-decision procedures in the quantum setting, where the goal is a quantum *state synthesis* algorithm that outputs a *target* quantum state (e.g. a ground state of a local Hamiltonian) by making quantum queries to a classical decision oracle. We investigate how the complexity of the state synthesis algorithm and the complexity of the decision oracle depend on the type of states we want to generate. We consider both the generalized state synthesis problem for abritrary states in the Hilbert space $(\mathbb{C}^2)^{\otimes n}$ as well as the specific task of generating solutions to QMA problems.

We construct state synthesis and search-to-decision procedures for the quantum setting using only one or two superposition queries as opposed to $O(n)$ superposition queries; for QMA witnesses, the synthesis procedure requires only one query to a PP oracle. Simultaneously, we prove results suggesting the impossibility of any search-to-decision reduction for QMA. More precisely, we show that there exists a *quantum* oracle $\mathcal{O}$ relative to which *all* efficient query algorithms fail to be a good search-to-decision reduction for $\mathsf{QMA}^{\mathcal{O}}$, the relativization of QMA. This stands in contrast to classes such as NP, MA, and QCMA, which all have efficient search to decision reductions, relative to any oracle. As a consequence, proving impossibility of QMA search-to-decision without an oracle is at least as hard as separating QCMA and QMA which is at least as hard as separating P and PP. We believe that the juxtaposition of our results lend further weight to the view that the complexity of tasks where the outputs (and inputs) are quantum states cannot be directly explained by the traditional study of decision problems (which has been the main focus of quantum complexity theory to date). In particular, we believe our results suggest that the relationship between search and decision problems is much more mysterious in the quantum setting. As suggested by Aaronson in [1] and others in some recent works [11, 14], the complexity of quantum states (and more generally, quantum state transformations) deserves to be studied more deeply as a subject in its own right.

1.1 Starting point

Before describing our results in more detail, we first explain the starting point for our investigations, which is a simple state synthesis algorithm described by Aaronson [1] in his lecture notes. He shows that there exists a $\mathsf{poly}(n)$-time quantum algorithm A which makes $O(n)$ quantum queries to a classical oracle such that for every n-qubit state $|\psi\rangle = \sum_x \alpha_x |x\rangle$, there exists a classical oracle f for which the algorithm $A^{\mathcal{O}_f}$ will output a state that is $\mathsf{exp}(-n)$-close to $|\psi\rangle$. In [1], Aaronson raises the question as to whether his protocol can be improved to a sublinear number of queries. We show, in fact, that 1 query is sufficient to achieve polynomially small error in synthesizing arbitrary states and 2-queries are sufficient for exponentially small error. Both the 1-query and the 2-query algorithms given here require exponential time and polynomial space.

To understand Aaronson's state synthesis algorithm, we first observe that we can write any quantum state in the form

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} e^{i\theta_x} \sqrt{\mathbf{Pr}[X=x]}\, |x\rangle \tag{1}$$

where $\mathbf{Pr}[X = x]$ is the probability distribution of some n-bit random variable X and $\{\theta_x\}_{\{0,1\}^n}$ are a set of phases. The synthesis algorithm performs $2n$ queries to synthesize the "QSample state".

$$\sum_{x\in\{0,1\}^n} \sqrt{\mathbf{Pr}[X = x]}\ |x\rangle \tag{2}$$

and then performs two additional queries at the end to apply the phases $e^{i\theta_x}$ to each basis state $|x\rangle$.

The $2n$-query procedure to build the QSample state works in n stages. Inductively assume that after the kth stage, for $k < n$, the intermediate state of the algorithm is the k-qubit state

$$\sum_{y\in\{0,1\}^k} \sqrt{\mathbf{Pr}[X_{\leq k} = y]}\ |y\rangle \tag{3}$$

where $\mathbf{Pr}[X_{\leq k} = y]$ denotes the marginal probability of the first k bits of X are equal to y. Controlled on the prefix $|y\rangle$ the algorithm queries the oracle f to obtain a (classical description of) the conditional probabilities $\mathbf{Pr}[X_{k+1} = 0 \mid X_{\leq k} = y]$ and $\mathbf{Pr}[X_{k+1} = 1 \mid X_{\leq k} = y]$, and prepares a $(k+1)$st qubit in the state

$$\sqrt{\mathbf{Pr}[X_{k+1} = 0 \mid X_{\leq k} = y]}\ |0\rangle + \sqrt{\mathbf{Pr}[X_{k+1} = 1 \mid X_{\leq k} = y]}\ |1\rangle\ . \tag{4}$$

The algorithm performs another query to f to uncompute the descriptions of the conditional probabilities. The resulting $k+1$ qubit state is then equal to

$$\sum_{y\in\{0,1\}^{k+1}} \sqrt{\mathbf{Pr}[X_{\leq k} = y_{\leq k}]} \cdot \sqrt{\mathbf{Pr}[X_{k+1} = y_{k+1} \mid X_{\leq k} = y_{\leq k}]}\ |y\rangle \tag{5}$$

$$= \sum_{y\in\{0,1\}^{k+1}} \sqrt{\mathbf{Pr}[X_{\leq k+1} = y]}\ |y\rangle \tag{6}$$

which maintains the desired invariant. After the nth stage, a similar process applies the phases $\{\theta_x\}$ to generate the output state. The approximations come in when the conditional probabilities and phases are specified with $\mathsf{poly}(n)$ bits of precision, which result in the final state being at most $\mathsf{exp}(-n)$ far from the ideal target state $|\psi\rangle$. With this $O(n)$-query state synthesis algorithm in mind, we now proceed to describe our results.

1.2 Our results

A one-query search-to-decision algorithm for QMA with a PP oracle. We show Section 2 that in the case of generating physically relevant states, i.e. solutions to QMA problems, such as the low-energy states of local Hamiltonians, that there exists a one-query search-to-decision algorithm using a PP oracle. While one would hope to find a search-to-decision reduction in which the oracle complexity is only QMA, PP is the smallest complexity class containing QMA for which we can construct an oracular algorithm for search problems. Furthermore, given our no-go result for QMA search-to-decision (see below), this may be the optimal search-to-decision algorithm.

▶ **Theorem 1** (QMA-search to PP-decision reduction)**.** *There exists a probabilistic polynomial time quantum algorithm making a single query to a* PP *phase oracle such that, given as input a* QMA *problem, either aborts or outputs a witness $|\phi\rangle$. The algorithm will succeed in outputting a witness (i.e. not abort) with all but inverse exponential (in the system size) probability.*

Table 1 Summary of past work and our results on upper bounds for search-to-decision reductions and state synthesis. The "complexity class" column refers to the complexity of the search problem (e.g. computing NP witnesses, or QMA witnesses). The other columns refer to the algorithmic results known for the specified number of queries; furthermore these are *quantum* queries performed by quantum algorithms in superposition.

Complexity class	1 query	2 queries	$O(n)$ queries
NP	NP **oracle,** $\Omega(n^{-1})$ **success probability**	←	NP oracle, classical queries (folklore)
QCMA	QCMA **oracle,** $\Omega(n^{-1})$ **success probability**	←	QCMA oracle, classical queries (folklore)
QMA	PP **oracle,** $1/\mathsf{poly}(n)$ **precision, Theorem 1**	← (Theorem 4 applies but is time-inefficient)	PP oracle, $1/\mathsf{exp}(n)$ precision [1]
$\mathsf{QMA}_{\mathsf{exp}}$ (= PSPACE)	PSPACE **oracle** $\Omega(1)$ **overlap, Theorem 1**	← (Theorem 4 applies but is time-inefficient)	PSPACE oracle, $1/\mathsf{exp}(n)$ precision [1]
Arbitrary states	**Arbitrary oracle,** $1/\mathsf{poly}(n)$ **precision, Theorem 3**	**Arbitrary oracle,** $1/\mathsf{exp}(n)$ **precision, 2 queries, Theorem 4**	Arbitrary oracle, $1/\mathsf{exp}(n)$ precision [1]

To start sketching the proof, it is fruitful to notice that a single oracle query $|x\rangle \overset{\mathcal{O}_f}{\mapsto} (-1)^{f(x)}|x\rangle$ for $x \in \{0,1\}^n$ potentially contains 2^n bits of information and a quantum state requires 2^n complex numbers to describe. Furthermore, the collection of 2^{2^n} states

$$|p_f\rangle \overset{\text{def}}{=} \mathcal{O}_f H^{\otimes n}|0^n\rangle = \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle \tag{7}$$

defined for any function $f : \{0,1\}^n \to \{0,1\}$ are a diverse set of states in the Hilbert space. These states, referred to as *phase states* henceforth, despite not forming an ϵ-net for $(\mathbb{C}^2)^{\otimes n}$, turn out to provide a good approximation for $(\mathbb{C}^2)^{\otimes n}$ when considering the Haar-random distribution[3]. It follows that if we wanted to synthesize the witness to a QMA-complete problem, such as a low-energy state $|\tau\rangle$ for a local Hamiltonian problem, it suffices to build phase state $|p_f\rangle$ with constant overlap with the low-energy subspace. Finding a state with constant overlap with the target state is sufficient because QMA is efficiently verifiable, and given a state with constant overlap with the low-energy subspace, it is possible to distill a low-energy state with constant probability (by performing an energy measurement). However, it is not necessarily the case that a low-energy state of QMA problem will have a good approximation by a phase state. To solve this issue, we prove that for any state $|\tau\rangle$, with high probability $C|\tau\rangle$ will have a good approximation by a phase state where C is a random

[3] Recall, the Haar-measure is the unique left- and right- invariant distribution over unitary matrices over $(\mathbb{C}^2)^{\otimes n}$ and the Haar-random distribution is the distribution over quantum states $U|0^n\rangle$ where U is sampled according to the Haar-measure.

Clifford unitary. Therefore, we can instead attempt to synthesize $C|\tau\rangle$ which is the result of Theorem 1. In particular, if we can synthesize a phase state $|p\rangle$ that has constant overlap with $C|\tau\rangle$, then $C^\dagger|p\rangle$ will have constant overlap with the target $|\tau\rangle$.

Furthermore, we show that, using a slight modification of the same algorithm, we can perform a somewhat weaker one-query search-to-decision reduction for $\mathsf{QMA}_{\mathsf{exp}}$ (see Theorem 2.4 of the full version [8] for details). Recall $\mathsf{QMA}_{\mathsf{exp}}$ is the class of non – deterministic quantum computations with only an inverse exponential gap between completeness and soundness and is known to equal PSPACE [7, 6]. Our algorithm prepares a witness state with constant overlap with a low-energy state with one query to a PSPACE oracle (note that here, we cannot efficiently amplify the overlap with an energy measurement due to the inverse-exponential energy gap). As a further observation, we also show that quantum query access to a classical oracle gives one-query search-to-decision reductions when the witness is classical: in particular, for QCMA and NP (see Theorem 2.7 of the full version [8] for details). The one-query algorithm preparing the witness first reduces QCMA to *unique* QCMA (UQCMA) using the Valiant-Vazirani reduction [4], and then uses the Bernstein-Vazirani algorithm to extract the unique polynomial length witness with a single query.

A no-go result for search-to-decision for QMA. The previous result shows that search-to-decision reductions for QMA are possible with a PP decision oracle. However, the optimal search-to-decision reduction for QMA is with a QMA decision oracle (rather than a stronger PP oracle). We provide evidence that this is unlikely to exist: we prove that there is a quantum oracle relative to which QMA search-to-decision is impossible. This stands in contrast to classes such as NP, MA, and QCMA, which all have efficient search to decision reductions, relative to any oracle.

More precisely, we show that there exists a *quantum* oracle $\mathcal{O}$ relative to which *all* efficient query algorithms fail to be a good search-to-decision reduction for $\mathsf{QMA}^{\mathcal{O}}$, the relativization of QMA. The oracle $\mathcal{O}$ is a reflection $\mathbb{I} - 2|\psi\rangle\langle\psi|$ about a Haar-random state $|\psi\rangle$; we rely on the concentration of measure phenomenon of the Haar measure to prove this oracle no-go result. We formalize this result in Section 3.

▶ **Theorem 2** (Oracle impossibility for QMA search-to-decision)**.** *There exists a quantum oracle $\mathcal{O}$ relative to which* all $\mathsf{poly}(n)$*-time query algorithms fail to be a good search-to-decision reduction for* $\mathsf{QMA}^{\mathcal{O}}$*.*

A one-query state synthesis algorithm with inverse polynomial error. We also investigate the query complexity of synthesizing an arbitrary state, in the same spirit as Aaronson's adaptive state synthesis algorithm outlined in Section 1.1. In particular, we show that that every state $|\tau\rangle$ can be encoded into a classical oracle f_τ such that by making one query to $|\tau\rangle$, a quantum algorithm can prepare $|\tau\rangle$ with inverse polynomial error. The space complexity of the synthesis algorithm is polynomial in n, the number of qubits in the target state $|\tau\rangle$, but the time complexity is exponential. The starting point for the 1-query algorithm is the same observation used in the protocol for synthesizing QMA witnesses, which is that a random state has an expected constant overlap with some phase state. We can think of the oracle function f_τ as hard-coding the target $|\tau\rangle$, but parameterized by unitary U and standard basis state x. The oracle $f_\tau(U, x) = \mathrm{sgn}(\mathrm{Re}(\langle x|U\tau\rangle))$ can be used to create a phase state $|p_U\rangle$ which has constant overlap with $U|\tau\rangle$ with high probability for random U. The state $U^\dagger|p_U\rangle$ is already then a decent approximation for $|\tau\rangle$.

There are two remaining techniques to improve upon this basic synthesis protocol. First, we use a novel distillation procedure based on the swap test (explained below) to take a polynomial number of states generated in this manner, using unitaries $U_1, \ldots, , U_m$, to

create a single aggregated output state with greater overlap with the target state. Note that since the target state $|\tau\rangle$ is arbitrary, we do not have a means of measuring the overlap of an output state with $|\tau\rangle$ to boost the overlap as we did when the target state is a QMA witness. Secondly, we address the fact that the algorithm described above suffers from needing *exponential space complexity*; this is because specifying a Haar-random unitary on n qubits requires $\mathsf{exp}(\Omega(n))$ space, and thus the oracle $f_\tau(U, x)$ needs to act on exponentially many input bits. We derandomize this construction, and show via the probabilistic method that there exists a *single* choice of unitaries $U_{\star,1}, \ldots, , U_{\star,m}$ that works for *all* n-qubit states. This will reduce the space complexity of the algorithm to polynomial, although implementing the unitaries will still require exponential time.

▶ **Theorem 3** (One Query State Synthesis – Informal). *There is a* 1*-query algorithm that uses polynomial space and exponential time that synthesizes a state* ρ *such that* $\mathrm{Tr}\{\rho\,|\tau\rangle\langle\tau|\} \geq 1 - 1/q(n)$ *for some polynomial* q *and an arbitrary target state* $|\tau\rangle$*.*

The Swap Test Distillation Algorithm. This procedure takes in a polynomial number of states each of which has at least a constant overlap with the target state and outputs a state whose overlap with the target is at least $1 - 1/\mathrm{poly}$. In some sense, the Swap Test Distillation algorithm provides a way to take the "mean" of a collection of quantum samples where each state can be decomposed into a "signal" component and a "noise" component such that (1) the signal is some constant fraction of the mass and (2) the noise is roughly random. This may be useful in other contexts in quantum algorithms.

For formally, the algorithm requires that the sequence of input states $|\psi_1\rangle, \ldots, |\psi_m\rangle$ satisfies two properties. The first is that there is a constant a such that $|\langle\psi_j|\tau\rangle|^2 \geq a$ for all j. (We also show that this condition can be relaxed so that the expected overlap of each input state with the target state is at least a, as long as the input states are independently generated.) The second condition is that for every pair of input states, their components orthogonal to $|\tau\rangle$ are close to orthogonal to each other:

$$|\langle\psi_j|\,(\mathbb{I} - |\tau\rangle\langle\tau|)\,|\psi_i\rangle|^2 \leq \delta, \tag{8}$$

for δ exponentially small in n. Intuitively, one can imagine that if the $|\psi_j\rangle$ are generated independently, then the error vectors (the components perpendicular to $|\tau\rangle$) would be random and uncorrelated. We prove that under these two conditions, if the number of states is a sufficiently large polynomial, then the overlap of the resulting aggregated state with $|\tau\rangle$ is at least $1 - 1/\mathrm{poly}$. The algorithm is based on the observation that if the swap test is applied to a pair of states which each have overlap at least a with the target state, then conditioning on the swap test succeeding (measuring a 0 in the output bit), the state in each register has an overlap with the target state that is strictly larger than a. In each round of the algorithm, the surviving states are paired up and the swap test is applied to each pair. One state from every pair that succeeds the swap test advances to the next round.

A two-query state synthesis algorithm with inverse exponential error. While we do not know how to improve the error of the previous one-query algorithm beyond inverse polynomial, we show that there is a *two-query* state synthesis algorithm that achieves inverse exponential error.

▶ **Theorem 4** (Two Query State Synthesis – Informal). *There is a* 2*-query algorithm that uses polynomial space and exponential time that with high probability synthesizes a state* ρ *such that* $\mathrm{Tr}\{\rho\,|\tau\rangle\langle\tau|\} \geq 1 - 1/r(n)$ *for some function* $r = \mathsf{exp}(n)$ *and an arbitrary target state* $|\tau\rangle$*.*

Like with the one-query synthesis algorithm, we take advantage of the properties of Haar-random unitaries. Let $|\tau\rangle$ denote the target state to be synthesized. Whereas the basic building block of the one-query algorithm described is to synthesize the phase state corresponding to $U|\tau\rangle$ where U is a Haar-random unitary, the two-query algorithm attempts to directly synthesize the state $U|\tau\rangle$, and then apply the inverse unitary $U^\dagger$ to recover $|\tau\rangle$. Since U is Haar-random, the distribution of $U|\tau\rangle$ is that of a Haar-random state.

We then argue that with overwhelmingly high probability, a Haar-random state can be synthesized via two queries to a classical oracle. This relies on the observation that the *amplitude profile* of a Haar-random state concentrates extremely tightly around a fixed profile. By profile, we mean the list of absolute values of amplitudes of the state in sorted order. In other words, there exists a fixed, universal state $|\theta\rangle = \sum_x \beta_x |x\rangle$ such that, with very high probability, a Haar-random state $|\psi\rangle = \sum_x \alpha_x |x\rangle$ satisfies the following: there exists a permutation σ on the set of basis states $|x\rangle$ such that the distance

$$\left\| |\theta\rangle - \sum_x |\alpha_x| \, |\sigma(x)\rangle \right\| \tag{9}$$

is exponentially small. To prove this, we utilize bounds from the theory of optimal transport that control the convergence of the *Wasserstein distance* (also known as the *Earth Mover Distance*) between a log-concave distribution and the empirical distribution resulting from sampling from the distribution.

Given this, the two-query algorithm to synthesize $|\tau\rangle$ to exponential precision is clear: the algorithm first prepares the universal state $|\theta\rangle$. It then queries the classical oracle to determine how to permute the basis states $|x\rangle$ and what phase to apply to all the basis states. The algorithm applies the permutation and the phases in superposition. Finally, the algorithm queries the oracle again to uncompute the permutation/phase information.

Just as with the one-query algorithm, we also perform a derandomization step in order to make the query algorithm space-efficient (but not necessarily time-efficient). By expanding the dimension of the random unitary U, we show that there exists (via the probabilistic method) a *single* unitary $U_\star$ that maps *every* target state $|\tau\rangle$ to one whose amplitude profile is exponentially close to the universal one.

Open Questions. We conclude with some open questions which are elaborated in greater detail in Section 7. Can the 1- and 2-query algorithms for general state synthesis be improved to polynomial time by using random Cliffords instead of Haar-random unitaries? Is there a 1-query algorithm for state synthesis that also achieves inverse exponential error? What is the power of a QMA decision oracle? In particular, what states can be synthesized with queries to a QMA oracle in superposition? Is there a weaker oracle class than PP that can achieve search-to-decision for QMA witnesses?

Preliminaries. Preliminaries and definitions necessary are listed in Appendix A of the full version [8].

2 Search-to-decision for QMA problems

In the traditional search-to-decision paradigm, the goal is to create a witness $|\psi\rangle$ which could convince a verifier that indeed some string is in a particular QMA language. The creation of this witness should be carried out by a quantum machine running in polynomial time with access to a QMA oracle. There are multiple ways to relax this paradigm; here we consider

using a PP oracle instead of a QMA oracle and show that there is a polynomial time quantum algorithm which makes only one PP oracle call[4] and generates a solution to a QMA-complete problem.

Our algorithm proceeds from two observations:

1. Any phase state $|p_f\rangle = 2^{-n/2}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle$ for which the function f is computable in PP may be prepared by a single quantum query to a PP oracle.
2. *Any* state $|\tau\rangle$, after applying a random unitary U, looks like a phase state: in particular, with high probability over the choice of U, the state $U|\tau\rangle$ has constant overlap with some phase state.

2.1 One-query search-to-decision for QMA

We now consider the QMA search problem with respect to phase oracles. In general, the statement of the QMA search problem is to construct, given a verification circuit for some QMA language, and an input χ in the language, a state $|\psi_\chi\rangle$ that is accepted by the verification circuit with high probability. Rather than working with general verifiers, we will restrict to verifiers that measure the energy of a local Hamiltonian on the witness state up to inverse-polynomial precision. This restriction is almost without loss of generality, for two reasons. First, the local Hamiltonian problem with this precision is QMA-complete, so any QMA language has a verifier of this form. And secondly, the reduction to local Hamiltonian can be performed so that every low-energy state is very close to an accepting witness $|\psi\rangle$ for the original verifier. More precisely, given a general QMA verification circuit V, we can apply the padding trick of Nirkhe, Vazirani and Yuen [13] to generate a local Hamiltonian instance H such that any ground-state ρ of H, $\|\rho - \sigma\otimes\Phi\| \leq \delta$ where σ is an accepting witness of V and Φ is a fixed state independent of the instance. The size of the Hamiltonian instance H scales as $\mathsf{poly}(1/\delta)$ and therefore the approximation can be chosen as any inverse polynomial function of the system size.

Assume the input to the problem is an instance $\chi = (H, a, b)$ of the Local Hamiltonian problem with Hamiltonian H on n qubits and two thresholds $a < b$ such that $b - a = 1/\mathsf{poly}(n)$. Moreover, we assume that χ is a YES instance, so the minimum eigenvalue of H is at most a: $\lambda_{\min}(H) \leq a$. The goal is to construct a state $|\phi\rangle$ such that

$$\langle\phi|H|\phi\rangle \leq \frac{a+b}{2}. \tag{10}$$

While it would be ideal to construct a state for which the energy is at most a (since one exists), this may drastically increase the computational complexity of the function $f : \{0,1\}^n \to \{0,1\}$ defining the oracle. Instead, due to the promise gap in the problem, it suffices to construct a *witness* state which proves that the Hamiltonian has a state with energy at most $< b$. A state $|\phi\rangle$ satisfying eq. (10) is a proof that χ is a yes instance. We now prove the formal version of Theorem 1.

▶ **Theorem 5** (QMA-search to PP-decision reduction). *There exists a probabilistic polynomial time quantum algorithm with access to a single* PP *phase oracle query that, given as input an instance (H, a, b) of the local Hamiltonian problem on n qubits, either aborts or outputs a witness $|\phi\rangle$ with $\langle\phi|H|\phi\rangle \leq (a+b)/2$ for $b - a = 1/\mathsf{poly}(n)$. The algorithm will succeed in outputting a witness (i.e. not abort) with probability[5] $\geq 1/1024$.*

[4] The improvement over the algorithm of Aaronson [1] is in the number of oracle queries.

[5] We will later argue that this probability can be amplified through a variation of parallel repetition to improve to any function $1 - \mathsf{exp}(-n)$.

See Theorem 2.1 in the full version [8] for proof.

We remark that we see that the algorithm achieves something stronger: if the algorithm does not abort, then the output state is almost entirely supported on states of energy less than $a + (b-a)/4 + \epsilon$, where $\epsilon = 1/\mathsf{poly}(n)$ is a precision parameter much smaller than $b-a$. This is performed by using phase estimation to "check" the outcome of the query algorithm by measuring the energy.

At this point, it is useful to remember that in general, the notion of a QMA *witness* is defined only with reference to a specific verifier. The guarantee we achieve ensures that the "standard" verifier, which measures the energy of the local Hamiltonian H, has a high chance of accepting the given state. If one is willing to use a more sophisticated verifier, e.g. a verifier that performs the Marriott-Watrous amplification procedure [12], a witness of considerably worse quality could still be acceptable. Our theorem also sidesteps the issue of unique witnesses: we only guarantee that the energy of our state is low, not that it is the *unique* such state.

One can easily boost probability that our algorithm does not abort to $1 - \mathsf{exp}(-n)$ by repeating the construction in parallel with independent randomness and selecting any witness which did not cause the algorithm to abort. Furthermore, from the design of the algorithm, one can merge the oracle queries into a single larger PP query[6], so the query complexity does not increase.

Additional remarks. In Section 2.1 of the full version [8], we also provide remarks on why it is difficult to improve the oracle complexity as well as how to extend this argument for the class $\mathsf{QMA}_{\mathsf{exp}} = \mathsf{PSPACE}$ (see Section 2.2 of the full version [8]). Furthermore, Section 2.3 of the full version [8] includes a completely different procedure, which generates one oracle query search-to-decision for QCMA.

3 Impossibility of search-to-decision for QMA in oracle model

In this section we show that efficient search-to-decision reductions for QMA do not exist in general in the oracle setting, perhaps providing some evidence that QMA does not have efficient search-to-decision reductions "in the real world." More precisely, we show that there exists a *quantum* oracle $\mathcal{O}$ relative to which *all* polynomial-time quantum query algorithms fail to be a good search-to-decision reduction for $\mathsf{QMA}^{\mathcal{O}}$, the relativization of QMA. Equivalently, $\mathsf{QMA}^{\mathcal{O}}$-search problems are not reducible to $\mathsf{QMA}^{\mathcal{O}}$-decision problems. We contrast this impossibility result with the fact that complexity classes like NP, MA and QCMA all have efficient search-to-decision reductions, relative to any oracle (i.e. the reductions relativize)! For example, it is not hard to verify that the search-to-decision procedure for QCMA described in Section 2.3 of the full version [8] relativizes. Thus, Theorem 6 illustrates that, at least in the relativized setting, changing the proof model from classical to quantum nullifies the possibility of search-to-decision reductions.

We first define $\mathsf{QMA}^{\mathcal{O}}$ by way of a complete problem. Fix a small constant $\delta < \frac{1}{100}$. Define an $\mathcal{O}$-*verifier* circuit C to be a quantum circuit that can make queries to $\mathcal{O}$ (which can be viewed as applying a unitary gate for $\mathcal{O}$), and also takes as input a quantum proof state $|\phi\rangle$, as well as some ancilla qubits set to $|0\rangle$. Define the promise problem $\mathcal{O}$-QCircuitSAT

[6] One way to see that the merged oracle is also definable in PP is through the connection PP = PostBQP. The merged oracle can be seen as the logical exclusive-or (XOR) of multiple PP functions, and it is easy to create a new PostBQP function equal to the logical XOR of multiple PostBQP functions.

whose YES instances consist of $\mathcal{O}$-verifier circuits C for which there is a quantum proof state $|\phi\rangle$ such that $C(|\phi\rangle)$ accepts with probability at least $1-\delta$, and the NO instances are those circuits such that on all quantum witness states, C accepts with probability at most δ. Without access to $\mathcal{O}$, this is simply the canonical QMA-complete problem QCircuitSAT. The class $\mathsf{QMA}^{\mathcal{O}}$ is then the set of all promise decision problems that are polynomial-time reducible to $\mathcal{O}$-QCircuitSAT.

Now we formalize the notion of search-to-decision reductions for $\mathsf{QMA}^{\mathcal{O}}$. Consider quantum circuits that can make queries in superposition to both the quantum oracle $\mathcal{O}$ and a *classical* oracle $A^{\mathcal{O}}$ that decides the promise problem $\mathcal{O}$-QCircuitSAT as well as the controlled-versions of these oracles. Alternatively, we can consider a standard quantum circuit with special oracle "gates" implementing $\mathcal{O}$ and $A^{\mathcal{O}}$ unitary transformations. Specifically, the oracle $A^{\mathcal{O}}$ implements the unitary transformation

$$|C\rangle\,|b\rangle \mapsto |C\rangle\ |b \oplus A^{\mathcal{O}}(C)\rangle \tag{11}$$

where C is supposed to be a description of an $\mathcal{O}$-oracle circuit, $b \in \{0,1\}$, $A^{\mathcal{O}}(C) \in \{0,1\}$ with $A^{\mathcal{O}}(C) = 1$ if C is a YES instance of $\mathcal{O}$-QCircuitSAT, $A^{\mathcal{O}}(C) = 0$ if C is a NO instance, and otherwise $A^{\mathcal{O}}(C)$ is defined arbitrarily. This is sufficiently general as we previously remarked that all $\mathsf{QMA}^{\mathcal{O}}$ problems can be expressed as $\mathcal{O}$-oracle circuits C. We then say that such a quantum circuit S is an *ϵ-good search-to-decision reduction* for the problem $\mathcal{O}$-QCircuitSAT – or, alternatively, *ϵ-solves the search version* of $\mathcal{O}$-QCircuitSAT – if when given a YES instance C of $\mathcal{O}$-QCircuitSAT, it outputs a state that is accepted by C with probability at least $1-\delta-\epsilon$.

We now state the main result of this section (the technical version of Theorem 2).

▶ **Theorem 6.** *There exists a constant $\epsilon > 0$ and a quantum oracle $\mathcal{O}$ relative to which there is no $\mathsf{poly}(n)$-sized ϵ-good search-to-decision reduction for $\mathcal{O}$-QCircuitSAT.*[7]

See Theorem 3.1 in the full version [8] for proof.

4 1-query state synthesis algorithm with polynomially small error

We describe here a 1-query, polynomial-space algorithm that achieves polynomially small error. The state synthesis algorithm will not be efficient. We will start with a first attempt, which has exponential space complexity and then fix it so that it has polynomial space complexity. The algorithm makes use of the Swap Test Distillation algorithm described in 5 that takes as input a polynomial number of states, each with at least constant overlap with the target state, and uses successive applications of the Swap Test to produce a final state whose overlap with the target state is at least $1 - 1/\mathsf{poly}(n)$.

4.1 A space-inefficient algorithm

Let $d = 2^n$, $n' = n^2$, and $d' = 2^{n'}$. The m applications of the 1-query algorithm along with the Swap Test Distillation algorithm will be applied to n'-qubit registers with target state $|\tau'\rangle = |\tau\rangle \otimes |0\rangle^{\otimes(n'-n)}$. The expansion of the space is important for derandomizing the

[7] Technically, we should be considering an infinite family of oracles $\mathcal{O}$ and $A^{\mathcal{O}}$ where each oracle is parameterized by some input length n. However for simplicity we shall just deal with one input length; we will forgo the trouble of spelling out the details of stating our results for asymptotic n. To that end, let $\mathcal{O}$ be a unitary that acts on n qubits, and we only consider $\mathcal{O}$-verifier circuits who accept n-qubit quantum proof states; the verifier circuits themselves will be of size $\mathsf{poly}(n)$.

algorithm later on. In particular, we will show that there is a fixed sequence of unitaries that works for all $|\tau'\rangle$ of the form $|\tau\rangle \otimes |0\rangle^{\otimes(n'-n)}$. This will allow us to hard-code the unitaries into the oracle function. The resulting algorithm will still require exponential time to implement the unitaries, but the derandomized algorithm will require only polynomial space.

We will define a function $f_{\tau'} : \mathrm{U}(d') \times \{0,1\}^{n'} \to \{0,1\}$, where $\mathrm{U}(d')$ is the space of all unitaries on a d'-dimensional Hilbert space, and

$$f_{\tau'}(U,x) \stackrel{\text{def}}{=} \operatorname{sgn}\left(\operatorname{Re}\left(\langle x| U |\tau'\rangle\right)\right) \tag{12}$$

The corresponding phase state is

$$|p_U\rangle = \sum_{x\in\{0,1\}^{n'}} (-1)^{f_{\tau'}(U,x)} |x\rangle \tag{13}$$

OneQueryStateSynthesis (*space inefficient version*)
(1) **for** $j = 1,\ldots,m$ in parallel:
(2) Sample Haar-random n'-qubit unitary U_j.
(3) In the jth n'-qubit register, prepare the equal superposition $\sum_{x\in\{0,1\}^{n'}} |x\rangle$.
(3) Controlled on basis state $|x\rangle$, query the oracle on input (U_j, x) to apply $f_{\tau'}(U_j, x)$ and produce phase state $|p_{U_j}\rangle$ on n' qubits.
(4) Apply $U_j^\dagger$ to the phase state.
(5) Apply the SwapTestDistillation Algorithm (Figure 2) to the m resulting states $|\psi_1\rangle, \ldots, |\psi_m\rangle$.
(6) Output the first n qubits of any surviving register.

Figure 1 Pseudo-code for the OneQueryStateSynthesis query algorithm that uses exponential space complexity.

The algorithm will output m expanded registers on n' qubits. We will apply the Swap Test Distillation algorithm to m states on n' qubits generated by m parallel (and independent) applications of the 1-query algorithm and analyze the probability that the resulting state has at least $1 - 1/\mathsf{poly}(n)$ overlap with $|\tau'\rangle$. The mixed state ρ in the first n-qubits will also have $\mathrm{Tr}\{\rho\,|\tau\rangle\langle\tau|\} \geq 1 - 1/\mathsf{poly}(n)$.

The following lemma establishes that with high probability after step (4) of the algorithm, the conditions for the Swap Test Distillation Algorithm are met.

▶ **Lemma 7** (Probability Conditions Satisfied for Swap Test Distillation). *Let $|\psi_1\rangle \otimes \cdots \otimes |\psi_m\rangle$ be the states in the m registers after Step (4). There is a constant C such that*
1. $\mathbf{Pr}_{U_1,\cdots,U_m}\left[\min_j\{|\langle\psi_j|\tau'\rangle|^2\} \leq 1/8\right] \leq m \cdot \mathsf{exp}(-Cd')$
2. $\mathbf{Pr}_{U_1,\cdots,U_m}\left[\max_{i\neq j}\{|\langle\psi_i|\,(I - |\tau'\rangle\langle\tau'|)\,|\psi_j\rangle|^2\} \geq (d')^{-1/4}\right] \leq m^2 \cdot \mathsf{exp}(-C(d')^{1/2})$

See Lemma 4.1 in the full version [8] for proof.

4.2 A space-efficient algorithm

The algorithm described above suffers from needing *exponential space complexity*; this is because specifying a Haar-random unitary on n' qubits requires $\mathsf{exp}(\Omega(n'))$ space, and thus the oracle $f(U,x)$ needs to act on exponentially many input bits. We derandomize this construction, and show via the probabilistic method that there exists a *single* choice of

unitaries $U_{\star,1}, \ldots, U_{\star,m}$ that works for *all* n-qubit states – this is why we expanded the space to dimension d'.

Let $|v_1\rangle, \ldots, |v_D\rangle$ denote an ϵ-net for the space of n-qubit quantum states where $\epsilon = d^{-1}$. Then there at most $D \leq \epsilon^{-d} = d^d$ states in this enumeration. Fix an index $1 \leq i \leq D$. Imagine running the 1-query protocol in Figure 1 in parallel m times with target state $|v_i\rangle \otimes |0\rangle$. The probability that the protocol fails to satisfy the conditions for the Swap Test Distillation algorithm from Lemma 7 is at most $2m^2\mathsf{exp}(-\Omega((d')^{3/4}))$ over the choice of $U_1, \ldots, U_m$. By a union bound, the probability that a random choice of $U_1, \ldots, U_m$ fails to satisfy the conditions from Lemma 7 for *a single one* of the $|v_1\rangle, \ldots, |v_D\rangle$ is at most

$$d^d \cdot 2m^2\mathsf{exp}(-\Omega((d')^{1/2})) \leq 2^{n2^n} \cdot 2m^2\mathsf{exp}(-\Omega(2^{n^2/2})). \tag{14}$$

Since m is polynomial in n, for sufficiently large n, this probability is less than 1. Thus there exists a choice of unitaries $U_{\star,1}, \ldots, U_{\star,m}$ that results in a set of m states that satisfy the conditions for the Swap Test Distillation algorithm for *all* the $|v_1\rangle, \ldots, |v_D\rangle$. Hardcode these unitaries into the algorithm and oracles: $f_{\tau,i}(x) = f_\tau(U_{\star,i}, x)$. Now the oracles only take n' bits as input each, and the resulting query algorithm now only requires $\mathsf{poly}(n)$ space. Note that the implementation of the unitaries $U_{\star,1}, \ldots, U_{\star,m}$ will not be time-efficient in general, but they are still fixed unitary operators that act on n' qubits.

Thus for an arbitrary target state $|\tau\rangle$, use the oracles $f_{v_i}(U_{\star,1}, x), \ldots, f_{v_i}(U_{\star,m}, x)$ corresponding to the nearest state $|v_i\rangle$ in the ϵ-net, which is within d^{-1} of $|\tau\rangle$. Therefore, the one-query algorithm using unitaries $U_{\star,1}, \ldots, U_{\star,m}$, followed by the Swap Distillation Algorithm will incur an additional $O(d^{-1})$ error.

▶ **Theorem 8** (One Query State Synthesis Performance). *For every polynomial q, there is a polynomial p and constant C' such that if the* OneQueryStateSynthesis *is run with $m \geq p(n)$ registers, then with probability at least $1 - \mathsf{exp}(-C'n)$, the algorithm produces a state ρ such that $\mathrm{Tr}\{\rho |\tau\rangle\langle\tau|\} \geq 1 - 1/q(n)$. The oracle queried by the algorithm will depend on the closest state to $|\tau\rangle$ in the ϵ-net.*

See Theorem 4.2 in the full version [8] for proof.

5 Swap test distillation procedure

If a synthesis protocol is able to produce a state with at least constant overlap with the target state and the target state is a witness for a QMA verifier, then phase estimation can be used to boost the overlap and the probability of success. If the target state is an arbitrary state, we may not have the means to directly measure whether the output state is close to the target. In this section we describe a procedure that can take the output of m parallel applications of a state synthesis protocol, each of which has a constant overlap with the target state and apply a procedure to increase the overlap. The algorithm begins with m states, $|\psi_1\rangle, \ldots |\psi_m\rangle$, each of which is stored in an n-qubit register. We show that if the number of states m is a sufficiently large polynomial in n, then the overlap of the final output state will be at least $1 - 1/\mathsf{poly}(n)$, with high probability. The distillation process is based on the Swap Test and works subject to two conditions on the collection of input states:

1. For all j, $|\langle\psi_j|\tau\rangle|^2 \geq a$, for some constant a.
2. For all $i \neq j$ $|\langle\phi_j|\phi_i\rangle|^2 \leq \delta$, where δ is exponentially small in n and for all j

$$|\phi_j\rangle \stackrel{\text{def}}{=} \frac{|\psi_j\rangle - \langle\psi_j|\tau\rangle|\tau\rangle}{\||\psi_j\rangle - \langle\psi_j|\tau\rangle|\tau\rangle\|}. \tag{15}$$

The second condition is satisfied if the portion of each state $|\psi_j\rangle$ that is perpendicular to the target state $|\tau\rangle$ is essentially random. If the $|\psi_j\rangle$'s are generated according to some independent randomness, one might expect that the overlap between these perpendicular components to be (exponentially) small. In this section, we analyze the behavior of the Swap Test distillation procedure subject to these two properties. At the end of the section, we will show that the first condition can be relaxed to a lower bound on the expectation of the overlap as long as the m states are generated according to some independent randomness. In Section 4, we showed how the algorithm can be used in conjunction with a 1-query protocol to produce a state that has $1 - 1/\mathsf{poly}(n)$ overlap with the target state.

The Algorithm

Each round of the algorithm begins with some set of surviving registers. The surviving registers are paired up and the swap test is applied to each pair. An auxiliary qubit is used in each application of the swap test which is measured at the end of the swap test. If the outcome is 0 (a *successful* outcome), then one of the two registers is selected to survive to the next iteration. If the outcome is 1 (an *unsuccessful* outcome), neither register survives. Figure 2 shows the pseudocode for the procedure. Figure 3 shows an example of the procedure for one iteration applied to eight input states. Note that the state in a surviving register may be entangled with the other registers. If ρ is the reduced density matrix of the state in one of the surviving registers obtained by tracing out the other registers, we will refer to $\mathrm{Tr}\{\rho\,|\tau\rangle\langle\tau|\}$ as the *overlap* of ρ with $|\tau\rangle$. We will show that for m sufficiently large, with high probability the overlap of a surviving register with $|\tau\rangle$ is at least $1 - 1/\mathsf{poly}(n)$.

Consider one round of the algorithm applied to a particular pair of registers. We will prove that if the swap test succeeds, then the surviving register has an overlap with $|\tau\rangle$ that is at least the average of the overlap of the states in the two registers before the round. Moreover, if each of the two registers at the beginning of a round have overlap at least γ, then the overlap of a surviving register is strictly larger than γ and with enough successful rounds will tend towards 1.

The analysis of the swap test distillation is provided in Section 5.1-2 of the full version [8].

6 2-query state synthesis algorithm with exponentially small error

We now describe a 2-query state synthesis algorithm that achieves *exponentially small* error. Like with the 1-query algorithm from Section 4, it will be space-efficient, but not time-efficient. And also like in Section 4, we first describe a version of the algorithm with exponential space complexity, and then describe how to reduce the space complexity to polynomial.

6.1 A space-inefficient algorithm

Let $d = 2^n$, $n' = n^2$ and $d' = 2^{n'}$. Let $\{\sigma_{U,V}\}_{U,V}$ denote a set of permutations on $\{0,1\}^{n'}$, indexed by unitaries U, V on n' qubits. For all unitaries U, V and n'-bit strings x, let $\phi(U, V, x)$ denote a number in $[0, 2\pi)$, representable using $(n')^2$ bits. We will specify $\{\sigma_{U,V}\}$ and ϕ later. Define the oracles

$$f(U,V,x) \stackrel{\text{def}}{=} (\phi(U,V,x), \sigma_{U,V}(x)) \quad \text{and} \quad g(U,V,y) = (\sigma_{U,V}^{-1}(y), \phi(U,V,\sigma_{U,V}^{-1}(y))) \,. \tag{16}$$

These oracles have the property that if $f(U, V, x) = (\phi, y)$, then $g(U, V, y) = (x, \phi)$.

An analysis of the correctness of the algorithm is provided in Section 6.1 of the full version [8].

SwapTestDistillation
Input: m states $|\psi_1\rangle, \ldots |\psi_m\rangle$ stored in n-qubit registers numbered 1 through m
(1) Initialize $(R_1, \ldots, R_m) \leftarrow (1, \ldots, m)$
(2) $\ell = \lfloor \log_6(m/n) \rfloor$
(3) **for** $k = 1, \ldots, \ell$:
(4) count $= 0$
(5) **for** $j = 1, \ldots, \lfloor m/2 \rfloor$
(6) **if** SwapTest(R_{2j-1}, R_{2j}) returns 0
(7) count = count +1
(8) $R_{count} = R_{2j-1}$
(9) **end**
(10) $m =$ count
(11) **end**

SwapTest(R, R')
Start with auxiliary qubit b initialized to $|0\rangle$
(1) Apply:
(2) $H_b \otimes I_{R,R'}$
(3) Controlled SWAP operation on Registers R and R', controlled by qubit b
(4) $H_b \otimes I_{R,R'}$
(5) Measure qubit b and return the result
(6) **end**

Figure 2 Pseudo-code for SwapTestDistillation algorithm.

6.2 A space-efficient algorithm

The algorithm described above suffers from needing *exponential space complexity*; this is because specifying a Haar-random unitary on n' qubits requires $\mathsf{exp}(\Omega(n'))$ space, and thus the oracles f, g need to act on exponentially many input bits. We derandomize this construction, and show via the probabilistic method that there exists a *single* choice of unitary $U_\star, V_\star$ that works for *all* n-qubit states – this is why we expanded the space to dimension d'.

Let $|\psi_1\rangle, \ldots, |\psi_D\rangle$ denote an ϵ-net for the space of n-qubit quantum states. Set $\epsilon = d^{-1}$. Then there at most $D \leq \epsilon^{-d} = d^d$ states in this enumeration. Fix an index $1 \leq i \leq D$. Imagine running the aforementioned protocol on state $|\psi_i\rangle$. As discussed, the probability that the protocol fails to synthesize a $(d')^{-1/4}$-approximation to $|\psi_i\rangle$ is at most $\mathsf{exp}(-\Omega((d')^{1/4}))$ over the choice of U, V. By a union bound, the probability that a random choice of U, V fails to synthesize a $(d')^{-1/4}$-approximation to *a single one* of the $|\psi_1\rangle, \ldots, |\psi_D\rangle$ is at most

$$d^d \cdot \mathsf{exp}(-\Omega((d')^{1/4})) \leq 2^{n2^n} \cdot \mathsf{exp}(-\Omega(2^{n^2/4})) \tag{17}$$

which, for sufficiently large n, is less than 1. Thus there exists a choice of unitaries $U_\star, V_\star$ that enables successful synthesis of *all* the $|\psi_1\rangle, \ldots, |\psi_D\rangle$. Hardcode these unitaries into the algorithm and oracles in Figure 4; i.e., the oracles f and g only take n' bits as input. The resulting query algorithm now only requires $\mathsf{poly}(n)$ space. Note that the implementation of the unitaries $U_\star, V_\star$ will not be time-efficient in general, but they are still fixed n'-qubit unitary operators that are independent of the state being synthesized.

Thus for an arbitrary state $|\psi\rangle$, by letting the oracles A, B correspond to the nearest state $|\psi_i\rangle$ in the ϵ-net that is within d^{-1} of $|\psi\rangle$, the two-query algorithm synthesizes $|\psi\rangle$ with $O(d^{-1})$ error.

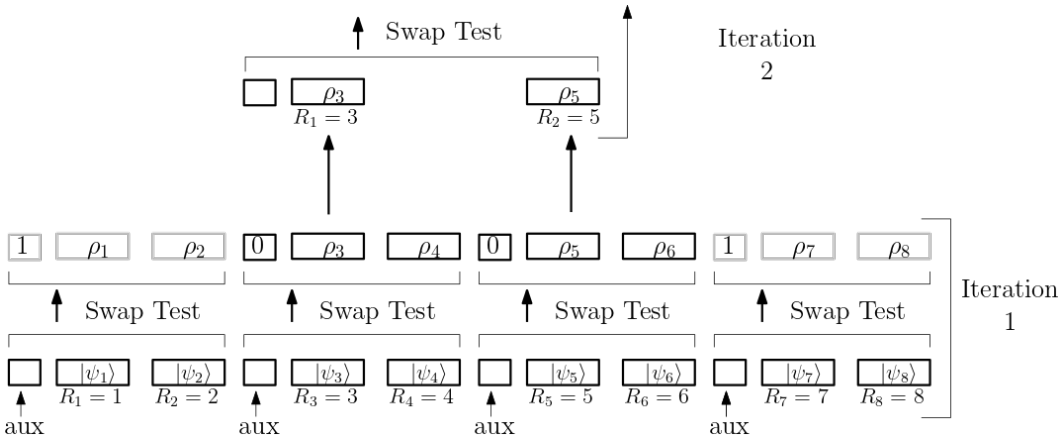


Figure 3 The Swap Test Distillation algorithm applied to eight input registers. The value measured in the auxiliary qubit indicates whether an application of the Swap Test is successful. In the first iteration, the swap tests applied to pairs $|\psi_3\rangle, |\psi_4\rangle$ and $|\psi_5\rangle, |\psi_6\rangle$ are successful. The resulting states in registers 3 and 5 advance to the next round. In each iteration, the sequence $(R_1, R_2, \ldots)$ indicates the indices of the surviving registers from left to right.

TwoQueryStateSynthesis (*space inefficient version*)
(1) Sample Haar-random n'-qubit unitaries U, V.
(2) Prepare the state $|\theta\rangle = U|0\rangle^{\otimes n'}$ in an n'-qubit register A.
(3) Controlled on basis state $|x\rangle$ in register A, call the oracle f on input (U, V, x) to obtain $\phi \in [0, 2\pi)$ in register B and $y \in \{0,1\}^{n'}$ in register C.
(4) Controlled on basis state $|\phi\rangle$ in register B, apply the phase $e^{i\phi}$.
(5) Controlled on basis state $|y\rangle$ in register C, call the oracle g on input (U, V, y) to uncompute $|x\rangle \otimes |\phi\rangle$ in registers A and B.
(6) Apply the inverse unitary $V^\dagger$ on register C.
(7) Output the first n qubits of register C.

Figure 4 Pseudo-code for the TwoQueryStateSynthesis query algorithm that uses exponential space complexity.

▶ **Theorem 9** (Two Query State Synthesis Performance). *For all n-qubit states $|\tau\rangle$, the algorithm* TwoQueryStateSynthesis *uses* poly(n) *space, makes two queries to a classical oracle depending on $|\tau\rangle$, and outputs a mixed state that is* exp$(-\Omega(n))$*-close in trace distance to $|\tau\rangle\langle\tau|$.*

7 Open Questions

We exhibited state synthesis algorithms for QMA witnesses and arbitrary states that only require a *single* query to a classical oracle, that generate the target state up to inverse polynomial error. We also presented a *two*-query state synthesis algorithm that generates the target state up to inverse *exponential* error. As mentioned, this resolves Open Question 3.3.6 of Aaronson [1]. However, there are several remaining open questions regarding these algorithms.

1. The one- and two-query algorithms for arbitrary states use polynomial space, but they aren't time efficient (because their existence is argued by sampling a Haar-random unitary and applying the probabilistic method). Can this probabilistic construction be derandomized (and thus be made time efficient) by using (approximate) unitary designs?
2. Is there a one-query algorithm for state synthesis that also achieves inverse exponential error?

7.1 The power of quantum queries to QMA oracles

Our impossibility result in Section 3 combined with our reduction of QMA-search to PP-decision problems leaves an interesting gap as to what exactly is the power of QMA oracle. More specifically, are there interesting computational tasks solvable only with quantum access to a QMA oracle? One question is to understand the collection of problems which have search-to-decision reductions where the oracle is a QMA oracle. Is this class strictly larger than QCMA, a class with known search-to-decision reductions (see Theorem 2.7 of the full version [8])?

7.2 The Unitary Synthesis Problem

In Aaronson's lecture notes [1] and his published list of open questions in quantum query complexity [2], he identifies the unitary synthesis problem as one of the major unresolved questions.

▶ **Conjecture 10** ([2, Problem 6]). *For every n-qubit unitary transformation U, does there exists an oracle $A : \{0,1\}^* \to \{0,1\}$ such that a BQP^A machine can implement U?*

While, we do not know how to synthesize the unitary U, we do know how to synthesize the Choi - Jamiolkowski state, [5, 9]

$$|g_U\rangle_{LR} \stackrel{\text{def}}{=} \sqrt{\frac{1}{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle_L U |x\rangle_R , \tag{18}$$

which can also be seen as applying the unitary $\mathbb{I}_L \otimes U_R$ to the maximally entangled state. This comes from the previously constructed state synthesis algorithms or the state synthesis algorithm of Aaronson [1]. While the Choi - Jamiolkowski state contains all the information about U, it is unclear how to use $|g_U\rangle$ to apply the unitary U. One idea is to use the gate-by-teleportation technique from measurement-based quantum computation to apply U. In this procedure, one measures an n-qubit input state $|\psi\rangle$ in register A and the half of $|g_U\rangle$ in register L in the generalized Bell basis, i.e. the POVM with elements $|g_{X^aZ^b}\rangle\langle g_{X^aZ^b}|_{AL}$ for $a, b \in \{0,1\}^n$. It is known that each outcome (a, b) is equally likely to occur and the resulting state on the R register is $UX^aZ^b |\psi\rangle$. Unfortunately, the Pauli twirl, X^aZ^b, has been applied inside of the unitary U.

However, note that whenever the measurement outcome $a = b = 0^n$ occurs, the resulting state is $U |\psi\rangle$ as desired. Therefore, there is a *post-selection* algorithm which generates the output $U |\psi\rangle$ by post-selecting on the outcome $a = b = 0^n$. The issue, of course, is that post-selection is a non-unitary operation. However, we note that while post-selection is non-unitary, the classes PostBQP and PostQMA have definitions as classical complexity theory classes PP and PSPACE, respectively. We previously outlined search-to-decision reductions for both of these classes through their equivalences with PGQMA and $\mathsf{QMA}_{\mathsf{exp}}$, respectively. While not obvious to us at the moment, we suspect that there may be an insight connecting these ideas together to generate a solution to the unitary state synthesis problem.

7.3 Improving the construction of witnesses for $\mathsf{QMA}_{\mathsf{exp}}$

We leave it as an open question as to whether the Swap Test Distillation algorithm can be used to improve the overlap with a $\mathsf{QMA}_{\mathsf{exp}}$ witness produced by the protocol described in Section 2.2 of the full version [8]. The challenge is establishing that the conditions for the distillation algorithm are met when t-designs (such as Clifford unitaries) are used to randomize the target state instead of Haar-random unitaries. We know that Clifford unitaries will produce a state whose expected overlap with the target state is at least a constant. Theorem 5.8 of the full version [8] shows that the Swap Test Distillation algorithm still works under this relaxed condition (instead of requiring that every input state have constant overlap with probability 1). The problem lies in the second condition: showing that with high probability, for two independently generated output states, their components orthogonal to the target state are close to orthogonal to each other. The proof in Lemma 7 showing that this holds for the 1-query protocol that uses Haar-random unitaries relies on the following fact: for any two orthogonal states $|\psi_1\rangle$ and $|\psi_2\rangle$, even when conditioning on the event that $U|\psi_1\rangle = |\phi\rangle$ for some specific $|\phi\rangle$, the state $U|\psi_2\rangle$ is still distributed in a manner that looks close to random. We leave it as an open question whether a similar fact can be shown when U is a t-design or whether there is a different way to establish the second requirement for the Swap Test Distillation algorithm. A proof that t-designs satisfy the second requirement for the distillation algorithm would also result in an improvement over the 1-query protocol for synthesizing arbitrary states shown in Section 4, by reducing the time complexity of the protocol from exponential to polynomial time.

References

1 Scott Aaronson. The complexity of quantum states and transformations: from quantum money to black holes. *arXiv preprint*, 2016. arXiv:1607.05256.

2 Scott Aaronson. Open problems related to quantum query complexity, 2021. arXiv:2109.06917.

3 Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. *Theory OF Computing*, 3:129–157, 2007.

4 Dorit Aharonov, Michael Ben-Or, Fernando G. S. L. Brandao, and Or Sattath. The pursuit of uniqueness: Extending Valiant-Vazirani theorem to the probabilistic and quantum settings, 2008. arXiv:0810.4840.

5 Man-Duen Choi. Completely positive linear maps on complex matrices. *Linear Algebra and its Applications*, 10(3):285–290, 1975. doi:10.1016/0024-3795(75)90075-0.

6 Abhinav Deshpande, Alexey V. Gorshkov, and Bill Fefferman. The importance of the spectral gap in estimating ground-state energies, 2020. arXiv:2007.11582.

7 Bill Fefferman and Cedric Lin. Quantum Merlin Arthur with Exponentially Small Gap, 2016. arXiv:1601.01975.

8 Sandy Irani, Anand Natarajan, Chinmay Nirkhe, Sujit Rao, and Henry Yuen. Quantum search-to-decision reductions and the state synthesis problem, 2021. arXiv:2111.02999.

9 A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275–278, 1972. doi:10.1016/0034-4877(72)90011-0.

10 Alexei Yu Kitaev, Alexander Shen, Mikhail N Vyalyi, and Mikhail N Vyalyi. *Classical and quantum computation*. Number 47 in Graduate Studies in Mathematics. American Mathematical Soc., 2002.

11 William Kretschmer. Quantum pseudorandomness and classical complexity. *arXiv preprint*, 2021. arXiv:2103.09320.

12 Chris Marriott and J. Watrous. Quantum Arthur–Merlin games. *computational complexity*, 14:122–152, 2004.

13 Chinmay Nirkhe, Umesh Vazirani, and Henry Yuen. Approximate Low-Weight Check Codes and Circuit Lower Bounds for Noisy Ground States. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 91:1–91:11, 2018.

14 Gregory Rosenthal and Henry Yuen. Interactive proofs for synthesizing quantum states and unitaries. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

Almost Polynomial Factor Inapproximability for Parameterized k-Clique

Karthik C. S.
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Subhash Khot
Courant Institute of Mathematical Sciences, New York University, NY, USA

Abstract

The k-Clique problem is a canonical hard problem in parameterized complexity. In this paper, we study the parameterized complexity of approximating the k-Clique problem where an integer k and a graph G on n vertices are given as input, and the goal is to find a clique of size at least $k/F(k)$ whenever the graph G has a clique of size k. When such an algorithm runs in time $T(k) \cdot \text{poly}(n)$ (i.e., FPT-time) for some computable function T, it is said to be an $F(k)$-*FPT-approximation algorithm* for the k-Clique problem.

Although, the non-existence of an $F(k)$-FPT-approximation algorithm for any computable sublinear function F is known under gap-ETH [Chalermsook et al., FOCS 2017], it has remained a long standing open problem to prove the same inapproximability result under the more standard and weaker assumption, W[1]≠FPT.

In a recent breakthrough, Lin [STOC 2021] ruled out constant factor (i.e., $F(k) = O(1)$) FPT-approximation algorithms under W[1]≠FPT. In this paper, we improve this inapproximability result (under the same assumption) to rule out every $F(k) = k^{1/H(k)}$ factor FPT-approximation algorithm for any increasing computable function H (for example $H(k) = \log^* k$).

Our main technical contribution is introducing list decoding of Hadamard codes over large prime fields into the proof framework of Lin.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Problems, reductions and completeness

Keywords and phrases Parameterized Complexity, k-clique, Hardness of Approximation

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.6

Related Version *Full Version*: `https://arxiv.org/abs/2112.03983` [39]

Funding *Karthik C. S.*: This work was supported by a grant from the Simons Foundation, Grant Number 825876, Awardee Thu D. Nguyen. Also, part of this work was done when the author was a postdoctoral researcher at NYU and supported by Subhash Khot's Simons Investigator Award.
Subhash Khot: This work was supported by the NSF Award CCF-1422159, 2130816, the Simons Collaboration on Algorithms and Geometry, and the Simons Investigator Award.

1 Introduction

In the *clique* problem (Clique), we are given an undirected graph G on n vertices and an integer k, and the goal is to decide whether there is a subset of vertices $S \subseteq V(G)$ of size k such that every two distinct vertices in S share an edge in G. Often regarded as one of the classical problems in computational complexity, Clique was first shown to be NP-complete in the seminal work of Karp [38]. Thus, its optimization variant, namely the *maximum clique*, where the goal is to find a clique of the largest possible size, is also NP-hard.

To circumvent this apparent intractability of the problem, the study of an approximate version was initiated. The quality of an approximation algorithm is measured by the *approximation ratio*, which is the ratio between the size of the maximum clique and the size

of the solution output by the algorithm. It is trivial to obtain an n/c factor approximation algorithm for any constant $c \in \mathbb{N}$. The state-of-the-art approximation algorithm is due to Feige [27] which yields an approximation ratio of $O(n(\log\log n)^2/\log^3 n)$. On the opposite side, Maximum Clique is arguably the first natural combinatorial optimization problem studied in the context of hardness of approximation; in a seminal work of Feige, Goldwasser, Lovász, Safra and Szegedy [28], a connection (hereafter referred to as the FGLSS reduction) was made between interactive proofs and hardness of approximating Clique. The FGLSS reduction, together with the PCP theorem [3, 2, 20] and gap amplification via randomized graph products [8], immediately implies n^{ε} ratio inapproximability of Clique for some constant $\varepsilon > 0$ under the assumption that NP$\not\subseteq$ BPP. Following [28], a long line of research on the inapproximability of Clique [6, 29, 5, 7], culminated in the works of Håstad [36, 35], wherein it was shown that Clique cannot be approximated to within a factor of $n^{1-\varepsilon}$ in polynomial time unless NP$\subseteq$ZPP; this was later derandomized by Zuckerman [57]. Since then, better inapproximability ratios are known [26, 42, 43], with the best ratio being $n/2^{(\log n)^{3/4+\varepsilon}}$ for every $\varepsilon > 0$ (assuming NP $\not\subseteq$ BPTIME($2^{(\log n)^{O(1)}}$)). Summarizing, our understanding of the limits of efficient computation of approximating clique in the NP world is almost complete.

Besides approximation, another widely-used technique to cope with NP-hardness is *parameterization*. The parameterized version of Clique, which we will refer to simply as k-Clique, is exactly the same as the original decision version of the problem except that now we are not looking for a polynomial time algorithm but rather a *fixed parameter tractable* (FPT) algorithm – one that runs in time $T(k) \cdot \text{poly}(n)$ for some computable function T (e.g., $T(k) = 2^k$ or even 2^{2^k}). Such running time will henceforth be referred to as *FPT time*. It turns out that even with this relaxed requirement, k-Clique still remains intractable: in the same work that introduced the W-hierarchy, Downey and Fellows [22] showed that k-Clique is complete for the class W[1], which is generally believed to not be contained in FPT, the class of fixed parameter tractable problems. Subsequently, stronger running time lower bounds have been shown for k-Clique under stronger assumptions. Specifically, Chen et al. [15] ruled out $T(k) \cdot n^{o(k)}$-time algorithms for k-Clique assuming the *Exponential Time Hypothesis* (ETH)[1]. Note that the trivial algorithm that enumerates through every k-tuple of vertices, and checks whether it forms a clique, runs in $\widetilde{O}(n^k)$ time. It is possible to speed up this running time using fast matrix multiplication [52, 25].

Given the strong negative results for k-Clique discussed in the previous paragraph, it is natural to ask whether one can come up with a *fixed parameter approximation (FPT-approximation) algorithm* for k-Clique. The notion of FPT-approximation algorithms is motivated primarily through the consideration of inputs with small sized optimal solutions. Case in point, the state-of-the-art polynomial time approximation ratio of $O(n(\log\log n)^2/\log^3 n)$ [27] would be meaningless if the size of the maximum clique (denoted OPT) was itself $O(n(\log\log n)^2/\log^3 n)$, as outputting a single vertex already guarantees an OPT-approximation ratio. In this case, a bound such as $o(\mathsf{OPT})$ would be more meaningful. Unfortunately, no approximation ratio of the form $o(\mathsf{OPT})$ is known even when FPT-time is allowed. We refer the reader to the textbooks [23, 19] for an excellent introduction to the area. On the other hand, inapproximability results in parameterized complexity aim to typically rule out algorithms running in FPT time (under the W[1]$\neq$FPT hypothesis) for various classes of computable functions F. This brings us to the main question addressed in our work:

[1] ETH [37] states that no subexponential time algorithm can solve 3-SAT.

Is there an $F(k)$-FPT-approximation algorithm for k-Clique for some computable function F which is $o(k)$?

This question, which dates back to late 1990s (see, e.g., remarks in [24]), has attracted significant attention in literature and continues to be repeatedly raised in workshops and surveys on parameterized complexity [16, 51, 18, 12, 55, 17, 40, 30]. This open problem is even listed[2] in the seminal textbook of Downey and Fellows [23].

Early attempts [34, 12] ruled out constant ratio FPT-approximation algorithms for k-Clique, but under very strong assumptions such as the combination of ETH and the existence of a linear-size PCP. However, a few years ago, the authors in [14] proved under the *Gap Exponential Time Hypothesis* (Gap-ETH)[3], that no $F(k)$-approximation algorithm for k-Clique exists for any computable function F. Such non-existence of FPT-approximation algorithms is referred to in literature as the *total FPT-inapproximability* of k-Clique.

While the result in [14] seems to settle the parameterized complexity of approximating k-Clique, there are a few disadvantages to their result. First, while Gap-ETH may be plausible, it is a strong conjecture and in their reduction, the hypothesis does much of the work in the proof. In particular, Gap-ETH itself already gives the gap in hardness of approximation; once they have such a gap, it suffices to design gap preserving reductions to prove other inapproximability results (although some care needs to be taken as they cannot directly use Raz's parallel repetition theorem [54] for gap amplification). This is analogous to the NP-world, where once one inapproximability result can be shown, many others follow via relatively simple gap-preserving reductions (see, e.g., [53]). However, creating a gap in the first place requires the PCP Theorem [3, 2, 20], which involves several new technical ideas such as local checkability and decodability of codes and proof composition. Hence, it is desirable to bypass Gap-ETH and prove total FPT-inapproximability under a standard assumption such as W[1]≠FPT, that does not inherently have a gap.

The last seven years have witnessed many significant inapproximability results in parameterized complexity that are only based on the assumption W[1]≠FPT. A key component in all these works is a gap creating technique. Elaborating, we now have strong inapproximability results under W[1]≠FPT for Set Cover [17, 40, 46, 41], Set Intersection [45, 13], Steiner Orientation problem [56], and problems in Coding theory and Lattice theory [9]. There have been even more strong inapproximability results under Gap-ETH proved in these last few years and we direct the reader to a recent survey [30] on the topic.

Returning to the discussion on the inapproximability of k-Clique, the difficulty in adopting the techniques from the NP world into parameterized complexity were discussed in many previous works, such as [16, 45, 17, 30], and it was also widely believed [16] that one needs to prove a PCP theorem analogue for parameterized complexity[4] in order to obtain any non-trivial inapproximability result for k-clique under the W[1]≠FPT assumption. Recently, in a remarkable breakthrough, Lin [47] negated this belief, and designed a different approach to prove constant ratio inapproximability for the k-Clique problem assuming W[1]≠FPT.

---

[2] In [23], the authors list proving hardness of approximation for dominating set as one of the six "most infamous" open questions in the area of Parameterized Complexity. Immediately, they clairfy that, "One can ask similarly about an FPT Approximation for Independent Set". Note that inapproximability results for independent set problem imply hardness of approximation of k-Clique and vice versa.

[3] Gap-ETH [21, 50] is a strengthening of ETH, and states that no subexponential time algorithm can distinguish satisfiable 3-CNF formulae from ones that are not even $(1-\varepsilon)$-satisfiable for some $\varepsilon > 0$.

[4] One such formulation is called the *Parameterized Inapproximability Hypothesis* (PIH) and was putforth by [49]. See Section 5 for a small discussion on PIH.

Lin's proof framework is briefly described in Section 1.1, but even given the result of [47], one remains very far from proving the total FPT-inapproximability of k-Clique. Thus, our result stated below, is a significant improvement over Lin's result.

▶ **Theorem 1** (Almost Polynomial Factor Inapproximability of k-Clique). *Let $H : \mathbb{N} \to \mathbb{N}$ be an increasing*[5] *computable function such that $\forall k \in \mathbb{N}$, we have $H(k) \leqslant k$. Given as input an integer k and a graph G on n vertices, it is W[1]-hard parameterized by k (under randomized reductions), to distinguish between the following two cases:*
Completeness: *G has a clique of size k.*
Soundness: *G does not have a clique of size $k/k^{1/H(k)}$.*

For example, if we plug in $H(k) = \log\log k$ in our theorem, we obtain $k^{1/\log\log k} = \omega(\text{polylog } k)$ ratio inapproximability of k-Clique. In fact, if we susbstitute H in the theorem statement with a *very* slowly growing function, then we *almost* obtain polynomial ratio inapproximability of k-Clique. We reiterate again that the only comparable result to the above theorem, is by Lin [47], who ruled out constant ratio (i.e., $H(k) = O(\log k)$) FPT-approximation algorithms.

Our result also rules out $k^{1/H(k)}$ ratio FPT-approximation algorithms for the k-Independent Set problem by using the well-known connection to the k-Clique problem.

We remark here that independent of our work, in [48], the authors assuming ETH, rule out FPT algorithms for approximating k-clique to the same hardness of approximation factors as in Theorem 1. Note that $W[1] \neq \text{FPT}$ is a weaker assumption than ETH as the latter is known to imply the former [15].

1.1 Proof Overview

In this subsection, we provide a proof overview of Theorem 1. In order to motivate our proof framework and ideas, we first describe a wishful thinking reduction to gap k-Clique, and then describe Lin's framework, and finally provide the details of our techniques.

From PIH to gap k-Clique

Suppose, our starting point was a gap 2-CSP[6] instance φ on k variables and alphabet $[n]$, which is either completely satisfiable (i.e., there exists an assignment that satisfies all the constraints) or every assignment to the variables violates at least 1% of the constraints. Furthermore, suppose that it was W[1]-hard, parameterized by k, to decide φ. This assumption is known as PIH and it was believed [16] that we need to first prove PIH in order to prove the hardness of gap k-Clique. Applying the well-known FGLSS reduction to φ, we obtain a graph in which finding a clique which is larger than 99% of the maximum clique size is W[1]-hard. Of course, the big problem with this reduction is that we do not know if PIH is true.

Lin's Framework

In [47], the author circumvents proving PIH, and instead makes the following surprising observation. Let φ be a 2-CSP instance where the variable set is thought of as $\{0,1\}^k$, and the constraints are only between a pair of points that differ on one coordinate. We call a

[5] A function $H : \mathbb{N} \to \mathbb{N}$ is said to be increasing if for all $k \in \mathbb{N}$ we have $H(k+1) \geqslant H(k)$, and $\lim_{k\to\infty} H(k) = \infty$.
[6] A t-CSP is a constraint satisfaction problem in which every constraint involves at most t variables.

constraint to be in direction $i \in [k]$ if the constraint is between a pair of points that differ on the i^{th} coordinate. Suppose we can show that it is W[1]-hard parameterized by $t := 2^k$, to distinguish between the cases when either φ is satisfiable or when, for every assignment to $\{0,1\}^k$, there exists $i \in [k]$, such that 1% of the constraints in the i^{th} direction are violated. Note that in the soundness case, there is no guarantee that for every assignment, 1% of the total constraints are violated, in fact, for every assignment we are only guaranteed that $\Omega(1/k)$ fraction of the total constraints are violated. Nevertheless, by applying the FGLSS reduction to φ, we obtain a gap t-Clique!

Therefore, informally speaking, a wishful version of Lin's framework comprises of two steps.

(i) Show W[1]-hardness of deciding 2-CSP on the Boolean hypercube host graph with the aforementioned soundness property.

(ii) Apply FGLSS reduction to reduce the above 2-CSP to the gap t-Clique problem.

Lin starts from the k-Vector Sum problem, where given k collections of n Boolean vectors each, the goal is to decide if there are k vectors, one in each collection, that sum to $\vec{0}$. Starting from the k-Vector Sum problem and by using the local testability and local decodability of Hadamard codes over $\mathbb{F}_2$, he shows the W[1]-hardness of deciding 3-CSP on some variant of the Boolean hypercube host graph, with the aforementioned soundness property.

However, since we have a CSP of arity three, applying the FGLSS directly becomes tricky, and he finds a critical modification to the FGLSS reduction, which allows him to reduce to the gap t-Clique problem. We note that the gap created is between the existence of a t-clique in the completeness case versus no $0.99t$-clique in the soundness case. In order to rule out FPT-approximation algorithms for all constant ratios, he applies the well-known technique of graph product, by taking an $O(1)$-wise product of the hard k-Clique instance and the size of the graph increases only to $n^{O(1)}$.

Our Framework

We are now ready to describe our proof framework. At a high level, the gap created by Lin mainly arrives from the distance of the Hadamard code. Since the gap generated by using Hadamard codes over $\mathbb{F}_2$ is at most $1/2$, in order to obtain larger gaps, we use Hadamard codes over $\mathbb{F}_q$, for some large q only depending on k. However, working with Hadamard codes over $\mathbb{F}_q$ in the low acceptance regime, has its own challenges, such as:

- First, in Lin's case, local testability of Hadamard codes in the high acceptance regime is just the standard BLR Linearity testing [11], which can be used off the shelf. However, we need to test the Hadamard code in the low acceptance regime over $\mathbb{F}_q$, and thus we prove results on the list decodability of Hadamard codes over $\mathbb{F}_q$. Such results appear implicitly in literature, and we make them explicit through our Theorems 4 and 5.
- Second, because we deal with list decoding instead of standard decoding, all the relationships in our proofs have some "noise" and therefore the arguments in our soundness analysis of Theorem 1 are very intricate.

We have described above, the challenges that we had to address to prove Theorem 1 over the result of [47]. Next, we sketch the outline of our proof.

Our starting point is the same as Lin, i.e., the k-Vector Sum problem, but over $\mathbb{F}_q$. The W[1]-hardness of the k-Vector Sum problem is known in literature [1], and in fact Lin provides a short proof in his paper. Then we create a 3-CSP on the variable set $\mathbb{F}_q^k$ and alphabet size $[n]$ with three types of constraints:

(i) We have 3-arity constraints arising from the 3-query list decoding of Hadamard codes. These constraints enforce that the assignments satisfying them can themselves be viewed as a Hadamard codeword. In particular, for every k-tuple of vectors of the k-Vector Sum instance, our assignment is supposed to be the Hadamard encoding of the sum of the k-tuple of vectors.

(ii) We have 2-arity constraints arising from a pair of points on any axis parallel line in $\mathbb{F}_q^k$. The constraints along the i^{th} direction enforce that the i^{th} vector in our k-tuple of vectors indeed comes from the i^{th} collection in the k-Vector Sum instance.

(iii) We have 2-arity constraints arising from a pair of points on specific lines through the origin, which enforce that the sum of the k-tuple of vectors is $\vec{0}$.

After constructing this CSP, we build an instance of the t-Clique problem, where $t := q^{2k}$, by building a graph on q^{2k} clouds of vertices, where each cloud is an independent set containing one vertex for each triple $(x, y, x + y) \in [n] \times [n] \times [n]$. Each cloud represents a pair of variables of our CSP, which are the queries to the linearity test. The satisfying pairs of the alphabet set of the constraints in items (ii) and (iii) appear directly as edges in the graph. Since every variable appear in multiple clouds of vertices, we only put an edge between pairs of vertices that are "consisitent" on their assigment to a variable.

Unlike [47], we do not analyze the reduction from k-Vector Sum problem to the 3-CSP and from the 3-CSP to the t-Clique problem, in two separate steps, but rather we analyze the instance of the t-Clique directly with respect to the k-Vector Sum problem, and this helps us keep the analysis clean and succinct. A more detailed overview of this reduction and analysis is given in Section 4.2.

1.2 Organization of Paper

The paper is organized as follows. First, in Section 2 we define the k-Vector Sum problem and state its known hardness result. Then, in Section 3 we prove linearity testing result in the low soundness regime (a.k.a. list decoding of Hadamard code) over fields of large prime order. Next, in Section 4 we prove our main result, i.e., Theorem 1. Finally, in Section 5 we highlight a couple of important open problems.

2 Preliminaries

First, we define the notion of relative Hamming distance that is used throughout this paper. Let q be a prime power and $n \in \mathbb{N}$. For any two vectors $x, y \in \mathbb{F}_q^d$ we define its relative Hamming distance, denoted $\|x - y\|$, as the fraction of coordinates in $[d]$ in which x and y differ, i.e.,

$$\|x - y\| := \frac{|\{i \in [d] : x_i \neq y_i\}|}{d}.$$

Next, we define the k-Vector Sum problem and state its known W[1]-hardness result.

▶ **Definition 2** (k-Vector Sum). *Let q be a prime. Given k sets $U_1, \ldots, U_k$ of vectors in $\mathbb{F}_q^m$, the goal of k-vector-sum problem is to decide whether there exist $\vec{u}_1 \in U_1, \ldots, \vec{u}_k \in U_k$ such that $\sum_{i \in [k]} \vec{u}_i = \vec{0}$.*

It is known that the above problem is W[1]-hard over finite fields [1]. We direct the reader to [47] for a short proof[7].

▶ **Theorem 3** ([1, 47]). *For every prime q (independent of n), k-Vector-Sum over $\mathbb{F}_q$ and $m = \Theta(k^2 \log n)$ is $W[1]$-hard parameterized by k.*

3 Low Soundness Linearity Testing over Large Characteristic Fields

In this section, we prove a linearity testing result which is a key technical component in proving our inapproximability result.

Let q be a prime number and $d, \ell \in \mathbb{N}$. Given a function $f : \mathbb{F}_q^d \to \mathbb{F}_q^\ell$, consider the following test $\mathcal{T}$. Pick $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_q^d$ uniformly and independently at random. Accept if $f(\vec{\alpha}) + f(\vec{\beta}) = f(\vec{\alpha} + \vec{\beta})$ and reject otherwise. Further, we define $S_{f,\mathcal{T}} \subseteq \mathbb{F}_q^d \times \mathbb{F}_q^d$ as follows:

$$S_{f,\mathcal{T}} := \{(\vec{\alpha}, \vec{\beta}) \in \mathbb{F}_q^d \times \mathbb{F}_q^d : f(\vec{\alpha}) + f(\vec{\beta}) = f(\vec{\alpha} + \vec{\beta})\}.$$

Furthermore, we define $\mathsf{var}(f, \mathcal{T}) \subseteq \mathbb{F}_q^d$ as follows:

$$\mathsf{var}(f, \mathcal{T}) := \{\vec{\alpha} \in \mathbb{F}_q^d : \exists \vec{\beta} \in \mathbb{F}_q^d \text{ such that } (\vec{\alpha}, \vec{\beta}) \in S_{f,\mathcal{T}}\}.$$

We say that a function $c : \mathbb{F}_q^d \to \mathbb{F}_q^\ell$ is linear if for all $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_q^d$ we have $c(\vec{\alpha}) + c(\vec{\beta}) = c(\vec{\alpha} + \vec{\beta})$. Moreover, we say that a function $f : \mathbb{F}_q^d \to \mathbb{F}_q^\ell$ is scalar respecting if for all $\vec{\alpha} \in \mathbb{F}_q^d$ and all $\gamma \in \mathbb{F}_q$ we have $f(\gamma \cdot \vec{\alpha}) = \gamma \cdot f(\vec{\alpha})$.

We prove below a couple of theorems in the flavor of the many list-decoding results known in literature for Hadamard codes [31, 32, 33].

▶ **Theorem 4** (Linearity Testing). *Let q be a prime number and $d \in \mathbb{N}$. Let $f : \mathbb{F}_q^d \to \mathbb{F}_q$ be a scalar respecting function. Let $\varepsilon, \delta > 0$ be parameters such that $\varepsilon \gg \delta \gg \frac{1}{q^{1/3}}$. If f passes $\mathcal{T}$ with probability ε, then there exists an integer $r = O(1/\delta^2)$ and* linear *functions $c_1, \ldots, c_r : \mathbb{F}_q^d \to \mathbb{F}_q$, such that the following holds.*

$$\Pr_{(\vec{\alpha},\vec{\beta}) \sim S_{f,\mathcal{T}}} \left[\exists \text{ unique } j \in [r] \text{ such that } f(\vec{\alpha}) = c_j(\vec{\alpha}), f(\vec{\beta}) = c_j(\vec{\beta})\right] \geqslant 1 - O\left(\frac{\delta}{\varepsilon}\right).$$

The proof of the above theorem follows by combining known ideas in literature, more precisely, we combine the arguments made in [4] and [44] to obtain the theorem. We include a proof of the above theorem in the full version [39], for the sake of completeness.

Next, we extend the above theorem to functions from $\mathbb{F}_q^d$ to $\mathbb{F}_q^\ell$ for any $\ell \in \mathbb{N}$ by using the above theorem as a blackbox result. The proof is deferred to the full version [39].

▶ **Theorem 5** (Piecing Together). *Let q be a prime number and $d, \ell \in \mathbb{N}$. Let $f : \mathbb{F}_q^d \to \mathbb{F}_q^\ell$ be a scalar respecting function. Let $\varepsilon, \tau > 0$ be parameters such that $\tau \geqslant \varepsilon \gg \frac{1}{q^{1/3}}$. If f passes $\mathcal{T}$ with probability ε, then there exists a* linear *function $c : \mathbb{F}_q^d \to \mathbb{F}_q^\ell$, such that the following holds.*

$$\Pr_{\vec{\alpha} \sim \mathsf{var}(f,\mathcal{T})} [\|f(\vec{\alpha}) - c(\vec{\alpha})\| \leqslant \tau] \geqslant \frac{\varepsilon^2}{3}.$$

[7] The proof in [47] is over $\mathbb{F}_2$ but it is easy to see that their reduction generalizes to fields of larger characteristic. Also, they prove the hardness result for a version of k-Vector Sum where a target vector is given as input, but that version reduces to the version given in this paper by simply including an extra collection containing only the negative of the target vector.

4 Almost Polynomial Factor FPT Inapproximability of k-Clique

In this section we prove Theorem 1. More precisely, we prove the following.

▶ **Theorem 6.** *Let $\mathbb{P}$ be the set of all prime numbers. For every increasing computable function $F : \mathbb{N} \to \mathbb{N}$, there exists computable functions $\Lambda : \mathbb{N} \to \mathbb{N}$ and $\hat{q} : \mathbb{N} \to \mathbb{P}$ such that the following holds. For every fixed parameter $k \in \mathbb{N}$, there is a randomized reduction running in $\Lambda(k)^{O(1)} \cdot poly(n)$ time which given an instance $(U_1, U_2, \ldots, U_k)$ of k-vector sum as input, where for all $i \in [k]$ we have that U_i is a collection of n vectors in $\mathbb{F}_{\hat{q}(k)}^{O(k^2 \log n)}$, outputs a graph G such that the following holds.*

Completeness: *If there exist $\vec{u}_1 \in U_1, \ldots, \vec{u}_k \in U_k$ such that $\sum_{i\in[k]} \vec{u}_i = \vec{0}$, then, there is a clique in G of size exactly $\Lambda(k)$.*

Soundness: *If for all $\vec{u}_1 \in U_1, \ldots, \vec{u}_k \in U_k$ we have that $\sum_{i\in[k]} \vec{u}_i \neq \vec{0}$, then, there is no clique in G of size $\Lambda(k)^{1-\frac{1}{F(\Lambda(k))}}$.*

Size: *The number of vertices in G is at most $\Lambda(k) \cdot poly(n)$.*

The proof of Theorem 1 then follows by invoking the above theorem and noting the W[1]-hardness of k-Vector Sum problem (Theorem 3).

The proof outline of Theorem 6 in the subsequent subsections is as follows. In Section 4.1 we introduce a few definitions and results which will be useful for the design and analysis of our reduction. In Section 4.2, we outline a randomized reduction from the k-vector sum to the $\Lambda(k)$-clique problem. In Section 4.3, we prove the completeness, soundness, and claims on the reduction parameters.

4.1 Notations and Definitions

In this subsection we introduce a few definitions and prove some basic results which will come in handy in the subsequent subsections.

For any finite field $\mathbb{F}$ we define the operator $\langle\cdot\rangle : \mathbb{F}^d \times \mathbb{F}^d \to \mathbb{F}$ (for every $d \in \mathbb{N}$) as follows. For all $\vec{a} := (a_1, \ldots, a_d), \vec{b} := (b_1, \ldots, b_d) \in \mathbb{F}^d$ we have $\langle \vec{a}, \vec{b} \rangle = \sum_{i\in[d]} (a_i \cdot b_i)$, where the sum is over $\mathbb{F}$.

Next, we define an operator $\mathcal{M}$ which mimics matrix multiplication but by treating the matrices as vectors. Formally, for any field $\mathbb{F}$ and $t, d \in \mathbb{N}$, we define $\mathcal{M} : \mathbb{F}^d \times \mathbb{F}^{t\cdot d} \to \mathbb{F}^t$ as follows. For all $\vec{a} \in \mathbb{F}^d, \vec{b} := (\vec{b}_1, \ldots, \vec{b}_t) \in \mathbb{F}^{t\cdot d}$ (where $\vec{b}_i \in \mathbb{F}^d$ for all $i \in [t]$) we have: $\mathcal{M}(\vec{a}, \vec{b}) := \left(\langle \vec{a}, \vec{b}_1 \rangle, \ldots, \langle \vec{a}, \vec{b}_t \rangle\right)$.

We now define a linear transformation g that will be useful later on. Let $k \in \mathbb{N}$ and $q \in \mathbb{P}$. Let $B \subseteq \mathbb{F}_q^m$, where $m = \Theta(k^2 \log n)$ and $|B| = n$. Let $\ell := 12 \log_q n$. In the next subsection, we will fix k, set q to be a prime depending on k, and use the notations m and ℓ as specified here.

Select ℓ matrices $A_1, A_2, \ldots, A_\ell \in \mathbb{F}_q^{k\times m}$ uniformly and independently at random. For every $\vec{b} \in \mathbb{F}_q^m$, let $g(\vec{b}) := (A_1\vec{b}, \cdots, A_\ell\vec{b}) \in \mathbb{F}_q^{k\cdot\ell}$.

Let $\widetilde{B}_r \subseteq \mathbb{F}_q^m$ be the r-sumset of B, i.e.,

$$\widetilde{B}_r := \left\{ \sum_{i\in[r]} \gamma_i \cdot \vec{b}_i \middle| \gamma_1, \ldots, \gamma_r \in \mathbb{F}_q, \text{ and } \vec{b}_1, \ldots, \vec{b}_r \in B \right\}.$$

We next show that if q is large but only a function of k (independent of n), then with very high probability, the relative Hamming weight of the images of all vectors in $\widetilde{B}_k$ under g is high.

▶ **Proposition 7.** *Suppose $q > 2^{12k}$ but $q = O_k(1)$. Then with probability at least $1 - \frac{O_k(1)}{n^k}$, for every $\vec{b} \in \widetilde{B}_k \setminus \{\vec{0}\}$, we have that $\|g(\vec{b})\| \geqslant 2/3$.*

Proof. For every $i \in [\ell]$, let $\vec{a}_i^1, \ldots, \vec{a}_i^k \in \mathbb{F}_q^m$ be the row vectors of A_i. Fix $\vec{b} \in \widetilde{B}_k \setminus \{\vec{0}\}$. For any fixed $i \in [\ell]$ and $j \in [k]$, we have $\Pr\left[\langle \vec{a}_i^j, \vec{b}\rangle \neq 0\right] = 1 - \frac{1}{q}$, where the probability is over the selection of the random matrix row $\vec{a}_i^j$.

Next, the probability that for a fixed $\vec{b}$ we have $\|g(\vec{b})\| < 2/3$ is upper bounded by the probability that there exists a subset $S \subseteq [\ell] \times [k]$ of size $\ell k/3$ such that for every $(i,j) \in S$ we have $\langle \vec{a}_i^j, \vec{b}\rangle = 0$. Therefore, $\Pr\left[\|g(\vec{b})\| < \frac{2}{3}\right] \leqslant \binom{\ell k}{|S|} \cdot q^{-\ell k/3}$. By union bound, the probability that for every $\vec{b} \in \widetilde{B}_k \setminus \{\vec{0}\}$, we have $\|g(\vec{b})\| \geqslant \frac{2}{3}$ is at least:

$$1 - |\widetilde{B}_k| \cdot \binom{\ell k}{\ell k/3} \cdot q^{-\ell k/3}. \tag{1}$$

Note that $|\widetilde{B}_k| \leqslant (qn)^k = O_k(1) \cdot n^k$, $\binom{\ell k}{\ell k/3} \leqslant 2^{\ell k} = n^{12k/\log q} < n$, and $q^{-\ell k/3} \leqslant n^{-4k}$. Thus we have expression in (1) is lower bounded by $1 - \frac{O_k(1)}{n^k}$. ◀

We saw above that any two vectors in B disagree on most coordinates under g. We see below that this continues to hold even when projected to a fixed smaller subspace.

▶ **Proposition 8.** *Suppose $q > 2^{12k}$ but $q = O_k(1)$. Then with probability at least $1 - \frac{O_k(1)}{n}$, for every distinct $\vec{b}_1, \vec{b}_2 \in \widetilde{B}_2$, and linealy independent $\vec{a}_1, \vec{a}_2 \in \mathbb{F}_q^k$, we have that*

$$\left\|\mathcal{M}\left(\vec{a}_1, g(\vec{b}_1)\right) - \mathcal{M}\left(\vec{a}_2, g(\vec{b}_2)\right)\right\| \geqslant \frac{1}{2}.$$

Proof. For fixed non-zero $\vec{a} \in \mathbb{F}_q^k$, $i \in [\ell]$, and any $\vec{\rho} \in \mathbb{F}_q^m$ we have $\Pr\left[\vec{a}^T A_i = \vec{\rho}^T\right] = \frac{1}{q^m}$, where the probability is over the selection of the random matrix A_i. Thus, for a fixed non-zero $\vec{b} \in \mathbb{F}_q^m$, and any fixed $\gamma \in \mathbb{F}_q$ we have

$$\Pr\left[\langle \vec{a}^T A_i, \vec{b}\rangle = \gamma\right] = \frac{1}{q}. \tag{2}$$

Next the probability that for fixed distinct $\vec{b}_1, \vec{b}_2 \in \widetilde{B}_2 \setminus \{\vec{0}\}$, and fixed linearly independent $\vec{a}_1, \vec{a}_2 \in \mathbb{F}_q^k$ we have $\left\|\mathcal{M}\left(\vec{a}_1, g(\vec{b}_1)\right) - \mathcal{M}\left(\vec{a}_2, g(\vec{b}_2)\right)\right\| < \frac{1}{2}$ is upper bounded by the probability that there exists a subset $S \subseteq [\ell]$ of size $\ell/2$ such that for every $i \in S$ we have $\langle \vec{a}_1^T A_i, \vec{b}_1\rangle = \langle \vec{a}_2^T A_i, \vec{b}_2\rangle$. However, for a fixed $i \in S$, we have from (2) that

$$\Pr\left[\langle \vec{a}_1^T A_i, \vec{b}_1\rangle = \langle \vec{a}_2^T A_i, \vec{b}_2\rangle\right] = \sum_{\gamma \in \mathbb{F}_q} \Pr\left[\langle \vec{a}_1^T A_i, \vec{b}_1\rangle = \langle \vec{a}_2^T A_i, \vec{b}_2\rangle = \gamma\right] = \sum_{\gamma \in \mathbb{F}_q} \frac{1}{q^2} = \frac{1}{q},$$

where we used the linear independence of $\vec{a}_1$ and $\vec{a}_2$ in the penultimate equality. Therefore we have,

$$\Pr\left[\left\|\mathcal{M}\left(\vec{a}_1, g(\vec{b}_1)\right) - \mathcal{M}\left(\vec{a}_2, g(\vec{b}_2)\right)\right\| < \frac{1}{2}\right] \leqslant \binom{\ell}{|S|} \cdot q^{-\ell/2}.$$

By union bound, the probability that for every distinct $\vec{b}_1, \vec{b}_2 \in \widetilde{B}_2 \setminus \{\vec{0}\}$, and every linearly independent $\vec{a}_1, \vec{a}_2 \in \mathbb{F}_q^k$, we have that the probability that $\left\|\mathcal{M}\left(\vec{a}_1, g(\vec{b}_1)\right) - \mathcal{M}\left(\vec{a}_2, g(\vec{b}_2)\right)\right\| \geqslant \frac{1}{2}$ is at least:

$$1 - n^4 \cdot q^{2k} \cdot \binom{\ell}{\ell/2} \cdot q^{-\ell/2}. \tag{3}$$

Note that $\binom{\ell}{\ell/2} \leqslant 2^{\ell} = n^{12/\log q} \leqslant n$ and $q^{-\ell/2} \leqslant 1/n^6$. Thus, we have expression in (3) is lower bounded by $1 - \frac{O_k(1)}{n}$.

Finally, we consider the case that either $\vec{b}_1$ or $\vec{b}_2$ is $\vec{0}$. Then the proposition amounts to proving that for every $\vec{b} \in \widetilde{B}_2 \setminus \{\vec{0}\}$, and $\vec{a} \in \mathbb{F}_q^k \setminus \{\vec{0}\}$, we have that $\left\|\mathcal{M}\left(\vec{a}, g(\vec{b})\right)\right\| \geqslant \frac{1}{2}$. For fixed $\vec{b} \in \widetilde{B}_2 \setminus \{\vec{0}\}$ and $\vec{a} \in \mathbb{F}_q^k \setminus \{\vec{0}\}$ we have the probability that $\left\|\mathcal{M}\left(\vec{a}, g(\vec{b})\right)\right\| < \frac{1}{2}$ is upper bounded by the probability that there exists a subset $S \subseteq [\ell]$ of size $\ell/2$ such that for every $i \in S$ we have $\langle \vec{a}^T A_i, \vec{b}\rangle = 0$. However, for a fixed $i \in S$, we have from (2) that this probability is $1/q$. Therefore we have, $\Pr\left[\left\|\mathcal{M}\left(\vec{a}, g(\vec{b})\right)\right\| < \frac{1}{2}\right] \leqslant \binom{\ell}{|S|} \cdot q^{-\ell/2}$. By union bound, and calculations similar to the one done previously, the proof is completed. ◀

4.2 Construction

In this subsection, we provide the reduction from the k-Vector Sum problem to the $\Lambda(k)$-Clique problem.

Fix $F : \mathbb{N} \to \mathbb{N}$ as in the statement of Theorem 6. Without loss of generality, we assume that F satisfies the following: for all $k \in \mathbb{N}$, we have that $F(k) \leqslant \lfloor \frac{\log k}{15} \rfloor$. This is because, suppose there is an FPT algorithm which can decide if a graph has a clique of size k or no clique of size $k^{1-1/F(k)}$, then we can use the same algorithm to decide if a graph has a clique of size k or no clique of size $k^{1-1/F'(k)}$, where $F'(k) := \min\left(F(k), \lfloor \frac{\log k}{15} \rfloor\right)$.

We define the functions $\hat{q} : \mathbb{N} \to \mathbb{P}$ and $\Lambda : \mathbb{N} \to \mathbb{N}$ as follows. For every $k \in \mathbb{N}$, we define $\hat{q}(k)$ as the smallest prime number greater than[8] 2^{12k}. Note that,

$$F(\hat{q}(k)^{2k^2}) \leqslant \left\lfloor \frac{\log \hat{q}(k)^{2k^2}}{15} \right\rfloor \leqslant \left\lfloor \frac{2k^2(12k+1)}{15} \right\rfloor < 2k^3, \tag{4}$$

where we used that $\hat{q}(k) < 2 \cdot 2^{12k}$, which follows from Bertrand's postulate. For every $k \in \mathbb{N}$, we define $\Lambda(k) := (\hat{q}(k))^{2k^2}$.

Fix $k \in \mathbb{N}$ and let $q := \hat{q}(k)$. Starting from an instance $(U_1, \ldots, U_k)$ of k-Vector Sum over $\mathbb{F}_q$ (where the vectors are m-dimensional for $m = \Theta(k^2 \log n)$) we construct a graph $G(V, E)$ as follows. For all $i \in [k]$, let $|U_i| = n/k$. Let $U := U_1 \cup \cdots \cup U_k$. Recall that $\ell = 12 \log_q n$. We next put together Propositions 7 and 8 as follows. We sample[9] ℓ matrices $A_1, A_2, \ldots, A_\ell \in \mathbb{F}_q^{k \times m}$ uniformly and independently at random and with probability at least $1 - o(1)$ we have (i) $\forall\ \gamma_1, \ldots, \gamma_k \in \mathbb{F}_q, \forall\ (\vec{u}_1, \ldots, \vec{u}_k) \in U_1 \times \cdots \times U_k$, if $\sum_{i \in [k]} \gamma_i \cdot \vec{u}_i \neq \vec{0}$ then:

$$\left\| g\left(\sum_{i \in [k]} \gamma_i \cdot \vec{u}_i \right) \right\| \geqslant \frac{2}{3}\ , \tag{5}$$

and (ii) $\forall\ i \in [k]$ and for every three vectors $\vec{u}_1, \vec{u}_2, \vec{u}_3 \in U_i$ such that $\vec{u}_3 - \vec{u}_1 \neq \vec{u}_2 - \vec{u}_3$, and every linearly independent $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_q^k$ we have:

$$\left\| \mathcal{M}\left(\vec{\alpha}, g(\vec{u}_3 - \vec{u}_1)\right) - \mathcal{M}\left(\vec{\beta}, g(\vec{u}_2 - \vec{u}_3)\right) \right\| \geqslant \frac{1}{2}. \tag{6}$$

Now we are ready to construct G. First we define the vertex set V of G:

$$V := \left\{ (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in \mathbb{F}_q^{k^2} \times \mathbb{F}_q^{k^2} \times \mathbb{F}_q^{\ell} \times \mathbb{F}_q^{\ell} \middle|\ \text{if } \vec{\boldsymbol{\alpha}} = \vec{\boldsymbol{\beta}} \text{ then } \vec{x} = \vec{y} \right\}.$$

[8] This lower bound on the choice of $\hat{q}(k)$ is needed as we would like to use Propositions 7 and 8 later in the section.

[9] The usage of these sampled matrices makes our reduction randomized.

Next, instead of defining the edge set E, we will define the graph through it's non-edges. But to do so in a clean way, we need a few additional notations and definitions.

We view every $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in V$ as a function from $\{\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{\beta}}\}$ to $\mathbb{F}_q^\ell$ where we define $v(\vec{\boldsymbol{\alpha}}) = \vec{x}$, $v(\vec{\boldsymbol{\beta}}) = \vec{y}$, and $v(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{\beta}}) = \vec{x} + \vec{y}$.

For a vertex $v = (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in V$, we define $\mathsf{var}(v) := \{\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{\beta}}\}$. Further, for any set $T \subseteq V$, we abuse notation and define $\mathsf{var}(T)$ to be $\bigcup_{v \in T} \mathsf{var}(v)$.

Finally, for every $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y})$ and $v' := (\vec{\boldsymbol{\alpha'}}, \vec{\boldsymbol{\beta'}}, \vec{x'}, \vec{y'}) \in V$ we do *not* have an edge between them if and only if at least one of the following conditions hold.

Type 1: $\vec{\boldsymbol{\alpha}} = \vec{\boldsymbol{\alpha'}}$ and $\vec{\boldsymbol{\beta}} = \vec{\boldsymbol{\beta'}}$.

Type 2: There exists $\vec{\boldsymbol{\rho}} \in \mathsf{var}(v) \cap \mathsf{var}(v')$ such that $v(\vec{\boldsymbol{\rho}}) \neq v'(\vec{\boldsymbol{\rho}})$.

Type 3: There exists some $\gamma \in \mathbb{F}_q$ such that $\vec{\boldsymbol{\alpha}} = \gamma \cdot \vec{\boldsymbol{\alpha}'}$ and $\vec{x} \neq \gamma \cdot \vec{x'}$.

Type 4: There exists some $i \in [k]$ and $\vec{\alpha} \in \mathbb{F}_q^k$, such that

$$\vec{\boldsymbol{\alpha}} - \vec{\boldsymbol{\alpha'}} = \vec{\alpha} \cdot \vec{\boldsymbol{e_i}} = (\underbrace{\vec{0}, \ldots, \vec{0}}_{\substack{i-1 \\ \text{coordinates}}}, \vec{\alpha}, \vec{0}, \ldots, \vec{0}),$$

and for all $\vec{u} \in U_i$ we have $\mathcal{M}(\vec{\alpha}, g(\vec{u})) \neq \vec{x} - \vec{x'}$. We emphasize here that we think of each coordinate as a vector in $\mathbb{F}_q^k$.

Type 5: There exists some $\vec{\alpha} \in \mathbb{F}_q^k$, such that $\vec{\boldsymbol{\alpha}} - \vec{\boldsymbol{\alpha'}} = (\vec{\alpha}, \ldots, \vec{\alpha})$ and $\vec{x} \neq \vec{x'}$.

The intuition behind specifying these non-edges is as follows. For every k-tuple of vectors $\vec{\boldsymbol{u}} := (\vec{u}_1, \ldots, \vec{u}_k) \in U_1 \times \cdots \times U_k$ we associate a unique subset of vertices $T_{\vec{\boldsymbol{u}}}$ as follows:

$$T_{\vec{\boldsymbol{u}}} := \left\{ \left(\vec{\boldsymbol{\alpha}} = (\vec{\alpha}_1, \ldots, \vec{\alpha}_k), \vec{\boldsymbol{\beta}} = (\vec{\beta}_1, \ldots, \vec{\beta}_k), \sum_{i \in [k]} \mathcal{M}(\vec{\alpha}_i, g(\vec{u}_i)), \sum_{i \in [k]} \mathcal{M}(\vec{\beta}_i, g(\vec{u}_i)) \right) \middle| \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}} \in \mathbb{F}_q^{k^2} \right\}.$$

The claim then is that if $\vec{u}_1 + \cdots + \vec{u}_k = \vec{0}$ then $T_{\vec{\boldsymbol{u}}}$ is a clique. On the other hand if $\vec{u}_1 + \cdots + \vec{u}_k \neq \vec{0}$ then the Type 5 non-edges ensure that there is no $|T_{\vec{\boldsymbol{u}}}|/q^{1/k}$ sized[10] clique in the graph induced by $T_{\vec{\boldsymbol{u}}}$.

On the other hand if we pick any subset $T' \subseteq V$ of size q^{2k^2} in G then one of the first four types of non-edges ensures that there is no $|T'|/q^{1/k}$ sized clique in the graph induced by T'. In other words, the first four types of non-edges incentivize to pick subset of vertices which corresponds to $T_{\vec{\boldsymbol{u}}}$ for some $\vec{\boldsymbol{u}} \in U_1 \times \cdots \times U_k$. Type 1 non-edges incentivize to include only one vertex in T' of the form $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y})$ for every $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}} \in \mathbb{F}_q^{k^2}$. Type 2 non-edges incentivize only to pick those vertices which are "consistent", i.e., we can extract an assignment $\sigma : \mathsf{var}(T') \to \mathbb{F}_q^\ell$ in a consistent manner. Type 3 non-edges are introduced for technical reasons, as we would like to invoke Theorem 5 in our analysis, i.e., to say that if T' contains a large clique, then it must have some "linear structure". Equipped with having an assignment σ and some linear structure, the dearth of Type 4 non-edges enables us to decode a vector $\vec{u}_i^* \in U_i$ such that T' has a large intersection with $T_{\vec{\boldsymbol{u}}^*}$ where $\vec{\boldsymbol{u}}^* := (\vec{u}_1^*, \ldots, \vec{u}_k^*) \in U_1 \times \cdots \times U_k$.

In summary, Types 1-4 non-edges ensure that any subset $T \subseteq V$ of size q^{2k^2} in G which contains a large clique must overlap significantly with $T_{\vec{\boldsymbol{u}}}$ for some $\vec{\boldsymbol{u}} \in U_1 \times \cdots \times U_k$. Then the lack of Type 5 non-edges ensure that if T has a large clique then the k-tuple of vectors represented by $\vec{\boldsymbol{u}}$ must sum to $\vec{0}$.

[10] In fact, we could claim that if $\vec{u}_1 + \cdots + \vec{u}_k \neq \vec{0}$ then there is no $|T_{\vec{\boldsymbol{u}}}|/q^\delta$ sized clique, for some tiny $\delta > 0$.

4.3 Analysis

In this section, we analyze the parameters of the reduction, and prove the completeness and soundness claims of the theorem statement.

Parameters of the reduction

The new graph has at most $|\mathbb{F}_q^{2k^2}| \cdot |\mathbb{F}_q^{2\ell}| = \Lambda(k) \cdot n^{24}$ many vertices. The time needed to construct this graph is $\Lambda(k) \cdot n^{25}$.

Completeness

Suppose there exist $\vec{u}_1 \in U_1, \ldots, \vec{u}_k \in U_k$ such that $\sum_{i\in[k]} \vec{u}_i = \vec{0}$. Then, we can find a clique of size $|\mathbb{F}_q|^{2k^2}$ in G as follows. Consider $T \subseteq V$ defined as below:

$$T := \left\{ \left(\vec{\alpha_1}, \ldots, \vec{\alpha_k}, \vec{\beta_1}, \ldots, \vec{\beta_k}, \sum_{i\in[k]} \mathcal{M}(\vec{\alpha_i}, g(\vec{u_i})), \sum_{i\in[k]} \mathcal{M}(\vec{\beta_i}, g(\vec{u_i})) \right) \middle| \vec{\alpha_i}, \vec{\beta_i} \in \mathbb{F}_q^k, i \in [k] \right\}.$$

We claim that every pair of distinct vertices in T have an edge in G and since $|T| = q^{2k^2}$, the completeness case follows.

First note that if we fix any $\vec{\alpha_1}, \ldots, \vec{\alpha_k}, \vec{\beta_1}, \ldots, \vec{\beta_k} \in \mathbb{F}_q^k$ then there are unique vectors $\vec{x}, \vec{y} \in \mathbb{F}_q^\ell$ such that $\left(\vec{\alpha_1}, \ldots, \vec{\alpha_k}, \vec{\beta_1}, \ldots, \vec{\beta_k}, \vec{x}, \vec{y}\right)$ is in T. Thus, there are no Type 1 non-edges in subgraph induced by T.

Next, for every two distinct vertices $v, v' \in T$, and for every $\vec{\boldsymbol{\rho}} := (\vec{\rho_1}, \ldots, \vec{\rho_k}) \in \mathsf{var}(v) \cap \mathsf{var}(v')$, we have $v(\vec{\boldsymbol{\rho}}) = v'(\vec{\boldsymbol{\rho}}) = \sum_{i\in[k]} \mathcal{M}(\vec{\rho_i}, g(\vec{u_i}))$, and thus there are no Type 2 non-edges in subgraph induced by T.

Then, we note that there are no Type 3 non-edges in the subgraph induced by T because for every $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in T$ and every $\gamma \in F_q$, if $v' := (\gamma \cdot \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}'}, \vec{x'}, \vec{y'}) \in T$, then we have:

$$\gamma \cdot \vec{x} = \gamma \cdot \sum_{i\in[k]} \mathcal{M}(\vec{\alpha_i}, g(\vec{u_i})) = \sum_{i\in[k]} \mathcal{M}(\gamma \cdot \vec{\alpha_i}, g(\vec{u_i}) = \vec{x'}.$$

In order to next show that there are no Type 4 non-edges in subgraph induced by T, we first fix $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in T$, $i \in [k]$, and $\vec{\alpha} \in \mathbb{F}_q^k$. Suppose there exists $v' := (\vec{\boldsymbol{\alpha}} - \vec{\alpha} \cdot \vec{\boldsymbol{e_i}}, \vec{\boldsymbol{\beta'}}, \vec{x'}, \vec{y'}) \in T$. Then we have

$$\begin{aligned}\vec{x} - \vec{x'} &= \sum_{j\in[k]} \mathcal{M}(\vec{\alpha_j}, g(\vec{u_j})) - \left(\sum_{\substack{j\in[k] \\ j\neq i}} \mathcal{M}(\vec{\alpha_j}, g(\vec{u_j})) \right) - \mathcal{M}(\vec{\alpha_i} - \vec{\alpha}, g(\vec{u_i})) \\ &= \mathcal{M}(\vec{\alpha_i}, g(\vec{u_i})) - \mathcal{M}(\vec{\alpha_i} - \vec{\alpha}, g(\vec{u_i})) = \mathcal{M}(\vec{\alpha}, g(\vec{u_i})).\end{aligned}$$

Thus, (v, v') is an edge in the subgraph induced by T.

Finally, we show that there are no Type 5 non-edges in subgraph induced by T. Let $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in T$ and $\vec{\alpha} \in \mathbb{F}_q^k$. Suppose there exists $v' := (\vec{\boldsymbol{\alpha}} - (\vec{\alpha}, \ldots, \vec{\alpha}), \vec{\boldsymbol{\beta'}}, \vec{x'}, \vec{y'}) \in T$. Then we have

$$\begin{aligned}\vec{x} - \vec{x'} &= \sum_{i\in[k]} \mathcal{M}(\vec{\alpha_i}, g(\vec{u_i})) - \sum_{i\in[k]} \mathcal{M}(\vec{\alpha_i} - \vec{\alpha}, g(\vec{u_i})) \\ &= \sum_{i\in[k]} \mathcal{M}(\vec{\alpha}, g(\vec{u_i})) = \mathcal{M}\left(\vec{\alpha}, g\left(\sum_{i\in[k]} \vec{u_i}\right)\right) = \mathcal{M}(\vec{\alpha}, g(\vec{0})) = \vec{0}.\end{aligned}$$

Thus, (v, v') is an edge in the subgraph induced by T.

Soundness

Let T be the set of vertices of the largest clique in G (breaking ties arbitrarily). Let[10] $\varepsilon := 1/q^{1/k}$. Suppose $|T| \geqslant \varepsilon \cdot q^{2k^2}$, then we shall show that for every $i \in [k]$ there exists $u_i^* \in U_i$ such that $u_1^* + \cdots + u_k^* = \vec{0}$. Note that the assertion in the theorem statement is satisfied as follows:

$$|T| \geqslant q^{2k^2 \cdot (1-1/(2k^3))} \geqslant q^{2k^2 \cdot \left(1-1/F\left(q^{2k^2}\right)\right)} = \Lambda(k)^{1-1/(F(\Lambda(k)))},$$

where the penultimate inequality follows from (4).

The proof strategy is as follows. First, using T, we construct a function Γ from $\mathbb{F}_q^{k^2}$ to $\mathbb{F}_q^{\ell}$ which we show passes the linearity test with probability at least ε (Claim 9). Then, we invoke Theorem 5 to say that there exists a collection of few linear functions on k variables over $\mathbb{F}_q^k$ with coefficients from $\mathbb{F}_q^{k \times \ell}$ with the following property: for many queries $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}) \in \mathbb{F}_q^{k^2} \times \mathbb{F}_q^{k^2}$ on which Γ passes the linearity test, we have a fixed linear function in our collection whose evaluation on $\vec{\boldsymbol{\alpha}}$ agrees with $\Gamma(\vec{\boldsymbol{\alpha}})$.

Then for every $i \in [k]$ and $\vec{\alpha} \in \mathbb{F}_q^k$, we will identify $\vec{u}_{\vec{\alpha}} \in U_i$ such that $\mathcal{M}(\vec{\alpha}, g(\vec{u}_{\vec{\alpha}}))$ is roughly equal to evaluating the linear function at $\vec{\boldsymbol{e}}_i \cdot \vec{\alpha}$ (Claim 10). Next, we show that for every $i \in [k]$, there is a single $\vec{u}_i \in U_i$ such that for all $\vec{\alpha} \in \mathbb{F}_q^k$ we have that $\mathcal{M}(\vec{\alpha}, g(\vec{u}_i))$ is roughly equal to evaluating the linear function at $\vec{\boldsymbol{e}}_i \cdot \vec{\alpha}$ (Claim 11). Finally, the proof follows by observing that there are no Type 5 non-edges in T, and thus these identified $\vec{u}_i$s must sum to $\vec{0}$.

We now begin the formal soundness case analysis. We claim that for every $\vec{\boldsymbol{\alpha}} \in \mathsf{var}(T)$, if there exist distinct $v, v' \in T$ such that $\vec{\boldsymbol{\alpha}} \in \mathsf{var}(v) \cap \mathsf{var}(v')$ then, $v(\vec{\boldsymbol{\alpha}}) = v'(\vec{\boldsymbol{\alpha}})$. Otherwise, (v, v') would be a non-edge of Type 2 which is not possible as the vertices in T form a clique.

We construct a function $\Gamma : \mathbb{F}_q^{k^2} \to \mathbb{F}_q^{\ell}$ in two phases. In the first phase, we define Γ only for vectors in $\mathsf{var}(T)$. For every $\vec{\boldsymbol{\alpha}} \in \mathsf{var}(T)$, we set $\Gamma(\vec{\boldsymbol{\alpha}}) = v(\vec{\boldsymbol{\alpha}})$ if $v \in T$ is such that $\vec{\boldsymbol{\alpha}} \in \mathsf{var}(v)$. In the second phase, we iteratively go over all the vectors in $\mathbb{F}_q^{k^2} \setminus \mathsf{var}(T)$ in some canonical order. In the j^{th} iteration, let $\vec{\boldsymbol{\alpha}}_j$ be the vector considered. If there exists $\gamma \in \mathbb{F}_q$ and $\vec{\boldsymbol{\alpha}}' \in \mathsf{var}(T)$ such that $\vec{\boldsymbol{\alpha}}_j = \gamma \cdot \vec{\boldsymbol{\alpha}}'$ then we define $\Gamma(\vec{\boldsymbol{\alpha}}_j) = \gamma \cdot \Gamma(\vec{\boldsymbol{\alpha}}')$; otherwise if there exists $j' < j$ such that $\vec{\boldsymbol{\alpha}}_j = \gamma \cdot \vec{\boldsymbol{\alpha}}_{j'}$ for some $\gamma \in \mathbb{F}_q$ then we define $\Gamma(\vec{\boldsymbol{\alpha}}_j) = \gamma \cdot \Gamma(\vec{\boldsymbol{\alpha}}_{j'})$; otherwise, we set $\Gamma(\vec{\boldsymbol{\alpha}}_j)$ to a uniformly random vector in $\mathbb{F}_q^{\ell}$.

Notice that by our construction and that there are no Type 3 non-edges in T, we have that for all $\vec{\boldsymbol{\alpha}} \in \mathbb{F}_q^{k^2}$ and for all $\gamma \in \mathbb{F}_q$ we have $\Gamma(\gamma \cdot \vec{\boldsymbol{\alpha}}) = \gamma \cdot \Gamma(\vec{\boldsymbol{\alpha}})$, i.e., Γ is scalar respecting.

Next, we have the following claim on Γ passing the linearity test.

▷ Claim 9. Γ passes the linearity test with probability at least ε.

To see the claim, first consider the set $S \subseteq \mathbb{F}_q^{k^2} \times \mathbb{F}_q^{k^2}$ defined as follows.

$$S := \bigcup_{(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y}) \in T} \{(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}})\}.$$

Notice that the probability of Γ passing the linearity test is lower bounded by:

$$\frac{|S|}{|\mathbb{F}|^{2k^2}} \cdot \Pr_{(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}) \sim S} \left[\Gamma(\vec{\boldsymbol{\alpha}}) + \Gamma(\vec{\boldsymbol{\beta}}) = \Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{\beta}})\right].$$

However, for any $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}) \in S$, we have that $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \Gamma(\vec{\boldsymbol{\alpha}}), \Gamma(\vec{\boldsymbol{\beta}}))$ is in T by construction of set S. Thus, we have $\Pr_{(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}) \sim S}\left[\Gamma(\vec{\boldsymbol{\alpha}}) + \Gamma(\vec{\boldsymbol{\beta}}) = \Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{\beta}})\right] = 1$ and Γ passes the linearity test with probability at least $|S|/|\mathbb{F}_q|^{2k^2}$. Since $|S| = |T|$, we have that the proof of Claim 9 is completed.

Next invoking Theorem 5 with $\tau = \frac{1}{8k}$ (since Γ is scalar respecting and $\tau \geqslant \varepsilon$), we have that there exist a linear function $c : \mathbb{F}_q^{k^2} \to \mathbb{F}_q^{\ell}$, such that the following holds.

$$\Pr_{\vec{\boldsymbol{\alpha}} \sim \mathsf{var}(T)}[\|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| \leqslant \tau] \geqslant \frac{\varepsilon^2}{3}.$$

Let $R^* \subseteq \mathsf{var}(T)$ be the largest sized subset such that the following holds:

$$\Pr_{\vec{\boldsymbol{\alpha}} \sim R^*}[\|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| \leqslant \tau] = 1. \tag{7}$$

We note that $|R^*| \geqslant \frac{\varepsilon^2}{3} \cdot |\mathsf{var}(T)|$, and since $|\mathsf{var}(T)| \geqslant \varepsilon \cdot |\mathbb{F}_q^{k^2}|$, we have that $|R^*| \geqslant \frac{\varepsilon^3}{3} \cdot q^{k^2}$. Next, we think of c as a linear function on k variables over $\mathbb{F}_q^k$ with coefficients in $\mathbb{F}_q^{k \times \ell}$: $c(\vec{\alpha}_1, \ldots, \vec{\alpha}_k) = \sum_{i \in [k]} \mathcal{M}(\vec{\alpha}_i, \vec{\Theta}_i)$, for some $\vec{\Theta}_1, \ldots, \vec{\Theta}_k \in \mathbb{F}^{k \times \ell}$.

▷ Claim 10. For every $i \in [k]$ and $\vec{\alpha} \in \mathbb{F}_q^k$, there exists $\vec{u}_i^* \in U_i$ such that $\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_i^*))\| \leqslant 2\tau$.

If $\vec{\alpha} = \vec{0}$, the claim trivially holds. Therefore we assume that $\vec{\alpha} \in \mathbb{F}_q^k \setminus \{\vec{0}\}$. For every $i \in [k]$ and every $\vec{\alpha} \in \mathbb{F}_q^k$, we show that there is a line in the direction of $\vec{\alpha} \cdot \vec{\boldsymbol{e_i}}$ which contains two vertices $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y})$ and $(\vec{\boldsymbol{\alpha}}', \vec{\boldsymbol{\beta}}', \vec{x}', \vec{y}')$ such that $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\alpha}}' \in R^*$. Then, by noting that these two vertices don't have a Type 4 non-edge between them, we identify $\vec{u}_i^* \in U_i$. The formal argument follows.

We say a line is linear if it passes through the origin and affine otherwise. Note that every linear line can be identified through one of the non-zero points on it. Also note that for every linear line in $\mathbb{F}_q^{k^2}$, the line and all its affine shifts always cover the entire space $\mathbb{F}_q^{k^2}$. Since $|R^*|/q^{k^2} > \varepsilon^3/3 > 1/q$, we have that by an averaging argument, for every linear line, either that line or one of it's affine shifts contains at least two points in R^*. We use this argument below and in the proof of Claim 12.

Fix $i \in [k]$ and $\vec{\alpha} \in \mathbb{F}_q^k \setminus \{\vec{0}\}$. Let L_i be a linear line in $\mathbb{F}_q^{k^2}$ containing the point $\vec{\alpha} \cdot \vec{\boldsymbol{e}}_i$. Then there exists two points $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha}) \in R^*$, for some $\vec{\boldsymbol{\alpha}} \in \mathbb{F}_q^{k^2}$ and $\gamma \in \mathbb{F}_q \setminus \{0\}$. From (7) we have

$$\|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| \leqslant \tau \text{ and } \|\Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})) - c(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha}))\| \leqslant \tau. \tag{8}$$

Let $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \Gamma(\vec{\boldsymbol{\alpha}}), \Gamma(\vec{\boldsymbol{\beta}}))$ and $v' := (\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha}), \vec{\boldsymbol{\beta}}', \Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})), \Gamma(\vec{\boldsymbol{\beta}}'))$ be the two vertices in T for some $\vec{\boldsymbol{\beta}}, \vec{\boldsymbol{\beta}}' \in \mathbb{F}_q^{k^2}$. Since there is no Type 4 non-edge between them, there exists $u_i^* \in U_i$ such that

$$\Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})) - \Gamma(\vec{\boldsymbol{\alpha}}) = \mathcal{M}(\gamma \cdot \vec{\alpha}, g(\vec{u}_i^*)). \tag{9}$$

On a different note, we have

$$c(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})) = c(\vec{\boldsymbol{\alpha}}) + \gamma \cdot c(\vec{\boldsymbol{e}}_i \cdot \vec{\alpha}) = c(\vec{\boldsymbol{\alpha}}) + \mathcal{M}(\gamma \cdot \vec{\alpha}, \vec{\Theta}_i). \tag{10}$$

Plugging in the simplification in (9) and (10) into (8), we have

$$\begin{aligned} 2\tau &\geqslant \|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| + \|\Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})) - c(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha}))\| \\ &\geqslant \|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}}) - \Gamma(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha})) + c(\vec{\boldsymbol{\alpha}} + \vec{\boldsymbol{e}}_i \cdot (\gamma \cdot \vec{\alpha}))\| \\ &= \|\mathcal{M}(\gamma \cdot \vec{\alpha}, \vec{\Theta}_i) - \mathcal{M}(\gamma \cdot \vec{\alpha}, g(\vec{u_i^*}))\| = \|\mathcal{M}(\gamma \cdot \vec{\alpha}, \vec{\Theta}_i - g(\vec{u_i^*}))\| = \|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u_i^*}))\|, \end{aligned}$$

where the last equality follows from noting that for any vector $\vec{a}$ and non-zero scalar ζ, we have $\|\zeta \cdot \vec{a}\| = \|\vec{a}\|$.

▷ Claim 11. For every $i \in [k]$, there exists $\vec{u}_i^* \in U_i$ such that for every $\vec{\alpha} \in \mathbb{F}_q^k$ we have

$$\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u_i^*}))\| \leqslant 2\tau.$$

We prove the claim for non-zero $\vec{\alpha}$ as the claim is trivial for the case $\vec{\alpha} = \vec{0}$.

For every $\vec{\alpha} \in \mathbb{F}_q^k \setminus \{\vec{0}\}$, let $\vec{u}_{\vec{\alpha}} \in U_i$ be the vector guaranteed in Claim 10, i.e., $\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}}))\| \leqslant 2\tau$.

Now consider any linearly independent $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_q^k$. We then have:

$$\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}}))\| \leqslant 2\tau,\ \|\mathcal{M}(\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\beta}}))\| \leqslant 2\tau,\ \text{and}\ \|\mathcal{M}(\vec{\alpha}+\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}+\vec{\beta}}))\| \leqslant 2\tau.$$

Putting these three inequalities together:

$$\begin{aligned}
6\tau &\geqslant \|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}}))\| + \|\mathcal{M}(\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\beta}}))\| + \|\mathcal{M}(\vec{\alpha}+\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}+\vec{\beta}}))\| \\
&\geqslant \|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}})) + \mathcal{M}(\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\beta}})) - \mathcal{M}(\vec{\alpha}+\vec{\beta}, \vec{\Theta}_i - g(\vec{u}_{\vec{\alpha}+\vec{\beta}}))\| \\
&= \|\mathcal{M}(\vec{\alpha}, g(\vec{u}_{\vec{\alpha}})) + \mathcal{M}(\vec{\beta}, g(\vec{u}_{\vec{\beta}})) - \mathcal{M}(\vec{\alpha}+\vec{\beta}, g(\vec{u}_{\vec{\alpha}+\vec{\beta}}))\| \\
&= \|\mathcal{M}(\vec{\alpha}, g(\vec{u}_{\vec{\alpha}}) - g(\vec{u}_{\vec{\alpha}+\vec{\beta}})) + \mathcal{M}(\vec{\beta}, g(\vec{u}_{\vec{\beta}}) - g(\vec{u}_{\vec{\alpha}+\vec{\beta}}))\| \\
&= \|\mathcal{M}(\vec{\alpha}, g(\vec{w})) - \mathcal{M}(\vec{\beta}, g(\vec{w}'))\|,
\end{aligned}$$

where $\vec{w} := \vec{u}_{\vec{\alpha}} - \vec{u}_{\vec{\alpha}+\vec{\beta}}$ and $\vec{w}' := \vec{u}_{\vec{\alpha}+\vec{\beta}} - \vec{u}_{\vec{\beta}}$.

If $\vec{w} \neq \vec{w}'$ then we arrive at a contradiction to (6) (since $\|\mathcal{M}(\vec{\alpha}, g(\vec{w})) - \mathcal{M}(\vec{\beta}, g(\vec{w}'))\| \leqslant 6\tau$ and $\tau = o(1)$).

Thus $\vec{w} = \vec{w}'$ which implies $\vec{u}_{\vec{\alpha}} + \vec{u}_{\vec{\beta}} = 2\vec{u}_{\vec{\alpha}+\vec{\beta}}$. Since the choice of $\vec{\alpha}$ and $\vec{\beta}$ were arbitrary linearly independent vectors, we also have:

$$\vec{u}_{\vec{\alpha}+\vec{\beta}} + \vec{u}_{\vec{\beta}} = 2\vec{u}_{\vec{\alpha}+2\vec{\beta}}, \vec{u}_{\vec{\alpha}} + \vec{u}_{\vec{\alpha}+\vec{\beta}} \quad = 2\vec{u}_{2\vec{\alpha}+\vec{\beta}}, \vec{u}_{2\vec{\alpha}+\vec{\beta}} + \vec{u}_{\vec{\beta}} = 2\vec{u}_{2\vec{\alpha}+2\vec{\beta}} = \vec{u}_{\vec{\alpha}} + \vec{u}_{\vec{\alpha}+2\vec{\beta}}.$$

We put these relationships together to obtain the following:

$$\begin{aligned}
\vec{u}_{\vec{\alpha}} &= \vec{u}_{\vec{\alpha}} + 4\vec{u}_{2\vec{\alpha}+2\vec{\beta}} - 4\vec{u}_{2\vec{\alpha}+2\vec{\beta}} = \vec{u}_{\vec{\alpha}} + \left(2\vec{u}_{2\vec{\alpha}+\vec{\beta}} + 2\vec{u}_{\vec{\beta}}\right) - \left(2\vec{u}_{\vec{\alpha}} + 2\vec{u}_{\vec{\alpha}+2\vec{\beta}}\right) \\
&= 2\vec{u}_{2\vec{\alpha}+\vec{\beta}} + 2\vec{u}_{\vec{\beta}} - 2\vec{u}_{\vec{\alpha}+2\vec{\beta}} - \vec{u}_{\vec{\alpha}} = \left(\vec{u}_{\vec{\alpha}} + \vec{u}_{\vec{\alpha}+\vec{\beta}}\right) + 2\vec{u}_{\vec{\beta}} - \left(\vec{u}_{\vec{\alpha}+\vec{\beta}} + \vec{u}_{\vec{\beta}}\right) - \vec{u}_{\vec{\alpha}} = \vec{u}_{\vec{\beta}}.
\end{aligned}$$

So we are only left to handle the cases when $\vec{\alpha}$ and $\vec{\beta}$ are linearly dependent, i.e., for some $\gamma \in \mathbb{F}_q \setminus \{0\}$ we have $\vec{\alpha} = \gamma \cdot \vec{\beta}$. In this case let $\vec{\beta}' \in \mathbb{F}_q^k$ such that it is linearly independent to $\vec{\beta}$ (and thus linearly independent to $\vec{\alpha}$ as well). From the above argument we have that $\vec{u}_{\vec{\alpha}} = \vec{u}_{\vec{\beta}'} = \vec{u}_{\vec{\beta}}$.

▷ Claim 12. We have $\vec{u}_1^* + \cdots + \vec{u}_k^* = \vec{0}$, where for all $i \in [k]$, $\vec{u}_i^*$ is the vector identified in Claim 11.

The proof idea of this claim is as follows. For every $i \in [k]$ and every $\vec{\alpha} \in \mathbb{F}_q^k$, we show that there is a line in the direction of $(\vec{\alpha}, \ldots, \vec{\alpha})$ which contains two vertices $(\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \vec{x}, \vec{y})$ and $(\vec{\boldsymbol{\alpha}}', \vec{\boldsymbol{\beta}}', \vec{x}', \vec{y}')$ such that $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\alpha}}' \in R^*$. Then, by noting that these two vertices don't have a Type 5 non-edge between them, we obtain that the linear function c evaluated at $(\vec{\alpha}, \ldots, \vec{\alpha})$ is almost $\vec{0}$. On the other hand, from Claim 11, we have that $\mathcal{M}\left(\vec{\alpha}, \sum_{i \in [k]} g(\vec{u}_i^*)\right)$ is close to $c(\vec{\alpha}, \ldots, \vec{\alpha})$. Thus, we obtain that $\mathcal{M}\left(\vec{\alpha}, \sum_{i \in [k]} g(\vec{u}_i^*)\right)$ has small relative Hamming weight for

all $\vec{\alpha} \in \mathbb{F}_q^k$. However, from (5) we know that if $\sum_{i\in[k]} \vec{u}_i^* \neq \vec{0}$ then there exists $\vec{\alpha} \in \mathbb{F}_q^k$ such that $\mathcal{M}\left(\vec{\alpha}, \sum_{i\in[k]} g(\vec{u}_i^*)\right)$ has large relative Hamming weight, and thus, we arrive at a contradiction. The formal argument follows.

Fix some non-zero $\vec{\alpha} \in \mathbb{F}_q^k$. Let L_0 be a linear line in $\mathbb{F}_q^{k^2}$ containing the point $(\vec{\alpha}, \ldots, \vec{\alpha})$. Since $|R^*|/q^{k^2} > 1/q$, there exists two points $\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha}) \in R^*$, for some $\vec{\boldsymbol{\alpha}} \in \mathbb{F}_q^{k^2}$ and $\gamma \in \mathbb{F}_q \setminus \{0\}$. From (7) we have

$$\|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| \leqslant \tau \text{ and } \|\Gamma(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha})) - c(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha}))\| \leqslant \tau. \tag{11}$$

Let $v := (\vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{\beta}}, \Gamma(\vec{\boldsymbol{\alpha}}), \Gamma(\vec{\boldsymbol{\beta}}))$ and $v' := (\vec{\boldsymbol{\alpha}} + (\vec{\alpha} \cdot \gamma, \ldots, \vec{\alpha} \cdot \gamma), \vec{\boldsymbol{\beta}}', \Gamma(\vec{\boldsymbol{\alpha}} + (\vec{\alpha} \cdot \gamma, \ldots, \vec{\alpha} \cdot \gamma)), \Gamma(\vec{\boldsymbol{\beta}}'))$ be the two vertices in T for some $\vec{\boldsymbol{\beta}}, \vec{\boldsymbol{\beta}}' \in \mathbb{F}_q^{k^2}$. Since there is no Type 5 non-edge between them, we have

$$\Gamma(\vec{\boldsymbol{\alpha}}) = \Gamma(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha})). \tag{12}$$

On a different note, we have

$$c(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha})) = c(\vec{\boldsymbol{\alpha}}) + \gamma \cdot c(\vec{\alpha}, \ldots, \vec{\alpha}) = c(\vec{\boldsymbol{\alpha}}) + \mathcal{M}\left(\gamma \cdot \vec{\alpha}, \sum_{i\in[k]} \vec{\Theta}_i\right). \tag{13}$$

Plugging in the simplification in (12) and (13) into (11), we have

$$\begin{aligned} 2\tau &\geqslant \|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}})\| + \|\Gamma(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha})) - c(\vec{\boldsymbol{\alpha}} + (\vec{\alpha} \cdot \gamma, \ldots, \vec{\alpha} \cdot \gamma))\| \\ &\geqslant \|\Gamma(\vec{\boldsymbol{\alpha}}) - c(\vec{\boldsymbol{\alpha}}) - \Gamma(\vec{\boldsymbol{\alpha}} + (\gamma \cdot \vec{\alpha}, \ldots, \gamma \cdot \vec{\alpha})) + c(\vec{\boldsymbol{\alpha}} + (\vec{\alpha} \cdot \gamma, \ldots, \vec{\alpha} \cdot \gamma))\| \\ &= \left\|\mathcal{M}\left(\gamma \cdot \vec{\alpha}, \sum_{i\in[k]} \vec{\Theta}_i\right)\right\| = \left\|\mathcal{M}\left(\vec{\alpha}, \sum_{i\in[k]} \vec{\Theta}_i\right)\right\|, \end{aligned} \tag{14}$$

where the last equality follows from noting that for any vector $\vec{a}$ and non-zero scalar ζ, we have $\|\zeta \cdot \vec{a}\| = \|\vec{a}\|$.

Next, to see the claim, we first define $\vec{z}^* \in \mathbb{F}_q^m$ as follows: $\vec{z}^* := \sum_{i\in[k]} \vec{u}_i^*$.

From Claim 11, we have that for every $i \in [k]$ and for all $\vec{\alpha} \in \mathbb{F}_q^k$ we have $\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u}_i^*))\| \leqslant 2\tau$. Fix some $\vec{\alpha} \in \mathbb{F}_q^k \setminus \{0\}$. Then,

$$2\tau k \geqslant \sum_{i\in[k]} \left\|\mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u_i^*}))\right\| \geqslant \left\|\sum_{i\in[k]} \mathcal{M}(\vec{\alpha}, \vec{\Theta}_i - g(\vec{u_i^*}))\right\| = \left\|\mathcal{M}\left(\vec{\alpha}, \sum_{i\in[k]} \left(\vec{\Theta}_i - g(\vec{u_i^*})\right)\right)\right\|.$$

Plugging in (14), we have

$$\frac{1}{2} \geqslant 2\tau(k+1) \geqslant \left\|\mathcal{M}\left(\vec{\alpha}, \sum_{i\in[k]} g(\vec{u_i^*})\right)\right\| = \left\|\mathcal{M}\left(\vec{\alpha}, g\left(\sum_{i\in[k]} \vec{u_i^*}\right)\right)\right\| = \left\|\mathcal{M}\left(\vec{\alpha}, g\left(\vec{z}^*\right)\right)\right\|.$$

Therefore, we have that for all $\vec{\alpha} \in \mathbb{F}_q^k$

$$\left\|\mathcal{M}\left(\vec{\alpha}, g\left(\vec{z}^*\right)\right)\right\| \leqslant 1/2. \tag{15}$$

From (5) we have that if $\vec{z}^* \neq \vec{0}$ then $\|g(\vec{z}^*)\| \geqslant 2/3$. We think of $g(\vec{z}^*)$ as $(\vec{b}_1, \ldots, \vec{b}_\ell)$, where $\vec{b}_i \in \mathbb{F}_q^k$, for all $i \in [\ell]$. Since $\|g(\vec{z}^*)\| \geqslant 2/3$, we have that $\Pr_{i\sim[\ell]}[\vec{b}_i = \vec{0}] \leqslant 1/3$. For every $i \in [\ell]$ and a uniformly random $\vec{\alpha} \in \mathbb{F}_q^k$ we have that

$$\Pr_{\vec{\alpha}\sim\mathbb{F}_q^k}[\langle \vec{\alpha}, \vec{b}_i\rangle = 0] = \begin{cases} \frac{1}{q} & \text{if } \vec{b}_i \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Thus, we have $\mathop{\mathbb{E}}_{\vec{\alpha}\sim\mathbb{F}_q^k}[\|\mathcal{M}(\vec{\alpha}, g(\vec{z}^*))\|] \geqslant \frac{q-1}{q} \cdot \frac{2}{3} > \frac{1}{2}$. This implies there exists $\vec{\alpha} \in \mathbb{F}_q^k$ such that $\|\mathcal{M}(\vec{\alpha}, g(\vec{z}^*))\| > 1/2$, which contradicts (15), and therefore we have $\vec{z}^* = \vec{0}$.

5 Open Problems

The main open problem left behind from this work is to prove the total FPT-inapproximability of the k-Clique problem. Apart from this open problem, we would like to highlight the following two open problems too.

- Parameterized Inapproximability Hypothesis (PIH): The PIH was putforth in [49] and asserts that it is W[1]-hard parameterized by k, to decide the satisfiability of gap 2-CSP on k variables and alphabet size n. It is easy to show that assuming Gap-ETH, the above gap 2-CSP instances do not admit FPT-approximation algorithms (for example see [10]). Previously, many researchers belived that the way to obtain inapproximability results for the parameterized k-Clique problem must be to first resolve PIH. However, Lin [47] surprisingly found a route to prove inapproximability of the k-Clique problem while circumventing past PIH. Nevertheless, since one may see PIH as a parameterized complexity analogue of the PCP theorem (for NP), it remains an outstanding open problem to be settled.
- ETH lower bound for approximating k-Clique: In [47] and this paper, we are primarily interested in proving strong hardness of approximation factors for the k-Clique problem under the W[1]≠FPT assumption. However, can we prove tighter running time lower bounds for approximating k-Clique problem under stronger assumptions such as ETH? For example, assuming ETH, can we rule out constant factor approximation algorithms for k-Clique problem running in $n^{o(k)}$ time? Both [47] and this paper can only prove a time lower bound of $n^{(\log k)^{\Omega(1)}}$ under ETH, for approximating the k-Clique to constant factors.

References

1 Amir Abboud, Kevin Lewi, and Ryan Williams. On the parameterized complexity of k-sum. *CoRR*, abs/1311.3054, 2013. arXiv:1311.3054.

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

3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

4 Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Comb.*, 23(3):365–426, 2003. doi:10.1007/s00493-003-0025-0.

5 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.

6 Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 294–304. ACM, 1993. doi:10.1145/167088.167174.

7 Mihir Bellare and Madhu Sudan. Improved non-approximability results. 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 184–193. ACM, 1994. doi:10.1145/195058.195129.

8 Piotr Berman and Georg Schnitger. On the complexity of approximating the independent set problem. *Inf. Comput.*, 96(1):77–94, 1992. doi:10.1016/0890-5401(92)90056-L.

9 Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *J. ACM*, 68(3):16:1–16:40, 2021.

10 Arnab Bhattacharyya, Suprovat Ghoshal, Karthik C. S., and Pasin Manurangsi. Parameterized intractability of even set and shortest vector problem from gap-eth. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 17:1–17:15, 2018. doi:10.4230/LIPIcs.ICALP.2018.17.

11 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.

12 Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On subexponential and fpt-time inapproximability. *Algorithmica*, 71(3):541–565, 2015. doi:10.1007/s00453-014-9889-1.

13 Boris Bukh, Karthik C. S., and Bhargav Narayanan. Inapproximability of clustering in lp metrics. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, 2021.

14 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-eth to fpt-inapproximability: Clique, dominating set, and more. *SIAM J. Comput.*, 49(4):772–810, 2020.

15 Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. On the computational hardness based on linear fpt-reductions. *J. Comb. Optim.*, 11(2):231–247, 2006. doi:10.1007/s10878-006-7137-6.

16 Yijia Chen, Martin Grohe, and Magdalena Grüber. On parameterized approximability. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2006. doi:10.1007/11847250_10.

17 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. *SIAM J. Comput.*, 48(2):513–533, 2019. doi:10.1137/17M1127211.

18 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Guy Kortsarz. Fixed-parameter and approximation algorithms: A new look. In Gregory Z. Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation - 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4-6, 2013, Revised Selected Papers*, volume 8246 of *Lecture Notes in Computer Science*, pages 110–122. Springer, 2013. doi:10.1007/978-3-319-03898-8_11.

19 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

20 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.

21 Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:128, 2016. URL: `http://eccc.hpi-web.de/report/2016/128`.

22 Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.*, 141(1&2):109–131, 1995. `doi:10.1016/0304-3975(94)00097-3`.

23 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. `doi:10.1007/978-1-4471-5559-1`.

24 Rodney G. Downey, Michael R. Fellows, and Catherine McCartin. Parameterized approximation problems. In Hans L. Bodlaender and Michael A. Langston, editors, *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006, Zürich, Switzerland, September 13-15, 2006, Proceedings*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer, 2006. `doi:10.1007/11847250_11`.

25 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`.

26 Lars Engebretsen and Jonas Holmerin. Clique is hard to approximate within $n^{1\text{-}o(1)}$. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*, volume 1853 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2000. `doi:10.1007/3-540-45022-X_2`.

27 Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discret. Math.*, 18(2):219–225, 2004. `doi:10.1137/S089548010240415X`.

28 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996. `doi:10.1145/226643.226652`.

29 Uriel Feige and Joe Kilian. Two-prover protocols - low error at affordable rates. *SIAM J. Comput.*, 30(1):324–346, 2000. `doi:10.1137/S0097539797325375`.

30 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. `doi:10.3390/a13060146`.

31 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989. `doi:10.1145/73007.73010`.

32 Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discret. Math.*, 13(4):535–570, 2000. `doi:10.1137/S0895480198344540`.

33 Venkatesan Guruswami. List decoding of binary codes-a brief survey of some recent results. In Yeow Meng Chee, Chao Li, San Ling, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology, Second International Workshop, IWCC 2009, Zhangjiajie, China, June 1-5, 2009. Proceedings*, volume 5557 of *Lecture Notes in Computer Science*, pages 97–106. Springer, 2009. `doi:10.1007/978-3-642-01877-0_10`.

34 Mohammad Taghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. The foundations of fixed parameter inapproximability. *CoRR*, abs/1310.2711, 2013. `arXiv:1310.2711`.

35 Johan Håstad. Clique is hard to approximate within $n^{1\text{-epsilon}}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636. IEEE Computer Society, 1996. `doi:10.1109/SFCS.1996.548522`.

36 Johan Håstad. Testing of the long code and hardness for clique. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 11–19. ACM, 1996. `doi:10.1145/237814.237820`.

37 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

38 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972. URL: http://www.cs.berkeley.edu/~luca/cs172/karp.pdf, doi:10.1007/978-1-4684-2001-2_9.

39 Karthik C. S. and Subhash Khot. Almost polynomial factor inapproximability for parameterized k-clique. *CoRR*, abs/2112.03983, 2021. arXiv:2112.03983.

40 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.

41 Karthik C. S. and Inbal Livni Navon. On hardness of approximation of parameterized set cover and label cover: Threshold graphs from error correcting codes. In Hung Viet Le and Valerie King, editors, *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 210–223. SIAM, 2021. doi:10.1137/1.9781611976496.24.

42 Subhash Khot. Improved inaproximability results for maxclique, chromatic number and approximate graph coloring. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 600–609. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959936.

43 Subhash Khot and Ashok Kumar Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 226–237. Springer, 2006. doi:10.1007/11786986_21.

44 Subhash Khot and Muli Safra. A two-prover one-round game with strong soundness. *Theory Comput.*, 9:863–887, 2013. doi:10.4086/toc.2013.v009a028.

45 Bingkai Lin. The parameterized complexity of the k-biclique problem. *J. ACM*, 65(5):34:1–34:23, 2018. doi:10.1145/3212622.

46 Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 81:1–81:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.81.

47 Bingkai Lin. Constant approximating k-clique is w[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021. doi:10.1145/3406325.3451016.

48 Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. On lower bounds of approximating parameterized k-clique. *CoRR*, abs/2111.14033, 2021. arXiv:2111.14033.

49 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200, 2020. doi:10.1137/1.9781611975994.134.

50 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. *CoRR*, abs/1607.02986, 2016. arXiv:1607.02986.

51 Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer journal*, 51(1):60–78, 2008.

52 Jaroslav Nesetril and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 026(2):415–419, 1985. URL: http://eudml.org/doc/17394.

53 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

54 Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

55 Saket Saurabh. What's next? Future directions in parameterized complexity. Workshop on Recent Advances in Parameterized Complexity, Tel-Aviv, Israel, 2017. URL: `https://rapctelaviv.weebly.com/uploads/1/0/5/3/105379375/future.pdf`.

56 Michal Wlodarczyk. Parameterized inapproximability for steiner orientation by gap amplification. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 104:1–104:19, 2020. `doi:10.4230/LIPIcs.ICALP.2020.104`.

57 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007. `doi:10.4086/toc.2007.v003a006`.

ℓ_p-Spread and Restricted Isometry Properties of Sparse Random Matrices

Venkatesan Guruswami
University of California, Berkeley, CA, USA

Peter Manohar
Carnegie Mellon University, Pittsburgh, PA, USA

Jonathan Mosheiff
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Random subspaces X of $\mathbb{R}^n$ of dimension proportional to n are, with high probability, well-spread with respect to the ℓ_2-norm. Namely, every nonzero $x \in X$ is "robustly non-sparse" in the following sense: x is $\varepsilon \|x\|_2$-far in ℓ_2-distance from all δn-sparse vectors, for positive constants ε, δ bounded away from 0. This "ℓ_2-spread" property is the natural counterpart, for subspaces over the reals, of the minimum distance of linear codes over finite fields, and corresponds to X being a Euclidean section of the ℓ_1 unit ball. Explicit ℓ_2-spread subspaces of dimension $\Omega(n)$, however, are unknown, and the best known explicit constructions (which achieve weaker spread properties), are analogs of *low density parity check* (LDPC) codes over the reals, i.e., they are kernels of certain *sparse* matrices.

Motivated by this, we study the spread properties of the kernels of *sparse random matrices*. We prove that with high probability such subspaces contain vectors x that are $o(1) \cdot \|x\|_2$-close to $o(n)$-sparse with respect to the ℓ_2-norm, and in particular are *not* ℓ_2-spread. This is strikingly different from the case of random LDPC codes, whose distance is asymptotically almost as good as that of (dense) random linear codes.

On the other hand, for $p < 2$ we prove that such subspaces *are* ℓ_p-spread with high probability. The spread property of sparse random matrices thus exhibits a threshold behavior at $p = 2$. Our proof for $p < 2$ moreover shows that a random sparse matrix has the stronger restricted isometry property (RIP) with respect to the ℓ_p norm, and in fact this follows solely from the unique expansion of a random biregular graph, yielding a somewhat unexpected generalization of a similar result for the ℓ_1 norm [6]. Instantiating this with suitable explicit expanders, we obtain the first explicit constructions of ℓ_p-RIP matrices for $1 \le p < p_0$, where $1 < p_0 < 2$ is an absolute constant.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Random projections and metric embeddings

Keywords and phrases Spread Subspaces, Euclidean Sections, Restricted Isometry Property, Sparse Matrices

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.7

Funding *Venkatesan Guruswami*: Supported in part by NSF grants CCF-1908125 and CCF-2210823, and a Simons Investigator Award.
Peter Manohar: Supported in part by an ARCS Scholarship, NSF Graduate Research Fellowship (under grant numbers DGE1745016 and DGE2140739), and NSF CCF-1814603. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Jonathan Mosheiff: Supported in part by NSF CCF-1814603.

Acknowledgements We thank Sidhanth Mohanty for helpful discussions about the works of [9, 7, 8, 26, 27, 29], and Yuval Peled for helpful discussions about convergence theorems for graph spectra. We thank Amir Shpilka for bringing the work of [23] to our attention, and Ioana Dumitriu for bringing the works of [9, 34] to our attention.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 7; pp. 7:1–7:17

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

1 Introduction

Classical results in asymptotic geometric analysis on the Gelfand/Kolmogorov widths of ℓ_2 balls [14, 24, 16] show that random subspaces X of $\mathbb{R}^n$ of dimension proportional to n (say, defined as the kernel of random $n/2 \times n$ matrices with i.i.d. Gaussian or ± 1 entries) are *good Euclidean sections* of ℓ_1^n: namely, $\|x\|_1 \geq \Omega(\sqrt{n})\,\|x\|_2$ for every $x \in X$. An elementary proof of this fact also follows from the Johnson-Lindenstrauss (JL) property of random matrices, its connection to the restricted isometry property (RIP) and compressed sensing, and their relationship to the Euclidean sections property [4].

The condition $\|x\|_1 \geq \Omega(\sqrt{n})\,\|x\|_2$ can equivalently be expressed as a "well-spreadness" criterion satisfied by every nonzero vector $x \in X$: the largest δn entries of x have at most $1-\varepsilon$ of its ℓ_2 mass, for some positive constants δ, ε bounded away from 0 as $n \to \infty$. Equivalently, this means that all nonzero vectors $x \in X$ are *incompressible* – there is no sparse vector that approximates x well in ℓ_2 norm (in other words, $\|x - y\|_2 \geq \varepsilon\,\|x\|_2$ for all δn-sparse vectors y). This can be naturally viewed as a robust analog, for subspaces of $\mathbb{R}^n$, of the distance property of linear error-correcting codes.

The above well-spreadness criterion can naturally be imposed with respect to any ℓ_p metric: a subspace X is said to be *ℓ_p-spread* if every nonzero vector $x \in X$ is $\varepsilon\,\|x\|_p$-far in ℓ_p-distance from all δn-sparse vectors. The ℓ_p-spread property is a more stringent requirement for larger p. For $p > 2$, the optimal asymptotic dimension of ℓ_p-spread subspaces is at most $O_p(n^{2/p})$ and thus $o(n)$ [17]. In this work, we therefore focus on $p \in [1, 2]$ where it is possible to have ℓ_p-spread subspaces of dimension proportional to n.

For a subspace X of $\mathbb{R}^n$, define its *ℓ_p-distortion* $\Delta_p(X)$ to be the following quantity:

$$\Delta_p(X) := \sup_{x \in X \setminus \{0^n\}} \frac{n^{1-\frac{1}{p}}\,\|x\|_p}{\|x\|_1} \ .$$

Note that $1 \leq \Delta_p(x) \leq n^{1-1/p}$. Good ℓ_p-spread of X can be captured by the condition that $\Delta_p(X)$ is bounded by a fixed constant independent of n; this generalizes the aforementioned equivalence of ℓ_2-spread and the Euclidean section property. The term *distortion* is used because the natural inclusion of X in $\mathbb{R}^n$ induces a bi-Lipschitz embedding of X, taken with the ℓ_p norm, into ℓ_1^n, with distortion $\Delta_p(X)$. The distortion/spread property of subspaces with respect to different ℓ_p norms has been extensively studied, owing to its connections to width properties in convex geometry [17, 25], embeddings between metric spaces [20], compressed sensing [13, 10, 25], error-correction over the reals [11, 18], and the restricted isometry (RIP) and dimensionality-reduction/Johnson-Lindenstrauss (JL) properties [25, 4, 1].

Despite a lot of interest and the abundance of probabilistic constructions, an outstanding question is to construct an *explicit* subspace $X \subseteq \mathbb{R}^n$ of dimension $\Omega(n)$ that is ℓ_2-spread, or equivalently has $\Delta_2(X) \leq O(1)$. By explicit, we mean deterministically constructing a basis for the subspace (or its dual) in poly(n) time.[1] This is a counterpart, for subspaces of $\mathbb{R}^n$, of the problem of constructing asymptotically good binary linear codes $C \subseteq \{0,1\}^n$: namely, codes whose dimension and minimum distance are both proportional to n. In addition to being a natural and basic challenge in pseudorandomness, explicit constructions are also valuable in applications of spread subspaces such as compressed sensing, as they provide a *guarantee* that the matrix will have the stipulated properties. This is particularly important since there are no known methods to efficiently certify the ℓ_2-spread (or even ℓ_p-spread) of random subspaces.

[1] Explicit constructions of ℓ_p-spread spaces of dimension $\Omega(n)$ are given in [6] ($p = 1$) and [23] ($1 \leq p < 2$).

1.1 Kernels of sparse matrices

In the case of $p = 2$, the best known explicit constructions of subspaces $X \subseteq \mathbb{R}^n$ with $\dim(X) \geq \Omega(n)$, in terms of their distortion $\Delta_2(X)$, are due to [19]. They give a construction analogous to Tanner codes from coding theory [32], combining appropriately chosen unbalanced bipartite expanders and local subspaces, to produce X with $\dim(X) \geq n - o(n)$ and $\Delta_2(X) \leq (\log n)^{O(\log\log\log n)}$ (so almost poly-logarithmic).[2] A simpler construction analogous to Sipser-Spielman codes [31], using s-regular spectral expanders and local well-spread subspaces of $\mathbb{R}^s$, was given in [18] and achieves[3] $\Delta_2(X) \leq n^{O(1/\log s)}$. An alternate probabilistic construction achieving similar parameters to [18] based on tensor products was given in [22]. The approach of [22] can further achieve distortion approaching 1 at the expense of making $\dim(X)$ smaller, but still $\Omega(n)$.

One notable attribute of the constructions above is that the subspace X can be expressed as the kernel of a matrix that is *sparse*. For instance, the construction of [18] picks a matrix where each row is s-sparse with ± 1 entries (that are chosen randomly for a probabilistic construction), and the construction in [22] defines the subspace $X \subseteq \mathbb{R}^n$ as the k-fold tensor product of another subspace, and so X can be defined as the kernel of an $n^{1/k}$-sparse matrix. Moreover, known explicit constructions of ℓ_p-spread subspaces for $1 \leq p < 2$ [6, 23] are also kernels of sparse matrices.

The sparsity of these constructions is inherited from the "underlying constructions" for codes; the constructions of [19, 18, 22] come from "lifting" constructions of linear codes (namely, Tanner codes [32], Sipser-Spielman codes [31], and tensor product codes, respectively) to this setting, and these constructions (for linear codes) are known to give good low density parity check (LDPC) codes: namely, codes that are the kernels of sparse matrices.

In light of these works, a natural question (and indeed one explicitly posed in [18]), is the following.

▶ **Question 1.** *Does there exist an $m \times n$ matrix A with $n - m \geq \Omega(n)$ whose rows are s-sparse for $s \leq O(1)$ (or even $s \leq \mathrm{polylog}(n)$) such that $\Delta_2(ker(A)) \leq O(1)$?*

The approaches of [18, 22] show that one can achieve $\Delta_2(\ker(A)) \leq \exp(O(1/\delta))$ when $s = n^\delta$. A positive answer to Question 1, even via random matrices, would likely yield good progress towards explicit constructions, as $O(1)$-sparse matrices are likely easier to derandomize than dense random ones. A negative answer to Question 1 would likely rule out explicit constructions based on the current state-of-the-art approaches of [19, 18, 22].

In addition to exploring the potential of the approaches behind the current best constructions, sparsity is desirable from a computational efficiency standpoint. Sparse matrices lead to faster algorithms, for example when used as measurement matrices in compressed sensing or to compute a sparse JL transform for dimensionality-reduction.

Motivated by these considerations, we study the ℓ_2-spread, and, more generally, ℓ_p-spread $(1 \leq p \leq 2)$ of subspaces defined as the kernel of *sparse random matrices*. Such subspaces are the continuous analogues of random low density parity check (LDPC) codes. Random LDPC codes have been studied in coding theory since Gallager's seminal work [15], with a renaissance since the mid 1990s [30] due to their fast iterative decoding algorithms and performance close to capacity.

[2] For sublinear dimension, an explicit construction of $X \subseteq \mathbb{R}^n$ with distortion $\Delta_2(X) \leq 1 + o(1)$ and $\dim(X) \geq n/2^{O((\log\log n)^2)}$ was given in [21].

[3] This construction is not explicit except for very small s, as the local subspace of $\mathbb{R}^s$ is either constructed by brute force or drawn at random.

Random LDPC codes are known to achieve rate vs. distance trade-offs approaching that of random (dense) linear codes [15]. Recently, even the list-decodability, and indeed any "local" property, of random LDPC codes was shown to be similar to that of random linear codes [28]. Given that random subspaces are well-spread and that random LDPC codes achieve similar properties to random (dense) codes, one might naturally expect, by analogy, that the kernels of sparse random matrices are also well-spread.

1.2 Our results

Our results paint a precise picture of the ℓ_p-spread of kernels X of sparse random matrices. Before stating our results, we first define ℓ_p-spread and state the random matrix model that we use.

▶ **Definition 2** (ℓ_p-spread). *Fix $p \in [1,\infty]$, $\varepsilon \in [0,1]$ and $k \le n \in \mathbb{N}$. A vector $y \in \mathbb{R}^n$ is* k-sparse *if $|\mathrm{supp}(y)| \le k$. A vector $x \in \mathbb{R}^n \setminus \{0^n\}$ is said to be* (k,ε)-ℓ_p-compressible *if there exists a k-sparse $y \in \mathbb{R}^n$ such that $\|x-y\|_p \le \varepsilon\|x\|_p$. Otherwise, we say that x is* (k,ε)-ℓ_p-spread.

A subspace $X \subseteq \mathbb{R}^n$ is (k,ε)-ℓ_p-spread *if every $x \in X \setminus \{0^n\}$ is (k,ε)-ℓ_p-spread.*

The random matrix model. A matrix $A \in \{0,1,-1\}^{m\times n}$ is said to be (s,t)-*biregular* if every row and column of A has exactly s and t nonzero entries, respectively. Let $\mathcal{M}_{m,n,s,t}$ denote the set of all (s,t)-biregular matrices in $\{0,1,-1\}^{m\times n}$.

All of our theorems for random matrices will be for a matrix A drawn uniformly at random from $\mathcal{M}_{m,n,s,t}$, where $\alpha = \frac{m}{n} = \frac{t}{s} \in (0,1)$ is a fixed constant and $n \to \infty$; for this exposition, we will use A to denote a random matrix from $\mathcal{M}_{m,n,s,t}$, and B to denote an arbitrary matrix in $\{0,1,-1\}^{m\times n}$. We additionally assume that $s := s(n) \le n^c$ for some absolute constant $0 < c < 1$, and $t = \alpha s \ge 3$. An event $\mathcal{E}$ is said to hold *with high probability* if $\lim_{n\to\infty} \Pr[\mathcal{E}] = 1$. All asymptotic notation refers to the regime of $n \to \infty$ and constant α. The constants implied by asymptotic notation are universal, unless stated otherwise. The symbols c, c', c_1 and c_2 always stand for positive universal constants, which may differ across different lemma and theorem statements.

1.2.1 Poor ℓ_2-spread of sparse random matrices

Our first theorem shows that, surprisingly, $\ker(A)$ is, with high probability, *not* ℓ_2-spread.

▶ **Theorem 3** (Poor ℓ_2-spread of $\ker(A)$). *With high probability over A, there exists an $(m^c, \frac{n^{-\Omega(\log(1/\alpha)/\log s)}}{1-\sqrt{\alpha}})$-$\ell_2$-compressible vector $x \in \ker(A)$, where $c < 1$ is an absolute constant. In particular,*

$$\Delta_2(\ker(A)) \ge (1-\sqrt{\alpha}) \cdot n^{\Omega(\log(1/\alpha)/\log s)} \ .$$

Moreover, there is a $\mathrm{poly}(n)$-time algorithm that, on input A, outputs such an x.

Choosing $s = O(1)$ in Theorem 3 (and letting α be bounded away from 1) implies[4] that $\Delta_2(\ker(A)) \ge n^{\Omega(1)}$ with high probability, and choosing $s = \mathrm{polylog}(n)$ implies $\Delta_2(\ker(A)) \ge n^{\Omega(\log(1/\alpha)/\log\log n)}$. We always trivially have $\Delta_2(\ker(A)) \le \sqrt{n}$, so not only does Theorem 3

[4] Note that since $\alpha s = t \ge 3$, we must have $\log s \ge \log \frac{1}{\alpha}$.

answer Question 1 in the negative for sparse random matrices, but it also does so in a very strong sense. For instance, when $s = O(1)$, Theorem 3 shows that $\Delta_2(\ker(A))$ is "maximally bad", up to a constant factor in the exponent.

Another point of interest is the choice $s = n^\delta$ for some fixed δ. This yields the tradeoff of $\Delta_2(\ker(A)) \geq (\frac{1}{\alpha})^{\Omega(\frac{1}{\delta})}$, which precisely matches the tradeoff (in terms of δ) achieved by both [18, 22]. While our matrix ensemble is "more random" compared to those in [18, 22], Theorem 3 can nonetheless be interpreted as giving evidence that this $\exp\left(O(\frac{1}{\delta})\right)$ tradeoff from [18, 22] is tight and inherent to sparse constructions.

Our proof of Theorem 3 is *constructive*, in the sense that we give a very simple, efficient algorithm to find such an $x \in \ker(A)$. This moreover shows that for sparse random matrices, one can efficiently *refute* the claim that $\Delta_2(\ker(A)) = O(1)$, as the vector x is a refutation witness. Our algorithm provides an interesting counterpoint to the work of [3], who gave an algorithm based on the sum-of-squares SDP hierarchy to certify that $\Delta_2(\ker(A)) \leq O(1)$ with high probability for *dense* matrices A where $\dim(\ker(A)) \leq O(\sqrt{n})$. In contrast, our algorithm succeeds when $\dim(\ker(A)) = \Omega(n)$ and the matrix A is *sparse*. The two results taken together suggest an interesting relationship between the density and $\dim(\ker(A))$ of matrices A for which we can efficiently certify or refute bounds on $\Delta_2(\ker(A))$.

We also note that, by the well-known duality formula relating Kolmorogov and Gelfand widths (see [KT07] and the references therein), Theorem 3 implies that the row span of A is far from approximating the ℓ_2-sphere in ℓ_∞ distance. Concretely, with high probability over A there exists $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$ that is $(1 - \sqrt{\alpha}) \cdot n^{\Omega(\log(1/\alpha)/\log s)}/\sqrt{n}$-far in ℓ_∞ norm from all vectors of the form $A^\top y$, where $y \in \mathbb{R}^m$.

The proof of Theorem 3 requires the following strong bound that we show on the singular values of A.

▶ **Theorem 4** (Singular value bound). *With high probability, the set of singular values $\sigma(A)$ of A satisfy*

$$\sigma(A) \subseteq \left[\sqrt{s-1} - (1+o(1)) \cdot \sqrt{t-1}, \sqrt{s-1} + (1+o(1)) \cdot \sqrt{t-1}\right] .$$

Moreover, the above bound holds even without our (otherwise global) assumption that $s \leq n^c$ for some absolute constant $c < 1$.

Theorem 4 should not be surprising, especially given the recent works of [9, 7, 8, 26, 27, 29, 34], and indeed our proof follows the same overall blueprint of these works. Most of these papers, however, only handle the case when the degree of the graph is *constant* as $n \to \infty$;[5] this corresponds to the case of $s = O(1)$ in Theorem 4. Theorem 4 thus differs as it allows for $s = \omega(1)$, and indeed we can even take $s = n^c$ for some absolute constant $c < 1$. On the other hand, most of the aforementioned works deal with the case of *unsigned* adjacency matrices, whereas we only prove Theorem 4 for *randomly signed* adjacency matrices. We note that proving the analogue of Theorem 4 for unsigned adjacency matrices (where $\sigma(A)$ now denotes the set of singular values, excluding the trivial value of $\sqrt{st}$) and for all (non-constant) s, t remains open.

The singular value bound in Theorem 4 is challenging to prove because it is so sharp. Indeed, it is not too difficult to show that $\sigma(A) \subseteq [\sqrt{s} - O(\sqrt{t}), \sqrt{s} + O(\sqrt{t})]$ with high probability via black-box applications of known results, e.g., [2]. However, this does not

[5] The exceptions are [26], which handles polylog(n) degree, and [34], which handles n^c degree but does not obtain as sharp bounds.

suffice for our use in the proof of Theorem 3, as the aforementioned weaker bound would only suffice to prove Theorem 3 provided that $\alpha \le c$ for some absolute constant c, where c depends on the absolute constant c' hidden in the "$O(\sqrt{t})$". We need the sharp bound of Theorem 4 in order to allow for α to be an arbitrary constant in $(0,1)$.

As a counterpart to Theorem 3, we give the following partial converse, which shows that $\ker(A)$ *is* (k,ε)-ℓ_2-spread for a weak choice of parameters k and ε.

▶ **Theorem 5** (Converse to Theorem 3)**.** *Assume that $t \ge 9$. Then, with high probability over A, the space* $\ker(A)$ *is* $\left(\Omega(\alpha^2 n/t^4), \alpha^{O(\log n/\log t)}\right)$*-$\ell_2$-spread.*

We note that in Theorem 5, the parameter k is $\Omega(\alpha^2 n/t^4) = m^{\Omega(1)}$ and the parameter ε is $\alpha^{O(\log n/\log s)}$.[6] Theorem 5 thus shows that the parameters in Theorem 3 are tight up to the universal constants in the exponent. Our proof of Theorem 5 is an adaptation of the proof of [5, Lemma 3.4].

1.2.2 ℓ_p-RIP and ℓ_p-spread for $p < 2$

We next focus on the ℓ_p norm for $p < 2$. For $p < 2$, there are known explicit constructions of ℓ_p-spread subspaces [23]. Because of this, for $p < 2$ we focus on the stronger, well-studied *Restricted Isometry Property (RIP)*. We also note that the constructions of [23] are highly structured, and so even though they also come from sparse matrices, they do not tell us anything about the ℓ_p-spread of sparse *random* matrices.

We prove that sparse random matrices are not only ℓ_p-spread, but are also ℓ_p-RIP, and, moreover, this follows merely from the expansion of the underlying bipartite graph of the random matrix A. In particular, we prove that *any* signed adjacency matrix B of a left-regular bipartite expander graph G is ℓ_p-RIP, provided that the maximum right degree $s_{\max}$ is above a small threshold independent of n.

The RIP is a well-studied property of matrices from the compressed sensing literature, defined as follows.

▶ **Definition 6** (ℓ_p-RIP)**.** *Let $B \in \mathbb{R}^{m\times n}$ be a matrix. We say that B is* (k,ε)-ℓ_p-RIP *if there exists $K > 0$ such that for every k-sparse $x \in \mathbb{R}^n$, it holds that*[7]

$$K(1-\varepsilon)\|x\|_p \le \|Bx\|_p \le K(1+\varepsilon)\|x\|_p \quad .$$

We note that ℓ_p-RIP implies ℓ_p-spread, and in fact it is a strictly stronger property [25].

RIP matrices have been studied extensively in the context of compressed sensing, as they yield a polynomial-time algorithm based on linear programming for the *robust sparse recovery problem*. Namely, given a "noisy measurement sketch" $y = Bx + e$ of a vector x, where B is (k,ε)-ℓ_p-RIP and $\|e\|_p \le \eta$, there is a polynomial-time algorithm to recover an estimate $\hat{x}$ for x with the so-called "ℓ_p-ℓ_1 guarantee," namely the estimate $\hat{x}$ satisfies $\|\hat{x} - x\|_p \le O\left(k^{-(1-\frac{1}{p})}\|x - x^*\|_1 + \eta\right)$, where x^* is a k-sparse vector minimizing $\|x - x^*\|_1$ (see Appendix A in [1] for details). We note that if B is merely ℓ_p-spread, then B suffices for the (non-robust) sparse recovery problem, i.e., when there is no noise e.

We now turn to formally stating our results. We first recall the definition of a (unique) bipartite expander.

---

[6] This follows since $t = \alpha s \le s$, $m = \alpha n$, and $s \le n^c$ for some absolute constant c.

[7] We note that the standard definition of RIP typically appears without the normalization factor K above. We include the parameter K for convenience, as the random sparse matrices we consider are not normalized.

▶ **Definition 7** (Unique expanders). *A bipartite graph $G = (V_L = [n], V_R = [m], E)$ is a t-left-regular (γ, μ)-unique expander if (1) $\deg(u) = t$ for all $u \in V_L$, and (2) for all $S \subseteq V_L$, $|S| \leq \gamma n$, there are at least $t(1-\mu)|S|$ vertices $v \in V_R$ which each have exactly one neighbor in S.*

A matrix $B \in \{0, 1, -1\}^{m \times n}$ is a *signed adjacency matrix* of a bipartite graph $G = (V_L = [n], V_R = [m], E)$ if

$$B_{r,u} \neq 0 \iff (u, r) \in E$$

for all $u \in V_L$, $r \in V_R$.

▶ **Theorem 8** (ℓ_p-RIP of expander graphs). *Let G be a bipartite t-left-regular (γ, μ)-unique expander with maximum right degree $s_{\max}$, and let B be any signed adjacency matrix of G. Let $0 < \varepsilon \leq 1$ and $1 \leq p < 2$ such that $\varepsilon^2 \geq 9\mu s_{\max}^{p-1}$. Then, B is $(\gamma n, \varepsilon)$-ℓ_p-RIP, i.e., for every γn-sparse $x \in \mathbb{R}^n$,*

$$t^{\frac{1}{p}}(1-\varepsilon)\|x\|_p \leq \|Bx\|_p \leq t^{\frac{1}{p}}(1+\varepsilon)\|x\|_p \quad .$$

Theorem 8 generalizes a result of [6], which shows that any signed adjacency matrix B of G is ℓ_1-RIP, provided that G is an expander. This is somewhat surprising, as the proof in [6] makes heavy use of properties specific to the ℓ_1 norm.[8]

The ℓ_p-RIP of matrices for general p has been studied in other contexts, most notably in [1]. As is typical when studying RIP matrices, they view the sparsity parameter k as a fixed function of n, and determine m as a function of k, n. However, the results in [1] are incomparable to ours, as they hold only for the low-sparsity case of $k = O(n^{1/p})$ (so $k = o(n)$ if $p > 1$), but we are concerned with the case of $k = \Omega(n)$, when the sparsity is a small constant fraction of n.

As a random t-left-regular bipartite graph is a good expander with high probability, we obtain the following corollary of Theorem 8, which shows that $\ker(A)$ for $A \leftarrow \mathcal{M}_{m,n,s,t}$ achieves very good ℓ_p-spread for every $p \in [1, 2)$. Thus, the poor ℓ_2-spread of $\ker(A)$ is in fact specific to the case of $p = 2$.

▶ **Corollary 9** (Good ℓ_p-RIP and ℓ_p-spread of A). *Fix $p \in [1, 2)$, $0 < \varepsilon < \frac{1}{2}$, and suppose that $s \geq \left(\frac{18}{\alpha\varepsilon^2}\right)^{\frac{1}{2-p}}$. Then, with high probability over A, the matrix A is $(\Omega(\gamma n), \varepsilon)$-$\ell_p$-RIP for $\gamma = \frac{\alpha^2}{t^4}$: for every $\Omega(\gamma n)$-sparse $x \in \mathbb{R}^n$, it holds that*

$$t^{\frac{1}{p}}(1-\varepsilon)\|x\|_p \leq \|Ax\|_p \leq t^{\frac{1}{p}}(1+\varepsilon)\|x\|_p \quad .$$

In particular, the subspace $\ker(A)$ is $\left(\Omega(\gamma n), \Omega\left(\gamma^{1-\frac{1}{p}}\right)\right)$-$\ell_p$-spread and $\Delta_p(\ker(A)) \leq O\left(1/\gamma^{2-\frac{2}{p}}\right)$.

Fixing p, α, ε to be constants and taking s to be a large enough constant, this shows that $\ker(A)$ is $(\Omega(n), \Omega(1))$-ℓ_p-spread with high probability, and therefore $\Delta_p(\ker(A)) = O(1)$. Together with Theorem 3, this shows that the ℓ_p-spread property of $\ker(A)$ exhibits an interesting *threshold phenomenon* at $p = 2$.

[8] They also show that their proof for ℓ_1-RIP extends to ℓ_p-RIP for $p \leq 1 + O(\frac{1}{\log n})$, because the "Hölder factor" of $n^{1-\frac{1}{p}}$ is $O(1)$, but it does not extend to ℓ_p for any constant $p > 1$.

We also combine Theorem 8 with the explicit constructions of expander graphs of [12] to obtain the following corollary, which gives an explicit construction of ℓ_p-RIP matrices for all $p \in [1, p_0)$, where $1 < p_0 < 2$ is an absolute constant. We thus obtain the first explicit construction of a matrix B achieving the "ℓ_p/ℓ_1 guarantee" for the robust sparse recovery problem, and our matrices are for the regime $k = \Theta(n)$ and any $p \in [1, p_0)$. Previously, such constructions were only known for $p \le 1 + O\left(\frac{1}{\log n}\right)$ [6]. Unlike Corollary 9, our explicit constructions only extend up to some threshold $p_0 < 2$. This is because the expanders of [12] achieve weaker expansion than random graphs. Concretely, the "expansion error" μ of the [12] expanders is $\mu = O(1/t)^\tau$ for some constant $\tau < 1$, which yields the threshold of $p_0 = 1 + \tau$, whereas random graphs achieve $\mu = O(1/t)$, allowing for $p_0 = 2$.

▶ **Corollary 10** (Explicit construction of ℓ_p-RIP matrices). *Let $0 < \varepsilon < \frac{1}{2}$, $\alpha \in (0,1)$, and let $n \in \mathbb{N}$ be sufficiently large. For some universal constant $1 < p_0 < 2$, there exists a deterministic algorithm which, given $p \in [1, p_0)$, ε, α and n, outputs in time $\mathrm{poly}(n/\delta) + 2^{O(1/\delta)}$ a matrix $B \in \{0,1\}^{m \times n}$, for some $m \le \alpha n$, such that B is $(\gamma n, \varepsilon)$-ℓ_p-RIP, for some $\delta, \gamma = \mathrm{poly}(\varepsilon, \alpha)^{\frac{1}{p_0 - p}}$. In particular, $\ker(B)$ is $(\gamma n, \gamma^{1-\frac{1}{p}})$-$\ell_p$-spread and $\Delta_p(\ker(B)) \le 1/\gamma^{2-\frac{2}{p}}$.*

Note that as ε, α and p are constants, the matrix B in Corollary 10 is $(\Omega(n), O(1))$-ℓ_p-RIP, $\ker(B)$ is $(\Omega(n), \Omega(1))$-ℓ_p-spread, and $\Delta_p(\ker(B)) \le O(1)$.

As noted earlier, [23] gives explicit constructions of $(\Omega(n), \Omega(1))$-ℓ_p-spread subspaces, for *all* $1 \le p < 2$. This is incomparable to Corollary 10: on one hand, [23] obtains the full range of $1 \le p < 2$, but on the other hand, his matrices only are ℓ_p-spread and do not satisfy the (strictly stronger) ℓ_p-RIP. Our construction is moreover the "simplest" black-box reduction to expansion: we show that the mere adjacency matrix of a bipartite expander is ℓ_p-RIP. While the constructions in [23] are themselves not too complicated, we think that this is nonetheless an interesting conceptual contribution of our work.

Finally, we also prove the following partial converse to Corollary 9, which shows that when $s^{2-p} \lessapprox \frac{1}{\alpha}$ (i.e., s^{2-p} is a constant factor below the threshold in Corollary 9), then A is *not* ℓ_p-RIP.

▶ **Theorem 11** (Partial converse to Corollary 9). *Let $p \in [1,2)$, $\varepsilon > 0$. If $s - 1 \le \left(\frac{1}{(1+\varepsilon)\alpha}\right)^{\frac{1}{2-p}}$, then with high probability over A, there exists an n^c-sparse vector $x \in \mathbb{R}^n \setminus \{0^n\}$ such that*

$$\frac{\|Ax\|_p}{\|x\|_p} \le t^{\frac{1}{p}} \cdot m^{-\Omega\left(\frac{\varepsilon}{\log s}\right)} .$$

Note that $\|Ae_1\|_p / \|e_1\|_p = t^{\frac{1}{p}}$ always holds, so Corollary 9 demonstrates that, given small enough s, the ratio $\frac{\|Ax\|_p}{\|x\|_p}$ has a large range over different choices of n^c-sparse x.

2 Proof overview

We outline the proofs of our results. For the purposes of this exposition, we will adopt the same convention as in Section 1.2 and use A and B to denote a uniformly sampled matrix from $\mathcal{M}_{m,n,s,t}$ and arbitrary matrix from $\{0,1,-1\}^{m \times n}$, respectively. Recall that $\mathcal{M}_{m,n,s,t}$ denotes the set of (s,t)-biregular matrices with entries in $\{0,1,-1\}$, and that $\frac{m}{n} = \frac{t}{s} = \alpha$ for some constant α, and $n \to \infty$. For simplicity of this exposition, in this section we restrict ourselves to the regime $s = O(1)$ unless stated otherwise.

We naturally associate with B the bipartite graph $G = G_B = (V_L, V_R, E)$ with $n = |V_L|$ left vertices, $m = |V_R|$ right vertices, and an edge between $u \in V_L$ and $r \in V_R$ if $B_{r,u} \neq 0$. We view the rows and columns of B as indexed by V_R and V_L, respectively, and identify $\mathbb{R}^n$ with $\mathbb{R}^{V_L}$, and $\mathbb{R}^m$ with $\mathbb{R}^{V_R}$. In addition, we define the function $\mathsf{sign} = \mathsf{sign}_B : E \to \{1, -1\}$, which maps an edge $\{u, r\}$ as above to $B_{r,u}$. We note that the combination of G_B and sign_B completely describes B.

2.1 Theorem 3: $\ker(A)$ in not ℓ_2-spread

For simplicity, we only sketch here why $\ker(A)$ is likely to contain an $(o(n), o(1))$-compressible vector.

The proof of Theorem 3 consists of two steps. In the first step we find an $o(n)$-sparse vector $x \in \mathbb{R}^n$ with $\|x\|_2 \geq 1$ and $\|Ax\|_2 \leq o(1)$. In the second step we find a vector $y \in \ker(A)$ with $\|y - x\|_2 \leq o(1) \cdot \|y\|_2$. In particular, y is $(o(n), o(1))$-compressible, so $\ker(A)$ cannot be ℓ_2-spread.

Below, we outline these two steps. It is straightforward to see, given the construction described below, that both x and y can be computed in polynomial time given A.

We also note that an ℓ_p analog of Step 1 is the main technical component in the proof of Theorem 11.

Step 1: constructing a sparse x with small $\|Ax\|_2$. To obtain the vector x, we first prove that G is highly likely to contain a vertex $v^* \in V_L$ such that the ball of radius $2\ell + 1$ about v^*, for some $\ell \leq O(\log n)$, contains no cycles. That is, the radius-ℓ neighborhood of v^* is a complete (t, s)-biregular tree T rooted at v^*. Recall that a rooted tree is (t, s)-biregular if the even depth (resp. odd depth) inner vertices have degree t (resp. s). The existence of such a vertex v^* is the only random property of A needed in this step of the proof. In particular, assuming that G has the aforementioned property, our construction of x is always possible, regardless of the sign function.

To describe the construction of x itself, we assume for simplicity that $\mathsf{sign}(e) = 1$ for all $e \in E$. Namely, all the non-zero entries of A are 1. In this setting, let $v \in V_L \cap T$ be a vertex of depth $2k$ in the tree for some $k \geq 0$ (note that a vertex in V_L must have even depth), and set $x_v = (-(s-1))^{-k}$. For any $x \in V_L \setminus T$, set $x_v = 0$. Note that $\mathrm{supp}(x) \subseteq T$. We choose ℓ above to be as large as possible, i.e., $O(\log n)$, so that the size of T is roughly n^c for some $c < 1$. In particular, x is $\approx n^c$-sparse. Also, note that $\|x\|_2^2 \geq x_{v^*}^2 = 1$. We informally refer to the vector x produced by this construction as a *tree vector*.

Our construction guarantees that $(Ax)_r = 0$ for every internal node $r \in T \cap V_R$. Indeed, suppose that r is of depth $2k + 1$. Then, it has one neighbor of depth $2k$, and $s - 1$ neighbors of depth $2k + 2$. As $(Ax)_r$ is the sum of x_v over neighbors v of r, we have $(Ax)_r = (-(s-1))^{-k} + (s-1) \cdot (-(s-1))^{-(k+1)} = 0$.

To compute $\|Ax\|_2$, it thus suffices to compute $|(Ax)_r|$ when r is one of the $t(t-1)^\ell(s-1)^\ell$ leaves of T. It is not hard to see that in this case $|(Ax)_r| = (s-1)^\ell$, and so

$$\|Ax\|_2^2 = t(t-1)^\ell(s-1)^\ell \cdot (s-1)^{-2\ell} = e^{-\Omega(\ell)} = o(1) \ .$$

We note that our tree vector construction is similar in spirit to a construction by Noga Alon [18, Theorem 8], which demonstrates the limitations of expander-based analysis of the spread property. In [18], however, they *choose* their graph G so that (their analog of) the tree vector x will lie in (their analog of) $\ker(A)$ *by design*. Our graph is random and not

up to our choice, so we cannot simply orchestrate the graph so that our tree vector x to belong to $\ker(A)$. This necessitates that we perform the nontrivial step of rounding x to some $y \in \ker(A)$, which we discuss next.

Step 2: finding $y \in \ker(A)$ close to x. Our main goal in this step is to establish the following lemma:

▶ **Lemma 12** (Informal). *With high probability over A, it holds that every $x \in \mathbb{R}^n$ with $\|Ax\|_2 \le o(1)$ is $o(1)$-close to some vector $y \in \ker(A)$.*

Indeed, let x be the tree vector constructed in Step 1. Then x is $o(n)$-sparse with $\|x\|_2 \ge 1$ and $\|Ax\|_2 \le o(1)$. By Lemma 12, there exists a vector $y \in \ker(A)$, which is $o(1)$-close to x. This vector y is $(o(n), o(1))$-compressible, which yields Theorem 3 in the present parameter setting.

One may naively try to prove Lemma 12 by locally perturbing x to try to make $Ax = 0^m$, e.g. by designing a greedy algorithm for this task. This approach, however, seems difficult to execute, especially given that Lemma 12 is in fact not true in general. For example, it could be the case that x is a (unit norm) right singular vector of A with singular value $o(1)$. Then, $\|Ax\|_2 = o(1)$, but $\|x - y\|_2 \ge 1$ for all $y \in \ker(A)$, and in fact the closest vector in $\ker(A)$ to x is 0^n.

Instead, we set y to be the orthogonal projection of x onto $\ker(A)$. In hindsight, this is the obvious choice for y, as then $y \in \ker(A)$ is the vector that minimizes $\|x - y\|_2$. How large can $\|x - y\|_2$ be? Intuitively, we would like to say that $\|Ax\|_2$ being small implies that $\|x - y\|_2$ is small as well. As the earlier example shows, this is not true for a general matrix A, as A could have small singular values. However, the implication *does* hold provided that all singular values of A are $\Omega(1)$.[9] Indeed, the singular value decomposition of A implies that

$$\|Ax\|_2 = \|A(x-y)\|_2 \ge \sigma_{\min}(A)\,\|x-y\|_2 \quad ,$$

where $\sigma_{\min}(A)$ is the minimum singular value of A and the inequality holds as $x - y$ is orthogonal to $\ker(A)$. Hence, $\|x - y\|_2 \le \frac{\|Ax\|_2}{\sigma_{\min}(A)}$.

The main technical component of Step 2 is therefore the lower bound on $\sigma_{\min}(A)$, given by Theorem 4. As we have argued above, the crude lower bound of $\sigma_{\min}(A) \ge \Omega(1)$ suffices to yield Lemma 12. Indeed, if $\sigma_{\min}(A) \ge \Omega(1)$, then $\|x - y\|_2 \le o(1)$, and so $\ker(A)$ contains an $(o(n), o(1))$-compressible vector. The precise high-probability lower bound on $\sigma_{\min}(A)$ established in Theorem 4 implies a finer quantitative version of Lemma 12, which yields the full Theorem 3. The latter gives a much sharper bound on the $o(1)$ term, and also applies to sparsity all the way up to $O(n^c)$ for some $c > 0$.

We remark that one can easily show that $\sigma_{\min}(A) \ge \sqrt{s} - O(\sqrt{t})$ via "off-the-shelf" methods, such as [2]. However, this would only allow us to prove Theorem 3 provided that $\alpha \le c$ for some absolute constant $c < 1$ (related to the $O(1)$ factor in front of $\sqrt{t}$ above), and thus would not allow us to take α to be *any* constant in $(0, 1)$, e.g., $\alpha = 0.999$. Our sharper bound also highlights the difficulty in lower bounding the minimum singular value when $\alpha = m/n$ is close to 1.

We postpone our discussion of the proof of Theorem 4 to Section 2.3, and turn next to our positive result for ℓ_p-spread for $p < 2$.

[9] Technically, what matters is the minimum *nonzero* singular value. However, with high probability the matrix A will be full rank (i.e., rank m), so that $\sigma_{\min}(A) > 0$. Indeed, this is trivially implied by Theorem 4.

2.2 Theorem 8: ℓ_p-RIP for $p < 2$ from vertex expansion

We sketch the proof of Theorem 8. For simplicity, we will assume that $B \in \mathcal{M}_{m,n,s,t}$, i.e., that the bipartite graph G_B is t-left-regular and s-right-regular (and hence $s_{\max} = s$) and also that G_B is a (γ, μ)-unique expander. For this exposition, we only discuss the claimed lower bound on $\|Bx\|_p$ stated in the theorem, namely,

$$\|Bx\|_p \geq t^{\frac{1}{p}}(1-\varepsilon)\|x\|_p \tag{1}$$

for all γn-sparse $x \in \mathbb{R}^n$, as the upper bound is obtained via a variation on the same method.

Theorem 8 for tree vectors. As a warm-up for the proof of Theorem 8, we show why the $o(n)$-sparse tree vector x constructed in Section 2.1 does not yield a counterexample to Equation (1). Let $x^{(k)}$ $(0 \leq k \leq \ell)$ denote the restriction of x to the vertices in the $2k$-th level of the tree T. Then,

$$\left\|x^{(k)}\right\|_p^p = t(t-1)^{k-1}(s-1)^{(1-p)k} \ .$$

For $p = 2$, this expression decreases exponentially in k, and thus the ℓ_2-mass of x is concentrated at the top of the tree. For $p < 2$, however, $\left\|x^{(k)}\right\|_p^p$ actually *grows* exponentially in k provided that s is large enough (concretely, one needs $s^{2-p} \gtrapprox \frac{1}{\alpha}$).[10] In this case, the ℓ_p-mass is concentrated towards the bottom of the tree. Moreover, one can take s large enough so that all but an $\frac{\varepsilon}{2}$-fraction of the mass lies in the bottom layer. Then,

$$\frac{\|Bx\|_p^p}{\|x\|_p^p} \geq \left(1-\frac{\varepsilon}{2}\right)\cdot\frac{\|Bx\|_p^p}{\left\|x^{(\ell)}\right\|_p^p} = \left(1-\frac{\varepsilon}{2}\right)\cdot\frac{t(t-1)^{\ell}(s-1)^{(1-p)\ell}}{t(t-1)^{\ell-1}(s-1)^{(1-p)\ell}} = \left(1-\frac{\varepsilon}{2}\right)\cdot(t-1) \geq (1-\varepsilon)t \ .$$

Hence, x is not a counterexample to Equation (1).

Theorem 8 for general vectors with tree-shaped support. Fix a set $S \subseteq V_L$ such that the subgraph induced by $T := S \cup N(S)$ is a (t,s)-biregular tree. We generalize the above discussion of tree vectors by explaining why Equation (1) holds for any vector $x \in \mathbb{R}^n$ supported on S.

Given $r \in N(S)$, let v_r denote the parent of r in the tree T. In an overly optimistic scenario, if we could show that $|(Bx)_r| \approx |v_r|$ for all $r \in N(S)$, then we would be done, as each vertex $v \in S$ has $t-1$ children.[11] Each of these children then contributes $\approx |x_v|^p$ mass to $\|Bx\|_p^p$, so that $\|Bx\|_p^p \approx (t-1)\cdot\sum_{v\in S}|x_v|^p = (t-1)\cdot\|x\|_p^p$, implying Equation (1). As the tree vector case shows, one cannot, in fact, hope to guarantee $|(Bx)_r| \approx |v_r|$ for all $r \in N(S)$. Indeed, for a tree vector x we have $(Bx)_r = 0$ for any non-leaf $r \in N(S)$. Thus, a more delicate analysis is required.

For intuition, let us consider the viewpoint of an adversary seeking to construct an x supported on S such that $\|Bx\|_p^p$ is small. We shall think of the adversary as assigning values to $\{x_v\}_{v\in S}$ starting from the root, and then moving down the tree.

For each non-leaf $r \in N(S)$, let W_r denote the set of $s-1$ children of r. Recall that v_r is the parent of r. Note that $|(Bx)_r| \geq |x_{v_r}| - \sum_{u\in W_r}|x_u|$ due to the triangle inequality. Hence, when assigning values to the vertices in W_r, the adversary morally has two choices: (1) either

[10] And, indeed, if instead $s^{2-p} \lessapprox \frac{1}{\alpha}$, then we have $\|Bx\|_p = o(1)$, and this gives us Theorem 11.
[11] Except for the root, which has t children.

make $\sum_{u\in W_r}|x_u| \ll |x_{v_r}|$, in which case $|(Bx)_r| \approx |x_{v_r}|$, or (2) make $|(Bx)_r| \approx 0$, in which case $\sum_{u\in W_r}|x_u| \approx |x_{v_r}|$.[12] Let us fix some $\beta < 1$, and suppose that the adversary chooses values for $\{x_u\}_{u\in W_r}$ such that $\sum_{u\in W_r}|x_u| \le \beta|x_v|$, i.e., the adversary chooses Case (1). We then have that

$$|(Bx)_r| \ge \left(|x_{v_r}| - \sum_{u\in W_r}|x_u|\right) \ge (1-\beta)|x_{v_r}| \ ,$$

so $|(Bx)_r| \approx |x_{v_r}|$, which is what we wanted. Next, suppose that the adversary makes $\sum_{u\in W_r}|x_u| \ge \beta|x_v|$, i.e., the adversary chooses Case (2). Then, $|(Bx)_r|$ can be small, but applying Hölder's inequality, we have

$$\sum_{u\in W_r}|x_u|^p \ge \frac{\beta^p}{(s-1)^{p-1}}\cdot|x_v|^p \ .$$

Now, suppose that all children r of v_r have this property. Then, the total ℓ_p^p mass of all of the grandchildren of v_r must be at least $\frac{(t-1)\beta^p}{(s-1)^{p-1}}\cdot|x_v|^p \gg |x_v|^p$. We thus see that, intuitively, the adversary has merely pushed its task down to the grandchildren of v_r, and in doing so has not made any progress towards its overall goal. Indeed, this is precisely what happens in the case of a tree vector!

The above informal argument shows that the adversary does not "win" in either case. We can concretely capture this intuition via the following potential function:

$$a_r(x) := |(Bx)_r|^p + \Theta(1)\cdot\frac{(s-1)^{p-1}}{\varepsilon^{p-1}}\sum_{u\in W_r}|x_u|^p \ .$$

In the actual proof, this choice of $a_r(x)$ allows us to cleanly express the intuition that either $|(Bx)_r|$ is large or $\sum_{u\in W_r}|x_u|$ is large, and further extends beyond the "toy case" of tree-supported vectors.

Using expansion when $S\cup N(S)$ is not a tree. We now turn to the general case, where the subgraph induced by $S\cup N(S)$ (where $S = \mathrm{supp}(x)$) is not necessarily a tree. We observe that above, we are only using the tree structure to show that the rooted tree $S\cup N(S)$ trivially has a 1-to-$(t-1)$ "matching" with the following properties: (1) every vertex $r\in N(S)$ is matched with exactly one vertex $v\in S$, and (2) every vertex $v\in S$ is matched with at least $t-1$ vertices in $N(S)$. Indeed, when $S\cup N(S)$ is a tree, such a matching exists by matching each vertex $r\in N(S)$ with its parent v_r.

To generalize the above, we use the (unique) expansion of G to construct a similar matching that suffices for the proof. Recall that G is a (γ,μ)-unique expander, meaning that every set $S\subseteq V_L$ of size $\le \gamma n$ has at least $t(1-\mu)|S|$ unique neighbors, i.e., neighbors of a unique element of S. We construct the matching by "peeling off" vertices one at a time from S, each time matching a vertex with $\ge t(1-\mu)$ vertices in $N(S)$, namely its neighbors that are not neighbors of any of the remaining "unpeeled" vertices in S.

The above step can be viewed as extracting a "tree-like" subgraph from $S\cup N(S)$, where each vertex $v\in S$ has at least $t(1-\mu)$ "children" (the vertices it was matched with), and at most μt "parents" (its neighbors that it was *not* matched with). Each vertex $r\in N(S)$

[12] Note that the adversary has the third choice of setting $\sum_{u\in W_r}|x_u| \gg |x_{v_r}|$, but this is worse for the adversary.

still has exactly one "parent" and $\leq s-1$ "children". Once we have the above "tree-like" subgraph, the argument for trees goes through with only minor modifications, so this finishes the proof.

We note that the existence of this "tree-like" subgraph for any set S with $|S| \leq \gamma n$ immediately implies that G is a (γ, μ)-vertex expander, and hence a $(\gamma, 2\mu)$-unique expander. Thus, the existence of such a subgraph for every S of size at most γn is equivalent to unique expansion, up to a factor of 2 loss in the parameter μ.

Comparison with [6]. We briefly summarize the proof in [6] for the case of $p = 1$, and explain why their proof does not extend to the case of $p > 1$.

The proof in [6] proceeds as follows. For a vector x supported on S, let E_0 denote the set of edges between S and $N(S)$. First, they match each $r \in N(S)$ to its neighbor $v \in S$ with $|x_v|$ maximized. Let E_1 be the set of edges in this matching, and let $E_2 = E_0 \setminus E_1$. For any $r \in N(S)$, it then follows that $|(Bx)_r| \geq |x_v| - \sum_{u \in W_r} |x_u|$, where $(v, r) \in E_1$ and $W_r = \{u : (u, r) \in E_2\}$. Hence, $\|Bx\|_1 \geq \sum_{(v,r)\in E_1} |x_v| - \sum_{(u,r)\in E_2} |x_u|$. We observe that this step of the proof is specific to the ℓ_1 norm, and does not generalize to larger ℓ_p norms.

The main step in the proof is to argue that $\sum_{(v,r)\in E_2} |x_v| \leq t\varepsilon \|x\|_1$ using expansion. With this in hand, it immediately follows that $\sum_{(v,r)\in E_1} |x_v| \geq t(1-\varepsilon)\|x\|_1$, because $\sum_{(v,r)\in E_0} |x_v| = t\|x\|_1$ by regularity. It then follows that $\|Bx\|_1 \geq t(1-2\varepsilon)\|x\|_1$. Note that the upper bound $\|Bx\|_1 \leq t\|x\|_1$ is trivial, so this shows that B is ℓ_1-RIP.

One may attempt to generalize this proof to $p > 1$ by replacing $|x_v|$ with $|x_v|^p$. For example, using expansion it follows that $\sum_{(v,r)\in E_2} |x_v|^p \leq t\varepsilon \|x\|_p^p$, and as $\sum_{(v,r)\in E_0} |x_v|^p = t\|x\|_p^p$, we then have $\sum_{(v,r)\in E_1} |x_v|^p \geq t(1-\varepsilon)\|x\|_p^p$. But this is not enough to complete the proof, as it does *not* follow that $\|Bx\|_p^p \geq \sum_{(v,r)\in E_1} |x_v|^p - \sum_{(v,r)\in E_2} |x_v|^p$. Indeed, this is a fundamental barrier, and is the reason why our analysis for $p \geq 1$ proceeds by analyzing the "local" potential function $a_r(x)$, rather than the two "global" sums over E_1 and E_2 above.

2.3 Theorem 4: bounds on the singular values of A

We give a brief overview of the proof of Theorem 4. First, we observe that in order to bound the singular values of A, it suffices to bound the spectrum of $M := AA^\top - s \cdot I$, as each singular value of A is the square root of an eigenvalue of $AA^\top$. Note that M is a square matrix with an all-0 diagonal, by regularity of A.

Step 1: reducing to the nomadic walk matrix via a modified Ihara–Bass formula. The first step in the proof is to relate bounds on the spectrum of M to the spectral radius (i.e., maximum eigenvalue in absolute value) $\rho(B)$ of B, the *nomadic walk matrix* introduced in [27].[13] The nomadic walk matrix B is indexed by pairs of edges[14] (e_1, e_2) in G that form a length 2 non-backtracking walk in G, and its $((e_1, e_2), (e_1', e_2'))$-th entry is $\mathsf{sign}(e_1')\mathsf{sign}(e_2')$ if $e_1 \to e_2 \to e_1' \to e_2'$ forms a non-backtracking walk of length 4 in G, and is 0 otherwise. Note that B is *not* symmetric.

[13] We note that one could most likely also prove Theorem 4 using the standard nonbacktracking walk matrix and Ihara–Bass formula, e.g., with similar methods as in [9].

[14] In [27], the nomadic walk matrix is indexed by *directed* edges. In our context, this is equivalent to a length 2 oriented walk $e_1 \to e_2$, which is equivalent to a pair (e_1, e_2) of *undirected* edges, as the ordering in the pair gives the unique orientation $e_1 \to e_2$ in the walk.

▶ **Theorem 13** (Modified Ihara–Bass formula, Theorem 3.1 of [27], informal). *If* $\rho(B) \le (1+o(1))\sqrt{(s-1)(t-1)}$*, then the spectrum* $\mathrm{Spec}(M)$ *of* M *satisfies:*

$$\mathrm{Spec}(M) \subseteq [t-2-2(1+o(1))\sqrt{(s-1)(t-1)}, t-2+2(1+o(1))\sqrt{(s-1)(t-1)}] \ .$$

The above theorem thus shows that it suffices to prove that $\rho(B) \le (1+o(1))\sqrt{(s-1)(t-1)}$ with high probability.

We remark that bounds on the spectra of matrices of the form of M were studied in [27] for the case of $s = O(1)$. Unfortunately, this is insufficient to prove Theorem 4, as we wish to allow s to be any function of n (provided that $s \le n^c$ for some absolute constant c). However, [27, Theorem 3.1] is a general statement that holds regardless of s, so we can make use of it in our setting.

Step 2: bounding $\rho(B)$ via the trace method by counting hikes. The natural approach to bound $\rho(B)$ is by applying the trace method. As the matrix B is not symmetric, we compute:

$$T := \mathbb{E}_{\mathsf{sign}}[\mathrm{tr}(B^\ell (B^\top)^\ell)] \ ,$$

where the expectation is taken over the function sign that determines the signs of the entries of A. By carefully expanding this expectation, one can show that the nonzero contributions to T roughly come from length $4(\ell-1)$ closed walks in G where (1) each edge in the walk appears an even number of times, and (2) the walk is non-backtracking, except possibly at the middle step in the walk. Such walks (of length 4ℓ) are commonly referred to as (2ℓ)-*hikes* [26].

To finish the proof of Theorem 4, we thus turn to obtaining a careful bound on the number of such walks.

Counting these walks requires extra care in our setting as our graph is bipartite, and so the bound needs to be sensitive to the difference in right/left degree. The counting of such hikes also differs greatly depending on whether $s \le \mathrm{polylog}(n)$ or $s = \omega(\mathrm{polylog}(n))$.

Step 3: counting the number of hikes when $s \le \mathrm{polylog}(n)$. [26, Section 3] counts the number of such hikes when $s = O(1)$, provided that G is bicycle-free at radius $O(\log n)$. Namely, any vertex v participates in at most one cycle of length $O(\log n)$. By repeating their proof, one can show that their bounds can be extended to the case when $s \le \mathrm{polylog}(n)$. However, we still cannot use their bound on the number of such hikes naively, as their counting is for non-bipartite graphs and thus yields a bound of $m(1+o(1))^\ell (s-1)^{2\ell}$, simply because it treats left and right vertices the same, and the maximum degree of a vertex is s. We refine their approach to ensure that right and left vertices contribute roughly equally, which will yield the desired bound. One may, at first glance, be tempted to assume that this is trivial because a closed walk in a bipartite graph has an equal number of left and right vertices, but this is not the case, as we shall see.

We adopt the bookkeeping approach of [26]. We think of a hike as discovering the graph G as one traverses the hike. A step in a hike is *fresh* if uses an edge for the first time and ends at a previously undiscovered vertex; it is *boundary* if it uses an edge for the first time but ends up at an old vertex; finally, it is *stale* if it uses an old edge.

Because each edge must appear an even number of times, a hike can have at most 2ℓ fresh steps. Each fresh step "pays" a factor of $(s-1)$ (if we move from a right to a left vertex) or $(t-1)$ (if we move from a left to a right vertex) in our bound in the number of hikes, as this is the number of choices for the next vertex that the hike moves to. [26, Theorem 2.13]

implies that, since G is bicycle-free, the number of boundary steps is $\ll \ell$; they also show that one can bound the "number of choices" for the stale steps by some small factor, which we ignore here.

We need to augment the argument of [26] with the following addition: if a hike has c fresh steps, then the number of fresh steps c_R that start at a right vertex is $\approx \frac{c}{2}$, and similarly for c_L for left vertices. Note that by definition, $c = c_R + c_L$. The key observation is that a fresh step from a right (resp. left) vertex must be followed by either a fresh step from a left (resp. right) vertex, or by a boundary step. Indeed, after we take a fresh step we are at a previously unvisited vertex, so the next step must use a new edge; in particular, it cannot be *stale*. This implies that the deviation of each of c_L, c_R from $\frac{c}{2}$ is bounded by the number of boundary steps, which is $\ll \ell$.

This implies a bound of $m(1+o(1))^\ell (s-1)^\ell (t-1)^\ell$ on the number of hikes, provided that $s \leq \text{polylog}(n)$. The m comes from the number of start vertices in the hike, and the $(1+o(1))^\ell$ comes from the stale and boundary steps, as well as our new deviation term. This yields the desired bound for sparse s.

Step 4: counting the number of hikes when $s = \omega(\text{polylog}(n))$. For s this large, the graph G is "dense", and so it will not be bicycle-free at radius $O(\log n)$. This rules out the approach of [26], which relies on G being bicycle-free. Instead, we adapt a standard counting approach (for bounding the operator norm of a random $n \times n$ Gaussian matrix) given in [33, Section 2.3.6] to our bipartite setting. Our crucial observation here is to note that any hike can have at most ℓ distinct left vertices. As we pay a factor of $(s-1)$ every time we move to a left vertex, it then follows that the "power" of $(s-1)$ in our bound can only be at most ℓ. The standard counting argument of [33, Section 2.3.6] for Gaussian matrices then goes through, yielding the desired bound.

References

1 Zeyuan Allen-Zhu, Rati Gelashvili, and Ilya P. Razenshteyn. Restricted isometry property for general p-norms. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 451–460. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

2 Afonso S Bandeira and Ramon Van Handel. Sharp nonasymptotic bounds on the norm of random matrices with independent entries. *Annals of Probability*, 44(4):2479–2506, 2016.

3 Boaz Barak, Fernando G. S. L. Brandão, Aram Wettroth Harrow, Jonathan A. Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 307–326. ACM, 2012.

4 Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the Restricted Isometry property for random matrices. Available at `https://www.math.tamu.edu/~rdevore/publications/131.pdf`, 2007.

5 Anirban Basak and Mark Rudelson. Invertibility of sparse non-hermitian matrices. *Advances in Mathematics*, 310:426–483, 2017.

6 R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 798–805, 2008. `doi:10.1109/ALLERTON.2008.4797639`.

7 Charles Bordenave. A new proof of friedman's second eigenvalue theorem and its extension to random lifts. In *Annales scientifiques de l'Ecole normale supérieure*, 2019.

8 Charles Bordenave and Benoît Collins. Eigenvalues of random lifts and polynomials of random permutation matrices. *Annals of Mathematics*, 190(3):811–875, 2019.

9 Gerandy Brito, Ioana Dumitriu, and Kameron Decker Harris. Spectral gap in random bipartite biregular graphs and applications. *arXiv preprint*, 2018. arXiv:1804.07808.

10 Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59:1207–1223, 2006.

11 Emmanuel J. Candès and Terence Tao. Decoding by linear programming. *IEEE Trans. Inf. Theory*, 51(12):4203–4215, 2005.

12 Michael R. Capalbo, Omer Reingold, Salil P. Vadhan, and Avi Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 659–668. ACM, 2002.

13 David L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582.

14 T. Figiel, J. Lindenstrauss, and V. D. Milman. The dimension of almost spherical sections of convex bodies. *Acta Math.*, 139(1-2):53–94, 1977.

15 R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, 1963.

16 A. Garnaev and E. D. Gluskin. The widths of Euclidean balls. *Doklady An. SSSR.*, 277:1048–1052, 1984.

17 E. D. Gluskin. Norms of random matrices and diameters of finite-dimensional sets. *Mat. Sb. (N.S.)*, 120(162)(2):180–189, 286, 1983.

18 V. Guruswami, J. Lee, and A. Wigderson. Euclidean sections of with sublinear randomness and error-correction over the reals. In *12th International Workshop on Randomization and Combinatorial Optimization: Algorithms and Techniques (RANDOM)*, pages 444–454, 2008.

19 Venkatesan Guruswami, James R. Lee, and Alexander A. Razborov. Almost Euclidean subspaces of ℓ_1^N via expander codes. *Combinatorica*, 30(1):47–68, 2010. doi:10.1007/s00493-010-2463-9.

20 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

21 Piotr Indyk. Uncertainty principles, extractors, and explicit embeddings of L_1 into L_2. In *Proceedings of the 39th Annual ACM Symposium on the Theory of Computing*, pages 615–620, 2007.

22 Piotr Indyk and Stanislaw J. Szarek. Almost-euclidean subspaces of ℓ_1^n via tensor products: A simple approach to randomness reduction. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 632–641. Springer, 2010.

23 Zohar Shay Karnin. Deterministic construction of a high dimensional $\mathrm{l_p}$ section in $\mathrm{l_1^n}$ for any $p<2$. 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 645–654. ACM, 2011. doi:10.1145/1993636.1993722.

24 B. S. Kashin. The widths of certain finite-dimensional sets and classes of smooth functions. *Izv. Akad. Nauk SSSR Ser. Mat.*, 41(2):334–351, 478, 1977.

25 Boris S Kashin and Vladimir N Temlyakov. A remark on compressed sensing. *Mathematical notes*, 82(5):748–755, 2007.

26 Sidhanth Mohanty, Ryan O'Donnell, and Pedro Paredes. Explicit near-ramanujan graphs of every degree. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 510–523. ACM, 2020.

27 Sidhanth Mohanty, Ryan O'Donnell, and Pedro Paredes. The SDP value for random two-eigenvalue csps. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 50:1–50:45. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

28 Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. LDPC codes achieve list decoding capacity. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science*, pages 458–469, 2020.

29 Ryan O'Donnell and Xinyu Wu. Explicit near-fully X-ramanujan graphs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1045–1056. IEEE, 2020. doi:10.1109/FOCS46700.2020.00101.

30 Tom Richardson and Ruediger Urbanke. *Modern Coding Theory*. Cambridge University Press, USA, 2008.

31 Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6, part 1):1710–1722, 1996. Codes and complexity.

32 Robert M. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

33 Terence Tao. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

34 Yizhe Zhu. On the second eigenvalue of random bipartite biregular graphs. *arXiv preprint*, 2020. arXiv:2005.08103.

Trading Time and Space in Catalytic Branching Programs

James Cook
Independent Researcher[1], Toronto, Canada

Ian Mertz
University of Toronto, Canada

Abstract

An *m-catalytic branching program* (Girard, Koucký, McKenzie 2015) is a set of m distinct branching programs for f which are permitted to share internal (i.e. non-source non-sink) nodes. While originally introduced as a non-uniform analogue to catalytic space, this also gives a natural notion of amortized non-uniform space complexity for f, namely the smallest value $|G|/m$ for an m-catalytic branching program G for f (Potechin 2017).

Potechin (2017) showed that every function f has amortized size $O(n)$, witnessed by an m-catalytic branching program where $m = 2^{2^n-1}$. We recreate this result by defining a catalytic algorithm for evaluating polynomials using a large amount of space but $O(n)$ time. This allows us to balance this with previously known algorithms which are efficient with respect to space at the cost of time (Cook, Mertz 2020, 2021). We show that for any $\epsilon \geq 2n^{-1}$, every function f has an m-catalytic branching program of size $O_\epsilon(mn)$, where $m = 2^{2^{\epsilon n}}$. We similarly recreate an improved result due to Robere and Zuiddam (2021), and show that for $d \leq n$ and $\epsilon \geq 2d^{-1}$, the same result holds for $m = 2^{\binom{n}{\leq \epsilon d}}$ as long as f is a degree-d polynomial over $\mathbb{F}_2$. We also show that for certain classes of functions, m can be reduced to $2^{\text{poly}\, n}$ while still maintaining linear or quasi-linear amortized size.

In the other direction, we bound the necessary length, and by extension the amortized size, of any *permutation branching program* for an arbitrary function between $3n$ and $4n - 4$.

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

Keywords and phrases complexity theory, branching programs, amortized, space complexity, catalytic computation

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.8

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/026/

Funding *Ian Mertz*: Partially funded by NSERC.

Acknowledgements The authors would like to thank Toniann Pitassi, Robert Robere, Jeroen Zuiddam, and Aaron Potechin for many helpful discussions.

1 Introduction

In computational complexity, there is often a focus on analyzing the *worst-case* scenario for a given computation model C and a given function f, but there are other natural cases to consider. One such case is *amortized* computation, where our C algorithm computes many copies of f in such a way that the average cost per copy may be much less than the worst-case cost of computing f a single time.

[1] Now at Amazon, Toronto, Canada.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 8; pp. 8:1–8:21

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

Amortized analysis has typically been used in the context of (time-bounded) Turing Machines, but studying the amortized complexity of syntactic – and in particular non-uniform – computation models goes back just as far, such as Uhlig's results on circuits computing f on m different inputs. The study of amortized analysis for *branching programs*, a model corresponding to non-uniform space-bounded complexity, was initiated by Potechin [11] and later standardized by Robere and Zuiddam [12] for branching programs and other syntactic models.

The amortized model was introduced by [8] in a different context, namely as a non-uniform version of *catalytic space*, originally introduced in the uniform setting by Buhrman et al. [4] as a new type of space-bounded complexity class. In a catalytic Turing Machine, there are four tapes: as in a traditional space-bounded Turing Machine there is a read-only input tape of length n, a write-only output tape of length 1, and a read-write work tape of length $s(n)$, but additionally we have a read-write *catalytic tape* of length $2^{O(s(n))}$. Ordinarily a tape of this length would allow us to capture $SPACE(2^{O(s(n))})$ rather than $SPACE(s(n))$, but the catalytic tape comes with a catch: the entire tape is initialized to an arbitrary string τ, and at the end of the computation it must contain that same string τ. Since the algorithm must work for every string τ (so, for example, τ cannot be compressed), it seems as if we should get no additional power over $SPACE(s(n))$. Buhrman et al. defied this intuition, showing that when $s(n) = O(\log n)$, the catalytic tape allows us to compute any function in TC^1, a class which as far as we know is larger even than $NL = NSPACE(s(n))$.

Going to the non-uniform setting, an m-catalytic branching program for f [8] is defined as having m start nodes, m 1-end nodes, and m 0-end nodes, where if we restrict to any particular start node we get an ordinary branching program for f with a single start, 1-end, and 0-end node, all of which are distinct for each different choice of the start node. To see the connection to catalytic space, we can think of each start node as corresponding to a different setting of a $\log m$-length catalytic tape, where the "restoration" of the catalytic tape is captured by the fact that the two end nodes corresponding to any start node are unique to that start node. After defining this model and showing multiple results extending the uniform arguments in [4], Girard, Koucký, and McKenzie [8] left open the question of how large of an m-catalytic branching program is required to compute an *arbitrary* function f.

As observed by Potechin [11], this definition is also a natural interpretation of branching programs in the amortized world, as our m-catalytic branching program can be thought of as computing the function f m times. Thus in approaching the question left open by [8], they also settled the question of the amortized space required for an arbitrary function f; they showed that *every* function f has an m-catalytic branching program of size $O(mn)$, or in other words amortized size $O(n)$. The only catch is that the number of copies is doubly exponential; specifically, there exists a (layered) m-catalytic branching program of width $2m$ and length $4n$, where $m = 2^{2^n-1}$. The branching program also has the nice property of reading each input variable exactly 4 times, and thus this also has implications for *read-k branching programs* for $k = O(1)$.

In terms of amortized size, the result of [11] is clearly optimal up to constant factors, and so following [8] the central open question they posed is to understand whether or not m can be improved while maintaining linear amortized size, and what the implications of this result may be. Taking up this challenge, Robere and Zuiddam [12] showed that any function f can be computed by an m-catalytic branching program with the same parameters as [11] even when $m = 2^{\binom{n}{\leq d}-1}$, where d is the degree of f as an $\mathbb{F}_2$ polynomial. Unfortunately this doesn't allow us to go beyond [11] for most functions, but it provides a much sharper analysis for many functions that still appear quite difficult. The proof uses properties of $\mathbb{F}_2$ polynomials under permutations of the input variables.

1.1 Our results

1.1.1 Main result

While the m-catalytic branching programs of [11, 12] can be viewed as catalytic algorithms by definition, our initial aim is to restate these algorithms using the basic catalytic tools derived in [4] and follow-up works. In particular we reprove their results using the natural non-uniform variant of *register programs*, which were defined by Ben-Or and Cleve [3] as space-bounded machines for computing simple arithmetic operations and were also the model used in [4] for their results. Our non-uniform register program follows by extending previous work on catalytic algorithms for the *tree evaluation problem* [5, 6], by adapting their register program to optimize time rather than space.

More importantly, as a result of this connection, we can also exploit a trade-off between space and time – here corresponding to m and length, respectively – in order to strongly break the 2^{2^n-1} barrier for arbitrary functions. In Section 3, we show:

▶ **Theorem 1.** *For any function $f : \{0,1\}^n \to \{0,1\}$ and any $\epsilon \geq 2n^{-1}$, there exists an m-catalytic branching program of width $2m$ and length $2^{1/\epsilon} \cdot 2\epsilon n$ computing f, where $m = 2^{n+\frac{1}{\epsilon}\cdot 2^{\epsilon n}}$.*

Furthermore, if f is a degree-d polynomial over $\mathbb{F}_2$ and $\epsilon \geq 2d^{-1}$, there is an m-catalytic branching program of width $2m$ and length $2^{1/\epsilon} \cdot 2n$ and computing f, where $m = 2^{n+\frac{1}{\epsilon}\binom{n}{\leq \epsilon d}}$.

Focusing on the case when $\epsilon = \Omega(1)$, this gives us a read-$O(1)$ m-catalytic branching program with m significantly less than 2^{2^n-1} for every function, as well as significantly less than $2^{\binom{n}{\leq d}-1}$ for degree d functions.

1.1.2 Other results

As a bonus, this interpretation also allows us to show significant improvements on m (while still maintaining linear amortized size) for some functions not covered by [12], in particular all functions in TC^0, the class of functions computable by low-depth threshold circuits of polynomial size. By allowing the amortized size to increase to quasilinear, we can capture the much larger class VP, which is the class of all polynomials computable by poly-size arithmetic circuits. See the full version of the paper for proofs and discussions of these results.

▶ **Theorem 2.** *Let f be a function which can be computed by a Boolean circuit C which has depth $O(1)$, size $\mathrm{poly}(n)$, and consists of unbounded fan-in MAJ gates. Then f can be computed by an m-catalytic branching program of length $O(n)$ and width $3m$, where $m = 2^{\mathrm{poly}(n)}$.*

Let $\mathbb{F} \in \{\mathbb{F}_{p\in[\mathrm{poly}\, n]}, \mathbb{Z}, \mathbb{Q}\}$, and let f be a function which can be computed by an arithmetic circuit[2] C over $\mathbb{F}$ which has depth $O(\log n)$, size $\mathrm{poly}(n)$, and consists of unbounded fan-in $+$ gates and fan-in 2 $\times$ gates. Then for any $\epsilon > 0$, f can be computed by an m-catalytic branching program of length $O(n^\epsilon) \cdot n$ and width $3m$, where $m = 2^{n^{O(1/\epsilon)}}$.

In Section 4, we study whether tradeoffs can be made in the other direction, namely whether length $4n$ is optimal for m-catalytic branching programs of width $O(m)$. We show that a few modifications can bring the length down even while slightly improving the original

[2] Our notion of an arithmetic circuit computing a Boolean function is the *Boolean part* of the circuit class, meaning that for all inputs coming from $\{0,1\}^n$, the polynomial the circuit computes will either evaluate to 0 or 1 over $\mathbb{F}$. See e.g. [1] for more discussion of such models.

parameter $m = 2^{2^n-1}$. As with the results of [11, 12] and our improvements, this program is not only layered with optimal width $2m$, but in fact can be made a *permutation branching program*, meaning that each layer has exactly $2m$ nodes and the transition between layers is restricted to be a permutation.[3]

▶ **Theorem 3.** *For every function f, there is a read-4 permutation branching program of width $2^{n+2^{n-1}}$ and length $4n-4$ computing f.*

To complement this result, we show that for such restricted programs and *any* m, even the AND function cannot be computed by permutation programs of length less than $3n$ once $n \geq 4$. In fact our lower bound is very suggestive; it shows that any attempt to read any variable less than three times results in a program exactly matching Theorem 3.

▶ **Theorem 4.** *Any permutation branching program computing $AND(x_1, \ldots, x_n)$ which reads some variable at most twice must have length at least $4n-4$.*

Together these two results leave three questions: 1) what, between $3n$ and $4n-4$, is the shortest length of a permutation branching program for an arbitrary function?; 2) can m can be improved for programs of length $3n$ – or in general any fixed length?; and 3) can we get any improvements by moving to more general programs?

2 Preliminaries

We introduce two space-bounded models of computation. Our first model is a variation of *branching programs*, which are the standard syntactic (i.e. non-uniform) notion of space-bounded computation (see [7]).

▶ **Definition 5** (m-catalytic branching program [8])**.** *Let $n \in \mathbb{N}$ and let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary function. An* m-catalytic branching program *is a directed acyclic graph G with the following properties:*

- *There are m source nodes and $2m$ sink nodes.*
- *Every non-sink node is labeled with an input variable x_i for $i \in [n]$, and has two outgoing edges labeled 0 and 1.*
- *For every source node v there is one sink node labeled with $(v, 0)$ and one with $(v, 1)$.*

We say that G computes *f if for every $x \in \{0,1\}^n$ and source node v, the path defined by starting at v and following the edges labeled by the value of the x_i labeling each node ends at the sink labeled by $(v, f(x))$. The* size *of G is the number of non-sink nodes[4] in G.*

We also consider m-catalytic branching programs with standard restrictions:

- *layered branching programs*: for some $\ell \in \mathbb{N}$, there exists a function $\sigma : G \to [\ell+1]$ such that for all $u \in G$, the outgoing edges of u go to nodes v, w such that $\sigma(v) = \sigma(w) = \sigma(u) + 1$; we call the set of nodes $\{v \in \sigma^{-1}(j)\}$ the jth *layer*. Furthermore for each $j \in [\ell]$ there exists an input variable x_{j_i} which is the variable labeling every $v \in \sigma^{-1}(j)$.[5] The *width* of G is $\max_{j \in [\ell]} |\sigma^{-1}(j)|$ and the *length* of G is ℓ. Note that the size of G is at most the product of the length and width of G.

[3] It is likely m can be improved to $2^{2^{n-1}}$ by arguing directly about the branching program. E.g. our Theorem 9 loses a similar factor compared to Potechin's Theorem 3.1 [11], by using register programs.

[4] This is somewhat non-standard, but when talking about layered branching programs this simplifies things by defining the length as the number of times we read variables, which will in turn be connected to the number of instructions in our register program model. This choice does not affect the asymptotics of any results.

[5] By construction layer $\ell + 1$ will contain exactly the sink nodes of G. See the previous footnote for an explanation of this somewhat non-standard convention.

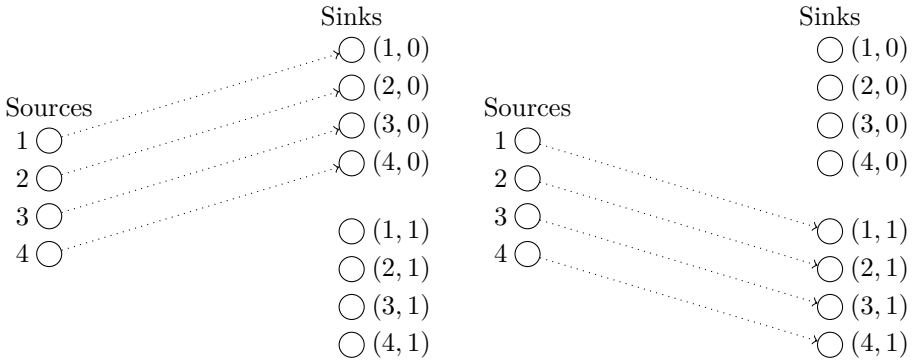


Figure 1 The computation of an m-catalytic branching program starting at source node i ends at sink node $(i, 0)$ if $f(x) = 0$ (left) or $(i, 1)$ if $f(x) = 1$ (right). In these diagrams, $m = 4$.

- *read-k branching programs*: for any start node v and any input x, each variable x_i is read at most k times during the computation starting at v. We note that for layered branching programs, this corresponds to each variable x_j labeling at most k layers.
- *permutation branching programs*: consider a layered branching program of width $2m$ with $2m$ source nodes $S = \{s_1 \dots s_{2m}\}$ and $2m$ sink nodes $T = \{t_1 \dots t_{2m}\}$, and let $P(x) : [2m] \to [2m]$ be the function such that $P(x)(i) = i'$ where the computation path starting from s_i on input x goes to $t_{i'}$. Then there exists a permutation $\sigma^* : [2m] \to [2m]$ such that $P(x)$ is the identity function when $f(x) = 0$ and $P(x) = \sigma^*$ when $f(x) = 1$. This is not an m-catalytic branching program in the strict sense, and we will address this model more precisely in Section 4.

Our second model comes from a line of work starting with [3], more recently fleshed out in [4] and used in many follow-up works on catalytic computation [5, 6].

▶ **Definition 6** (Transparent register program). *Let $\mathcal{R}$ be a ring and $f \in \{0,1\}^n \to \{0,1\}$ a function. A* register program *P is defined by a set of registers $\mathcal{S} = \{R_1 \dots R_s, R^{\text{out}}\}$, each storing a value in $\mathcal{R}$, plus an ordered list of t instructions where for every $j \in [t]$ the jth instruction is $R \leftarrow R + p_j(x_i, \mathcal{S} \setminus \{R\})$ for some $i \in [n]$, $R \in \mathcal{S}$, and polynomial $p_j \in \mathcal{R}[x, y_0 \dots y_s]$.*

We say that P computes f if for every $x \in \{0,1\}^n$, after initializing $R = 0$ for all registers R and then executing all instructions in order, the value stored in R^{out} is $f(x)$. We say that P transparently *computes f if instead of initializing all R to 0, each R begins in an arbitrary initial state τ_i, and at the end of the program we have $R^{\text{out}} = \tau^{\text{out}} + f(x)$ and $R_i = \tau_i$ for all i. The* size *of P is the number of registers $s + 1$ and the* time *of P is the number of instructions t.*

We use register programs as an abstraction for understanding m-catalytic branching programs, which will be the model our main results will refer to. The two models are connected through the following observation (see Figure 2 for an example of our construction).

▶ **Observation 7.** *Let $f_n : \{0,1\}^n \to \{0,1\}$ be a family of functions and let P_n be a family of register programs over a family of rings $\mathcal{R}_n$ with size $s(n)+1$ and time $t(n)$ each transparently computing f_n. Then f_n can be computed by a family of m-catalytic branching programs of width $|\mathcal{R}_n|m$ and length $t(n)$, where $m = |\mathcal{R}_n|^{s(n)}$.*

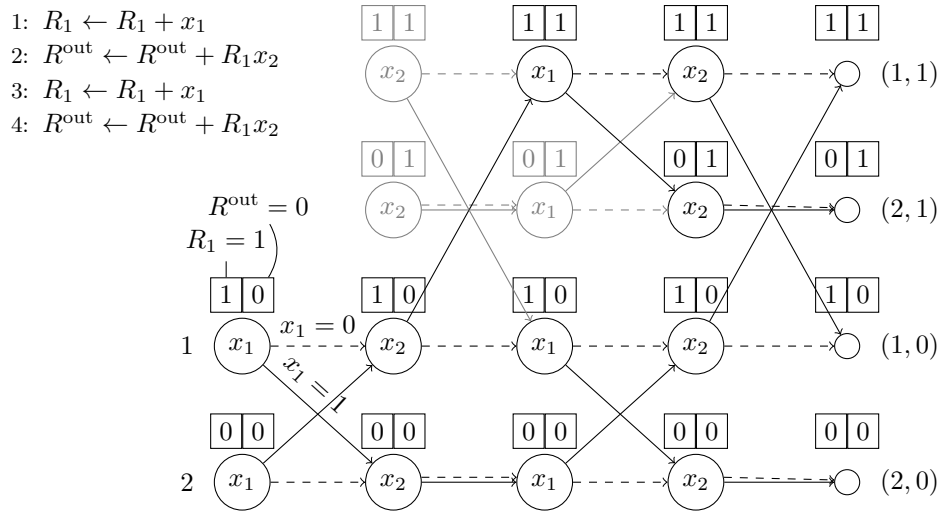


Figure 2 A transparent register program computing AND(x_1, x_2), and its realization as a 2-catalytic branching program using Observation 7. Each node is annotated (above, in boxes) with the register values it stores, and each non-sink node is labelled (inside the circle) with the input to be read. Dashed lines are transitions taken when that input is zero, and solid lines are taken when the input is one. Finally, the source nodes are assigned the numbers 1 and 2 (since $m = 2$), and the sink nodes are assigned pairs of numbers, so that like in the more abstract Figure 1, a computation starting at source node v will end at end at sink node $(v, f(x))$. Note that nodes/edges that appear in gray are unreachable from the start states and thus would not appear in the final branching program; they appear only for reference as to how the register program translates to a branching program.

Proof. We will define a branching program with width $|\mathcal{R}_n|^{s(n)+1}$ and length $t(n)$. Each layer will represent a stage in the register program and each node in a layer will represent a setting to the registers at that time. Since each register program step only requires us to read one input variable, at layer k we query the variable associated with step k in the register program and create edges from each node in layer k to the nodes at layer $k+1$ representing the state of our memory after the step has completed. We label each input node as τ for some distinct initial configuration $\tau = (\tau_1 \dots \tau_{s(n)})$ to all registers except R^{out}, and we treat R^{out} as being initialized to 0. Then by the fact that P transparently computes f, starting at node τ we are guaranteed to reach a node $(\tau, f(x))$. Since in each layer we have a node for every setting of the $s(n)+1$ registers, the width of our branching program is $|\mathcal{R}_n|^{s(n)+1}$, and since each non-output layer corresponds to a unique instruction from our register program, the length of our branching program is $t(n)$. ◀

▶ Note 8. In our computation we often include instructions of the form $R \leftarrow R + p_j(\mathcal{R}/\{R\})$, i.e. instructions that do not require reading a variable x_i. By observing the above proof, it is clear that such instructions do not add any length to the branching program, as they can be computed at the same time as an adjacent instruction.

3 Saving Space

In this section we show that every function can be computed by an m-catalytic branching program with size $O(mn)$ for $m \ll 2^{2^n - 1}$ (improving on [11]) and $m \ll 2^{\binom{n}{\leq d} - 1}$ (improving on [12]). We present our algorithm in three steps:

- In Section 3.1 we show a simpler version of our algorithm which is sufficient to reproduce – with a negligible loss in parameters – Potechin's result [11] that any function can be computed with a linear-amortized-size m-catalytic branching program. Our program has length $4n$ and width $2m$, where $m = 2^{2^n+n}$.
- In Section 3.2 we show how to trade off between m and amortized size, yielding for every $k \in [d]$ an m-catalytic branching program of length $2^k \cdot 4\lceil n/k \rceil$ and width $2m$, where $m = 2^{k \cdot 2^{\lceil n/k \rceil}+n}$.
- In Section 3.3 we show a simple modification of our first algorithm which reproduces – again with a negligible loss – the result of Robere and Zuiddam [12] that m can be made as small as $2^{\binom{n}{\leq d}-1}$, where d is the degree of f as an $\mathbb{F}_2$ polynomial, with no cost to the length. Our program has length $4n$ and width $2m$, where $m = 2^{\binom{n}{\leq d}+n}$. We then show that the tradeoff algorithm gives us an m-catalytic branching program of length $2^k \cdot 2n$ and width $2m$, where $m = 2^{k \cdot \binom{n}{\leq \lceil d/k \rceil}+n}$.[6]

In these sections, our register programs will all operate over the field of two elements: $\mathcal{R}_n = \mathbb{F}_2$ for all n.

3.1 Basic Algorithm

In this section, we will prove:

▶ **Theorem 9.** *For any function $f : \{0,1\}^n \to \{0,1\}$ there is an m-catalytic branching program with length $4n$ and width $2m$ that computes f, where $m = 2^{n+2^n}$.*

This is proved by Algorithm 1 below. This nearly reproduces Potechin's Theorem 3.1 [11], but with a worse value $m = 2^{n+2^n}$ instead of 2^{2^n-1}. Nonetheless, we will find it a useful starting point for our algorithm that trades space for time in Section 3.2.

Proof. We will design a program that uses $n+2^n$ work registers plus one output register R^{out}, which is sufficient by Observation 7. First, we have n registers $R_1^{\text{in}}, \ldots, R_n^{\text{in}}$, corresponding to the n input bits. This correspondence is given by the following subroutine:

```
1: procedure ToggleInput
2:     for i = 1, ..., n do
3:         R_i^in ← R_i^in + x_i
4:     end for
5: end procedure
```

After ToggleInput runs, the registers have values $R_i^{\text{in}} = \tau_i^{\text{in}} + x_i$, where τ_i^{in} stands for the initial value of R_i^{in}. If we run it a second time, the registers are restored to their original values: $R_i^{\text{in}} = \tau_i^{\text{in}}$. Since we query all n variables once, ToggleInput requires time n to run once.

Before defining our other 2^n registers, we introduce an algebraic view of f, which will be our main focus. We can view f as a multilinear polynomial $p_f \in \mathbb{F}_2[x_1, \ldots, x_n]$ using basic interpolation:

▶ **Lemma 10.** *For any function $f : \{0,1\}^n \to \{0,1\}$ there is a multilinear polynomial $p_f \in \mathbb{F}_2[x_1, \ldots, x_n]$ such that $p_f(\vec{x}) = f(\vec{x})$ for all $\vec{x} \in \{0,1\}^n$.*

[6] While the program of [12] matches or beats [11] for all d, our improved version of [12] is worse than our improved version of [11] when $d = \Omega(n)$ (although still an improvement over the original results of both papers), and thus we state and prove both results separately rather than subsuming our improved version of [11].

Proof. For any $\vec{y} \in \{0,1\}^n$, we can algebraically define the indicator function $[\vec{x} = \vec{y}]$ as $\prod_{i=1}^{n}(x_i + y_i + 1) \in \mathbb{F}_2[x_1, \ldots, x_n]$. Then we can set

$$p_f = \sum_{\vec{y} \in \{0,1\}^n : f(\vec{y})=1} [\vec{x} = \vec{y}]$$ ◀

Now define $y_i = \tau_i^{\text{in}} + x_i$; in other words, y_i is the value of R_i^{in} after running ToggleInput. Define $q_f(y_1, \ldots, y_n, \tau_1^{\text{in}}, \ldots, \tau_n^{\text{in}}) = p_f(y_1 - \tau_1^{\text{in}}, \ldots, y_n - \tau_n^{\text{in}})$ to be the result of rewriting p_f using the y_i and τ_i variables.[7] For $S, S' \subseteq [n]$, let $c_{S,S'}$ be the coefficient of $\left(\prod_{i \in S} \tau_i^{\text{in}}\right) \cdot \left(\prod_{i \in S'} y_i\right)$ in q_f, so that

$$q_f(\overrightarrow{\tau^{\text{in}}}, \vec{y}) = \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}}\right) \left(\prod_{i \in S'} y_i\right)$$

Note that S and S' are disjoint whenever $c_{S,S'} \neq 0$, because p_f is multilinear. We now introduce our other registers: we have 2^n registers R_S indexed by subsets $S \subseteq [n]$. Our next subroutine prepares us to use these registers to compute q_f:

```
1: procedure ToggleMonomials
2:     ToggleInput
3:     for S' ⊆ [n] do
4:         R_{S'} ← R_{S'} + ∏_{i∈S'} R_i^in
5:     end for
6:     ToggleInput
7: end procedure
```

After ToggleMonomials runs, we have $R_S = \tau_S + \prod_{i \in S} y_i$ for each $S \subseteq [n]$, where τ_S stands for the register's initial value. The R^{in} registers have their initial values $R_i^{\text{in}} = \tau_i^{\text{in}}$. We run ToggleInput twice and have 2^n additional instructions, but since the additional instructions do not query any x variables they can be computed in the last x query of ToggleInput, for a total runtime of $2n$.

Our final algorithm for computing f is:

Algorithm 1 Transparently compute f in time $4n$ with $n + 2^n + 1$ registers.

```
1: ToggleMonomials
2: R^out ← R^out + Σ_{S,S'⊆[n]} c_{S,S'} (∏_{i∈S} R_i^in) R_{S'}
3: ToggleMonomials
4: R^out ← R^out − Σ_{S,S'⊆[n]} c_{S,S'} (∏_{i∈S} R_i^in) R_{S'}
```

When Line 2 executes, we have $R_{S'} = \tau_{S'} + \prod_{i \in S'} y_i$, so after that line,

$$R^{\text{out}} = \tau^{\text{out}} + \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}}\right) \left(\tau_{S'} + \prod_{i \in S'} y_i\right).$$

Then when Line 4 executes, we have $R_{S'} = \tau_{S'}$, so the final value is

$$R^{\text{out}} = \tau^{\text{out}} + \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}}\right) \left(\prod_{i \in S'} y_i\right) = \tau^{\text{out}} + f(x_1, \ldots, x_n).$$

[7] For example, if f is the OR function $f(x_1, x_2) = x_1 \vee x_2$, we have $p_f(x_1, x_2) = x_1 + x_2 + x_1 x_2$ and $q_f(\tau_1^{\text{in}}, \tau_2^{\text{in}}, y_1, y_2) = y_1 + \tau_1^{\text{in}} + y_2 + \tau_2^{\text{in}} + y_1 y_2 + y_1 \tau_2^{\text{in}} + \tau_1^{\text{in}} y_2 + \tau_1^{\text{in}} \tau_2^{\text{in}}$, and both are equal to $f(x_1, x_2)$ so long as $\vec{y}$ have the correct values.

So Algorithm 1 correctly computes f. The space of the program is $n + 2^n$ by construction, and as before we can ignore the instructions on lines 2 and 4 since they do not use x, giving us a total runtime of $4n$ from the two calls to ToggleMonomials. ◀

3.2 Trading Space for Time

In this section, we will modify Algorithm 1 to make m dramatically smaller, in exchange for making the branching program longer.

▶ **Theorem 11.** *For any $k \in \mathbb{N}$ and any function $f : \{0,1\}^n \to \{0,1\}$ there is an m-catalytic branching program with length $2^k \cdot 2\lceil n/k \rceil$ and width $2m$ that computes f, where $m = 2^{n+k\cdot 2^{\lceil n/k \rceil}}$.*

Before jumping into the proof of Theorem 11, we will address the main innovation of our work, namely trading off time for space. Namely we begin by building a register program that takes time $n2^{n+1}$ but uses only the $n+1$ registers $R_1^{\text{in}} \dots R_n^{\text{in}}, R^{\text{out}}$. This is similar to what Theorem 11 guarantees when $k = n$.

As in Sections 3.1 and 3.3, let $p_f \in \mathbb{F}_2[x_1, \dots, x_n]$ be f as a polynomial, let $q_f \in \mathbb{F}_2[\tau_1^{\text{in}}, \dots, \tau_n^{\text{in}}, y_1, \dots, y_n]$ be equal to p_f so long as $y_i = \tau_i^{\text{in}} + x_i$ for all i, and let $c_{S,S'}$ be the coefficient of $\left(\prod_{i\in S} \tau_i^{\text{in}}\right)\left(\prod_{i\in S'} y_i\right)$ in q_f. We define a small generalization of ToggleInput, where we can choose to toggle only a subset of our inputs:

1: **procedure** ToggleSomeInputs(S')
2: **for** $i \in S'$ **do**
3: $R_i^{\text{in}} \leftarrow R_i^{\text{in}} + x_i$
4: **end for**
5: **end procedure**

Using ToggleSomeInputs(S'), we can replace the register $R_{S'}$ in Algorithm 1 with a separate set of instructions that computes the corresponding term of q_f:

Algorithm 2 A slow algorithm for computing q_f.

1: **for** $S' \subseteq [n]$ **do**
2: ToggleSomeInputs(S')
3: $R^{\text{out}} \leftarrow R^{\text{out}} + \sum_{S\subseteq[n]} c_{S,S'} \cdot \prod_{i\in S\cup S'} R_i^{\text{in}}$
4: ToggleSomeInputs(S')
5: **end for**

Whenever Line 3 is executed, $R_i^{\text{in}} = y_i$ for $i \in S'$, and $R_i^{\text{in}} = \tau_i^{\text{in}}$ for $i \notin S'$. By construction of q_f, S and S' are disjoint whenever $c_{S,S'} \neq 0$, from which it follows that $R_i^{\text{in}} = \tau_i^{\text{in}}$ for $i \in S$. Thus the effect of Line 3 is to add $\sum_{S\subseteq[n]} c_{S,S'} \left(\prod_{i\in S} \tau_i^{\text{in}}\right)\left(\prod_{i\in S'} y_i\right)$ to R^{out}. Since this is run for every subset S', the overall effect of the program is to add

$$\sum_{S,S'\subseteq[n]} c_{S,S'} \left(\prod_{i\in S} \tau_i^{\text{in}}\right)\left(\prod_{i\in S'} y_i\right) = p_f(x_1 \dots x_n)$$

to R^{out}.

We now give an overview of the full proof. Our goal is to balance Algorithms 1 and 2 by removing the registers R_S corresponding to large subsets S and using slow multiplication to build the polynomial q_f from the remaining small subsets. In particular, if we divide the input bits into k groups each of size $\lceil n/k \rceil$, and only store all subsets within each group, then

any monomial $c_{S,S'} \prod_{i \in S} \tau_i^{\text{in}} \prod_{i \in S'} y_i$ can be computed by multiplying together one subset from each group, namely the restriction of S to the group. Instead of 2^n registers for all subsets, we use only $k \cdot 2^{\lceil n/k \rceil}$ registers corresponding to subsets in the k groups, and we can compute all the corresponding monomials into these registers in time $2n$ using the first half of Algorithm 1. Then since we are only multiplying k monomials together, we can compute q_f using Algorithm 2 in time $2^k \cdot 2 \cdot 2n$, since each call to ToggleSomeInputs is replaced with our $2n$ time execution of Algorithm 1.

As a slight last speedup, we use a Gray code to order our subsets in Algorithm 2, replacing two executions of ToggleSomeInputs with a subroutine toggling a single group on or off and only spending $2\lceil n/k \rceil$ time to do so. This allows us to compute q_f in $2^k \cdot 2\lceil n/k \rceil$ time rather than $2^k \cdot 4n$ time.

Proof of Theorem 11. For $j \in [k]$ let $b_j = \lceil nj/k \rceil$, and divide the range $[n]$ into k groups: $G_1 = \{1, \ldots, b_1\}, G_2 = \{b_1 + 1, \ldots, b_2\}, \ldots, G_k = \{b_{k-1} + 1, \ldots, b_n = n\}$. For each group G_j, we have $2^{|G_j|}$ registers $R_{j,S}$ indexed by subsets $S \subseteq G_j$. As in all previous algorithms we also use n registers $R_1^{\text{in}}, \ldots, R_n^{\text{in}}$, corresponding to the n input bits, for a total of $n + \sum_{j=1}^{k} 2^{|G_j|}$ registers plus the output register R^{out}.

We'll begin by rewriting the polynomial q_f in a new form. Recall from Section 3.1 that $q_f(\overrightarrow{\tau^{\text{in}}}, \vec{y}) = f(x)$ so long as $\vec{y} = \vec{x} + \overrightarrow{\tau^{\text{in}}}$, and

$$q_f(\overrightarrow{\tau^{\text{in}}}, \vec{y}) = \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{i \in S'} y_i \right)$$

Now, for every $S' \subseteq [n]$, define $S'_j := S' \cap G_j$. For each $j \in [k]$ and $S \subseteq G_j$, let $z_{j,S} = \tau_{j,S} + \prod_{i \in S} y_i$, where $\tau_{j,S}$ is the initial value of register $R_{j,S}$. Now for every monomial in q_f, we split the term $\prod_{i \in S'} y_i$ in the monomial into k different products $\prod_{i \in S'_j} y_i$, each of which we can replace with $z_{j,S'_j} - \tau_{j,S'_j}$. This gives us a new polynomial

$$r_f(\overrightarrow{\tau^{\text{in}}}, \vec{\tau}, \vec{z}) = \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{j=1}^{k} (z_{j,S'} - \tau_{j,S'}) \right).$$

As we did with q_f, for $S, S' \subseteq [n]$ and $T \subseteq [k]$, let $d_{S,S',T}$ be the coefficient of $(\prod_{i \in S} \tau_i^{\text{in}}) \cdot (\prod_{j \in T} z_{j,S'_j}) \cdot (\prod_{j \in [k] \setminus T} \tau_{j,S'_j})$ in r_f, so that

$$r_f(\overrightarrow{\tau^{\text{in}}}, \vec{\tau}, \vec{z}) = \sum_{S \subseteq [n]} \sum_{S' \subseteq [n]} \sum_{T \subseteq [k]} d_{S,S',T} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{j \in T} z_{j,S'_j} \right) \left(\prod_{j \in [k] \setminus T} \tau_{j,S'_j} \right)$$

which is equivalent to $f(x_1 \ldots f_n)$ as long as $z_{j,S'_j} = \tau_{j,S'_j} + \prod_{i \in S'_j} (x_i + \tau_i^{\text{in}} + 1)$.

Following ToggleSomeInputs(S'), we define new versions of ToggleInput and ToggleMonomials from Section 3.1 which focus on some groups and not others. In fact we will only focus on a single group G_j rather than a subset of the groups, as we will order our subsets S' in such a way that we will only ever need to toggle one group at a time:

```
1: procedure ToggleInputForGroup(j)
2:     for i ∈ G_j do
3:         R_i^in ← R_i^in + x_i
4:     end for
5: end procedure
```

```
1: procedure ToggleMonomialsForGroup(j)
2:     ToggleInputForGroup(j)
3:     for S ⊆ G_j do
4:         R_{j,S} ← R_{j,S} + ∏_{i∈S} R_i^in
5:     end for
6:     ToggleInputForGroup(j)
7: end procedure
```

We are now ready to assemble Algorithm 4, which completes the proof of Theorem 11. To incorporate our improvement using Gray codes [9], let $T_0 = \emptyset, \ldots, T_{2^k-1}$ be an ordering of all subsets of $[k]$ such that each consecutive pair of sets $T_\ell, T_{\ell+1 \bmod 2^k}$ differs by exactly one element $e_\ell \in [k]$; thus we will only need to toggle the group G_{e_ℓ} as claimed:

Algorithm 3 Transparently compute f in time $2^k \cdot 2\lceil n/k \rceil$ with $n + k \cdot 2^{\lceil n/k \rceil}$ registers.

```
1: for ℓ = 0, ..., 2^k − 1 do
2:     R^out ← R^out + Σ_{S⊆[n]} Σ_{S'⊆[n]} d_{S,S',T_ℓ} (∏_{i∈S} R_i^in) (∏_{j=1}^k R_{j,S'_j})
3:     ToggleMonomialsForGroup(e_ℓ)
4: end for
```

Each time Line 2 is reached, we have $R_{j,S} = \tau_{j,S} + \prod_{i \in S} y_j$ for $j \in T_\ell$, and $R_{j,S} = \tau_{j,S}$ for $j \in [k] \setminus T_\ell$. We also have $R_i^{\text{in}} = \tau_i^{\text{in}}$ for each $i \in [n]$. So the effect of the line is to add

$$\sum_{S \subseteq [n]} \sum_{S' \subseteq [n]} d_{S,S',T_\ell} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{j \in T_\ell} z_{j,S'_j} \right) \left(\prod_{j \in [k] \setminus T_\ell} \tau_{j,S'_j} \right)$$

to R^{out}. Summing this expression over all possible subsets $T_\ell \subseteq [k]$ gives $R^{\text{out}} = \tau^{\text{out}} + r_f(\cdots) = \tau^{\text{out}} + f(x_1, \ldots, x_n)$, and so our algorithm transparently computes f. ◀

▶ Note 12. It is not difficult to save k registers by removing $R_{1,\emptyset}, \ldots, R_{k,\emptyset}$, as we simply add each value $d_{S,\emptyset,T}(\prod_{i \in S} R_i^{\text{in}})$ to our polynomial without concerning ourselves with any x_i (and by extension any y_i or z_i) variables.

3.3 Bounded-Degree Polynomials

Robere and Zuiddam [12, Theorem 5.13] showed that if f is a polynomial of degree $d < n$, it is possible to improve on Potechin's theorem by decreasing $m = 2^{2^n-1}$ down to $m = 2^{\binom{n}{\leq d}-1}$. Here we show how to adapt Algorithm 1 to get a similar result, and then at the end of the section we build a tradeoff algorithm to improve it.

▶ **Theorem 13.** *For any function $f : \{0,1\}^n \to \{0,1\}$ which is a degree-d polynomial, there is an m-catalytic branching program with length $4n$ and width $2m$ that computes f, where $m \leq 2^{n+\binom{n}{\leq d}}$.*

Again, while this is slightly worse than Robere and Zuiddam's original result, we include it to show the flexibility of our approach and as a stepping stone to our tradeoff result.

Proof. The proof can be summarized as follows. We start with Algorithm 1, and make the following change: *for every $S' \in [n]$ such that $c_{S,S'} = 0$ for all S, remove the register $R_{S'}$.* We then argue that only $n + \binom{n}{\leq d}$ registers remain.

Let $p_f = f \in \mathbb{F}_2[x_1, \ldots, x_n]$ – since f is already a polynomial, there's no need to define p_f using interpolation this time. As before, let

$$q_f(\overrightarrow{\tau^{\text{in}}}, \vec{y}) = p_f(\vec{y} - \overrightarrow{\tau^{\text{in}}}) = \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}}\right) \left(\prod_{i \in S'} y_i\right)$$

Let $\mathcal{M}_f \subseteq 2^{[n]}$ be the set of all S' such that there exists an S where $c_{S,S'} \neq 0$. We define the following subroutine:

```
1: procedure ToggleUsefulMonomials
2:     ToggleInput
3:     for S ∈ M_f do
4:         R_S ← R_S + ∏_{i∈S} R_i^in
5:     end for
6:     ToggleInput
7: end procedure
```

The only difference from ToggleMonomials is that we ignore subsets S which are not in $\mathcal{M}_f$ (not "useful"). Our final algorithm is

Algorithm 4 Transparently compute a degree-d polynomial f in time $4n$ with $n + \binom{n}{\leq d}$ registers.

```
1: ToggleUsefulMonomials
2: R^out ← R^out + Σ_{S⊆[n]} Σ_{S'∈M_f} c_{S,S'} (∏_{i∈S} R_i^in) R_{S'}
3: ToggleUsefulMonomials
4: R^out ← R^out − Σ_{S⊆[n]} Σ_{S'∈M_f} c_{S,S'} (∏_{i∈S} R_i^in) R_{S'}
```

To conclude the proof of Theorem 13, we need to show $|\mathcal{M}_f| \leq \binom{n}{\leq d}$. Indeed, since p_f is a degree-d polynomial, q_f also has degree d, which means $c_{S,S'} = 0$ whenever $|S| + |S'| > d$. So, $\mathcal{M}_f$ only contains sets S' with size at most d, of which there are $\binom{n}{\leq d}$. ◀

▶ Note 14. The original algorithms of [11, 12] rely on the *symmetries* of f as an $\mathbb{F}_2$ polynomial, in essence having each start state represent a possible function g which can be obtained from f by negating input variables or taking $\oplus$ with f itself. [11] takes this set of functions to be the space of all n-variable functions, while [12] analyzes these rules and obtains a more exact characterization. While this characterization is phrased in terms of orbit closures, it can also be described in terms of polynomials as the span of the set of all monomials appearing in f as an $\mathbb{F}_2$ polynomial along with all *submonomials* of this set; this exactly coincides with our notion as $\prod_{i \in S} y_i$ generates all submonomials $\prod_{i \in S' \subseteq S} x_i$ for $y_i := x_i + \tau_i$, which leads to the quantitative results being essentially the same despite taking two completely different approaches.

Now we state our tradeoff algorithm, which goes much in the same way as Theorem 11 but without breaking the variables into groups.

▶ **Theorem 15.** *For any function $f : \{0,1\}^n \to \{0,1\}$ which is a degree-d polynomial and any $k \in \mathbb{N}$, there is an m-catalytic branching program with length $2^k \cdot 2n$ and width $2m$ that computes f, where $m = 2^{n+k \cdot \binom{n}{\leq \lceil d/k \rceil}}$.*

Proof. For any $\Delta \in \mathbb{N}$, let $\mathcal{M}_f^{\Delta} \subseteq \binom{n}{\leq \Delta}$ be the set of all S'' of size at most Δ such that there exists an $S \subseteq [n]$ and $S' \supseteq S''$ where $c_{S,S'} \neq 0$. We will have k registers $R_{j,S''}$ for every $S'' \in \mathcal{M}_f^{\lceil d/k \rceil}$, as well as the usual registers $R_1^{\text{in}} \ldots R_n^{\text{in}}, R^{\text{out}}$. Note that this gives us our target space, as $|\mathcal{M}_f^{\lceil d/k \rceil}| \leq \binom{n}{\leq \lceil d/k \rceil}$.

Following our proof of Theorem 11, let $z_{j,S} = \tau_{j,S} + \prod_{i \in S} y_i$, where $\tau_{j,S}$ is the initial value of register $R_{j,S}$, and for every monomial in q_f we split the term $\prod_{i \in S'} y_i$ in the monomial arbitrarily into k different products $\prod_{i \in S'_j} y_i$ – each of which we can replace with $z_{j,S'_j} - \tau_{j,S'_j}$ – where $S'_j \in \mathcal{M}_f^{\lceil d/k \rceil}$ and $\cup_{j \in [k]} S'_j = S'$. This is possible because each non-zero term in q_f has degree at most d, meaning that $|S'| \leq d$ and furthermore every subset of S' of size at most $\lceil d/k \rceil$ appears in $\mathcal{M}_f^{\lceil d/k \rceil}$ by construction.[8]

Fixing some particular partition $(S'_j)_{j \in [k]}$ for each S', this gives us a new polynomial

$$r_f(\overrightarrow{\tau^{\text{in}}}, \vec{\tau}, \vec{z}) = \sum_{S \subseteq [n]} \sum_{S' \subseteq [n]} \sum_{T \subseteq [k]} d_{S,S',T} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{j \in T} z_{j,S'_j} \right) \left(\prod_{j \in [k] \setminus T} \tau_{j,S'_j} \right)$$

which is equivalent to $f(x_1 \dots f_n)$ as long as $z_{j,S'_j} = \tau_{j,S'_j} + \prod_{i \in S'_j} (x_i + \tau_i^{\text{in}} + 1)$. We define ToggleMonomialsForGroup as before, using ToggleInput instead of ToggleInputForGroup since the variables are no longer split into groups, and using a Gray code we get our final algorithm:

Algorithm 5 Transparently compute f in time $2^k \cdot 2n$ with $n + k \cdot \binom{n}{\leq \lceil d/k \rceil}$ registers.

1: **for** $\ell = 0, \dots, 2^k - 1$ **do**
2: $\quad R^{\text{out}} \leftarrow R^{\text{out}} + \sum_{S \subseteq [n]} \sum_{S' \subseteq [n]} d_{S,S',T_\ell} \left(\prod_{i \in S} R_i^{\text{in}} \right) \left(\prod_{j=1}^{k} R_{j,S'_j} \right)$
3: $\quad$ ToggleMonomialsForGroup(e_ℓ)
4: **end for**

The analysis is identical to that of Algorithm 4, except the runtime is $2^k \cdot 2n$ rather than $2^k \cdot 2\lceil n/k \rceil$ because we do not split the variables into groups. ◀

4 Saving Time

In the previous section we took the length $4n$ branching programs of [11, 12] as a starting point to analyze whether m could be significantly reduced while still maintaining a linear amortized size. In this section we investigate the opposite question: namely, is $4n$ optimal? If we do not restrict the amortized size of our program, then every function has a branching program of length n regardless of m. We will not only consider branching programs of linear amortized size, but the stricter model of *permutation branching programs*.

We focus on permutation branching programs in this section for two reasons. First, all our algorithms in the previous section can be converted to permutation branching programs. To see this, we resist our connection between register programs and m-catalytic branching programs, as proven in Observation 7. Our observation in fact says something stronger, which is that f_n can be computed by a family of permutation branching programs of width $2^{s(n)+1}$. This follows because we can choose to not fix R^{out} to be 0, and instead have one source node corresponding to each initial configuration $\tau_1 \dots \tau_s, \tau^{\text{out}}$; by Definition 6 this

[8] Our use of j here is slightly different than in our previous proof; namely, j is not linked to a specific block of variables, and rather we arbitrarily partitioned S' into k sets and assigned them each a distinct j. This will result in us having to spend time n to load the monomials in, rather than time $\lceil n/k \rceil$ as in the previous proof, but this is necessary as we have no guarantee that there is a partition of the variables such that every monomial of degree at most d is split into k monomials of degree at most $\lceil d/k \rceil$. Note that this is where our algorithm performs worse than Algorithm 4 when $d = \Omega(n)$.

source reaches the sink node labeled $\tau_1 \dots \tau_s, \tau^{\text{out}} + f(x)$, which is the identity permutation when $f(x) = 0$ and a fixed set of $2^{s(n)}$ transpositions otherwise. Thus as a corollary of [11, 12] we get the following:

▶ **Theorem 16.** *Every function f can be computed by a read-4 permutation branching program of width 2^{2^n-1} (or $2^{\binom{n}{\leq d}-1}$, where d is the $\mathbb{F}_2$-degree of p_f).*

Given the strength of these results, only a factor of 4 away from the best conceivable amortized size, there is no reason *a priori* to believe that permutation branching programs are too weak to consider even better upper bounds. Indeed we will show that improvements are possible.

This leads to our other reason for focusing on permutation branching programs: lower bounds against general branching programs are notoriously difficult. Besides the basic counting argument, the best known branching program lower bounds for an explicit function are not even quadratic, using techniques known to go no further [10]. Considering the amortized branching program size needed to compute any function f is always at most the basic branching program size, and considering the upper bound of $4n$ given by [11], proving lower bounds for concrete functions seems exceedingly difficult. Furthermore, even if we were to seek refuge in focusing on non-constructive lower bounds, the basic counting argument fails to prove *any* non-trivial lower bounds in the case of $m \geq 2^n/n$.

4.1 Notation and tools

Before going into our results, we formally define permutation branching programs along with the notation we will use in the rest of this section. These programs will have a more specialized form than when we introduced them in Section 2, which we subsequently justify. We assume basic familiarity with permutations, and we write $\sigma_1\sigma_2$ as a shorthand for $\sigma_2 \circ \sigma_1$.

▶ **Definition 17.** *Let $n, m, s \in \mathbb{N}$ and let $f : \{0,1\}^n \to \{0,1\}$ be a function such that $f(0 \dots 0) = 0$. A* permutation branching program *is a sequence $P = \pi_1 \dots \pi_\ell$, where each π_j is a pair $\langle i_j, \sigma_j \rangle$ where $i_j \in [0..n]$ and σ_j is a permutation of $[m]$. We refer to each π_j as an* instruction *of P. The* width *of P is m and the* length *of P is ℓ.*

For any $\alpha \in \{0,1\}^n$ we define $P(\alpha)$ as follows: fix $\sigma = id$, and for every $j = 1 \dots s$, we set σ to $\sigma\sigma_j$ if $\pi_j = \langle 0, \sigma_j \rangle$ or $\pi_j = \langle i_j, \sigma_j \rangle$ where $\alpha_{i_j} = 1$, and leave σ unchanged otherwise (that is, if $\pi_j = \langle i_j, \sigma_j \rangle$ where $\alpha_{i_j} = 0$). Our output is the final value of σ. We say that P computes f if there exists a permutation $\sigma^* \neq id$ such that $P(\alpha) = id$ if $f(\alpha) = 0$ and $P(\alpha) = \sigma^*$ if $f(\alpha) = 1$.

We make three observations about our choices in Definition 17. First, the restriction that $f(0 \dots 0) = 0$ will be a convenience; we can always compute $\neg f$ instead if this condition does not hold, or change Definition 17 such that $P(\alpha) = id$ if $f(\alpha) = 1$ and vice versa.[9] Second, in a layer $\langle i, \sigma_j \rangle$ reading variable x_i, we only fix a permutation in the case that x_i is set to 1. This is without loss of generality, as adding a layer of the form $\langle 0, \sigma'_j \rangle$ before an instruction can be thought of as choosing a permutation in the case that x_i is set to 0 (while the permutation for $x_i = 1$ can be adjusted accordingly).

Before going on to our third observation, we state and prove four simple lemmas which will allow us to conveniently restructure our programs P.

[9] We also note that if P computes $\neg f$, we can compute f by appending the instruction $\langle 0, (\sigma^*)^{-1} \rangle$ to P. We avoid taking this route because a later observation will allow us to remove these fixed layers, but only when $f(\alpha) = 0$, which would cause our logic to become circular.

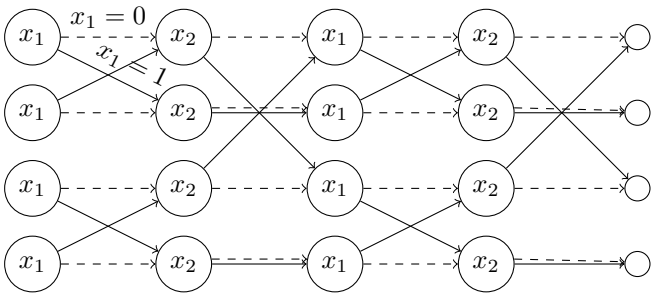


Figure 3 A permutation branching program computing $\mathrm{AND}(x_1, x_2)$. It is identical to the one from Figure 2, except that two more source nodes have been added so that all layers have the same width. As before, each non-sink node is labelled with the input to be read, dashed lines represent transitions taken when that input is zero, and solid lines are taken when the input is one. In the terminology of Definition 17, this program can be written as $\langle 1, (12)(34)\rangle, \langle 2, (13)\rangle, \langle 1, (12)(34)\rangle, \langle 2, (13)\rangle$ (using cycle notation for the permutations).

▶ **Lemma 18.** *Let P be a permutation branching program computing f and let j be such that $i_j = i_{j+1}$. Then the program P' resulting from replacing π_j, π_{j+1} with $\pi'_j = \langle i_j, \sigma_j\sigma_{j+1}\rangle$ is also a valid program for computing f.*

Proof. In both P and P', the permutations σ_j and σ_{j+1} are both applied when $i_j = 1$ and neither are applied when $i_j = 0$. ◀

▶ **Lemma 19.** *Let P be a permutation branching program computing f and let j be such that $\sigma_j\sigma_{j+1} = \sigma_{j+1}\sigma_j$. Then the program P' resulting from switching the order of π_j and π_{j+1} is also a valid program for computing f.*

Proof. Consider any assignment α to x. In the case that either α_{i_j} or $\alpha_{i_{j+1}}$ is set to 0, these programs compute identical permutations as either σ_j or σ_{j+1} will not be applied. If both are set to 1, then

$$P'(\alpha) = \Sigma_1\sigma_{j+1}\sigma_j\Sigma_2 = \Sigma_1\sigma_j\sigma_{j+1}\Sigma_2 = P(\alpha)$$

where Σ_1, Σ_2 are the permutations corresponding to the rest of the instructions on input α. ◀

▶ **Lemma 20.** *Let $P = \pi_1 \dots \pi_s$ be a permutation branching program computing f, let $\pi_j = \langle i_j, \sigma_j\rangle$ for all j, and let $j^* \in [\ell]$ be such that $i_{j^*} = 0$. Then there exists a permutation branching program $P' = \pi'_1 \dots \pi'_{j^*-1}\pi'_{j^*+1} \dots \pi'_\ell\pi_{j^*}$ computing f, where $\pi'_j = \langle i_j, \sigma'_j\rangle$ for some permutation σ'_j.*

Proof. For $j < j^*$ define $\sigma'_j = \sigma_j$, and for $j > j^*$ define $\sigma'_j = \sigma_{j^*}\sigma_j\sigma_{j^*}^{-1}$. Clearly because $\sigma_{j^*}^{-1}$ and σ_{j^*} cancel out for every adjacent pair of permutations σ'_j, $P'(\alpha)$ contains exactly the same permutations as $P(\alpha)$ in the same order regardless of the assignment α.[10] ◀

Our next observation is that the layers of the form $\langle 0, \sigma_j\rangle$ are only a convenience and are not necessary to our definition. Let P be our program for f and let $\sigma_1 \dots \sigma_k$ be the permutations corresponding to instructions π_j where $i_j = 0$. By our restriction that $f(0 \dots 0) = 0$ we get

[10] This argument actually allows us to move π_{j^*} to any spot in the program we want, but we are content with just moving them to the end, for reasons which will become immediately clear.

$\sigma_1 \dots \sigma_k = P(0 \dots 0) = id$, and by Lemma 20 we can move the instructions π_j with $i_j = 0$ to the end of the program, in order, at which point we can simply remove them all using Lemma 18 as they compose to the identity for any input α.

We can also generalize Lemma 20 for *restrictions* of the function f, meaning when we fix the values of some variables and consider the function on the remaining variables. This is simply the observation that fixing variable x_i turns all instructions of the form $\langle i, \sigma_j \rangle$ into fixed layers $\langle 0, \sigma_j \rangle$.

▶ **Corollary 21.** *Let $\rho \in \{0, 1, *\}^n$ and let f_ρ be the function f with x_i fixed to $\rho(i)$ wherever $\rho(i) \neq *$. Let $P = \pi_1 \dots \pi_\ell$ be a permutation branching program computing f, let $\pi_j = \langle i_j, \sigma_j \rangle$ for all j, and let $j^* \in [s]$ be such that $i_{j^*} = 0$ or $\rho(i_{j^*}) \neq *$. Then there exists a permutation branching program $P' = \pi'_1 \dots \pi'_{j^*-1} \pi'_{j^*+1} \dots \pi'_\ell \pi_{j^*}$ computing f_ρ, where $\pi'_j = \langle i'_j, \sigma'_j \rangle$ for some permutation σ'_j and $i'_j = i_j$ iff $\rho(i_j) = *$ and 0 otherwise.*

Proof. Let program P'' be the result of replacing i_j with 0 in each instruction $\pi_j \in P$ such that $\rho(i_j) \neq *$. Clearly this program computes f_ρ, and so applying Lemma 20 to P'' completes the proof. ◀

Assuming that $f_\rho(0 \dots 0) = 0$, by our previous observation this allows us to remove all layers that read variables fixed by ρ. We also note that the other three lemmas hold for f_ρ with no changes.

Finally, when f is the AND function, we can *rotate* our program by moving the first instruction to the end as many times as we like:

▶ **Lemma 22.** *Let $P = \Pi_1, \Pi_2$ be a permutation branching program computing AND$(x_1, \dots, x_n)$, where Π_1 and Π_2 are sequences of instructions. Then $P' = \Pi_2, \Pi_1$ also computes AND$(x_1, \dots, x_n)$.*

Proof. Let $\sigma^* = P'(1, \dots, 1)$ (that is, P' applied to the all-ones input). Referring to Definition 17, to prove that P' computes AND, we must show (a) that $\sigma^* \neq id$, (b) that $P'(\alpha) = id$ whenever $\text{AND}(\alpha) = 0$, and (c) that $P'(\alpha) = \sigma^*$ whenever $\text{AND}(\alpha) = 1$.

(c) follows from the fact that the only string α satisfying $\text{AND}(\alpha) = 1$ is the all-ones string.

Now, let $\alpha \in \{0, 1\}^n$. Note that $P(\alpha) = \Pi_1(\alpha)\Pi_2(\alpha)$. If $\text{AND}(\alpha) = 0$, then $P(\alpha) = id$, so $\Pi_1(\alpha) = (\Pi_2(\alpha))^{-1}$, and so $P'(\alpha) = id$ as well: this proves (b).

Finally, note that $P(1, \dots, 1) \neq id$, so $\Pi_1(1, \dots, 1) \neq (\Pi_2(1, \dots, 1))^{-1}$, and so $\sigma^* = P'(1, \dots, 1) \neq id$, proving (a). ◀

4.2 Upper bounds

For our main upper bound, we modify our algorithm recreating the result of [11] (and analogously [12]) to have length $4n - 4$. In particular, our program will read all but two variables four times, while the last two variables will be read twice.

Proof of Theorem 3. First, we make an easy change to Theorem 9 which allows us to achieve $4n - 3$. Observe that in ToggleInput the order in which we add the inputs is irrelevant, and so consider ToggleMonomials where we reverse the order of toggling on Line 6. Then notice that the last query on Line 2 and the first query on Line 6 are both made to x_n, and so we can merge these two layers along with our entire for loop (which reads no variables) into a single layer querying x_n. Moving to Algorithm 1, this means we

only query x_n twice, and furthermore the last query on Line 1 and the first query on Line 3 are both made to x_1, and so again by merging these two queries along with Line 2 we only query x_1 three times.

Now we will change our program so that x_1 is only read twice. Consider two new functions obtained by fixing the value of x_1, namely $f^0 = f(0, x_2 \dots x_n)$ and $f^1 = f(1, x_2 \dots x_n)$. Recall that we used the following polynomial to compute f, where $y_i = \tau_i^{\text{in}} + x_i$:

$$q_f = \sum_{S,S' \subseteq [n]} c_{S,S'} \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{i \in S'} y_i \right)$$

If we choose $b \in \{0, 1\}$ and fix $\tau_1^{\text{in}} = 0$ and $y_1 = x_1 = b$, we get the following, which can be used to compute (since it is equal to) f^b:

$$q_{f^b} = \sum_{S,S' \subseteq [2..n]} (c_{S,S'} + b \cdot c_{S,S' \cup \{1\}}) \left(\prod_{i \in S} \tau_i^{\text{in}} \right) \left(\prod_{i \in S'} y_i \right)$$

We will use Algorithm 1 to compute q_{f^b}, where $b = x_1$, by removing all reference to x_1 from ToggleInput and ToggleMonomials, and querying x_1 whenever we execute Lines 2 or 4 to determine whether to compute q_{f^0} or q_{f^1} in place of q_f. More specifically, ToggleInput will now only loop over $i = 2 \dots n$, while ToggleMonomials will now only loop over $S \subseteq [2..n]$. Finally in Algorithm 1 we change Lines 2 and 4 to

$$R^{\text{out}} \leftarrow R^{\text{out}} \pm \sum_{S,S' \subseteq [2..n]} (c_{S,S'} + x_1 \cdot c_{S,S' \cup \{1\}}) \left(\prod_{i \in S} R_i^{\text{in}} \right) R_{S'}$$

where Line 2 uses $+$ and Line 4 uses $-$. Note that to execute these lines correctly, we will query x_1 and perform the corresponding instruction; thus we no longer ignore these two lines in calculating our program length.

By our earlier definition of q_{f^b}, this exactly computes q_{f^b} for $x_1 = b$ as claimed. As above we will reverse the order of the queries in ToggleInput the second time it is called in ToggleMonomials, which allows us to read x_n only once per execution for a total of two reads. x_1 will be queried in Lines 2 and 4, and all other variables will be queried four times.

Since we no longer use register R_1, or R_S when $1 \in S$, the number of registers is reduced from the original $n + 2^n$ in Theorem 9 to $n - 1 + 2^{n-1}$. The output register R^{out} is now catalytic, so we add one register for a total $m = 2^{n+2^{n-1}}$. ◀

▶ Note 23. This strategy also allows us to save an exponential number of registers, as we only need a register R_S for each $S \subseteq [2..n]$. While it may be tempting to extend this trick to more variables, say by fixing the values of both x_1 and x_2, the fact that Lines 2 and 4 depend on the value(s) of the fixed variable(s) means that we will have to store at least one of these values in a non-catalytic register, which will add to our width and take us out of the realm of permutation programs. If we go back to m-catalytic branching programs, this gives us another way to save over [11, 12], but with worse parameters; for any $k \in [n]$ by fixing k values we can get a program of length $2(k + 1) + 4(n - k - 1)$ and amortized size $2^k \cdot O(n)$ as before, but for $m = 2^{2^{n-k}-1}$ instead of $2^{2^{n/k}-1}$.

There are two known cases in which we can achieve better than read-4 for AND: $n = 2, 3$. The $n = 2$ case is unsurprising, as our argument allows for two variables to be read twice; it has appeared in many previous works (see c.f. [2]). The case of $n = 3$ is more surprising, and suggests that read-3 may be achievable in general. Note that because of the small values of n involved, neither result gives a program smaller than length $4n - 4$.

▶ **Lemma 24.** *There is a read-2 permutation branching program of width* 3 *computing* $AND(x_1, x_2)$.

Proof. Choose any two permutations σ_1 and σ_2 such that $\sigma_1\sigma_2\sigma_1^{-1}\sigma_2^{-1} \neq id$; for example we can choose $\sigma_1 = (12)$ and $\sigma_2 = (23)$. Then consider the following program:

$$\langle 1, \sigma_1\rangle, \langle 2, \sigma_2\rangle, \langle 1, \sigma_1^{-1}\rangle, \langle 2, \sigma_2^{-1}\rangle$$

By definition of σ_1 and σ_2, $P(1,1) \neq id$, and if either variable is set to 0 then the only permutations left are σ_j and σ_j^{-1} for some $j \in \{1, 2\}$, and the composition of these permutations is id. ◀

▶ **Lemma 25.** *There is a read-3 permutation branching program of width* 3 *computing* $AND(x_1, x_2, x_3)$.

Proof. We state the program and leave the reader to check correctness.[11] Our permutations σ_j are given in cycle notation.

$$\langle 1, (23)\rangle, \langle 2, (12)\rangle, \langle 3, (123)\rangle,$$
$$\langle 1, (12)\rangle, \langle 2, (13)\rangle, \langle 3, (23)\rangle,$$
$$\langle 1, (132)\rangle, \langle 2, (132)\rangle, \langle 3, (13)\rangle$$ ◀

▶ Note 26. We could also consider a stronger model of permutation branching programs where we only require that $P(\alpha) \neq id$ whenever $f(\alpha) = 1$, instead of requiring $P(\alpha)$ always equal the same permutation when $f(\alpha) = 1$. This is the model used by e.g. [2]. In this case, it is not hard to show that for any n, if we can compute $\mathrm{AND}(x_1 \dots x_n)$ in length ℓ, we can also compute any function $f(x_1 \dots x_n)$ in length ℓ by "tensoring" the permutations in P with themselves for each $\alpha \in f^{-1}(1)$. Our lower bounds in the following section will still hold against this model.

4.3 Lower bounds

In this section we show that if one tries to get a program of length less than $3n$, one cannot beat Theorem 3.

Proof of Theorem 4. Let $P = \pi_1\pi_2 \dots \pi_s$ be any program computing $\mathrm{AND}(x_1, \dots, x_n)$. We will write σ_i^j to refer to the permutation in the jth instruction in P that reads variable x_i; in other words, the instructions corresponding to x_i will be $\langle i, \sigma_i^1\rangle \dots \langle i, \sigma_i^k\rangle$ for some k.

▷ Claim 27. Any program P computing AND of more than one variable must read every variable at least twice

Proof. Assume that some variable x_i is read only once in P. Then setting $x_{i'} = 0$ for all $i' \neq i$, we get $\sigma_i^1 = P(0, \dots, 0, 1, 0, \dots, 0) = id$. Therefore P acts identically whether x_i is 0 or 1, which is a contradiction because AND depends on x_1. ◁

[11] It should be noted that we found this program through an automated search, and it would be interesting to see what nice properties of the program – of which there are many candidates – could be useful in searching for read-3 programs for higher n.

Now consider when some variable x_i is read exactly twice. Then P has the form:

$$\Sigma_1, \langle i, \sigma_i^1 \rangle, \Pi_1, \langle i, \sigma_i^2 \rangle, \Sigma_2$$

Rotate this program to produce

$$P' = \langle i, \sigma_i^1 \rangle, \Pi_1, \langle i, \sigma_i^2 \rangle, \Pi_2$$

where $\Pi_2 = \Sigma_2, \Sigma_1$. By Lemma 22, P' also computes AND.

The following is our main claim.

▷ Claim 28. Every variable besides x_i is read at least once in Π_1, and there is at most one such variable $x_{i'}$ which is not read at least twice in Π_1.

The same holds for Π_2.

Proof. First, assume for contradiction that there exists $i' \neq i$ such that $x_{i'}$ does not appear in Π_1. Then if we fix $x_{i''} = 1$ for all $i'' \neq i, i'$, we can apply Corollary 21 to move all instructions querying variables other than x_i and $x_{i'}$ to the end of the program, to get an equivalent program of the following form which computes $\text{AND}(x_i, x_{i'})$:

$$\langle i, \sigma_i'^1 \rangle, \langle i, \sigma_i'^2 \rangle, \Sigma$$

where Σ is a sequence of instructions which do not query x_i. Applying Lemma 18, we can replace $\langle i, \sigma_i'^1 \rangle, \langle i, \sigma_i'^2 \rangle$ with a single instruction $\langle i, \sigma'' \rangle$ to we get an equivalent program

$$\langle i, \sigma_i'' \rangle, \Sigma$$

which also computes $\text{AND}(x_i, x_{i'})$, which contradicts Claim 27 as i is only read once.

Next, assume for contradiction that there exist $i' \neq i'' \neq i$ such that i' and i'' appear only once each in Π_1. If we fix $x_{i'''} = 1$ for all $i''' \neq i, i', i''$, applying Lemmas 21 and 18, without loss of generality the following program computes $\text{AND}(x_i, x_{i'})$:

$$\langle i, \sigma_i'^1 \rangle, \langle i', \sigma_{i'} \rangle, \langle i'', \sigma_{i''} \rangle, \langle i, \sigma_i'^2 \rangle, \Sigma$$

where $\sigma_{i'}$ and $\sigma_{i''}$ are some permutations and Σ is a set of instructions reading only the variables $x_{i'}$ and $x_{i''}$.

Define $\Sigma_{i'}$ to be the result of fixing $x_{i''} = 0$ in Σ, and define $\Sigma_{i''}$ to be the result of fixing $x_{i'} = 0$ in Σ. Note that if only one remaining variable is set to 1 then the program must output 0, so $\sigma_i'^2 = (\sigma_i'^1)^{-1}$, $\Sigma_{i'} = (\sigma_{i'})^{-1}$, and $\Sigma_{i''} = (\sigma_{i''})^{-1}$. Thus if we set $x_{i''} = 0$ our resulting program is

$$\langle i, \sigma_i'^1 \rangle, \langle i', \sigma_{i'} \rangle, \langle i, (\sigma_i'^1)^{-1} \rangle, \langle i', (\sigma_{i'}^1)^{-1} \rangle$$

and so setting $x_i = x_{i'} = 1$ we get that $\sigma_i'^1 \sigma_{i'} (\sigma_i'^1)^{-1} (\sigma_{i'})^{-1} = id$. Therefore by Lemma 19 we can swap the order of these two instructions and get an equivalent program for $\text{AND}(x_i, x_{i'}, x_{i''})$ of the form

$$\langle i', \sigma_{i'} \rangle, \langle i, \sigma_i'^1 \rangle, \langle i'', \sigma_{i''} \rangle, \langle i, (\sigma_i'^1)^{-1} \rangle, \Sigma$$

and similarly by fixing $x_{i'} = 0$ we have $\sigma_i'^1 \sigma_{i''} (\sigma_i'^1)^{-1} (\sigma_{i''})^{-1} = id$, which by Lemma 19 leaves us with the program

$$\langle i', \sigma_{i'} \rangle, \langle i'', \sigma_{i''} \rangle, \langle i, \sigma_i'^1 \rangle, \langle i, (\sigma_i'^1)^{-1} \rangle, \Sigma$$

and applying Lemma 18 on our two layers reading i gives us a program which never reads x_i, which is a contradiction.

Finally, to prove the claim for Π_2, observe that by Lemma 22,

$$P'' = \langle i, \sigma_i^2 \rangle, \Pi_2, \langle i, \sigma_i^1 \rangle, \Pi_1$$

also computes AND, and apply the above argument to P'', with Π_2 playing the role of Π_1. ◁

By a simple analysis of Claim 28, one of two cases must occur for the variables besides x_i: either 1) one variable $x_{i'}$ is read at least twice and all other variables are read at least four times or 2) two variables $x_{i'}, x_{i''}$ are read at least three times and all other variables are read at least four times. This is because a read-2 variable can only be read at most once in each of Π_1 and Π_2, while a read-3 variable will be read at most once in either Π_1 or Π_2. In either of these cases, our branching program must have length at least $4(n-2) + 2 \cdot 2 = 4(n-3) + 2 \cdot 1 + 3 \cdot 2 = 4n - 4$. ◀

▶ Note 29. Besides the fact that our lower bound in Theorem 4 quantitatively matches up with our upper bound in Theorem 3 in the case of reading any variable twice, qualitatively both cases in the analysis at the end of our lower bound proof match with a possible construction given by our upper bound. After fixing a read-2 variable to condition f on, we get two halves to our top level program, and in each of them we will merge two reads of a variable in order to save a further layer. The choice of which variable to merge the reads of is arbitrary, so consider our choices for the first and second half. If we choose the same variable for both halves, it will be read twice and all other variables will be read four times. If we choose different variables in each half, both will be read three times and the rest will be read four times.

5 Open Problems

The most obvious problem left open by our work is to figure out, for an arbitrary function f, the optimal value of m under which we can still achieve linear amortized size. In the upper bounds direction, any improved transparent register program for computing the polynomial q_f could potentially give an upper bound of $2^{2^{o(n)}}$ on m. In the other direction, nothing is known besides the basic counting argument ($m \geq 2^n/O(n)$), and even getting a lower bound of $m \geq 2^n$ for some function f could shed some light on where the correct answer should lie.

In terms of connecting uniform and non-uniform models of space, $\mathsf{L}/\operatorname{poly}$ is equivalent to the class of problems solvable by $\operatorname{poly} n$-size branching programs. However, this gets trickier for *catalytic logspace* (CL), as the corresponding object for $\mathsf{CL}/\operatorname{poly}$ would be m-catalytic branching programs of amortized size $\operatorname{poly}(n)$ for $m = 2^{\operatorname{poly}(n)}$, which has exponential size and thus cannot be written down in polynomial advice. It would be very interesting to understand the connection between such m-catalytic branching programs and $\mathsf{CL}/\operatorname{poly}$, as this would immediately give lower bounds on m for random functions.

Also of interest would be to study the same question for other restricted classes of functions. For example, it is possible that our result for VP could be extended to VNP, although such a result would presumably need to use non-uniformity in a stronger way lest we accidentally prove that uniform VNP is contained in CL – and by extension ZPP [4].

Finally, closing the gap between $3n$ and $4n - 4$ on the upper and lower bounds for the optimal length of permutation branching programs seems within reach. For example, a cursory machine search gave no read-3 permutation branching programs for AND on four variables, and if we could formally verify this then it would immediately lead to fully closing the gap at $4n - 4$.

References

1 Eric Allender, Anna Gál, and Ian Mertz. Dual VP classes. *Comput. Complex.*, 26(3):583–625, 2017.

2 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

3 Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.

4 Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 857–866. ACM, 2014.

5 James Cook and Ian Mertz. Catalytic approaches to the tree evaluation problem. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 752–760. ACM, 2020.

6 James Cook and Ian Mertz. Encodings and the tree evaluation problem. *Electron. Colloquium Comput. Complex.*, page 54, 2021. URL: `https://eccc.weizmann.ac.il/report/2021/054`.

7 Stephen A. Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and branching programs for tree evaluation. *ACM Trans. Comput. Theory*, 3(2):4:1–4:43, 2012.

8 Vincent Girard, Michal Koucky, and Pierre McKenzie. Nonuniform catalytic space and the direct sum for space. *Electronic Colloquium on Computational Complexity (ECCC)*, 138, 2015.

9 Frank Gray. Pulse code communication. `https://patents.google.com/patent/US2632058A/en`, 1953. US Patent 2632058A.

10 E.I. Nečiporuk. A boolean function. *Dokl. Akad. Nauk SSSR*, 169(4), 1966.

11 Aaron Potechin. A note on amortized branching program complexity, 2017.

12 Robert Robere and Jeroen Zuiddam. Amortized circuit complexity, formal complexity measures, and catalytic algorithms. In *FOCS*, pages 759–769. IEEE, 2021.

Subrank and Optimal Reduction of Scalar Multiplications to Generic Tensors

Harm Derksen
Northeastern University, Boston, MA, USA

Visu Makam
Radix Trading Europe B.V., Amsterdam, The Netherlands

Jeroen Zuiddam
Korteweg-de Vries Institute for Mathematics, University of Amsterdam, The Netherlands

Abstract

Since the seminal works of Strassen and Valiant it has been a central theme in algebraic complexity theory to understand the *relative* complexity of algebraic problems, that is, to understand which algebraic problems (be it bilinear maps like matrix multiplication in Strassen's work, or the determinant and permanent polynomials in Valiant's) can be reduced to each other (under the appropriate notion of reduction).

In this paper we work in the setting of bilinear maps and with the usual notion of reduction that allows applying linear maps to the inputs and output of a bilinear map in order to compute another bilinear map. As our main result we determine precisely how many independent scalar multiplications can be reduced to a given bilinear map (this number is called the *subrank*, and extends the concept of matrix diagonalization to tensors), for essentially all (i.e. generic) bilinear maps. Namely, we prove for a generic bilinear map $T : V \times V \to V$ where $\dim(V) = n$ that $\theta(\sqrt{n})$ independent scalar multiplications can be reduced to T. Our result significantly improves on the previous upper bound from the work of Strassen (1991) and Bürgisser (1990) which was $n^{2/3+o(1)}$. Our result is very precise and tight up to an additive constant. Our full result is much more general and applies not only to bilinear maps and 3-tensors but also to k-tensors, for which we find that the generic subrank is $\theta(n^{1/(k-1)})$. Moreover, as an application we prove that the subrank is not additive under the direct sum.

The subrank plays a central role in several areas of complexity theory (matrix multiplication algorithms, barrier results) and combinatorics (e.g., the cap set problem and sunflower problem). As a consequence of our result we obtain several large separations between the subrank and tensor methods that have received much interest recently, notably the slice rank (Tao, 2016), analytic rank (Gowers–Wolf, 2011; Lovett, 2018; Bhrushundi–Harsha–Hatami–Kopparty–Kumar, 2020), geometric rank (Kopparty–Moshkovitz–Zuiddam, 2020), and G-stable rank (Derksen, 2020).

Our proofs of the lower bounds rely on a new technical result about an optimal decomposition of tensor space into structured subspaces, which we think may be of independent interest.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases tensors, bilinear maps, complexity, subrank, diagonalization, generic tensors, random tensors, reduction, slice rank

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.9

Related Version *Full Version*: https://arxiv.org/abs/2205.15168

Funding *Harm Derksen*: NSF grants IIS-1837985 and DMS-2001460.
Visu Makam: NSF grants CCF-1900460 and the University of Melbourne.
Jeroen Zuiddam: Simons Junior Fellowship and NWO Veni grant VI.Veni.212.284.

Acknowledgements The authors thank Swastik Kopparty for helpful discussions, and JM Landsberg and Jan Draisma for comments.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 9; pp. 9:1–9:23

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

1 Introduction

We solve a fundamental problem in algebraic complexity theory about a notion of complexity on bilinear maps (tensors) called the *subrank*, which was introduced by Strassen in [39] in the study of fast matrix multiplication algorithms, and which later found close connections to several hypergraph independence and Ramsey-type problems in combinatorics and tensor methods in these areas (e.g., analytic rank [28, 3] and related notions). Our results improve significantly on previous bounds from the work of Bürgisser [9] and Strassen [41].

In high-level terms, the subrank of a bilinear map is the largest number of independent scalar multiplications that can be reduced to (i.e. "embedded in") the bilinear map, under the natural algebraic complexity notion of reduction (which we elaborate on in a moment). This definition extends the usual rank of matrices and moreover naturally extends further to multilinear maps (k-tensors). In this paper, we:

- determine the subrank for almost all bilinear maps (i.e. generic bilinear maps),
- prove precise bounds that are accurate up to a small additive constant,
- extend the above results from bilinear maps (3-tensors) also to multilinear maps (k-tensors),
- prove as a technical result (used in the proofs of the above) an optimal decomposition theorem for tensor subspace, decomposing tensor space into very structured subspaces,
- prove, as an application of our upper bound on generic subrank, that the subrank is not additive under the direct sum.

Let us biefly state our asymptotic results here. (We will return to these in more detail in Subsection 1.2.) We prove for any vector space[1] V with $\dim(V) = n$ that almost all bilinear maps $T : V \times V \to V$ have subrank equal to $\theta(\sqrt{n})$. That is, $\theta(\sqrt{n})$ independent scalar multiplications can be reduced to T. For k-tensors we find the subrank to be $\theta(n^{1/(k-1)})$ on almost all k-tensors.

To prove our results we use methods from algebraic geometry as well as linear algebraic arguments. Our upper bound proof relies on an efficient parametrization of the set of tensors with subrank larger than a given number (as a collection of orbits) and a good dimension estimate of this set. Our lower bound proof relies on arguments involving differentials. At the core we prove a technical result about a very structured decomposition of tensor space which we think may be of independent interest. The main goal here (in the simplest case) is to write tensor space as a sum of tensor subspaces as efficiently as possible (meaning with smallest as possible sum of dimensions) such that each subspace has the special form of a matrix subspace tensored with n-space. Our technical result is that we can do this optimally for any order of tensor space. We discuss the proof methods further in Subsection 1.3.

To get some intuition for the subrank it is instructive to ask how the subrank relates to the better known complexity notion *tensor rank*. Subrank and rank are indeed closely linked, and in a sense they are dual to each other. Indeed, whereas the subrank of a bilinear map T is the maximal number of independent scalar multiplications that can be reduced to T, the rank of T is (among several equivalent definitions) the minimal number of independent scalar multiplications that T reduces to. In this way, the rank measures the computational "cost" of T in terms of scalar multiplications while the subrank measures the "value" of T in terms of independent scalar multiplications.

[1] We will require the base field of the vector space to have the mild property of being algebraically closed (but it will be clear from our techniques that this assumption can be weakened considerably).

1.1 Subrank and generic subrank

Let us now discuss more precisely several equivalent formulations and the meaning of the definition of subrank, and the sense in which we determine the subrank for almost all tensors (the generic subrank).

Subrank of bilinear and multilinear maps

We first stay in the realm of bilinear maps, and will define subrank via the notion of a reduction between bilinear maps, which is really a reduction in the sense of computational complexity. Given two bilinear maps $S : V_1 \times V_2 \to V_3$ and $T : W_1 \times W_2 \to W_3$ we say S reduces to T and write $S \leq T$ if there are linear maps $L_1 : V_1 \to W_1$, $L_2 : V_2 \to W_2$ and $L_3 : W_3 \to V_3$ such that $S(v_1, v_2) = L_3(T(L_1(v_1), L_2(v_2))$ for all $v_1 \in V_1, v_2 \in V_2$. (In the literature, this relation $\leq$ on tensors is often called the *restriction preorder* and when $S \leq T$ one says that "S is a restriction of T" [39, 8].) In other words, if $S \leq T$, then any algorithm for T also gives an algorithm for S (with only small computational overhead, namely applying the linear maps L_i to inputs and output). Next, an important basic bilinear map that we use to define subrank is (for any positive integer r) the bilinear map I_r that computes r independent scalar multiplications with scalars from some[2] field K:

$$I_r : K^r \times K^r \to K^r : (v_1, v_2) \mapsto ((v_1)_1(v_2)_1, \ldots, (v_1)_r(v_2)_r).$$

With these notions set up, the subrank $\mathrm{Q}(T)$ of a bilinear map $T : V_1 \times V_2 \to V_3$ is defined as the largest number r such that I_r reduces to T, that is, $I_r \leq T$. On the other hand, the tensor rank of T is the smallest number r such that $T \leq I_r$. Thus, the subrank $\mathrm{Q}(T)$ is indeed the largest number of independent scalar multiplications that can be reduced to T, while the rank is the smallest number of independent scalar multiplications that T reduces to. The above definition of subrank extends directly to multilinear maps $V_1 \times \cdots \times V_{k-1} \to V_k$ by naturally extending the reduction and defining I_r as the multilinear map that computes r independent $(k-1)$-wise products.

Subrank of tensors.

Alternatively we may phrase the definition of the subrank in the language of tensors, as bilinear maps (and multilinear maps) naturally correspond to these. Let K^{n_1,n_2,n_3} be the space of 3-tensors (3-dimensional arrays) $T = (T_{i,j,k})_{i,j,k}$ with $i \in [n_1]$, $j \in [n_2]$, $k \in [n_3]$. For tensors $S \in K^{n_1,n_2,n_3}$ and $T \in K^{m_1,m_2,m_3}$ we write $S \leq T$ if there are matrices $A \in \mathrm{Mat}_{n_1,m_1}$, $B \in \mathrm{Mat}_{n_2,m_2}$, $C \in \mathrm{Mat}_{n_3,m_3}$ such that S is obtained from T by applying A, B, C to the slices of T, in the sense that

$$S_{i,j,k} = \sum_{a,b,c} A_{i,a} B_{j,b} C_{k,c} T_{a,b,c}$$

for all $i \in [n_1], j \in [n_2], k \in [n_3]$. Let $I_r \in K^{r,r,r}$ be the tensor for which the diagonal entries $(I_r)_{i,i,i}$ are 1 and the other entries are 0. In this language the subrank $\mathrm{Q}(T)$ is again the largest number r such that $I_r \leq T$. This definition of subrank also naturally extends to higher-order tensors $K^{n_1,\ldots,n_k}$.

[2] We will put some mild restrictions on the field in some parts of the paper.

Subrank on "almost all" tensors; generic subrank

It follows from a short argument (which we give later; Proposition 9) that for vector spaces V_1, V_2, V_3, there is a subset U of all bilinear maps $V_1 \times V_2 \to V_3$ that is nonempty and Zariski-open (the complement is the zero-set of a finite collection of polynomials) and with constant subrank. This set U thus contains almost all bilinear maps $V_1 \times V_2 \to V_3$. The subrank on U is called the *generic subrank*. We denote the generic subrank of bilinear maps $K^{n_1} \times K^{n_2} \to K^{n_3}$ (or equivalently of tensors in K^{n_1,n_2,n_3}) by $\mathrm{Q}(n_1, n_2, n_3)$ (and by $\mathrm{Q}(n_1, \ldots, n_k)$ generally for higher-order tensors).

1.2 Our Results

As our main results we determine (1) the subrank of generic tensors of order three, and, more generally, (2) the subrank of generic tensors of any order $k \geq 3$. Moreover, as a core technical ingredient in our proof of (1) and (2), we prove (3) an optimal decomposition of tensor space into highly structured subspaces, which we think is of independent interest and which may have further applications in algebraic complexity theory. We will now describe each of these results in detail.

(1) The generic subrank of tensors of order three.

We will begin by discussing our results for tensors of order three (or equivalently, for trilinear maps or bilinear maps on vector spaces of appropriate dimensions). Recall that $\mathrm{Q}(n) = \mathrm{Q}(n, n, n)$ denotes the generic subrank of tensors in $K^{n,n,n}$. In other words, $\mathrm{Q}(n)$ is the value that the subrank takes on "essentially all" tensors, or on "randomly chosen" tensors with probability 1. We prove that $\mathrm{Q}(n)$ grows as $\sqrt{n}$.

▶ **Theorem 1.** *We have* $\mathrm{Q}(n) = \theta(\sqrt{n})$.

We thus solve the problem of determining the generic subrank of tensors of order three (and also, with more work, of order k in general as we will discuss below). In computational terms, Theorem 1 states that for "essentially all" bilinear maps $T : K^n \times K^n \to K^n$ we can reduce the problem of multiplying $\sqrt{n}$ independent pairs of numbers to T, and that this is optimal (under the usual notion of reduction from algebraic complexity theory in which linear maps are applied to the inputs and output of T).

In particular, with Theorem 1 we significantly improve on the previous best upper bound on $\mathrm{Q}(n)$ of Strassen and Bürgisser (obtained via the method of "lower support functionals") which was $\mathrm{Q}(n) \leq n^{2/3+o(1)}$.

As an application of Theorem 1 we prove that the subrank is not additive under the direct sum. Namely, we prove that there are bilinear maps $S, T : K^n \times K^n \to K^n$ such that $\mathrm{Q}(T) + \mathrm{Q}(S) = \theta(\sqrt{n})$ while $\mathrm{Q}(T \oplus S) \geq n$. In other words, it is sometimes possible to reduce many more independent scalar multiplications to a direct sum of bilinear maps than to the bilinear maps separately.[3]

[3] This result is the subrank analogue of the recent result of Shitov [36] that disproved Strassen's additivity conjecture for tensor rank [37]. For this he constructed a complicated example of 3-tensors S, T (over any infinite field) such that $\mathrm{R}(S \oplus T) < \mathrm{R}(S) + \mathrm{R}(T)$. We discuss additivity results further in Subsection 1.4.

As a direct consequence of Theorem 1, we separate the generic subrank $\mathrm{Q}(n)$ from the generic asymptotic subrank $\underset{\sim}{\mathrm{Q}}(n)$[4], for which Strassen proved that the generic value satisfies $\underset{\sim}{\mathrm{Q}}(n) \geq n^{2/3}$ [40, Prop. 3.6]. Moreover, our result gives a large separation between the subrank on the one hand, and the slice rank (and partition rank, for higher order tensors), geometric rank, and G-stable rank on the other hand (whose generic value is full).

Let us now discuss the bound of Theorem 1 in more detail, and in particular get the precise constants right, after which we discuss the extension to higher-order tensors. We (naturally) obtain Theorem 1 in two parts, namely by first proving the following upper bound:

▶ **Theorem 2.** *We have* $\mathrm{Q}(n) \leq \lfloor \sqrt{3n-2} \rfloor$.

And then proving the following essentially matching lower bound:

▶ **Theorem 3.** *We have* $\mathrm{Q}(n) \geq 3(\lfloor \sqrt{n/3+1/4} - 1/2 \rfloor)$.

It is not difficult to see that our upper and lower bounds on the generic subrank are very precise. Namely, the difference between the upper bound $\lfloor \sqrt{3n-2} \rfloor$ and the lower bound $3(\lfloor \sqrt{n/3+1/4} - 1/2 \rfloor)$ is at most a small additive constant.[5]

We conjecture that our upper bound in Theorem 2 is tight. Through a more sophisticated analysis (which requires as a final component a computer verification that a determinant is nonzero) we prove that for all $1 \leq n \leq 100$ our upper bound is in fact tight, that is: for all $1 \leq n \leq 100$ we have $\mathrm{Q}(n) = \lfloor \sqrt{3n-2} \rfloor$.

(2) The generic subrank of k-tensors.

So far we have discussed only tensors of order three. With more elaborate methods we are able to completely extend our above results from tensors of order three to tensors of order k for every $k \geq 3$. Denoting the subrank of a generic tensor in $K^{n,\dots,n}$ of order k by $\mathrm{Q}(n,\dots,n)$, we find that $\mathrm{Q}(n,\dots,n)$ grows as $n^{1/(k-1)}$.

▶ **Theorem 4.** *We have* $\mathrm{Q}(n,\dots,n) = \theta(n^{1/(k-1)})$.

Again this result directly leads to a large separation between the subrank and the asymptotic subrank (for which the generic value, extending a construction of Strassen, satisfies $\underset{\sim}{\mathrm{Q}}(n) \geq n^{2/k}$), and a separation between the subrank and the slice rank, partition rank, geometric rank and G-stable rank (all of whose generic value is n).

Regarding the precise bounds, we prove Theorem 4 in two parts again. In the first we extend the upper bound of Theorem 2 to order k, in a fully general manner as an upper bound on the generic subrank $\mathrm{Q}(n_1,\dots,n_k)$ where the n_i need not be equal, as follows.

▶ **Theorem 5.** *We have* $\mathrm{Q}(n_1,\dots,n_k) \leq \left(\sum_{i=1}^k n_i - (k-1)\right)^{\frac{1}{k-1}}$.

Then finally we extend Theorem 3 to order k tensors, which leads to the asymptotically matching lower bound (which for conciseness we will not write down here more explicitly).

▶ **Theorem 6.** *We have* $\mathrm{Q}(n,\dots,n) \geq \Omega(n^{1/(k-1)})$.

Our proof of this last theorem (and also Theorem 3) makes crucial use of the technical results on tensor space that we discuss in the next section.

---

[4] The asymptotic subrank is defined as $\underset{\sim}{\mathrm{Q}}(T) = \lim_{n\to\infty} \mathrm{Q}(T^{\otimes n})^{1/n}$ and $\underset{\sim}{\mathrm{Q}}(n)$ denotes the value of $\underset{\sim}{\mathrm{Q}}(T)$ for a generic tensor $T \in K^{n,n,n}$.

[5] Indeed, a straightforward direct computation shows that $\lfloor \sqrt{3n-2} \rfloor - 3(\lfloor \sqrt{n/3+1/4} - 1/2 \rfloor) \leq 5$.

(3) Technical result: structured subspace decomposition of tensor space

The proofs of our lower bounds (Theorem 3 and Theorem 6) rely on a technical result about a very structured decomposition of tensor space which we think may be of independent interest, so we will describe it here. The main goal here (in the simplest case) is to write tensor space $K^{n,n,n}$ as a sum of tensor subspaces as efficiently as possible (meaning with smallest as possible sum of dimensions) such that each subspace has the form of an $n \times n$ matrix subspace tensored with K^n (so that it becomes a subspace of $K^{n,n,n}$) with the tensoring being applied in any of the three possible directions. Our result is that we can do this optimally for any order of tensor space.

We will begin our discussion with the simplest version, which is for the tensor space $K^{3,3,3}$, as it is the easiest to explain (and in fact also not hard to prove) and forms the basis for the proof of the (harder to prove) general version for the tensor space $K^{n,\dots,n} = (K^n)^{\otimes n}$ of tensors of order n and dimension n in each direction (in our application to the generic subrank we use a blow-up argument so that we can deal with tensors of order k and dimension n in each direction).

Recall that K^{n_1,n_2,n_3} denotes the space of $n_1 \times n_2 \times n_3$ tensors with coefficients in the field K, which we require to be large enough in this part. Let $\mathrm{Mat}_{n,m}$ denote the space of $n \times m$ matrices with coefficients in K. We use a special notation to denote a certain construction of a tensor subspaces given a matrix subspace. Namely, for any given $n_1 \times n_2$ matrix subspace $\mathcal{W} \subseteq \mathrm{Mat}_{n_1,n_2}$ we denote by $\mathcal{W}[3]$ the tensor subspace $\mathcal{W} \otimes K^{n_3} \subseteq K^{n_1,n_2,n_3}$. Analogously, for any matrix subspace $\mathcal{W} \subseteq \mathrm{Mat}_{n_2,n_3}$ we denote by $\mathcal{W}[1] \subseteq K^{n_1,n_2,n_3}$ the tensor subspace obtained by appropriately tensoring $\mathcal{W}$ with K^{n_1}, and for any matrix subspace $\mathcal{W} \subseteq \mathrm{Mat}_{n_1,n_3}$ we denote by $\mathcal{W}[2] \subseteq K^{n_1,n_2,n_3}$ the tensor subspace obtained by appropriately tensoring W with K^{n_2}.

For the tensor space $K^{3,3,3}$ we have the following optimal decomposition theorem.

▶ **Theorem 7.** *There exist subspaces $\mathcal{X}_i \subseteq \mathrm{Mat}_{3,3} = K^3 \otimes K^3$, each of dimension 3, such that*

$$K^{3,3,3} = \mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3].^6$$

Comparing dimensions, we have $\dim K^{3,3,3} = 3 \cdot 3 \cdot 3 = \dim \mathcal{X}_1[1] + \dim \mathcal{X}_2[2] + \dim \mathcal{X}_3[3]$ and so the decomposition in Theorem 7 is optimal. In other words, it is a direct sum decomposition of $K^{3,3,3}$.

The requirement in Theorem 7 that the $\mathcal{X}_i$ all have dimension 3 is crucial to make the theorem interesting, as without this requirement we could "decompose" $K^{3,3,3}$ simply as $K^{3,3,3} = \mathcal{X}_1[1]$ with $\mathcal{X}_1 = \mathrm{Mat}_{3,3}$. Interestingly, the analogous statement of Theorem 7 for any matrix space $K^{n,n}$ is false. That is, for every positive integer n there are no subspaces $\mathcal{X}_i \subseteq K^n$, each of dimension $n/2$, such that $K^{n,n} = \mathcal{X}_1[1] + \mathcal{X}_2[2]$. This is saying that if we pick a row space R and a column space C of dimension $n/2$, then it is not possible to write every $n \times n$ matrix A as $A_1 + A_2$ where the row space of A_1 is in R and the column space of A_2 is in C. On the other hand, 3-tensors do not suffer from this malady: we can write any tensor as a sum of thee tensors, whose "row_i"-space are asked to be in a generic space of the right dimension. (Here we can think of $W[i]$ as tensors whose "row_i"-space is W.) Therefore, Theorem 7 is demonstrating an interesting phenomenon that occurs in tensor space but not in matrix space.

[6] We note that the existence of subspaces $\mathcal{X}_i$ with this property is equivalent to generic subspaces $\mathcal{X}_i$ having this property (i.e. there being a non-empty Zariski-open set of triples $\mathcal{X}_i$ with this property).

Theorem 7 (as opposed to the upcoming generalization to $(K^n)^{\otimes n}$) is not difficult to prove. Indeed, one may choose the matrix subspaces $\mathcal{X}_i$ randomly and then for such an explicit choice verify directly that they satisfy the claim (and this approach will work with high probability). In fact, there are even valid choices of the $\mathcal{X}_i$ such that each $\mathcal{X}_i$ is spanned by a matrix with coefficients in $\{0,1\}$.

Next we discuss the higher-dimensional version of Theorem 7 in which $K^{3,3,3}$ is generalized to n-tensor space $(K^n)^{\otimes n}$. We naturally extend our previous notation so that for every tensor subspace $\mathcal{W} \subseteq (K^n)^{\otimes(n-1)}$ we define, for any $i \in \{1,\dots,n\}$, the tensor subspace $\mathcal{W}[i] \subseteq (K^n)^{\otimes n}$ by tensoring $\mathcal{W}$ with K^n in one of the n possible ways.

By a recursive construction with $K^{3,3,3}$ as a base case, we find the following optimal decomposition of n-tensor space $(K^n)^{\otimes n}$ for all $n \geq 3$.

▶ **Theorem 8.** *For every integer $n \geq 3$ there exist subspaces $\mathcal{X}_i \subseteq (K^n)^{\otimes n-1}$ of dimension n^{n-2} such that*

$$(K^n)^{\otimes n} = \mathcal{X}_1[1] + \mathcal{X}_2[2] + \cdots + \mathcal{X}_n[n].$$

Again, since $\dim K^{n,\dots,n} = n^n = n{\cdot}n{\cdot}n^{n-2} = \sum_{i=1}^n \dim \mathcal{X}_i[i]$, the decomposition in Theorem 8 is optimal in terms of dimension and hence a direct sum decomposition of $(K^n)^{\otimes n}$.

Theorem 8 is the theorem we use to prove the general generic subrank lower bound Theorem 6. However, the methods we introduce in the process of proving Theorem 8 allow us much more generally for other choices of positive integers $n_1,\dots,n_k$ to construct optimal decompositions $K^{n_1,\dots,n_k} = \mathcal{X}_1[1] + \mathcal{X}_2[2] + \cdots + \mathcal{X}_k[k]$ from known decompositions. This leads to a natural fundamental mathematical question (with potentially other applications) of what choices of $n_1,\dots,n_k$ and $\dim \mathcal{X}_i$ allow such decompositions.

1.3 Technical Overview

We give a brief technical overview of the methods and ideas that we use in our proofs.

Upper bounds on generic subrank

The high-level approach in our proof of the upper bound on the generic subrank in Theorem 2 (and similarly for the general case of Theorem 5) is as follows. For any nonnegative integer we consider the set $\mathcal{C}_r$ of tensors in $K^{n,n,n}$ with subrank at least r. We argue that the generic subrank is precisely the largest r such that the dimension of $\mathcal{C}_r$ equals the dimension n^3 of the full space $K^{n,n,n}$. We then prove the core ingredient, namely the dimension upper bound $\dim(\mathcal{C}_r) \leq n^3 - r(r^2 - 3n + 2)$. This information leads to the desired result, since if we let t be the generic subrank $\mathrm{Q}(n)$, then we must by the above have $n^3 = \dim(\mathcal{C}_t) \leq n^3 - t(t^2 - 3n + 2)$, from which we directly deduce that $t \leq \sqrt{3n-2}$, that is, we obtain the bound $\mathrm{Q}(n) \leq \sqrt{3n-2}$ of Theorem 2.

To prove the aforementioned dimension upper bound on $\mathcal{C}_r$ that is the core ingredient in the above argument we employ the idea of providing a (non-injective) parametrization of $\mathcal{C}_r$, compute the dimension of the parameter space, and then subtracting the dimension "over-count" (the fiber dimension under the parametrization). For this we first define a set X_r of tensors in $K^{n,n,n}$ of a special form, namely whose $[r] \times [r] \times [r]$ subtensor is zero except for the diagonal which is nonzero. Then the elements of X_r clearly have subrank at least r (and are thus in $\mathcal{C}_r$). The important point is that, by applying all possible basis transformations to the tensors in X_r we obtain all of $\mathcal{C}_r$. Thus X_r together with the group of all basis transformations provide the parametrization of $\mathcal{C}_r$. Technically, we describe

this by saying that the map $\psi_r : \mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r \to K^{n,n,n}$ that maps (A, B, C, T) to $(A \otimes B \otimes C)T$ has image precisely $\mathcal{C}_r$. Now the computation to upper bound $\dim(\mathcal{C}_r)$ consists of computing the dimension of the domain $\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r$ and subtracting the dimension of a general fiber of ψ_r, which we carry out to arrive at the dimension upper bound stated earlier.

Lower bounds on generic subrank

The high-level approach in our proof of the lower bound on the generic subrank in Theorem 3 (and the general Theorem 6) is as follows. We use notation defined in the upper bound proof discussion and the results section (Subsection 1.2). Our proof reduces the problem of lower bounding the generic subrank $\mathrm{Q}(n_1, n_2, n_3)$ to a problem of constructing tensor space decompositions of a specific form (which we discuss further in the next section). Namely we prove that, for $r \leq n_1, n_2, n_3$ (with a technical condition), if $\mathcal{X}_i \subseteq \mathrm{Mat}_{r,r}$ are subspaces of dimension $n_i - r$ for $i = 1, 2, 3$ such that

$$\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{r,r,r},$$ [7]

then $\mathrm{Q}(n_1, n_2, n_3) \geq r$. (And we prove the analogous statement for higher-order tensors.) Finding such $\mathcal{X}_i$ we discuss in the next section. The proof of the lower bound given the $\mathcal{X}_i$ goes as follows.

Recall the map $\psi_r : \mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r \to K^{n,n,n}$ whose image we already claimed is the set $\mathcal{C}_r$ of tensors of subrank at least r. To reach our goal we want to find conditions that imply that the image of ψ_r has full dimension n^3 and thus is Zariski-dense in $K^{n,n,n}$. To do this we use the notion of the differential $d\psi_r$ of ψ_r and a general method that says that the dimension of a map can be computed as the rank of the differential at a "generic point".

The differential $d\psi_r$ at the point (g_1, g_2, g_3, T) is the map

$$(d\psi_r)_{(g_1,g_2,g_3,T)} : \mathrm{Mat}_{n,n} \times \mathrm{Mat}_{n,n} \times \mathrm{Mat}_{n,n} \times Y_r \to K^{n,n,n}$$

where Y_r is the tangent space of X_r, given by

$$(A, B, C, S) \mapsto ((A \otimes g_2 \otimes g_3) + (g_1 \otimes B \otimes g_3) + (g_1 \otimes g_2 \otimes C))T + (g_1 \otimes g_2 \otimes g_3)S.$$

Analyzing this map we compute its image which leads to the aforementioned lower bound statement on the generic subrank. In fact we prove a stronger lower bound, which in characteristic 0 characterizes the generic subrank precisely, but with a harder to analyze condition, in Theorem 21. In particular, if the characteristic of K is 0, then Theorem 21 says that $\mathrm{Q}(n_1, n_2, n_3)$ is given by the smallest number r such that $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r}$ for generic subspaces $\mathcal{X}_i \subseteq \mathrm{Mat}_{r,r}$ of dimension $n_i - r$, where $W_r \subseteq K^{r,r,r}$ is the subspace of tensors such that $T_{ijk} = 0$ if i, j, k are all different. (In other characteristics this number r gives a lower bound.)

Constructions of tensor space decompositions

To prove Theorem 8 we introduce general methods to construct the optimal tensor space decompositions as described in the theorem from existing ones. We then give some small constructions and combine these in multiple recursions to achieve the required object.

[7] where $\mathcal{X}_1[1] \subseteq K^{r,r,r}$ denotes $\mathcal{X}_1$ tensored with K^r in the first tensor leg, $\mathcal{X}_2[2]$ denotes $\mathcal{X}_2$ tensored with K^r in the second tensor leg, and $\mathcal{X}_3[3]$ denotes $\mathcal{X}_3$ tensored with K^r in the third tensor leg

To set this up we take a very general approach in which we study direct sum decompositions

$$K^{n_1,n_2,\ldots,n_k} = \mathcal{X}_1[1] + \cdots + \mathcal{X}_k[k]$$

where $\mathcal{X}_i \subseteq K^{n_1} \otimes \cdots \otimes K^{n_{i-1}} \otimes K^{n_{i+1}} \otimes \cdots \otimes K^{n_k}$ is a tensor subspace and $\mathcal{X}_i[i]$ denotes the subspace obtained by tensoring $\mathcal{X}_i$ with K^{n_i} as the ith tensor factor. Let a_i be the dimension of $\mathcal{X}_i$. We are interested in which values of $n_1, \ldots, n_k$ and $a_1, \ldots, a_k$ allow for a decomposition of the above form. Writing these numbers into a $2 \times k$ matrix

$$\begin{bmatrix} n_1 & n_2 & \cdots & n_k \\ a_1 & a_2 & \cdots & a_k \end{bmatrix}$$

we let $\mathcal{S}$ be the set of all such matrices for which a decomposition exists satisfying the parameters given in the matrix. Then Theorem 8 corresponds to proving that the $2 \times n$ matrix

$$\begin{bmatrix} n & n & \cdots & n \\ n^{n-2} & n^{n-2} & \ldots & n^{n-2} \end{bmatrix} \tag{1}$$

is an element of $\mathcal{S}$.

Next we observe that there are some simple constructions and properties of elements in $\mathcal{S}$, such as: if a matrix is in $\mathcal{S}$ then if we permute its columns it is still in $\mathcal{S}$ and the matrix $\left[\begin{smallmatrix} n \\ 1 \end{smallmatrix}\right]$ is in $\mathcal{S}$. With slightly more work we can give direct constructions for

$$\begin{bmatrix} 2 & 2 & 2 \\ 0 & 3 & 1 \end{bmatrix} \tag{2}$$

being an element of $\mathcal{S}$, for instance.

In order to construct more elements of $\mathcal{S}$ we prove a "direct sum construction" that given two elements in $\mathcal{S}$ combines them to get a new one. Namely, this result gives that, if

$$\begin{bmatrix} n_1 & n_2 & \ldots & n_{k-1} & n'_k \\ a'_1 & a'_2 & \cdots & a'_{k-1} & a_k \end{bmatrix} \in \mathcal{S}$$

and

$$\begin{bmatrix} n_1 & n_2 & \ldots & n_{k-1} & n''_k \\ a''_1 & a''_2 & \cdots & a''_{k-1} & a_k \end{bmatrix} \in \mathcal{S}$$

then we have

$$\begin{bmatrix} n_1 & n_2 & \cdots & n_k \\ a_1 & a_2 & \cdots & a_k \end{bmatrix} \in \mathcal{S}$$

where $n_k = n'_k + n''_k$, and $a_i = a'_i + a''_i$ for $i = 1, 2, \ldots, k-1$.

Finally, from some simple base cases including (2) we give a construction for

$$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

being an element of $\mathcal{S}$ and via an elaborate argument with multiple recursions arrive at the matrix in (1) being an element of $\mathcal{S}$, which is the ingredient required in our proof of Theorem 6. It is natural to ask what precisely are all elements of $\mathcal{S}$, which we leave as an open problem.

1.4 Related Work

The previous best bound on the generic subrank $\mathrm{Q}(n)$ was the upper bound $n^{2/3+o(1)}$ which follows from the work of Bürgisser [9, Satz 2.8][8] as part of the broader research program on the theory of asymptotic spectra of tensors (motivated by the study of matrix multiplication algorithms). The proof of this bound relies on the method of "lower support functionals" introduced by Strassen in [41] (see also the more recent surveys on this topic in [10, 44, 43]) and the properties of these that he proves there. This method recovers certain asymptotic information about tensors, which importantly is monotone under the restriction preorder and normalized on diagonal tensors so that it provides an upper bound on the subrank (in a manner that is very different from the approach that we take to prove our optimal upper bound). Bürgisser's analysis of this method on generic tensors consists of proving that the support of a generic tensor is large for any choice of basis and a combinatorial study of a certain type of covering of these supports, which leads to the aforementioned $n^{2/3+o(1)}$ upper bound.

Recent research has brought about a rich collection of tensor methods that are in a similar "regime" as the subrank (for instance, they are all monotone under restriction), each with their own properties and applications in complexity theory and combinatorics. Notable are the slice rank [42] and closely related partition rank [30], the analytic rank [20, 28, 3], the geometric rank [26, 18] and G-stable rank [14]. Some important applications of these methods include new bounds on cap sets [42, 16], new bounds on the sunflower problem [31], determining the border subrank of matrix multiplication [26], and proving matrix multiplication barriers [6, 1, 11]. Many strong connections have been shown among these parameters. In particular, Derksen [14] showed that the G-stable rank is equal to the slice rank up to a constant factor, and Cohen–Moshkovitz [12, 13] showed (over large fields) that the partition rank, analytic rank and geometric rank are equal up to a constant factor, the culmination of a long line of work on this topic [21, 24, 2, 22, 23, 29]. All of the aforementioned tensor parameters are lower bounded by the subrank.[9] Our results provide a large separation on almost all tensors between the subrank and all other aforementioned parameters, as the generic subrank satisfies $\mathrm{Q}(n) = \theta(\sqrt{n})$ whereas the generic value of all other parameters is the maximal value n.

The study of "generic" or "typical" complexity in algebraic complexity theory goes back to at least Strassen's paper "Rank and optimal computation of generic tensors" [38], in which he determines the tensor rank of almost all tensors (i.e. generic tensors). His result is that the rank $\mathrm{R}(n_1, n_2, n_3)$ of almost all tensors in K^{n_1,n_2,n_3} grows as $n_1n_2n_3/(n_1+n_2+n_3-2)$ and a description is given of "perfect shapes" (n_1, n_2, n_3) for which precisely equality $\mathrm{R}(n_1, n_2, n_3) = n_1n_2n_3/(n_1+n_2+n_3-2)$ holds. (It is still a fundamental open problem to find explicit tensors in $K^{n,n,n}$ with rank close to n^2 [5] with important implications to formula lower bounds [32].) For the shape $(n, n, 3)$ with n odd this work provides an equation that determines whether a tensor has typical rank or not. These equations lead to several later generalizations [27, 25] and subsequently precise barrier results for "rank methods" [15, 17]. Our result rather than generic rank determines the generic subrank. The rank of a tensor being the smallest number of scalar multiplications that the tensor reduces to, the subrank is a natural dual to the tensor rank. We note that combining the results on generic subrank and generic rank, we

[8] [9, Satz 2.8] determines the generic value of a tensor parameter called the "lower support functional" (unteren Trägerfunktional) which upper bounds the subrank as proven in [41].

[9] The analytic rank requires a natural normalization for this to be true, that is, one should normalize it by the analytic rank of a full-dimensional diagonal tensor.

see that reducing scalar multiplications to a generic tensor and reducing the generic tensor back to scalar multiplications necessarily induces a great loss, as $\mathrm{Q}(n,n,n) = \theta(\sqrt{n})$ is much smaller than $\mathrm{R}(n,n,n) = \theta(n^2)$.

The study of additivity results is a central theme in complexity theory and mathematics. In algebraic complexity, it has been long known (and crucial in the design of matrix multiplication algorithms) that the border rank of tensors (the approximative version of tensor rank) is not additive under the direct sum [35] (see also [4, Lemma 7.1] and [8, (15.12)]). Strassen conjectured the tensor rank to be additive under the direct sum, but this was disproved recently by Shitov [36]. On the other hand, the analytic rank [28], geometric rank [26], G-stable rank [14] and slice rank [19] were shown to be additive recently. The subrank, as we show, is however not additive. Our proof of this relies on writing a tensor as a sum of two generic tensors, which is reminiscent of methods that Razborov [33] uses to prove a linear upper bound on submodular complexity measures of boolean functions (see also [34]).

1.5 Paper Organization

In Section 2 we prove the upper bound theorems Theorem 2 and Theorem 5. In Section 3 we prove the characterization of generic subrank that allows us to reduce the problem of determining its value to the tensor subspace decomposition problem. In Section 4 we solve the tensor subspace decomposition problem thus completing the lower bound proof. In Section 5 we use our upper bounds to prove that the subrank is not additive under direct sum. In Section 6 we discuss several natural related open problems.

2 Upper bounds on generic subrank

In this section, we prove our upper bounds on the generic subrank of tensors. We give a detailed proof in the case of 3-tensors, and then generalize to tensors of all orders.

2.1 Tensors of order three

The techniques we use are familiar in invariant theory and representation theory. The main idea is to take advantage of the fact that subrank is invariant under a large group of symmetries, namely change of basis on each tensor leg. Further, this group of symmetries has excellent algebraic properties which can often be leveraged to remarkable effect.

First and foremost, we have to argue that generic subrank is a valid notion, which we do in the following proposition. The proof is a standard argument which we will discuss because it naturally uses some ingredients that we will use later on.

▶ **Proposition 9.** *For every n there is a non-empty Zariski-open subset $U \subseteq K^{n,n,n}$ and integer r such that for all $T \in U$ we have $\mathrm{Q}(T) = r$.*

For any n, the number r given by Proposition 9 is unique since any two non-empty Zariski-open subsets $U_1, U_2 \subseteq K^{n,n,n}$ must intersect ($K^{n,n,n}$ is irreducible). This number r we call the generic subrank $\mathrm{Q}(n)$. Similarly we define the generic subrank $\mathrm{Q}(n_1, \dots, n_k)$ of $K^{n_1,\dots,n_k}$.

We discuss a couple of preparatory results before giving the proof of Proposition 9. We define X_r to be the set of tensors in $K^{n,n,n}$ whose $[r] \times [r] \times [r]$ subtensor is zero except for the diagonal entries in $[r] \times [r] \times [r]$ which are all nonzero,

$$X_r = \left\{ T \in K^{n,n,n} \mid T_{ijk} = 0 \text{ for } (i,j,k) \in [r]^3 \setminus \{(i,i,i) \mid i \in [r]\} \text{ and } T_{i,i,i} \neq 0 \text{ for } i \in [r] \right\}.$$

We let ψ_r be the map that applies basis transformations to elements of X_r,

$$\psi_r : \mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r \longrightarrow K^{n,n,n}$$
$$(A, B, C, T) \longmapsto (A \otimes B \otimes C)T.$$

We define $\mathcal{C}_r$ to be the set of tensors in $K^{n,n,n}$ whose subrank is at least r,

$$\mathcal{C}_r = \{T \in K^{n,n,n} \mid \mathrm{Q}(T) \geq r\}.$$

▶ **Lemma 10.** *The image of ψ_r is precisely $\mathcal{C}_r$.*

Proof. To prove $\mathrm{im}(\psi_r) = \mathcal{C}_r$ we show both inclusions.

First we will prove $\mathrm{im}(\psi_r) \subseteq \mathcal{C}_r$. As a first step we will prove that $X_r \subseteq \mathcal{C}_r$. Let $T \in X_r$. Let $T_{i,i,i} = \lambda_i \neq 0$. Let

$$A = \begin{pmatrix} \mathrm{Id}_r & 0 \\ 0 & 0 \end{pmatrix}$$

where Id_r denotes the identity matrix of size $r \times r$ and let

$$B = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}$$

where D is a diagonal matrix of size $r \times r$ whose diagonal entries are $\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_r^{-1}$. It is easy to check that $(A \otimes A \otimes B) \cdot T = I_r$. Thus $X_r \subseteq \mathcal{C}_r$. Since subrank is invariant under the action of $\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n$, we see that $\mathcal{C}_r$ is $\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n$ invariant, so we deduce that $\mathrm{im}(\psi_r) = (\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n) \cdot X_r \subseteq \mathcal{C}_r$.

Now we will prove $\mathcal{C}_r \subseteq \mathrm{im}(\psi_r)$. Let $T \in \mathcal{C}_r$. Then, there exist $A, B, C \in \mathrm{Mat}_{r,n}$ such that $(A \otimes B \otimes C) \cdot T = I_r$. Let

$$\widetilde{A} = \begin{pmatrix} A \\ * \end{pmatrix}$$

be a completion of A to a full rank $n \times n$ matrix, and similarly define $\widetilde{B}$ and $\widetilde{C}$. This is possible because A, B, C must all have rank r. Then $(\widetilde{A} \otimes \widetilde{B} \otimes \widetilde{C}) \cdot T \in X_r$, which implies that $T \in \mathrm{im}(\psi_r)$. Thus $\mathcal{C}_r \subseteq \mathrm{im}(\psi_r)$ ◀

We now give the proof of the existence of the generic subrank (Proposition 9). The proof is standard and may safely be skipped.

Proof of Proposition 9. The map Q attains only finitely many values on $V = K^{n,n,n}$, namely $\{0, 1, \ldots, n\}$. Thus it has finitely many fibers $P_i = \mathrm{Q}^{-1}(i) \subseteq V$. Each P_i is a constructible set, since $P_i = \mathcal{C}_i \setminus \mathcal{C}_{i+1}$ and $\mathcal{C}_i$ is constructible as a consequence of Lemma 10. Then $\bigcup_{i=0}^{n} P_i = V$ and so $\bigcup_{i=0}^{n} \overline{P_i} = V$. However, V is irreducible so there must be an i such that $\overline{P_i} = V$. Since P_i is constructible it contains a subset $U \subseteq P_i$ that is non-empty and Zariski-open in $\overline{P_i} = V$ (this is a general fact, see e.g. [7]). This U and i satisfy the claim. ◀

Now that we have established that the generic subrank exists, we continue to prove our upper bound on it. The following simple lemma is straightforward, but crucial:

▶ **Lemma 11.** *The generic subrank* $\mathrm{Q}(n) = \max\{r \mid \dim(\mathcal{C}_r) = n^3\}$.

Proof. As above, let P_i denote the subset of tensors with subrank i. Then, by definition of $\mathrm{Q}(n)$ and the fact that $\mathcal{C}_r = \bigsqcup_{i=r}^{n} P_i$, we deduce that $\dim(\mathcal{C}_r) = n^3$ if and only if $\mathcal{C}_r \supseteq P_{\mathrm{Q}(n)}$ if and only if $r \leq \mathrm{Q}(n)$. ◀

▶ **Proposition 12.** *The following is an upper bound for the dimension of $\mathcal{C}_r$:*

$$\dim(\mathcal{C}_r) \leq 3n^2 + (n^3 - r^3 + r) - 3(n(n-r)+r) = n^3 - r(r^2 - 3n + 2).$$

Proof. Consider the map ψ above. The theorem on dimension of fibres says that

$$\dim(\mathcal{C}_r) = \dim(\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r) - \dim(\text{general fiber of } \psi_r).$$

We see that $\dim(\mathrm{GL}_n \times \mathrm{GL}_n \times \mathrm{GL}_n \times X_r) = 3n^2 + (n^3 - r^3 + r)$ because each GL_n contributes n^2 to the dimension and X_r is a Zariski-open subset of a linear subspace of $K^{n,n,n}$ of dimension $(n^3 - r^3 + r)$. Thus, to compute $\dim(\mathcal{C}_r)$, we only need to compute the dimension of the general fiber of ψ_r. Note that the dimension of any fiber is at most the dimension of the general fiber, so it suffices to find a lower bound on the dimension of all fibers.

Suppose $T \in \mathcal{C}_r$, then $T = (g_1 \otimes g_2 \otimes g_3) \cdot S$ for some $S \in X_r$ by Lemma 10. It is easy to see that $\dim(\psi^{-1}(T)) = \dim(\psi^{-1}(S))$ since $\psi^{-1}(T)$ and $\psi^{-1}(S)$ are isomorphic as varieties – indeed this follows from the observation that $(A, B, C, U) \in \psi^{-1}(T) \iff (g_1^{-1}A, g_2^{-1}B, g_3^{-1}C, U) \in \psi^{-1}(S)$. Thus, it suffices to lower bound the dimension of $\psi^{-1}(S)$ for $S \in X_r$.

Let $S \in X_r$. Let $L \subseteq \mathrm{GL}_m$ be the subset of matrices of the form

$$\begin{pmatrix} D & 0 \\ * & * \end{pmatrix}$$

where D is a diagonal matrix of size $r \times r$. It is easy to see that $\dim(L) = n(n-r) + r$. For $A, B, C \in L$, it is easy to see that $(A \otimes B \otimes C) \cdot S \in X_r$. Thus, $(A^{-1}, B^{-1}, C^{-1}, (A \otimes B \otimes C) \cdot S) \in \psi^{-1}(S)$. In particular, this means that $\dim(\psi^{-1}(S)) \geq 3\dim(L) = 3(n(n-r)+r)$. Thus, we conclude that $\dim(\text{generic fiber}) \geq 3(n(n-r)+r)$ and so

$$\dim(\mathcal{C}_r) \leq 3n^2 + (n^3 - r^3 + r) - 3(n(n-r)+r) = n^3 - r(r^2 - 3n + 2). \quad \blacktriangleleft$$

Now, we have everything necessary to prove the upper bound for the subrank of 3-tensors.

Proof of Theorem 2. Suppose the subrank of a generic tensor in $K^{n,n,n}$ is $t = \mathrm{Q}(n)$. Then we know that $n^3 = \dim(\mathcal{C}_t)$ by Lemma 11 and $\dim(\mathcal{C}_t) \leq n^3 - t(t^2 - 3n + 2)$ by Proposition 12. Thus, we must have $t^2 - 3n + 2 \leq 0$, so $t \leq \sqrt{3n-2}$. Hence, we have $\mathrm{Q}(n) \leq \sqrt{3n-2}$ as desired. ◀

2.2 Higher-order tensors

In the general case, where we look at tensors in $K^{n_1,n_2,\ldots,n_k}$, we define analogously the objects X_r and $\mathcal{C}_r$, the map $\psi : \prod_{i=1}^k \mathrm{GL}_{n_i} \times X_r \to \mathcal{C}_r$, etc.

Proof of Theorem 5. We obtain analogously that

$$\dim(\mathcal{C}_r) = \Big(\sum_{i=1}^k n_i^2\Big) + \Big(\prod_{i=1}^k n_i - r^k + r\Big) - \sum_{i=1}^k (n_i(n_i - r) + r) = \prod_{i=1}^k n_i - r\Big(r^{k-1} - \sum_{i=1}^k n_i + (k-1)\Big).$$

Suppose the subrank of a generic tensor in $K^{n_1,n_2,\ldots,n_k}$ is $t = \mathrm{Q}(n_1, \ldots, n_k)$. Then, we have

$$\prod_{i=1}^k n_i = \dim(\mathcal{C}_t) \leq \prod_{i=1}^k n_i - t\Big(t^{k-1} - \sum_{i=1}^k n_i + (k-1)\Big),$$

so we get that $t^{k-1} - \sum_{i=1}^{k} n_i + (k-1) \leq 0$, so that

$$\mathrm{Q}(n) = t \leq \Big(\sum_{i=1}^{k} n_i - (k-1)\Big)^{\frac{1}{k-1}}$$

as desired. ◀

3 Lower bounds on generic subrank

In this section, we describe the technique we use to show lower bounds on generic subrank. In order to make this technique effective, we will need some explicit constructions of linear subspaces which we postpone to the next section.

First, we introduce some notation. We identify Mat_{n_2,n_3} with $K^{n_2} \otimes K^{n_3}$ in the standard way. For a linear subspace $\mathcal{X} \subseteq \mathrm{Mat}_{n_2,n_3}$, we define a linear subspace

$$\mathcal{X}[1] = K^{n_1} \otimes \mathcal{X} \subseteq K^{n_1,n_2,n_3}.$$

The linear subspace $\mathcal{X}[1]$ consists precisely of the tensors whose slices in the first direction (i.e., matrices $(T_{1jk})_{j,k}, (T_{2jk})_{j,k}, \ldots, (T_{njk})_{j,k}$) are in $\mathcal{X}$. Similarly, we define $\mathcal{X}[2]$ (resp. $\mathcal{X}[3]$) as the set of tensors whose slices in the second (resp. third) direction are in $\mathcal{X}$.

The main idea behind proving our lower bounds is the following result:

▶ **Theorem 13.** *Let $r \leq n_1, n_2, n_3$ such that $n_i - r \leq r^2$.*[10] *Let $\mathcal{X}_i \subseteq \mathrm{Mat}_{r,r}$ be a generic subspace of dimension $n_i - r$ for $i = 1, 2, 3$. Suppose $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{r,r,r}$, then*

$$\mathrm{Q}(n_1, n_2, n_3) \geq r.$$

In fact, we can prove a stronger version of the above theorem, which we state as Theorem 21. However, the downside of this stronger version is that it has a hypothesis that is more difficult to work with.

To prove that r is a lower bound for the generic subrank, we need to show that the image of ψ_r has dimension n^3, that is, the image is Zariski-dense in $K^{n,n,n}$. The dimension of the image of a map can be captured by the rank of the differential at a generic point – an idea that is familiar to differential geometers and algebraic geometers alike. In arbitrary characteristic, we know that the rank of the differential at a generic point is a lower bound for the dimension of the image, which is sufficient for proving lower bounds. In characteristic 0, the image of the rank of the differential at a generic point is equal to the dimension of the image. Consequently, we are able to obtain an exact linear algebraic characterization for generic subrank in Theorem 21.

Recall that we defined the map

$$\psi_r : \mathrm{GL}_{n_1} \times \mathrm{GL}_{n_2} \times \mathrm{GL}_{n_3} \times X_r \to K^{n_1,n_2,n_3},$$

where

$$X_r = \left\{T \in K^{n_1,n_2,n_3} \mid T_{ijk} = 0 \text{ for } (i,j,k) \in [r]^3 \setminus \{(i,i,i) \mid i \in [r]\} \text{ and } T_{i,i,i} \neq 0 \text{ for } i \in [r]\right\}.$$

Observe that

$$Y_r = \left\{T \in K^{n_1,n_2,n_3} \mid T_{ijk} = 0 \text{ for } (i,j,k) \in [r]^3 \setminus \{(i,i,i) \mid i \in [r]\}\right\}$$

[10] Without this assumption it does not mean anything to have a generic subspace of dimension $n_i - r$. Moreover, if $n_i - r > r^2$, then it is easy to see that $\mathrm{Q}(n_1, n_2, n_3) \geq r$.

is the tangent space of X_r. The differential at a point (g_1, g_2, g_3, T) is

$$(d\psi_r)_{(g_1,g_2,g_3,T)} : \mathrm{Mat}_{n_1,n_1} \times \mathrm{Mat}_{n_2,n_2} \times \mathrm{Mat}_{n_3,n_3} \times Y_r \longrightarrow K^{n_1,n_2,n_3}$$

given by

$$(A, B, C, S) \longmapsto ((A \otimes g_2 \otimes g_3) + (g_1 \otimes B \otimes g_3) + (g_1 \otimes g_2 \otimes C))T + (g_1 \otimes g_2 \otimes g_3)S.$$

▶ **Lemma 14.** *If for generic $(g_1, g_2, g_3, T) \in \mathrm{GL}_{n_1} \times \mathrm{GL}_{n_2} \times \mathrm{GL}_{n_3} \times X_r$ we have*

$$\mathrm{im}((d\psi_r)_{(g_1,g_2,g_3,T)}) = K^{n_1,n_2,n_3},$$

then $\mathrm{Q}(n_1, n_2, n_3) \geq r$. The converse holds if the characteristic of K is 0.

Proof. If the rank of the differential $d\psi_r$ at a generic point is full, then the image of ψ_r must be full dimensional, that is, Zariski-dense (and constructible). Every tensor in the image of ψ_r has subrank $\geq r$. Hence, the generic subrank must be at least r.

Assume now that characteristic of K is zero. If the rank of $d\psi_r$ at a generic point is not full, then the image of ψ_r is not full dimensional, that is, the set of tensors having subrank $\geq r$ is not Zariski-dense, so we get $\mathrm{Q}(n_1, n_2, n_3) < r$, thereby proving the converse. ◀

We use the following equivariance property of the differential.

▶ **Lemma 15.** *We have for all $g_i \in \mathrm{GL}_{n_i}$ and $T \in X_r$ that*

$$\mathrm{im}((d\psi_r)_{(g_1,g_2,g_3,T)}) = (g_1 \otimes g_2 \otimes g_3)(\mathrm{im}((d\psi_r)_{(I,I,I,T)}).$$

Proof. We see directly that

$$(d\psi_r)_{(g_1,g_2,g_3,T)}(A, B, C, S) = (g_1 \otimes g_2 \otimes g_3)(d\psi_r)_{(I,I,I,T)}(g_1^{-1}A, g_2^{-1}B, g_3^{-1}C, S)$$

which implies the claim. ◀

▶ **Corollary 16.** *If for generic $T \in X_r$ we have that $\mathrm{im}((d\psi_r)_{(I,I,I,T)}) = K^{n_1,n_2,n_3}$, then $\mathrm{Q}(n_1, n_2, n_3) \geq r$. The converse holds if the characteristic of K is 0.*

So, let us now analyze more carefully the image of $(d\psi_r)_{(I,I,I,T)}$ for a generic tensor T. Below, we write d for $(d\psi_r)_{(I,I,I,T)}$ for notational simplicity. Thus d is given by

$$d(A, B, C, S) = ((A \otimes I \otimes I) + (I \otimes B \otimes I) + (I \otimes I \otimes C)) \cdot T + S.$$

The map d is a linear map, so we see that

$$\mathrm{im}\ d = \sum_{i=1}^{3} d(\mathrm{Mat}_{n_i,n_i}) + d(Y_r)$$

where we use the notation $d(Y_r) = d(0, 0, 0, Y_r)$, $d(\mathrm{Mat}_{n_1,n_1}) = d(\mathrm{Mat}_{n_1,n_1}, 0, 0, 0)$, etc.

▶ **Lemma 17.** *The image of Y_r under the differential map d is*

$$d(Y_r) = Y_r$$

Proof. Observe that for any $S \in Y_r$, we have $d(0, 0, 0, S) = S$. ◀

Let $L_i = [T_{ijk}]_{j,k}$ for $1 \leq i \leq n_1$. These just split the tensor in the first direction as a stack of n matrices. Let $\mathcal{L}$ denote the span of the L_is.

▶ **Lemma 18.** *The image of* Mat_{n_1,n_1} *under the differential* d *is*

$$d(\mathrm{Mat}_{n_1,n_1}) = \mathcal{L}[1]. \tag{3}$$

Proof. For any $A \in \mathrm{Mat}_{n_1,n_1}$, let $T' = d(A,0,0,0) = (A \otimes I \otimes I) \cdot T$. Consider the slices $L_i = [T_{ijk}]_{j,k}$. Then, it is straighforward to compute that for $1 \le i \le n_1$, the slice

$$[T'_{ijk}]_{j,k} = \sum_{t=1}^{n_1} a_{it} L_i. \qquad \blacktriangleleft$$

Similar to L_i, define the slices in the other directions as $M_j = [T_{ijk}]_{i,k}$ and $N_k = [T_{ijk}]_{i,j}$. Let $\mathcal{M}$ and $\mathcal{N}$ denote the spans of the M_js and N_ks respectively.

▶ **Corollary 19.** *The image of the differential* d *is*

$$\mathrm{im}\ d = \mathcal{L}[1] + \mathcal{M}[2] + \mathcal{N}[3] + Y_r.$$

Proof. This follows from Lemma 18 (and its counterparts in the other two directions) and Lemma 17. ◀

Now, we refine the above corollary. Denote by $\widehat{L_i}$ the top-left $r \times r$ submatrix of L_i. Then, let $\widehat{\mathcal{L}} = \mathrm{span}(\widehat{L_1}, \widehat{L_2}, \ldots, \widehat{L_{n_1}})$. Similarly define $\widehat{M_j}, \widehat{N_k}$ and $\widehat{\mathcal{M}}$ and $\widehat{\mathcal{N}}$.

▶ **Proposition 20.** *The image of the differential* d *is*

$$\mathrm{im}\ d = \widehat{\mathcal{L}}[1] + \widehat{\mathcal{M}}[2] + \widehat{\mathcal{N}}[3] + Y_r.$$

Proof. This follows directly from Corollary 19 and the definition of the coordinate subspace Y_r. ◀

Now, we can finally prove Theorem 13.

Proof of Theorem 13. Let $T \in X_r$ be generic. Define L_i, M_i, N_i as above. Let

$$\mathcal{X}_1 = \mathrm{span}(\widehat{L_{r+1}}, \widehat{L_{r+2}}, \ldots, \widehat{L_{n_1}}).$$

Then $\mathcal{X}_1$ is a generic subspace of $\mathrm{Mat}_{r,r}$ of dimension $n_1 - r$. We similarly define

$$\mathcal{X}_2 = \mathrm{span}(\widehat{M_{r+1}}, \widehat{M_{r+2}}, \ldots, \widehat{M_{n_2}})$$
$$\mathcal{X}_3 = \mathrm{span}(\widehat{N_{r+1}}, \widehat{N_{r+2}}, \ldots, \widehat{N_{n_3}}).$$

By hypothesis, $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{r,r,r}$. Hence, by Proposition 20, we see that

$$\mathrm{im}(d) = \widehat{\mathcal{L}}[1] + \widehat{\mathcal{M}}[2] + \widehat{\mathcal{N}}[3] + Y_r \supseteq \mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + Y_r = K^{r,r,r} + Y_r = K^{n_1,n_2,n_3}.$$

Thus, we get that $\mathrm{Q}(n_1,n_2,n_3) \ge r$ by Corollary 16. ◀

We observe that the idea from the proof of Theorem 13 actually yields the stronger version below. To state this we define the linear subspace

$$W_r = \{T \in K^{r,r,r} \mid T_{ijk} = 0 \text{ if } i,j,k \text{ are all different}\} \subseteq K^{r,r,r}.$$

▶ **Theorem 21.** *Let* $r \le n_1, n_2, n_3$ *such that* $n_i = r < r^2$ *for* $i = 1,2,3$. *Let* $\mathcal{X}_i \subseteq \mathrm{Mat}_{r,r}$ *be a generic subspace of dimension* $n_i - r$ *for* $i = 1,2,3$. *Suppose*

$$\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r},$$

then $\mathrm{Q}(n_1,n_2,n_3) \ge r$. *Further, if characteristic of* K *is* 0, *then we have the converse.*

In other words, if characteristic of K is 0, then Theorem 21 says that $\mathrm{Q}(n_1, n_2, n_3)$ is given by the smallest number r such that $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r}$ for generic subspaces $\mathcal{X}_i \subseteq \mathrm{Mat}_{r,r}$ of dimension $n_i - r$.

Proof. The proof is similar to that of Theorem 13. Take a generic $T \in X_r$. Define L_i, M_i, N_i as above. Set $\mathcal{X}_1 = \mathrm{span}(\widehat{L_{r+1}}, \widehat{L_{r+2}}, \ldots, \widehat{L_{n_1}})$. Then $\mathcal{X}_1$ is a generic subspace of $\mathrm{Mat}_{r,r}$ of dimension $n_1 - r$. Let $P_1 = \mathrm{span}(\widehat{L_1}, \widehat{L_2}, \ldots, \widehat{L_r})$. Similarly define $\mathcal{X}_2 = \mathrm{span}(\widehat{M_{r+1}}, \widehat{M_{r+2}}, \ldots, \widehat{M_{n_2}})$ and $\mathcal{X}_3 = \mathrm{span}(\widehat{N_{r+1}}, \widehat{N_{r+2}}, \ldots, \widehat{N_{n_3}})$ and also $P_2 = \mathrm{span}(\widehat{M_1}, \ldots, \widehat{M_r})$ and $P_3 = \mathrm{span}(\widehat{N_1}, \ldots, \widehat{N_r})$.

It is a straightforward computation to see that since T is generic in X_r, we have $P_1[1] + P_2[2] + P_3[3] = W_r$. Hence, using Proposition 20, we see that

$$\begin{aligned}\mathrm{im}(d) &= \widehat{\mathcal{L}}[1] + \widehat{\mathcal{M}}[2] + \widehat{\mathcal{N}}[3] + Y_r \\ &= \mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + P_1[1] + P_2[2] + P_3[3] + Y_r \\ &= \mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r + Y_r\end{aligned}$$

Now, we claim that

$$\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r} \Leftrightarrow \mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r + Y_r = K^{n_1,n_2,n_3}.$$

The proof of the claim is rather straightforward, the only subtle point being that Y_r and $K^{r,r,r}$ have an intersection (i.e., the main diagonal of $K^{r,r,r}$), but this intersection is included in W_r anyway, so the claim goes through.

Thus, we get that $\mathrm{im}(d) = K^{n_1,n_2,n_3}$ if and only if $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r}$. Now, the theorem follows from applying Corollary 16. ◀

In the next section we will prove that the requirement of Theorem 13 is satisfied. This goes as follows. We will show (Lemma 26) that:

▶ **Lemma 22.** *Let $\mathcal{X}_i \subseteq \mathrm{Mat}_{3,3}$ be a generic subspace of dimension 3 for $i = 1, 2, 3$. Then $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{3,3,3}$.*

Then from a blow-up argument we obtain:

▶ **Lemma 23.** *Let $\mathcal{X}_i \subseteq \mathrm{Mat}_{3d,3d}$ be a generic subspace of dimension $3d^2$ for $i = 1, 2, 3$. Then $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{3d,3d,3d}$.*

Proof. It is enough to construct one such choice of $\mathcal{X}_i$. By Lemma 22 there exist subspaces $\mathcal{Y}_i \subseteq \mathrm{Mat}_{3,3}$ of dimension 3 such that $\mathcal{Y}_1[1] + \mathcal{Y}_2[2] + \mathcal{Y}_3[3] = K^{3,3,3}$. Then $\mathcal{X}_i = \mathcal{Y}_i \otimes \mathrm{Mat}_{d,d}$ satisfy the claim. ◀

▶ **Theorem 24.** *Let d be a positive integer such that $n - 3d \geq 3d^2$. Then $\mathrm{Q}(n, n, n) \geq 3d$.*

Proof. Taking $r = 3d$, this follows from Theorem 13 and Lemma 23. ◀

Proof of Theorem 3. If we take $d = \lfloor \sqrt{n/3 + 1/4} - 1/2 \rfloor$, then we have $d \leq \sqrt{n/3 + 1/4} - 1/2$, so $(d+1/2)^2 \leq n/3 + 1/4$. Thus, $d^2 + d + 1/4 \leq n/3 + 1/4$, so $d^2 + d \leq n/3$ or equivalently $3d^2 \leq n - 3d$. Thus, it follows from Theorem 24 that $\mathrm{Q}(n) \geq 3d$. ◀

For the higher-order lower bound Theorem 6 we prove (Lemma 27) the analogous higher-order version of Lemma 22 which we can similarly blow up and apply Theorem 13 to.

4 Constructions of tensor space decompositions

We assume that the base field K is infinite.

Let $\mathcal{C}$ be the set of all matrices $\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d \\ a_1 & a_2 & \cdots & a_d \end{smallmatrix}\right]$ for which $n_1, n_2, \ldots, n_d$ are positive integers, and $a_1, a_2, \ldots, a_d$ are nonnegative integers, and $\sum_{i=1}^{d} a_i n_i = \prod_{i=1}^{d} n_i$. Let $\mathcal{S}$ be the subset of all matrices $\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d \\ a_1 & a_2 & \cdots & a_d \end{smallmatrix}\right] \in \mathcal{C}$ with the following property: If $V_1, V_2, \ldots, V_d$ are vector spaces of dimensions $n_1, n_2, \ldots, n_d$ respectively, and W_i is a general subspace of $\widehat{V}_i := V_1 \otimes V_2 \otimes \cdots \otimes V_{i-1} \otimes V_{i+1} \otimes \cdots \otimes V_d$ of dimension a_i for all i, then $\sum_{i=1}^{d} \Phi_i(W_i \otimes V_i) = V_1 \otimes \cdots \otimes V_d$, where $\Phi : \widehat{V}_i \otimes V_i \to V_1 \otimes V_2 \otimes \cdots \otimes V_d$ is the isomorphism given by permuting the factors.

4.1 General construction methods

We will use the following simple facts. It is obvious that if a matrix lies in $\mathcal{S}$ and we permute its columns, then it will still lie in $\mathcal{S}$. It is also clear that $\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d \\ a_1 & a_2 & \cdots & a_d \end{smallmatrix}\right]$ lies in $\mathcal{S}$ if and only if $\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d & 1 \\ a_1 & a_2 & \cdots & a_d & 0 \end{smallmatrix}\right]$ lies in $\mathcal{S}$. The vector $\left[\begin{smallmatrix} n \\ 1 \end{smallmatrix}\right]$ lies in $\mathcal{S}$ for all positive integers n. Finally,

$$\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d & 1 & 1 \\ a_1 & a_2 & \cdots & a_d & b_1 & b_2 \end{smallmatrix}\right] \in \mathcal{S} \quad \text{if and only if} \quad \left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d & 1 \\ a_1 & a_2 & \cdots & a_d & b_1+b_2 \end{smallmatrix}\right] \in \mathcal{S} \tag{4}$$

▶ **Lemma 25** (Direct sum construction). *Suppose that $n_d = n'_d + n''_d$, and $a_i = a'_i + a''_i$ for $i = 1, 2, \ldots, d-1$, and*

$$\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_{d-1} & n'_d \\ a'_1 & a'_2 & \cdots & a'_{d-1} & a_d \end{smallmatrix}\right], \left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_{d-1} & n''_d \\ a''_1 & a''_2 & \cdots & a''_{d-1} & a_d \end{smallmatrix}\right] \in \mathcal{S}.$$

Then we have $\left[\begin{smallmatrix} n_1 & n_2 & \cdots & n_d \\ a_1 & a_2 & \cdots & a_d \end{smallmatrix}\right] \in \mathcal{S}$.

Proof. Suppose $V_1, V_2, \ldots, V_d$ are vector spaces of dimensions $n_1, n_2, \ldots, n_d$ respectively and choose a general subspace W_d of V_d of dimension a_d. We can write $V_d = V'_d \oplus V''_d$ where V'_d and V''_d have dimensions n'_d and n''_d respectively. For $i = 1, 2, \ldots, d-1$ choose a general subspace $W'_i \subseteq V_1 \otimes \cdots \otimes V_{i-1} \otimes V_{i+1} \otimes \cdots \otimes V_{d-1} \otimes V'_d$ of dimension a'_i such that $V_1 \otimes V_2 \otimes \cdots \otimes V_{d-1} \otimes V'_d$ is equal to $\left(\sum_{i=1}^{d-1} \Phi_i(W'_i \otimes V_i)\right) + W_d \otimes V'_d$. Similarly, for $i = 1, 2, \ldots, d-1$, choose a general subspace $W''_i \subseteq V_1 \otimes \cdots \otimes V_{i-1} \otimes V_{i+1} \otimes \cdots \otimes V_{d-1} \otimes V''_d$ of dimension a''_i such that $V_1 \otimes V_2 \otimes \cdots \otimes V_{d-1} \otimes V''_d$ is equal to $\left(\sum_{i=1}^{d-1} \Phi_i(W''_i \otimes V_i)\right) + W_d \otimes V''_d$. Set $W_i = W'_i \oplus W''_i \subseteq V_1 \otimes \cdots \otimes V_{i-1} \otimes V_{i+1} \otimes \cdots \otimes V_d$ for $i = 1, 2, \ldots, d-1$. Then we have

$$\sum_{i=1}^{d} \Phi_i(W_i \otimes V_i) = \left(\sum_{i=1}^{d-1} \Phi_i((W'_i \oplus W''_i) \otimes V_i)\right) + W_d \otimes (V'_d \oplus V''_d) =$$
$$\left(\left(\sum_{i=1}^{d-1} \Phi_i(W'_i \otimes V_i)\right) + W_d \otimes V'_d\right) \oplus \left(\left(\sum_{i=1}^{d-1} \Phi_i(W''_i \otimes V_i)\right) + W_d \otimes V''_d\right) =$$
$$(V_1 \otimes \cdots \otimes V_{d-1} \otimes V'_d) \oplus (V_1 \otimes \cdots \otimes V_{d-1} \otimes V''_d) = V_1 \otimes V_2 \otimes \cdots \otimes V_d.$$

This finishes the proof. ◀

4.2 Recursive construction for every order

The following lemma gives Lemma 22 which we used in the proof of the generic subrank lower bound for order three. After that we will recursively obtain what is needed for the higher-order case generic subrank lower bound.

▶ **Lemma 26** (Base case). *We have $\left[\begin{smallmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \end{smallmatrix}\right] \in \mathcal{S}$.*

Proof. We can verify explicitly that $\left[\begin{smallmatrix}2&2&2\\0&3&1\end{smallmatrix}\right] \in \mathcal{S}$, as follows. For a 3-dimensional subspace W_1 of $V_2 \otimes V_3 = K^{2\times 2}$, take the space of all matrices of the form

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

and for the 1-dimensional subspace W_3 of $V_1 \otimes V_2 = K^{2\times 2}$, take the span of the identity matrix. A tensor in the intersection of $W_1 \otimes V_3 \cap V_1 \otimes W_3 \subseteq V_1 \otimes V_2 \otimes V_3$ is of the form

$$\left(\begin{array}{cc|cc} a_1 & b_1 & a_2 & b_2 \\ b_1 & c_1 & b_2 & c_2 \end{array}\right) = \left(\begin{array}{cc|cc} p_1 & 0 & 0 & p_1 \\ p_2 & 0 & 0 & p_2 \end{array}\right)$$

for some $a_1, a_2, b_1, b_2, c_1, c_2, p_1, p_2$. Clearly, $b_1 = c_1 = a_2 = b_2 = 0$. So we get $p_1 = b_2 = 0$ and $p_2 = b_1 = 0$. So $W_1 \otimes V_3 \cap V_1 \otimes W_3 = 0$ and $W_1 \otimes V_3 + V_1 \otimes W_3$ has dimension $3 \cdot 2 + 2 \cdot 1 = 8$ and therefore must be equal to $V_1 \otimes V_2 \otimes V_3 = K^{2\times 2\times 2}$. Using Lemma 25, we now deduce:

- $\left[\begin{smallmatrix}2&2&2\\0&3&1\end{smallmatrix}\right], \left[\begin{smallmatrix}1&2&2\\0&0&2\end{smallmatrix}\right] \in \mathcal{S}$, so $\left[\begin{smallmatrix}3&2&2\\0&3&3\end{smallmatrix}\right] = \left[\begin{smallmatrix}2+1&2&2\\0&3+0&1+2\end{smallmatrix}\right] \in \mathcal{S}$
- $\left[\begin{smallmatrix}3&1&1\\1&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}3&1&1\\0&3&0\end{smallmatrix}\right] \in \mathcal{S}$, so $\left[\begin{smallmatrix}3&1&2\\1&3&0\end{smallmatrix}\right] = \left[\begin{smallmatrix}3&1&1+1\\1+0&0+3&0\end{smallmatrix}\right] \in \mathcal{S}$
- $\left[\begin{smallmatrix}3&2&2\\0&3&3\end{smallmatrix}\right], \left[\begin{smallmatrix}3&1&2\\1&3&0\end{smallmatrix}\right] \in \mathcal{S}$, so $\left[\begin{smallmatrix}3&3&2\\1&3&3\end{smallmatrix}\right] = \left[\begin{smallmatrix}3&2+1&2\\0+1&3&3+0\end{smallmatrix}\right] \in \mathcal{S}$
- $\left[\begin{smallmatrix}3&1&1\\1&0&0\end{smallmatrix}\right], \left[\begin{smallmatrix}3&1&1\\0&0&3\end{smallmatrix}\right] \in \mathcal{S}$, so $\left[\begin{smallmatrix}3&3&1\\2&0&3\end{smallmatrix}\right] = \left[\begin{smallmatrix}3&1+1+1&1\\1+1+0&0&0+0+3\end{smallmatrix}\right] \in \mathcal{S}$
- $\left[\begin{smallmatrix}3&3&1\\2&0&3\end{smallmatrix}\right], \left[\begin{smallmatrix}3&3&2\\1&3&3\end{smallmatrix}\right] \in \mathcal{S}$, so $\left[\begin{smallmatrix}3&3&3\\3&3&3\end{smallmatrix}\right] = \left[\begin{smallmatrix}3&3&1+2\\2+1&0+3&3\end{smallmatrix}\right] \in \mathcal{S}$

The final line finishes the proof. ◀

To continue, we define the "concatenation" notation:

$$\left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d\\ a_1 & a_2 & \cdots & a_d\end{smallmatrix}\right] \odot \left[\begin{smallmatrix}m_1 & m_2 & \cdots & m_e\\ b_1 & b_2 & \cdots & b_e\end{smallmatrix}\right] = \left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d & m_1 & m_2 & \cdots & m_e\\ a_1 & a_2 & \cdots & a_d & b_1 & b_2 & \cdots & b_e\end{smallmatrix}\right]$$

and the k-fold repeated version of this notation:

$$\left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d\\ a_1 & a_2 & \cdots & a_d\end{smallmatrix}\right]^{\odot k} = \underbrace{\left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d\\ a_1 & a_2 & \cdots & a_d\end{smallmatrix}\right] \odot \left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d\\ a_1 & a_2 & \cdots & a_d\end{smallmatrix}\right] \odot \cdots \odot \left[\begin{smallmatrix}n_1 & n_2 & \cdots & n_d\\ a_1 & a_2 & \cdots & a_d\end{smallmatrix}\right]}_{k}.$$

We will now use Lemma 25 and Lemma 26 to prove:

▶ **Lemma 27.** *For all $n \geq 3$, we have $\left[\begin{smallmatrix}n\\ n^{n-2}\end{smallmatrix}\right]^{\odot n} \in \mathcal{S}$.*

Proof. We already know the case $n = 3$ (Lemma 26), so assume that $n \geq 4$.

We will first show that $\left[\begin{smallmatrix}n\\ n\end{smallmatrix}\right]^{\odot 3} \odot \left[\begin{smallmatrix}1\\ n^2\end{smallmatrix}\right]^{\odot(n-3)} \in \mathcal{S}$. We note that a straightforward direct construction gives

$$\left[\begin{smallmatrix}n & n & 1\\ k & 0 & n^2-kn\end{smallmatrix}\right] \in \mathcal{S}$$

for $k = 1, 2, 3$. To proceed, we choose nonnegative integers $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n \in \{0, 1, 2, 3\}$ such that $a_1 + a_2 + \cdots + a_n = n$, $b_1 + b_2 + \cdots + b_n = n$ and $a_i b_i = 0$ for all i. Indeed this can be done as follows: If $n = 2m$ is even, then we can take $a_1 = a_2 = \cdots = a_m = b_{m+1} = b_{m+2} = \cdots = b_n = 2$ and $b_1 = b_2 = \cdots = b_m = a_{m+1} = a_{m+2} = \cdots = a_n = 0$. If $n = 2m + 1$ is odd, then we take $a_1 = 3$, $b_{m+1} = 1$, $a_2 = a_3 = \cdots = a_m = b_{m+2} = b_{m+3} = \cdots = b_n = 2$ and $b_1 = b_2 = \cdots = b_m = a_{m+1} = a_{m+2} = \cdots = a_n = 0$. Because $a_i b_i = 0$ we get that

$$\left[\begin{smallmatrix}n & n & 1\\ a_i & b_i & n^2-(a_i+b_i)n\end{smallmatrix}\right] \in \mathcal{S}$$

and then using (4) we get

$$\left[\begin{smallmatrix} n & n & 1 & & 1 \\ a_i & b_i & n & n^2-(a_i+b_i+1)n \end{smallmatrix}\right] \in \mathcal{S} \tag{5}$$

for $i = 1, 2, \ldots, n$. Applying the direct sum construction Lemma 25 to the elements of (5) we get

$$\left[\begin{smallmatrix} n & n & n & & 1 \\ n & n & n & n^3-3n^2 \end{smallmatrix}\right] = \left[\begin{smallmatrix} \sum_i^n a_i & \sum_i^n b_i & & 1+\cdots+1 & & \sum_i 1 \\ & & & n & & n^2-(a_i+b_i+1)n \end{smallmatrix}\right] \in \mathcal{S}.$$

Applying (4) repeatedly to this we obtain $\left[\begin{smallmatrix} n \\ n \end{smallmatrix}\right]^{\odot 3} \odot \left[\begin{smallmatrix} 1 \\ n^2 \end{smallmatrix}\right]^{\odot(n-3)} \in \mathcal{S}$.

We will now show by induction on d that

$$\left[\begin{smallmatrix} n \\ n^{d-2} \end{smallmatrix}\right]^{\odot d} \odot \left[\begin{smallmatrix} 1 \\ n^{d-1} \end{smallmatrix}\right]^{\odot(n-d)} \in \mathcal{S} \tag{6}$$

for $d = 3, 4, \ldots, n$. We already know that the base case $d = 3$ is true. Suppose for the induction step that $\left[\begin{smallmatrix} n \\ n^{d-2} \end{smallmatrix}\right]^{\odot d} \odot \left[\begin{smallmatrix} 1 \\ n^{d-1} \end{smallmatrix}\right]^{\odot(n-d)} \in \mathcal{S}$. Then we have using (4) that

$$\left[\begin{smallmatrix} n \\ n^{d-1} \end{smallmatrix}\right]^{\odot(d+1)} \odot \left[\begin{smallmatrix} 1 \\ n^{d} \end{smallmatrix}\right]^{\odot(n-d-1)} = \left[\begin{smallmatrix} n \\ n^{d-2}+\cdots+n^{d-2} \end{smallmatrix}\right]^{\odot d} \odot \left[\begin{smallmatrix} 1+1+\cdots+1 \\ n^{d-1} \end{smallmatrix}\right] \odot \left[\begin{smallmatrix} 1 \\ n^{d-1}+\cdots+n^{d-1} \end{smallmatrix}\right]^{\odot(n-d)-1} \in \mathcal{S}.$$

This proves (6).

Finally, by setting $d = n$ in (6) we obtain the claim $\left[\begin{smallmatrix} n \\ n^{n-2} \end{smallmatrix}\right]^{\odot n} \in \mathcal{S}$. ◀

5 Application: Subrank is not additive

We discuss in this section an application of our upper bound Theorem 2 on the generic subrank, namely that the subrank is not additive under the direct sum. That is, there are tensors S, T such that $\mathrm{Q}(T) + \mathrm{Q}(S) < \mathrm{Q}(T \oplus S)$. In fact, we obtain a large gap between $\mathrm{Q}(T) + \mathrm{Q}(S)$ and $\mathrm{Q}(T \oplus S)$. The proof relies on the idea of writing the diagonal tensor I_n as a sum of two generic tensors and on basic properties of the subrank.[11]

▶ **Theorem 28.** *There are tensors $S, T \in K^{n,n,n}$ such that we have $\mathrm{Q}(T), \mathrm{Q}(S) \leq \sqrt{3n-2}$ and $\mathrm{Q}(T \oplus S) \geq n$.*

Proof. By Theorem 2 there is a non-empty Zariski-open subset $U \subseteq K^{n,n,n}$ such that for all $T \in U$ we have $\mathrm{Q}(T) \leq \sqrt{3n-2}$. Recall that $I_n \in K^{n,n,n}$ is the tensor with ones in the diagonal entries and zeroes elsewhere. The subset $I_n - U \subseteq K^{n,n,n}$ is also non-empty and Zariski-open, and thus the intersection $U \cap (I_n - U)$ is non-empty. This means that there is a $T \in U$ such that $I_n - T$ is in U. Fix this T and let $S = I_n - T$. Then $\mathrm{Q}(T), \mathrm{Q}(S) \leq \sqrt{3n-2}$. For their direct sum, we observe the simple general fact that $T \oplus S \geq T + S$ where the left-hand side is the direct sum and the right-hand side is the coordinate-wise sum. Since subrank is monotone under $\geq$ we find that $\mathrm{Q}(T \oplus S) \geq \mathrm{Q}(T + S) = \mathrm{Q}(I_n) = n$. ◀

The proof of Theorem 28 extends directly so that similarly from Theorem 5 we get the following higher-order non-additivity result.

▶ **Theorem 29.** *There are k-tensors $S, T \in K^{n,\ldots,n}$ such that $\mathrm{Q}(T), \mathrm{Q}(S) \leq (kn-(k-1))^{\frac{1}{k-1}}$ while $\mathrm{Q}(T \oplus S) \geq n$.*

[11] This idea was used earlier in [9] to show that the "lower support functionals" [41] are not additive. Their results can be used to find a weaker non-additivity for subrank (with a smaller gap, and only for tensors of order three), namely that there are tensors $S, T \in K^{n,n,n}$ such that $\mathrm{Q}(T), \mathrm{Q}(S) \leq n^{2/3+o(1)}$ and $\mathrm{Q}(T \oplus S) \geq n$.

6 Open problems

There are several natural open problems that arise from or are closely related to our study and results on the generic subrank in this paper. We briefly list some of these problems here.

- While we determine the generic subrank very precisely up to a small additive constant, it is natural to ask whether our upper bound on the generic subrank $\mathrm{Q}(n)$ is exactly tight. We know that this is the case for all small cases ($n \leq 100$).
- Closely related to the above, but for higher-order and unbalanced formats, what is the exact value of the generic subrank $\mathrm{Q}(n_1, \ldots, n_k)$ in $K^{n_1,\ldots,n_k}$ when the dimensions n_i are not all equal? In particular, is our upper bound tight?
- To attack the previous two problems the natural approach would be to use the stronger Theorem 21. Our current lower bound on the generic subrank uses Theorem 13 which relies on our constructions of decompositions $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] = K^{r,r,r}$. Theorem 21 suggests to instead construct decompositions $\mathcal{X}_1[1] + \mathcal{X}_2[2] + \mathcal{X}_3[3] + W_r = K^{r,r,r}$. What are the "best" constructions that can be obtained of this form?
- There is a natural approximative version of subrank called *border* subrank[12] which plays a central role in algebraic complexity theory (in particular the study of matrix multiplication algorithms). The border subrank is at least the subrank. What value does the border subrank take on generic tensors?
- What is the value of the generic asymptotic subrank $\underset{\sim}{\mathrm{Q}}(n)$? The state of the art is that $n^{2/3} \leq \underset{\sim}{\mathrm{Q}}(n) \leq n$, so in particular it remains open whether generic tensors have "full" generic asymptotic subrank or not. We note that we do know of explicit tensors for which the asymptotic subrank is not full (the so-called W-tensor, for example) and also of explicit tensors for which the asymptotic subrank is full in a non-trivial way (matrix multiplication tensors, for example). The aforementioned lower bound of $n^{2/3}$ is by Strassen's elegant construction (which makes use of the matrix multiplication tensors). Our results show that one cannot obtain better lower bound on $\underset{\sim}{\mathrm{Q}}(n)$ by improving the lower bound on the generic subrank $\mathrm{Q}(n)$ since $\mathrm{Q}(n) = \theta(\sqrt{n})$.
- With the notation we introduced in Section 4 on constructions of tensor space decompositions, it is natural to ask what precisely are the elements of $\mathcal{S}$. Many elements can be constructed from simple base cases and the direct sum construction. However, we do not know whether a finite number of generators in this sense suffices to generate all of $\mathcal{S}$.
- We found tensors S, T for which there is a large gap between $\mathrm{Q}(S \oplus T)$ and $\mathrm{Q}(S) + \mathrm{Q}(T)$. Is this the largest possible gap? More speculatively, we may ask: is there a general relation between direct sum problems (i.e. additivity under direct sum) and parameter values at generic instances?

References

1 Josh Alman. Limits on the universal method for matrix multiplication. In *34th Computational Complexity Conference (CCC 2019)*, pages 12:1–12:24, 2019. doi:10.4230/LIPIcs.CCC.2019.12.

2 Abhishek Bhowmick and Shachar Lovett. Bias vs structure of polynomials in large fields, and applications in effective algebraic geometry and coding theory, 2015. arXiv:1506.02047.

[12] The difference with subrank is that the restriction preorder is replaced by the *degeneration* preorder, which results in the border subrank $\underline{\mathrm{Q}}(T)$ of T being the largest number r such that I_r is in the $\mathrm{GL}_n^{\times 3}$-orbit closure of T.

3 Abhishek Bhrushundi, Prahladh Harsha, Pooya Hatami, Swastik Kopparty, and Mrinal Kumar. On multilinear forms: Bias, correlation, and tensor rank. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, pages 29:1–29:23, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.29.

4 Markus Bläser. *Fast Matrix Multiplication.* Number 5 in Graduate Surveys. Theory of Computing Library, 2013. doi:10.4086/toc.gs.2013.005.

5 Markus Bläser. Explicit tensors. In *Perspectives in computational complexity*, pages 117–130. Springer, 2014.

6 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.*, pages Paper No. 3, 27, 2017. doi:10.19086/da.1245.

7 Armand Borel. *Linear algebraic groups*, volume 126 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1991. doi:10.1007/978-1-4612-0941-6.

8 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.

9 Peter Bürgisser. *Degenerationsordnung und Trägerfunktional bilinearer Abbildungen.* PhD thesis, Universität Konstanz, 1990. URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-opus-20311.

10 Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Universal points in the asymptotic spectrum of tensors (extended abstract). In *Proceedings of the 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC 2018)*, 2018. doi:10.1145/3188745.3188766.

11 Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Barriers for fast matrix multiplication from irreversibility. In *34th Computational Complexity Conference (CCC 2019)*, pages 26:1–26:17, 2019. doi:10.4230/LIPIcs.CCC.2019.26.

12 Alex Cohen and Guy Moshkovitz. Structure vs. randomness for bilinear maps. In *53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, 2021. doi:10.1145/3406325.3451007.

13 Alex Cohen and Guy Moshkovitz. Partition rank and analytic rank are uniformly equivalent, 2021. arXiv:2102.10509.

14 Harm Derksen. The G-stable rank for tensors, 2020. arXiv:2002.08435.

15 Klim Efremenko, Ankit Garg, Rafael Mendes de Oliveira, and Avi Wigderson. Barriers for rank methods in arithmetic complexity. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, pages 1:1–1:19, 2018. doi:10.4230/LIPIcs.ITCS.2018.1.

16 Jordan S. Ellenberg and Dion Gijswijt. On large subsets of $\mathbf{F}_q^n$ with no three-term arithmetic progression. *Ann. of Math. (2)*, 185(1):339–343, 2017. doi:10.4007/annals.2017.185.1.8.

17 Ankit Garg, Visu Makam, Rafael Mendes de Oliveira, and Avi Wigderson. More barriers for rank methods, via a "numeric to symbolic" transfer. In *60th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2019)*, pages 824–844, 2019. doi:10.1109/FOCS.2019.00054.

18 Runshi Geng and J. M. Landsberg. On the geometry of geometric rank, 2021. arXiv:2012.04679.

19 W. T. Gowers. The slice rank of a direct sum, 2021. arXiv:2105.08394.

20 W. T. Gowers and J. Wolf. Linear forms and higher-degree uniformity for functions on $\mathbf{F}_p^n$. *Geom. Funct. Anal.*, 21(1):36–69, 2011. doi:10.1007/s00039-010-0106-3.

21 Ben Joseph Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the gowers norms. *Contributions Discret. Math.*, 4(2), 2009. URL: http://cdm.ucalgary.ca/cdm/index.php/cdm/article/view/133.

22 Oliver Janzer. Low analytic rank implies low partition rank for tensors, 2018. arXiv:1809.10931.

23 Oliver Janzer. Polynomial bound for the partition rank vs the analytic rank of tensors. *Discrete Anal.*, 2020. arXiv:1902.11207.

24 Tali Kaufman and Shachar Lovett. Worst case to average case reductions for polynomials. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 166–175, 2008. doi:10.1109/FOCS.2008.17.

25 Pascal Koiran. On tensor rank and commuting matrices, 2020. arXiv:2006.02374.

26 Swastik Kopparty, Guy Moshkovitz, and Jeroen Zuiddam. Geometric rank of tensors and subrank of matrix multiplication. In *35th Computational Complexity Conference (CCC 2020)*, 2020. doi:10.4230/LIPIcs.CCC.2020.35.

27 J. M. Landsberg and Giorgio Ottaviani. Equations for secant varieties of Veronese and other varieties. *Ann. Mat. Pura Appl. (4)*, 192(4):569–606, 2013. doi:10.1007/s10231-011-0238-6.

28 Shachar Lovett. The analytic rank of tensors and its applications. *Discrete Anal.*, 2019. doi:10.19086/da.8654.

29 Luka Milićević. Polynomial bound for partition rank in terms of analytic rank. *Geom. Funct. Anal.*, 29(5):1503–1530, 2019. doi:10.1007/s00039-019-00505-4.

30 Eric Naslund. The partition rank of a tensor and k-right corners in $\mathbb{F}_q^n$. *Journal of Combinatorial Theory, Series A*, 174:105190, 2020. doi:10.1016/j.jcta.2019.105190.

31 Eric Naslund and Will Sawin. Upper bounds for sunflower-free sets. *Forum Math. Sigma*, 5:Paper No. e15, 10, 2017. doi:10.1017/fms.2017.12.

32 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *Journal of the ACM (JACM)*, 60(6):1–15, 2013.

33 Alexander A. Razborov. On submodular complexity measures. In *Poceedings of the London Mathematical Society symposium on Boolean function complexity*, pages 76–83, 1992. URL: https://dl.acm.org/doi/10.5555/167687.167709.

34 Robert Robere and Jeroen Zuiddam. Amortized circuit complexity, formal complexity measures, and catalytic algorithms. *Electron. Colloquium Comput. Complex.*, page 35, 2021. URL: https://eccc.weizmann.ac.il/report/2021/035.

35 Arnold Schönhage. Partial and total matrix multiplication. *SIAM J. Comput.*, 10(3):434–455, 1981. doi:10.1137/0210032.

36 Yaroslav Shitov. Counterexamples to Strassen's direct sum conjecture. *Acta Math.*, 222(2):363–379, 2019. doi:10.4310/ACTA.2019.v222.n2.a3.

37 Volker Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.*, 264:184–202, 1973.

38 Volker Strassen. Rank and optimal computation of generic tensors. *Linear Algebra Appl.*, 52/53:645–685, 1983. doi:10.1016/0024-3795(83)80041-X.

39 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.

40 Volker Strassen. The asymptotic spectrum of tensors. *J. Reine Angew. Math.*, 384:102–152, 1988. doi:10.1515/crll.1988.384.102.

41 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.

42 Terence Tao. A symmetric formulation of the Croot-Lev-Pach-Ellenberg-Gijswijt capset bound. URL: https://terrytao.wordpress.com, 2016.

43 Avi Wigderson and Jeroen Zuiddam. Asymptotic spectra: Theory, applications and extensions, 2021. URL: https://staff.fnwi.uva.nl/j.zuiddam/papers/convexity.pdf.

44 Jeroen Zuiddam. *Asymptotic spectra, algebraic complexity and moment polytopes*. PhD thesis, University of Amsterdam, 2018.

New Near-Linear Time Decodable Codes Closer to the GV Bound

Guy Blanc
Computer Science Department, Stanford University, CA, USA

Dean Doron
Department of Computer Science, Ben Gurion University of the Negev, Beer Sheva, Israel

Abstract

We construct a family of binary codes of relative distance $\frac{1}{2} - \varepsilon$ and rate

$$\varepsilon^2 \cdot 2^{-\log^{\alpha}(1/\varepsilon)}$$

for $\alpha \approx \frac{1}{2}$ that are decodable, probabilistically, in near-linear time. This improves upon the rate of the state-of-the-art near-linear time decoding near the GV bound due to Jeronimo, Srivastava, and Tulsiani, who gave a randomized decoding of Ta-Shma codes with $\alpha \approx \frac{5}{6}$ [34, 20]. Each code in our family can be constructed in probabilistic polynomial time, or deterministic polynomial time given sufficiently good explicit 3-uniform hypergraphs.

Our construction is based on a new graph-based bias amplification method. While previous works start with some base code of relative distance $\frac{1}{2} - \varepsilon_0$ for $\varepsilon_0 \gg \varepsilon$ and amplify the distance to $\frac{1}{2} - \varepsilon$ by walking on an expander, or on a carefully tailored product of expanders, we walk over very sparse, highly mixing, hypergraphs. Study of such hypergraphs further offers an avenue toward achieving rate $\widetilde{\Omega}(\varepsilon^2)$. For our unique- and list-decoding algorithms, we employ the framework developed in [20].

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

Keywords and phrases Unique decoding, list decoding, the Gilbert–Varshamov bound, small-bias sample spaces, hypergraphs, expander walks

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.10

Funding *Guy Blanc*: Supported by NSF CAREER Award 1942123.
Dean Doron: Part of this work was done while at Stanford, supported by NSF Award CCF-1763311.

Acknowledgements We are grateful to Victor Lecomte and Omer Reingold for stimulating discussions and collaboration in the early stages of the project. We also thank Ori Parzanchevski, Madhur Tulsiani, and Mary Wootters and for interesting and useful discussions.

1 Introduction

The Gilbert–Varshamov (GV) bound, for binary codes, tells us that there exist codes, even linear ones, with relative distance $\frac{1-\varepsilon}{2}$ and rate $\Omega(\varepsilon^2)$ [12, 36]. Namely, there exist codes $\mathcal{C} \subseteq \mathbb{F}_2^n$ such that for any two distinct codewords $x, y \in \mathcal{C}$ it holds that $\Delta(x, y) \geq \frac{1-\varepsilon}{2}$, for Δ being the normalized Hamming distance, such that $\frac{\log|\mathcal{C}|}{n} = \Omega(\varepsilon^2)$. Finding such small-redundancy codes, hopefully accompanied by an efficient decoding algorithm, has been subject to extensive and fruitful research in the past decades (see, e.g., [30, 2, 3, 5, 15, 34]). In a breakthrough result, Ta-Shma [34] constructed explicit linear codes of relative distance $\frac{1-\varepsilon}{2}$ having rate $\varepsilon^{2+o(1)}$. Ta-Shma's codes are also ε-*balanced*, i.e., $\Delta(x, y) \in \left[\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right]$, and thus give rise to explicit ε-biased sample spaces, which are ubiquitous in pseudorandomness and derandomization.

No decoding algorithm was given in [34], and this was later ameliorated by Jeronimo, Quintana, Srivastava, and Tulsiani [19, 20], who showed that a slight variant of Ta-Shma's codes are indeed efficiently decodable, and even in time $\widetilde{O}_\varepsilon(n)$.

▶ **Theorem 1** ([34, 20]). *There exists an explicit family of ε-balanced binary linear codes $\mathcal{C}_{\mathsf{TS}} \subseteq \mathbb{F}_2^n$ of rate*

$$r_{\mathsf{TSD}} = \varepsilon^2 \cdot 2^{-O\left(\log(1/\varepsilon)^{5/6}\right)},$$

such that:

1. *There exists a randomized algorithm that uniquely decode $\mathcal{C}_{\mathsf{TS}}$ up to half the distance in time $c_1(\varepsilon) \cdot \widetilde{O}(n)$. That is, given a noisy word $\tilde{z} \in \mathbb{F}_2^n$, the algorithm returns, with high probability, the unique $z \in \mathcal{C}$ such that $\Delta(z, \tilde{z}) \leq \frac{1-\varepsilon}{4}$ (if such exists).*
2. *There exists a randomized algorithm that list-decodes $\mathcal{C}_{\mathsf{TS}}$ up to radius*

$$\rho_{\mathsf{TSD}} = \frac{1}{2} - 2^{-O\left((\log(1/\varepsilon))^{1/6}\right)}$$

in time $c_2(\varepsilon) \cdot \widetilde{O}(n)$. That is, given a noisy word $\tilde{z} \in \mathbb{F}_2^n$, the algorithm returns, with high probability, a list $\mathcal{L} = \{z \in \mathcal{C} : \Delta(\tilde{z}, z) \leq \rho\}$ of size $|\mathcal{L}| = O(1/\varepsilon)$.[1]

We note that without any guarantee on the decoding capabilities, the codes in [34] achieve a better rate of

$$r_{\mathsf{TS}} = \varepsilon^2 \cdot 2^{-\widetilde{O}\left(\log(1/\varepsilon)^{2/3}\right)}.$$

Randomized constructions of binary codes, namely, randomized algorithms that output a good code with high probability, are also well-studied, where the goal is to achieve enough structure to allow for efficient decoding. If we focus on decoding in time $n^{1+o(1)}$, the current state-of-the-art is due to Hemenway, Wootters, and Ron-Zewi, that reaches the GV bound with a randomized construction.[2]

▶ **Theorem 2** ([17]). *There exists a family of ε-balanced binary codes $\mathcal{C}_{\mathsf{HRW}} \subseteq \mathbb{F}_2^n$ of rate $\Omega(\varepsilon^2)$ that can be constructed in probabilistic polynomial time, such that:*

1. *There exists a randomized algorithm that uniquely decodes $\mathcal{C}_{\mathsf{HRW}}$ up to half the distance in time $c_3(\varepsilon) \cdot n^{1+1/t}$, where $t \approx \log\log\log n$.*
2. *There exists a randomized algorithm that list-decodes*[3] *$\mathcal{C}_{\mathsf{HRW}}$ up to radius $\frac{1}{2} - O(\sqrt{\varepsilon})$ in time $c_3(\varepsilon) \cdot n^{1+1/t}$.*

In this work, we continue the study of near-linear time decodable binary codes near the GV bound, and give a randomized construction with improved rate.

▶ **Theorem 3** (see also Theorems 31 and 36). *There exists a family of ε-balanced binary codes $\mathcal{C} \subseteq \mathbb{F}_2^n$ that can be constructed in probabilistic polynomial time, of rate*

$$r = \varepsilon^2 \cdot 2^{-\widetilde{O}\left(\sqrt{\log(1/\varepsilon)}\right)},$$

such that:

---

[1] The guarantee on the list size is not a unique property of $\mathcal{C}_{\mathsf{TS}}$, but follows from the Johnson bound (see, e.g., [16, Section 7.3]), observing that $\rho_{\mathsf{TSD}} \leq \frac{1}{2} - \sqrt{\varepsilon}$.

[2] The foregoing theorem appears in the arXiv version, and some of the parameters are only implicit there.

[3] The randomized list decoding algorithm of [17] was later derandomized in [21].

1. *There exists a randomized algorithm that uniquely decodes $\mathcal{C}$ up to half the distance in time $c_1(\varepsilon) \cdot \widetilde{O}(n)$.*
2. *There exists a randomized algorithm that list-decodes $\mathcal{C}$ up to radius*

$$\rho = \frac{1}{2} - 2^{-O(\sqrt{\log(1/\varepsilon)})}$$

in time $c_2(\varepsilon) \cdot \widetilde{O}(n)$.

Thus, our codes achieve a better rate (and a better list decoding radius) than in [34, 20], while maintaining the $\widetilde{O}(n)$ runtime. Compared to state-of-the-art randomized constructions, we do not reach the GV bound, nor do we reach the Johnson radius for list decoding, but our decoding is faster, and as we shall soon see, our code is more structured. (The [17] result concatenate an outer code over a large alphabet with uniformly and independently chosen inner binary codes.)

In terms of the dependence on ε, for Theorems 1 and 3, $c_1(\varepsilon)$ is doubly-exponential in $\log^{\alpha}(1/\varepsilon)$ for some $\alpha < 1$ (that is slightly better in Theorem 3), and $c_2(\varepsilon)$ is triply-exponential in $\log^{\alpha}(1/\varepsilon)$. In Theorem 2, $c_3(\varepsilon)$ is triply-exponential in $\mathrm{poly}(1/\varepsilon)$.[4]

Our construction, which we shall soon describe, is arguably *simpler* than the constructions of Theorems 1 and 2.[5] Moreover, it gives an avenue toward achieving an even better rate of $\widetilde{\Omega}(\varepsilon^2)$ if we assume the existence of better primitives. In slightly more details, our construction utilizes hypergraphs with a strong mixing property, dubbed λ-*mixing*, and we show that a random 3-regular hypergraph achieves a good enough λ. A better dependence between λ and the regularity of the hypergraph readily gives better rate (for the details, see Section 6). We thereby put forward a challenge that warrants revisiting mixing properties of 3-uniform hypergraphs, which is interesting in itself.

1.1 Our construction

Our construction goes via distance amplification. We start with some base code $\mathcal{C}_0 \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ and construct our code $\mathcal{C} \colon \mathbb{F}_2^k \to \mathbb{F}_2^{\bar{n}}$ so that for every coordinate $i \in [\bar{n}]$ and $x \in \mathbb{F}_2^k$, $\mathcal{C}(x)_i$ is a function of the bits $\mathcal{C}_0(x)_{\Gamma(i)}$, where $\Gamma(i) \subseteq [n]$ is a small, carefully chosen, subset of the coordinates of $\mathcal{C}_0$. When we take the aforementioned function to be the parity function, i.e.,

$$\mathcal{C}(x)_i = \bigoplus_{j \in \Gamma(i)} \mathcal{C}_0(x)_j,$$

the code $\mathcal{C}$ is called the direct sum lift of $\mathcal{C}_0$ w.r.t. Γ. The goal is thus to start with $\mathcal{C}_0$ that is ε_0-balanced and argue that the lifted code $\mathcal{C}$ is ε-balanced, for $\varepsilon \ll \varepsilon_0$. A good Γ, that would fulfil this goal, is dubbed a *parity sampler*. See Section 2.1 for a slightly more general definition. Also, see [30, 2, 8, 34] for previous works that utilize direct sum lifting for distance amplification.

Our Γ, roughly speaking, consists of short walks over a hypergraph over n vertices. Toward giving a more detailed overview, let us define the desired hypergraphs more formally.

[4] More accurately, it is also doubly-exponential in $\log(1/\varepsilon) \cdot t$. The original analysis of [17] implies a quadruple-exponential dependence on $\mathrm{poly}(1/\varepsilon)$, but a better bound on the output list size of random list decodable codes, given in [24], can be used to reduce it to triply-exponential. Using the concatenation scheme of [17] with a different outer code given in [22] may be used to reduce the dependence on ε but at a cost of making the dependence on n worse. Finally, we note that the failure probability in Theorem 2 is sub-exponentially small, whereas the failure probability in Theorems 1 and 3 is exponentially small.

[5] By this we refer to our probabilistic construction, in which we draw a favorable hypergraph H at random. Admittedly, making our construction deterministic by constructing an explicit family of good H-s is likely to make it less simple.

Mixing hypergraphs

Let $H = (V, E)$ be a d-regular 3-uniform hypergraph over $V = [n]$. That is, E contains "hyperedges" of the form (w_1, w_2, w_3), and for each $v \in V$ and $j \in [3]$ we have that $v = w_j$ for exactly d hyperedges. We say H is λ-*mixing* if for any $S_1, S_2, S_3 \subseteq V$, it holds that

$$\left| \frac{E(S_1, S_2, S_3)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2} \right| \leq \lambda \cdot \sqrt{|S_1| \cdot |S_3|},$$

where $E(S_1, S_2, S_3)$ is the number of hyperedges $(w_1, w_2, w_3) \in E$ where $w_j \in S_j$ for $j = 1, 2, 3$. This notion and some variants of it were studied before, and we refer to Section 3 for the relevant discussion. In this work we show that a random hypergraph is $\lambda = O(1/\sqrt{d})$-mixing (see Corollary 21), but unfortunately we are not aware of any explicit construction that achieves such a good dependence on d.

Walks on hypergraphs

Set $t = t(\varepsilon)$ be a desired walk length. Starting from a random $\boldsymbol{v}_0 \sim V$, we walk on H according to uniformly random $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_t \sim [d]$ as follows. For each $j \in [t]$,

1. Let $\boldsymbol{e}_j$ be the $\boldsymbol{i}_j$-th hyperedge that touches $\boldsymbol{v}_{j-1}$ according to some fixed ordering. In particular, we require that $\boldsymbol{v}_{j-1} = (\boldsymbol{e}_j)_1$.
2. Denote $\boldsymbol{v}_j = (\boldsymbol{e}_j)_3$.
3. Denote $\boldsymbol{w}_j = (\boldsymbol{e}_j)_2$.

Γ comprises all the walks $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$. Note that we choose to query $\boldsymbol{w}_j$, but use $\boldsymbol{v}_j$ to determine the next step of our walk. Our lifted $\mathcal{C} \subseteq \mathbb{F}_2^{\bar{n}}$ will therefore have blocklength $\bar{n} = n \cdot d^t$.[6] Choosing parameters appropriately and using a λ-mixing H that satisfies $\lambda = O(1/\sqrt{d})$, we achieve the rate $\frac{k}{\bar{n}}$ that is given in Theorem 3.

In Section 1.2 below we briefly discuss how we analyze these walks, and how we are able to improve upon previous constructions that are also based on random walk over expanders.

Non-backtracking walks on λ-spectral hypergraphs

It turns out that we can get an even better rate, of $\widetilde{\Omega}(\varepsilon^2)$, by walking over hypergraphs with an even better dependence on d. Toward this end, we need a strengthening of our λ-mixing property, which we call λ-spectral. We say that H is λ-spectral if for any $x, y, z \in \mathbb{R}^n$, it holds that

$$\left| \frac{1}{d} \cdot \sum_{(i,j,k) \in E} x_i y_j z_k - \frac{1}{n^2} \cdot \sum_{i \in V} x_i \cdot \sum_{i \in V} y_i \cdot \sum_{i \in V} z_j \right| \leq \lambda \cdot \|x\|_2 \cdot \|y\|_\infty \cdot \|z\|_2.$$

In Section 3, we show that a λ-spectral hypergraph is readily a λ-mixing one, and that a λ-mixing hypergraph is λ'-spectral for $\lambda' = O(\lambda \log(1/\lambda))$.

Conjecturing the existence of λ-spectral hypergraphs with λ approaching $2/\sqrt{d}$ (see Open Problem 22), we can slightly modify the above construction to yield a rate of $\widetilde{\Omega}(\varepsilon^2)$, bringing us astonishingly close to the GV bound.

[6] We defer subtleties regarding sampling a single walk multiple times to the technical sections.

▶ **Theorem 4** (informal; see Corollary 39). *Assuming the existence of explicit λ-spectral hypergraphs with λ approaching $\frac{2}{\sqrt{d}}$, there exists an explicit family of ε-balanced codes $\mathcal{C} \subseteq \mathbb{F}_2^n$ of rate*

$$\varepsilon^2 \cdot \frac{1}{\mathrm{poly}(\log(1/\varepsilon))}$$

that are list- and uniquely-decodable in (probabilistic) near-linear time. The list decoding radius is $\frac{1}{2} - \frac{1}{\mathrm{poly}(\log(1/\varepsilon))}$.

For our modified construction, we replace the above random walks over H with *non-backtracking* walks, thus not "wasting" any randomness on returning steps. Analyzing the refined construction naturally requires working with non-symmetric operators, and in Section 6 we extend upon spectral decomposition results of Lubetzky and Peres [25]. We remark that we are not aware of many cases in which *directed* spectral graph theory is used in TCS, and our work demonstrates such an application.[7]

Explicitness

An ε-biased sample space over $\{0,1\}^k$ is a set $S \subseteq \{0,1\}^k$ such that for any nonzero test $\alpha \subseteq [k]$, it holds that

$$\left| \Pr_{\boldsymbol{s} \sim S}\left[\bigoplus_{i \in \alpha} \boldsymbol{s}_i = 0\right] - \Pr_{\boldsymbol{s} \sim S}\left[\bigoplus_{i \in \alpha} \boldsymbol{s}_i = 1\right] \right| \leq \varepsilon.$$

It is well-known that linear ε-balanced codes are equivalent to ε-biased sample space, by letting the elements of S correspond to rows in the $n \times k$ generator matrix of a binary code $\mathcal{C}$. Thus, an *explicit* ε-balanced code $\mathcal{C} \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ gives rise to an explicit ε-biased sample space $S \subseteq \{0,1\}^k$ of cardinality n. In our construction, the only non-explicit ingredient is the λ-mixing, or λ-spectral, hypergraph. Thus, coming up with such explicit hypergraphs would readily yield explicit (or even fully-explicit) small-biased spaces with better dependence on ε than the ones implied by Theorem 1.[8]

1.2 On (re)breaking the rate-$\Omega(\varepsilon^4)$ barrier of random walks

Recall that our construction uses a parity sampler Γ to amplify the distance of a base code that is ε_0-balanced to a one that is ε-balanced. Toward that goal, we will require that for a worst-case $S \subseteq [n]$ satisfying $|\mathbb{E}_{\boldsymbol{i} \in [n]}[(-1)^{\mathbb{1}[\boldsymbol{i} \in S]}| \leq \varepsilon_0$, that

$$\left| \mathop{\mathbb{E}}_{\boldsymbol{w} \in \Gamma}\left[\prod_{j=1}^{|\boldsymbol{w}|} (-1)^{\mathbb{1}[\boldsymbol{w}_j \in S]}\right] \right| \leq \varepsilon. \tag{1}$$

There is a simple, albeit inefficient, way to construct such a parity sampler: Take Γ to include all elements of $[n]^t$. This ensures $\varepsilon = \varepsilon_0^t$, however, then $|\Gamma| \triangleq \bar{n} = n^t$, which is obviously too large and would lead to a code with a vanishing rate. Thus, we seek a sparsification of the trivial parity sampler.

---

[7] One might wonder about using the non-backtracking walk with a $(\lambda = O(\log d/\sqrt{d}))$-spectral hypergraph, which is what we prove a random hypergraph satisfies. However, this does not lead to any meaningful improvement in parameters compared to the standard backtracking walk unless λ is close to $2/\sqrt{d}$.

[8] For ε-biased sample spaces we don't need to take a base code $\mathcal{C}_0$ that is efficiently encodable. Thus, given explicit good hypergraph and a suitable $\mathcal{C}_0$ (say, from [30]), we would be able to construct our ε-biased sample spaces in time polynomial in n and $1/\varepsilon$ for any $\varepsilon > 0$.

Random walks on graphs and the ε^4-barrier

Building off of ideas of Rozenman and Wigderson, Ta-Shma [34] suggested replacing the above fully independent construction with a random walk of length t on an n-vertex expander, associating each vertex of the expander with an element of $[n]$. If the graph used, G, is d-regular, then this construction leads to $\bar{n} = nd^{t-1}$, a substantial improvement from $|\Gamma| = n^t$.

We overload G to represent the graph's normalized adjacency matrix and let Π be the diagonal matrix in which $\Pi_{i,i} = (-1)^{\mathbb{1}[i \in S]}$. One can verify that if Γ consists of all length-t walks on G, Equation (1) is satisfied for

$$\varepsilon = \left| \frac{1}{n} \mathbf{1}^\dagger \left(\Pi G\right)^t \Pi \mathbf{1} \right| \leq \left\| \left(\Pi G\right)^t \right\|_{\mathrm{op}},$$

where $\mathbf{1}$ is the all-ones vector and $\|\cdot\|_{\mathrm{op}}$ is the operator norm $\|A\|_{\mathrm{op}} = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2$. As a first attempt, we could try to bound $\left\| \left(\Pi G\right)^t \right\|_{\mathrm{op}} \leq \|\Pi G\|_{\mathrm{op}}^t$. When a vector v is perpendicular to $\mathbf{1}$, we have that $\|\Pi G v\|_2 \leq \|Gv\|_2 \leq \lambda \|v\|_2$. Unfortunately, when a vector v is parallel to $\mathbf{1}$, we have that $\|\Pi G v\|_2 = \|Gv\|_2 = \|v\|_2$ because $G\mathbf{1} = \mathbf{1}$, meaning that $\|\Pi G\|_{\mathrm{op}} = 1$.

Ta-Shma observed that in the latter case, the *second* step works in our favor. This is because $\Pi\mathbf{1}$ is "mostly" (depending on how small ε_0 is) perpendicular to $\mathbf{1}$. In particular, he showed that $\|\Pi G \Pi G \mathbf{1}\|_2 \leq (\lambda + \varepsilon_0)\|\mathbf{1}\|_2$. Intuitively, at least one out of every two steps "works",[9] which is sufficient to guarantee a rate of $\approx \varepsilon^4$ by taking a good enough G. That is still far from the GV bound of $\approx \varepsilon^2$.

Breaking the ε^4-barrier

To break the barrier, Ta-Shma uses an intricately-designed random walk on a graph product called the *s-wide replacement product* (introduced in [4]), to guarantee that $s - O(1)$ out of every s steps work, for some $s < t$. Here, we diverge from Ta-Shma's approach. We will only aim for one out of every two steps to work, but will share randomness between the two steps in order to make them as cheap as a single step.

Specifically, let G_1 and G_2 be two degree-d expanders on the same n vertices. In order to take two coupled steps from a vertex v_1, we draw a random $\boldsymbol{j} \in [d]$ and move to v_2, the $\boldsymbol{j}^{\text{th}}$ neighbor of v_1 in G_1 (according to some fixed ordering). Then, we move to v_3, the $\boldsymbol{j}^{\text{th}}$ neighbor of v_2 in G_2. As we use the same label $\boldsymbol{j}$ for both steps, this walk can take ℓ "double steps" with a support size of only nd^ℓ. In contrast, if the steps were chosen independently, the support size would be $nd^{2\ell}$. If we could guarantee that a double step is as productive as two independent steps, the rate of the resulting code would be $\varepsilon^{2+o(1)}$.

For the double step to work, clearly there must be some relation between the two expanders. Otherwise, G_2 could always reverse the step taken by G_1. Hence, we would like to think of G_1 and G_2 together as a single primitive: For each vertex v_1, there are d choices for the pair (v_2, v_3). As a result, we can think of G_1 and G_2 together as a single d-regular 3-uniform hypergraph, and consider walks on that hypergraph, $H = (V, E_H)$, motivating the construction in Section 1.1.

[9] That phenomenon, of losing one λ factor in every two steps, is not a mere artifact of the proof, at least if one makes not further assumptions on the construction's primitives. See [4, 34] for relevant discussions.

To analyze this walk, we introduce an operator, $A^{(S)} \in \mathbb{R}^{V \times V}$. For each $i,k \in [n]$, we set

$$A^{(S)}_{i,k} \triangleq \frac{1}{d} \cdot \sum_{j:(i,j,k)\in E_H} (-1)^{\mathbb{1}[j\in S]}.$$

Then, for Γ corresponding to the length-t "double-step" construction, we show Equation (1) holds for

$$\varepsilon = \left|\frac{1}{n}\mathbf{1}^\dagger \left(\Pi A^{(S)}\right)^t \mathbf{1}\right| \leq \left\|\left(\Pi A^{(S)}\right)^t\right\|_{\mathrm{op}}.$$

If H is λ-spectral, it is simple to bound $\|\Pi A^{(S)}\|_{\mathrm{op}} = \|A^{(S)}\|_{\mathrm{op}} \leq \lambda + \varepsilon_0$ (see Proposition 28), which gives a bound of $\varepsilon = (\lambda+\varepsilon_0)^t$ and is sufficient for rate $\approx \varepsilon^2$.

Lastly, we note that because Π is unitary, the entire analysis goes through if instead of bounding $\|(\Pi A^{(S)})^t\|_{\mathrm{op}}$, we instead bound $\|(A^{(S)})^t\|_{\mathrm{op}}$. This corresponds to the a double-step construction where we only record every other vertex visited (starting with the second), which is exactly our construction in Section 1.1.

Bounding $A^{(S)}$, given the right notion of hypergraph expansion, is easier than analyzing Ta-Shma's s-wide replacement product, so we think of our construction as conceptually simpler. For our approach, the challenge is to construct sufficiently good hypergraphs. We are not aware of any explicit constructions, but are able to show that a random hypergraph suffices for decoding our code and obtaining Theorem 3.

1.3 Decoding our codes

Our decoding result in Theorem 3 follows the framework of Jeronimo et al. [20]. They used a novel algorithmic weak regularity lemma to show that direct sum lifts are decodable, roughly speaking, given that the parity sampler Γ used for the lifting satisfies the *splittability* condition (we refer the reader to [20] for the precise definition). While we suspect that our Γ is *not* splittable, we distill a weaker property that suffices for the [20] framework to work.

This property, which we call τ-sampling, tells us that we can use $\Gamma \subseteq [n]^t$ to sample any set $S \subseteq [n]$, starting from any prefix. Namely, for every $i \in [t]$ and $X \subseteq [n]^{i-1}$, we require that

$$\left|\Pr_{\boldsymbol{w}\in\Gamma}\left[\boldsymbol{w}_i \in S \mid (\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{i-1}) \in X\right] - \rho(S)\right| \leq \frac{\tau}{\rho(X)},$$

where $\rho(A)$, for some subset $A \subseteq [m]$, is its density $\frac{|A|}{m}$. For the more general definition, and further discussion, see Section 5.2. We believe that this strong mixing property, which still falls short of full-fledged splittability, is an interesting notion in itself. In Section 5.2, we show that our Γ is indeed τ-sampling, thereby allowing us to unique- and list-decode our code $\mathcal{C}$ in $\widetilde{O}_\varepsilon(n)$ time.

2 Preliminaries

For integers a,b, we use $[a,b]$ to denote the set $\{a,\ldots,b\}$ and $[n]$ as a shorthand for $[1,n]$. Given a set $S \subseteq [n]$, when the ground set $[n]$ is clear from context, we denote $\rho(S) = \frac{|S|}{n}$, and its ± 1 variant as $\mathrm{bias}(S) = 1 - 2\rho(S)$. For $z \in \mathbb{F}_2^n$, we similarly denote $\mathrm{bias}(z)$ as the bias of its characteristic set, i.e., $\mathbb{E}_{i\in[n]}[(-1)^{z_i}]$. We use boldface letters to denote random variables, except for $\mathbf{1} \in \mathbb{R}^n$, which we use for the all-ones vector. Also, when bounding running time, by writing $g(n) = \exp(f(n))$ we mean that $g(n) \leq 2^{c\cdot f(n)}$ for some universal constant $c > 0$.

▶ **Definition 5** (discretizable distribution). *For $M \in \mathbb{N}$, We say that a distribution $\mathcal{W}$ is M-discretizable if it satisfies either of the following two equivalent properties.*

1. *For any x in the support of $\mathcal{W}$, $\Pr_{\boldsymbol{x}\sim\mathcal{W}}[\boldsymbol{x} = x] = i/M$ for some $i \in \mathbb{N}$.*
2. *Let $\mathcal{U}_M$ be the uniform distribution over $[M]$. Then, there is some function f mapping $[M]$ to the support of $\mathcal{W}$ for which $f(\boldsymbol{i})$, where $\boldsymbol{i} \sim \mathcal{U}_M$, has the same distribution as a sample from $\mathcal{W}$.*

We say that $\mathcal{W}$ is computable in (deterministic or probabilistic) time T if f above is computable in time T. In particular, $\mathcal{W}$ is explicit if it is computable in deterministic time $\mathrm{poly}(M)$*, and fully explicit if it is computable in deterministic time* $\mathrm{poly}(\log M)$*.*

▶ **Definition 6** (homogeneous distribution). *For $n, t \in \mathbb{N}$, we say that a distribution $\mathcal{W}$ over $[n]^t$ is* homogeneous *if its restriction to any coordinate is uniform over $[n]$, i.e., if for any $i \in [t]$ and $a \in [n]$ it holds that $\Pr_{\boldsymbol{w}\sim\mathcal{W}}[\boldsymbol{w}_i = a] = \frac{1}{n}$.*

For any domain $\mathcal{X}$ and two distributions $\mathcal{D}, \mathcal{D}'$ over $\mathcal{X}$ we define the *total variation distance* of $\mathcal{D}$ and $\mathcal{D}'$ in terms of the optimal test distinguishing the distributions, i.e.,

$$d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \triangleq \sup_{T\colon \mathcal{X}\to[0,1]} \left\{ \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}}[T(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}'}[T(\boldsymbol{x}')] \right\}.$$

For a matrix $A \in \mathbb{R}^{n\times n}$, we denote by $\|A\|_{\mathrm{op}}$ its operator norm $\|A\|_{\mathrm{op}} = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2}$, which is also the maximum of $x^\dagger A y$ over all norm-1 vectors $x, y \in \mathbb{R}^n$.

Error correcting codes

A binary error correcting code of message length k and blocklength n is a mapping $\mathcal{C}\colon \mathbb{F}_2^k \to \mathbb{F}_2^n$, which we will often identify with its image $\mathrm{Im}(\mathcal{C}) \subseteq \mathbb{F}_2^n$. The *rate* of $\mathcal{C}$ is $\frac{k}{n}$, and its *relative distance* is $\Delta(\mathcal{C}) = \frac{1}{n}\min_{z\neq z'} \Delta(z, z')$ for $z, z' \in \mathcal{C}$, and $\Delta(z, z') = |\{i \in [n] : z_i \neq z'_i\}|$ being the Hamming distance. The Hamming ball of (relative) radius β centered at z is the set $B(z, \beta) = \{z' \in \mathbb{F}_2^n : \Delta(z, z')/n \leq \beta\}$.

We denote $\mathrm{bias}(\mathcal{C})$ as the maximal bias, in absolute value, of every nonzero $z \in \mathcal{C}$. Thus, $\mathrm{bias}(\mathcal{C}) \leq \varepsilon$ if the Hamming weight of any nonzero codeword is in $\left[\frac{1-\varepsilon}{2}, \frac{1+\varepsilon}{2}\right]$.

▶ **Definition 7** (balanced codes). *A linear binary error correcting code is ε-balanced if* $\mathrm{bias}(\mathcal{C}) \leq \varepsilon$*.*

In particular, an ε-balanced code $\mathcal{C}$ has distance at least $\frac{1-\varepsilon}{2}$.

Unique and list decoding

We say that $\mathcal{C}$ is (combinatorially) (β, L) list decodable if for every $z \in \mathbb{F}_2^n$, $|\mathcal{C} \cap B(z, \beta)| \leq L$. The Johnson bound tells us that any ε-balanced code is $(1/2 - \sqrt{\varepsilon}, L)$ list decodable for $L = O(1/\varepsilon)$. The *algorithmic* list decoding problem aims at finding the list $\mathcal{L}_{\mathcal{C},\beta}(z) \triangleq \mathcal{C} \cap B(z, \beta)$. When $\beta \leq \frac{1-\varepsilon}{4}$ and $\mathcal{C}$ is ε-balanced, we know that $\mathcal{L}_{\mathcal{C},\beta}$ always contains at most one codeword, which corresponds to the unique decoding problem.

2.1 Parity samplers and direct sum codes

We will be interested in constructing a binary linear code $\mathcal{C} \subseteq \mathbb{F}_2^{\bar{n}}$ with small bias by amplifying the (moderate) bias of some base code $\mathcal{C}_0\colon \mathbb{F}_2^k \to \mathbb{F}_2^n$. One natural way to do so is by XORing t-tuples of $\mathcal{C}_0$ according to some distribution $\mathcal{W} \sim [n]^t$.

▶ **Definition 8** (direct sum codes). *For $t, n, \bar{n} \in \mathbb{N}$, let $\mathcal{W} \sim [n]^t$ be an $\bar{n}$-discretizable distribution equipped with a corresponding mapping function $f_{\mathcal{W}} \colon [\bar{n}] \to [n]^t$. For $z \in \mathbb{F}_2^n$, we let $\mathrm{dsum}_{\mathcal{W}}(z) \in \mathbb{F}_2^{\bar{n}}$ be such that*

$$\mathrm{dsum}_{\mathcal{W}}(z)[\ell] = \sum_{i=1}^{t} z\left[f_{\mathcal{W}}(\ell)_i\right]$$

where the addition is taken over $\mathbb{F}_2$. Given a code $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$, the direct sum lift of $\mathcal{C}_0$ according to $\mathcal{W}$ is the code $\mathrm{dsum}_{\mathcal{W}}(\mathcal{C}_0) = \{\mathrm{dsum}_{\mathcal{W}}(z) : z \in \mathcal{C}_0\} \subseteq \mathbb{F}_2^{\bar{n}}$.

▶ **Definition 9** (parity sampler). *For $t, n \in \mathbb{N}$, and $0 \leq \varepsilon < \varepsilon_0 \leq 1$, we say that $\mathcal{W} \sim [n]^t$ is an $(\varepsilon_0, \varepsilon)$ parity sampler if for every $z \in \mathbb{F}_2^n$ with $|\mathrm{bias}(z)| \leq \varepsilon_0$ it holds that $|\mathrm{bias}(\mathrm{dsum}_{\mathcal{W}}(z))| \leq \varepsilon$.*

Clearly, if $\mathcal{C}_0 \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ is ε_0-balanced and $\mathcal{W} \sim [n]^t$ is an $\bar{n}$-discretizable $(\varepsilon_0, \varepsilon)$ parity sampler, the lifted code $\mathcal{C} = \mathrm{dsum}_{\mathcal{W}}(\mathcal{C}_0)$ is ε-balanced with rate $\frac{k}{\bar{n}}$.

3 Expanding 3-Uniform Hypergraphs

Our construction uses a family of expanding d-regular 3-uniform hypergraphs.

▶ **Definition 10** (d-regular 3-uniform hypergraph). *A 3-uniform hypergraph consists of a set of vertices, V, and hyperedges $E \subseteq V^3$. The hypergraph $H = (V, E)$ is d-regular if, for each $v \in V$ and $j \in [3]$, the number of hyperedges $(w_1, w_2, w_3) \in E$ for which $v = w_j$ is d.*

We will set d carefully in our construction, but for now, it can be thought of as an arbitrary constant. For the remainder of this section, we will use "hypergraph" as shorthand for d-regular 3-uniform hypergraph.

There are various notions of expansion for hypergraphs and they are not all equivalent. In this work, we consider two such notions.

▶ **Definition 11** (λ-mixing hypergraph). *A d-regular hypergraph $H = (V, E)$ on n vertices is λ-mixing if, for any $S_1, S_2, S_3 \subseteq V$,*

$$\left| \frac{E(S_1, S_2, S_3)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2} \right| \leq \lambda \cdot \sqrt{|S_1| \cdot |S_3|}, \tag{2}$$

where $E(S_1, S_2, S_3)$ is the number of hyperedges $(w_1, w_2, w_3) \in E$ where $w_j \in S_j$ for $j = 1, 2, 3$.

Note that the right-hand side of the above definition does not depend on the size of S_2.[10] That parallels the definition below, in which we use $\|y\|_\infty$ instead of $\|y\|_2$.

▶ **Definition 12** (λ-spectral hypergraph). *A d-regular hypergraph $H = (V, E)$ on n vertices is λ-spectral if, for any $x, y, z \in \mathbb{R}^n$,*

$$\left| \frac{1}{d} \cdot \sum_{(i,j,k) \in E} x_i y_j z_k - \frac{1}{n^2} \cdot \sum_{i \in V} x_i \cdot \sum_{j \in V} y_j \cdot \sum_{k \in V} z_k \right| \leq \lambda \cdot \|x\|_2 \cdot \|y\|_\infty \cdot \|z\|_2. \tag{3}$$

[10] Some works consider the stronger requirement of $\sqrt{|S_{\sigma(1)}| \cdot |S_{\sigma(2)}|}$ instead of $\sqrt{|S_1| \cdot |S_3|}$, for $S_{\sigma(1)}$ and $S_{\sigma(2)}$ being the two smallest sets.

We will only care about cases where $y \in \{\pm 1\}^n$.[11] In those cases, we have $\|y\|_\infty = 1$ and $\|y\|_2 = \sqrt{n}$. It is thus tempting to replace the $\|y\|_\infty$ in the right-hand side of Equation (3) with $\frac{\|y\|_2}{\sqrt{n}}$, as ℓ_2 norms are often easier to work with than ℓ_∞ norms. Unfortunately, no "good" λ-spectral hypergraphs would exist with that modification: Whenever $n \gg d^2$, it is straightforward to bound $\lambda = \Omega(\sqrt{n}/d)$. For our definition, as we shall soon see, it is possible to achieve $\lambda \approx 1/\sqrt{d}$.

A variant of the spectral definition, where indeed one takes $\|y\|_2$ instead of $\|y\|_\infty$, was first studied by Friedman and Wigderson [11], who also showed that the spectral definition implies combinatorial mixing. They considered much larger edge densities than us (corresponding to $d > n$ for our definition). Hypergraphs with similar combinatorial mixing properties were previously constructed from random walks on expanders [6], from Ramanujan complexes and other spectral properties of simplicial complexes (e.g., [27, 23, 32, 9, 31, 14]), and from Cayley graphs [10]. However, to the best of our knowledge, no explicit construction achieves λ smaller than $\approx \frac{1}{d^{1/3}}$. A simple hypergraph construction would be to take all length-2 walks on a Ramanujan expander as the hyperedges. This was considered by [6], who showed it can achieve $\lambda \approx \frac{1}{d^{1/4}}$.[12]

Similar to standard graphs, spectral expansion implies mixing.

▶ **Proposition 13** (spectral $\implies$ mixing)**.** *For any $\lambda > 0$, if H is a λ-spectral hypergraph, it is also a λ-mixing hypergraph.*

Proof. For any $S_1, S_2, S_3 \subseteq V$, let $x_i = \mathbb{1}[i \in S_1]$, $y_i = \mathbb{1}[i \in S_2]$, and $z_i = \mathbb{1}[i \in S_3]$. Then,

$$E(S_1, S_2, S_3) = \sum_{(i,j,k)\in E} x_i y_j z_k.$$

Equation (2) follows directly from Equation (3). ◀

By applying the converse to the expander mixing lemma for ordinary graphs [7], we can show that for symmetric hypergraphs, mixing implies spectral expansion with only a minor quantitative gap.

▶ **Definition 14** (symmetric 3-uniform hypergraph)**.** *We say a 3-uniform hypergraph $H = (V, E)$ is* symmetric *if, for any edge $e = (v_1, v_2, v_3) \in E$, the edge (v_3, v_2, v_1) is also in E.*

▶ **Proposition 15** (mixing $\implies$ spectral)**.** *There exists a universal constant $c_{\mathsf{spec}} > 0$ for which the following holds. Let $H = (V = [n], E)$ be any d-regular hypergraph, and $\lambda \leq \frac{1}{2}$. If H is λ-mixing and symmetric, then H is a λ'-spectral for $\lambda' = c_{\mathsf{spec}} \cdot \lambda \log(1/\lambda)$.*

The proof of Proposition 15 is a simple application of a similar result for graphs.

▶ **Lemma 16** (Lemma 3.3 of [7])**.** *There exists a universal constant c_{BL} for which the following hods. For any $n \times n$ real symmetric matrix A and $\lambda \leq \frac{1}{2}$, suppose each row has an ℓ_1 norm of at most 1 and for any two vectors $u, v \in \{0,1\}^n$,*[13]

$$|u^\dagger A v| \leq \lambda \cdot \|u\| \cdot \|v\|. \tag{4}$$

Then, the spectral radius of A is at most $c_{\mathsf{BL}} \cdot \lambda \log(1/\lambda)$.

---

[11] In fact, for any fixed choice of x and z, the left-hand side of Equation (3) is linear in y. Therefore, it is maximized for some $y \in \{\pm 1\}^n$ and so in general it is sufficient to consider only such y.

[12] In [6], Bilu and Hoory used hypergraphs for the construction of asymptotically good codes, generalizing Tanner's expander codes [35] and their decoding [33, 37].

[13] Note that Bilu and Linial's Lemma statement only has the weaker requirement that this hold for orthogonal u and v. As a result, they have an additional condition that the diagonal entries of A not be too large, which is not needed for our version of the Lemma.

If G is the (normalized) transition matrix of a d-regular graph, and $A = G - \frac{1}{n}J$ for J being the all-ones matrix, then Lemma 16 shows a converse to the expander mixing lemma: Any graph with good mixing is also a good spectral expander. We show that their result can be lifted to 3-uniform hypergraphs.

Proof of Proposition 15. Fix any $y \in \mathbb{R}^n$ and let $A^{(y)} \in \mathbb{R}^{n \times n}$ be defined as

$$A^{(y)}_{i,k} \triangleq \frac{1}{2d} \cdot \left(\sum_{j \in [n]} \mathbb{1}[(i,j,k) \in E] \cdot y_j \right) - \frac{1}{2n^2} \cdot \sum_{j \in [n]} y_j.$$

Then, for any $x, z \in \mathbb{R}^n$,

$$2 \cdot (z^\dagger A^{(y)} x) = \frac{1}{d} \cdot \sum_{(i,j,k) \in E} x_i y_j z_k - \frac{1}{n^2} \cdot \sum_{i \in V} x_i \cdot \sum_{i \in V} y_i \cdot \sum_{i \in V} z_j.$$

Therefore, in order to prove that H is an λ'-spectral expander, it is sufficient to show that for all $y \in \mathbb{R}^n$, the operator norm of $A^{(y)}$ is at most $\|y\|_\infty \cdot \frac{\lambda'}{2}$. We observe that:

1. For any fixed $x, z \in \mathbb{R}^n$, the quantity $z^\dagger A^{(y)} x$ is a linear function of y. Thus, we can consider y-s with $\|y\|_\infty = 1$ without loss of generality. Furthermore, the y maximizing $z^\dagger A^{(y)} x$ among those with $\|y\|_\infty = 1$ will be in $\{\pm 1\}^n$. Therefore we assume $y \in \{\pm 1\}^n$ also without loss of generality.
2. Since H is symmetric, the operator norm and spectral radius of $A^{(y)}$ are equal, so we instead bound the spectral radius.

We will apply Lemma 16 to each $A^{(y)}$. Note that:

- $A^{(y)}$ is symmetric, which follows immediately from the fact that H is symmetric.
- The ℓ_1 norm of each row of $A^{(y)}$ is bounded by 1:

$$\sum_{k \in [n]} \left| \frac{1}{2d} \cdot \left(\sum_{j \in [n]} \mathbb{1}[(i,j,k) \in E] \cdot y_j \right) + \frac{1}{2n^2} \cdot \sum_{j \in [n]} y_j \right| \leq \frac{1}{2d} \cdot d + \frac{1}{2n^2} \cdot n \leq 1.$$

Finally, fix some $u, v \in \{0,1\}^n$, and define the sets

$$\begin{aligned} S_1 &= \{i \in V : u_i = 1\} \\ S_3 &= \{i \in V : v_i = 1\} \\ S_2^+ &= \{i \in V : y_i = 1\} \\ S_2^- &= \{i \in V : y_i = -1\}. \end{aligned}$$

Then,

$$\begin{aligned} \left| u^\dagger A^{(y)} v \right| &= \frac{1}{2} \cdot \left| \frac{1}{d} \cdot \left(E(S_1, S_2^+, S_3) - E(S_1, S_2^-, S_3) \right) - \frac{|S_1| \cdot |S_3| \cdot (|S_2^+| - |S_2^-|)}{n^2} \right| \\ &\leq \frac{1}{2} \cdot \left| \frac{1}{d} \cdot E(S_1, S_2^+, S_3) - \frac{|S_1| \cdot |S_3| \cdot |S_2^+|}{n^2} \right| \\ &\quad + \frac{1}{2} \cdot \left| \frac{1}{d} \cdot E(S_1, S_2^-, S_3) - \frac{|S_1| \cdot |S_3| \cdot |S_2^-|}{n^2} \right| \leq \lambda \sqrt{|S_1| \cdot |S_3|}, \end{aligned}$$

where the last inequality follows from fact that H is λ-mixing (applied to both expressions). Thus, $|u^\dagger A^{(y)} v| \leq \lambda \cdot \|u\| \cdot \|v\|$ and we can apply Lemma 16 to obtain $\|A^{(y)}\|_{\text{op}} = c_{\mathsf{BL}} \cdot \lambda \log(1/\lambda)$, implying that H is $(\lambda' = 2c_{\mathsf{BL}} \cdot \lambda \log(1/\lambda))$-spectral. ◀

3.1 Random hypergraphs mix well

In this section, we'll show that given an expanding graph G, its *random hypergraph completion* is, with high probability, a good expander. Throughout, we say that an undirected regular graph G is a λ-expander if the second largest eigenvalue of its normalized adjacency matrix, in magnitude, is at most λ.

▶ **Definition 17** (random hypergraph completion of a graph.)**.** *Let $G = (V = [n], E_G)$ be a d-regular graph. To sample a random hypergraph completion, $\boldsymbol{H}$ of G, we choose a uniformly random ordering of G's edges, $\{(\boldsymbol{u}_1, \boldsymbol{v}_1), \ldots, (\boldsymbol{u}_{nd}, \boldsymbol{v}_{nd})\} = E_G$. Then, we set $\boldsymbol{H} = (V, \boldsymbol{E_H})$, where*

$$\boldsymbol{E_H} \triangleq \{(\boldsymbol{u}_i, \lceil i/d \rceil, \boldsymbol{v}_i) \mid i \in [nd]\}.$$

Equivalently, for each $(u, v) \in E_G$, we choose an independent and uniform $\boldsymbol{w} \in V$ and add the hyperedge $(u, \boldsymbol{w}, v)$ to E_H, conditioned on the resulting hypergraph $\boldsymbol{H}$ being d-regular.

Next, we prove that the random hypergraph completion mixes well with high probability.

▶ **Lemma 18.** *Let $G = (V = [n], E)$ be a d-regular λ-expander and $\boldsymbol{H}$ be a random hypergraph completion of G. With probability at least $1 - 2^{-n}$, $\boldsymbol{H}$ is a λ'-mixing hypergraph for $\lambda' = 2\lambda + \frac{2}{\sqrt{d}}$.*

In order to prove Lemma 18, we'll use Hoeffding's inequality for sampling *without* replacements.

▶ **Fact 19** (Hoeffding's inequality, [18])**.** *For any integers $a, k \leq m$, suppose there are m items, of which k of them are marked. Let $\boldsymbol{x}$ be a random variable indicating the number of marked items when a of the m items are sampled uniformly and independently* without replacement*. Then, for any $t \geq 0$,*

$$\Pr\left[\left|\boldsymbol{x} - \frac{k}{m} \cdot a\right| \geq t\right] \leq 2\exp\left(\frac{-2t^2}{a}\right).$$

Proof of Lemma 18. Fix arbitrary $S_1, S_2, S_3 \subseteq V$. We will show that with probability at least $1 - 2^{-4n}$ it holds that

$$\left|\frac{E_{\boldsymbol{H}}(S_1, S_2, S_3)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| \leq \left(\frac{2}{\sqrt{d}} + 2\lambda\right) \cdot \sqrt{|S_1| \cdot |S_3|} \tag{5}$$

where $E_{\boldsymbol{H}}(S_1, S_2, S_3)$ is the number of edges (v_1, v_2, v_3) in $\boldsymbol{H}$ where $v_j \in S_j$ for each $j \in [3]$. The desired result then follows from a union bound over the $(2^n)^3 = 2^{3n}$ choices for S_1, S_2, S_3. Let $E_G(S_1, S_3)$ be the number of edges, (u, v), of G, such that $u \in S_1$ and $v \in S_3$. By the expander mixing lemma applied to G, we have that

$$\left|\frac{E_G(S_1, S_3)}{d} - \frac{|S_1||S_3|}{n}\right| \leq \lambda\sqrt{|S_1||S_3|}.$$

Let us define $\mu \triangleq \frac{|S_1||S_3|}{n}$ and $\Delta \triangleq \lambda\sqrt{|S_1||S_3|}$. We consider two cases.

1. In the first case, $\mu \leq \Delta$. Here, we use the simple bound

$$0 \leq \frac{E_{\boldsymbol{H}}(S_1, S_2, S_3)}{d} \leq \frac{E_G(S_1, S_3)}{d} \leq 2\Delta$$

that holds with probability 1. This, along with the fact $\frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2} \leq \mu \leq \Delta$ implies that Equation (5) always holds.

2. In the second case, $\mu > \Delta$. Here, we will apply Fact 19. G has a total of nd edges, of which $d \cdot |S_2|$ are matched to a vertex in S_2. $E_{\boldsymbol{H}}(S_1, S_2, S_3)$ samples $E_G(S_1, S_3)$ of those nd edges (without replacement) and counts how many were among the $d \cdot |S_2|$ assigned to S_2. Hence, by Hoeffding's inequality, we have for any $t \geq 0$,

$$\Pr\left[\left|E_{\boldsymbol{H}}(S_1, S_2, S_3) - \frac{d \cdot |S_2|}{nd} \cdot E_G(S_1, S_3)\right| \geq t\right] \leq 2\exp\left(\frac{-2t^2}{E_G(S_1, S_3)}\right).$$

Then, setting $t = 2\sqrt{d|S_1| \cdot |S_3|}$ and using the fact that $E_G(S_1, S_3) \leq d(\mu + \Delta) \leq 2d\mu$,

$$\Pr\left[\left|E_{\boldsymbol{H}}(S_1, S_2, S_3) - \frac{|S_2|}{n} \cdot E_G(S_1, S_3)\right| \geq 2\sqrt{d|S_1| \cdot |S_3|}\right] \leq$$
$$2\exp\left(\frac{-8d|S_1| \cdot |S_3|}{2d \cdot \frac{|S_1||S_3|}{n}}\right) = 2\exp(-4.5n) \leq 2^{-4n}.$$

As the above shows, $E_{\boldsymbol{H}}(S_1, S_2, S_3)$ is within $\pm t$ of its expectation with probability at least 2^{-4n}. When that occurs, we have that

$$\begin{aligned}
\left|\frac{E_{\boldsymbol{H}}(S_1, S_2, S_3)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| &= \frac{1}{d}\left|E_{\boldsymbol{H}}(S_1, S_2, S_3) - \frac{|S_2|d\mu}{n}\right| \\
&\leq \frac{t}{d} + \frac{|S_2|}{n} \cdot \left|\frac{E_G(S_1, S_3)}{d} - \mu\right| \\
&\leq \frac{t}{d} + \frac{|S_2|}{n} \cdot \Delta \qquad \text{(expander mixing lemma)} \\
&\leq 2\sqrt{\frac{|S_1| \cdot |S_3|}{d}} + \lambda\sqrt{|S_1| \cdot |S_3|}. \qquad (|S_2|/n \leq 1)
\end{aligned}$$

Hence, Equation (5) holds with probability at least $1 - 2^{-4n}$ in both cases, so we can union bound over the 2^{3n} choices for S_1, S_2, S_3. ◀

For our purposes, we will want the hypergraph to be symmetric. It is quite easy to "symmetrize" any hypergraph at only a modest cost to the degree.

▶ **Proposition 20.** *For any $n, d \in \mathbb{N}$, there is an algorithm running in time $O(nd)$ that takes as input any d-regular λ-mixing hypergraph $H = (V = [n], E_H)$ and outputs a $2d$-regular λ-mixing hypergraph $H' = (V = [n], E_{H'})$ over the same vertices.*

Proof. For every edge $(u, v, w) \in E_H$ we include both (u, v, w) and the reverse edge (w, v, u) in $E_{H'}$.[14] Clearly, this results in the degree of H' being $2d$. Then, for any $S_1, S_2, S_3 \subseteq V$,

$$\begin{aligned}
&\left|\frac{E_{H'}(S_1, S_2, S_3)}{2d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| \\
&= \left|\frac{E_H(S_1, S_2, S_3) + E_H(S_3, S_2, S_1)}{2d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| \\
&\leq \frac{1}{2}\left|\frac{E_H(S_1, S_2, S_3)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| + \frac{1}{2}\left|\frac{E_H(S_3, S_2, S_1)}{d} - \frac{|S_1| \cdot |S_2| \cdot |S_3|}{n^2}\right| \\
&\leq \lambda\sqrt{|S_1| \cdot |S_3|}.
\end{aligned}$$

Therefore, H' is λ-mixing, as desired. ◀

[14] Note that, we do this even if it results in duplicated hyperedges: If (u, v, w) and (w, v, u) are both in E_H, then there will be two copies of (u, v, w) and two copies of (w, v, u) in $E_{H'}$.

We have explicit (and even fully explicit) constructions of Ramanujan graphs, i.e., λ-expanders for $\lambda \leq \frac{2\sqrt{d-1}}{d}$, albeit with some restrictions on d [26, 28]. By manipulating Ramanujan graphs, Alon gave a construction of d-regular λ-expanders over n vertices for any d and n while suffering only a tiny loss in λ (see [1], and also [29, 13] for weaker constructions). In particular, there exist explicit expanders with $\lambda = O(1/\sqrt{d})$ for all n-s. We thus get the following corollary.

▶ **Corollary 21.** *There exists a probabilistic algorithm such that for any integer n and even integer $6 \leq d \leq n$, runs in time $\mathrm{poly}(n)$ and with probability at least $1 - 2^{-n}$ outputs a 3-uniform symmetric d-regular hypergraph that is $\lambda = \frac{c_{\mathsf{rand}}}{\sqrt{d}}$-mixing, where $c_{\mathsf{rand}} \geq 2$ is some universal constant.*

We will refer to the above probabilistic construction as our *preprocessing step*.

Unfortunately, we do not know how to construct *explicit* mixing, or spectral, hypergraphs with $\lambda \approx 2/\sqrt{d}$. We put forward a concrete goal of constructing "nearly Ramanujan" spectral hypergraphs.

▶ **Open Problem 22.** *Construct a sufficiently dense infinite family of explicit 3-uniform d-regular hypergraphs which are λ-spectral for $\lambda \leq \frac{2}{\sqrt{d}} \cdot (1 + d^c)$, where $c < 0$ is any absolute constant.*

Getting such hypergraphs is an interesting goal on its own right (and in particular, it is not clear if one can get them from good high-dimensional complexes). As we will later see, fulfilling Open Problem 22 would readily give *explicit* ε-balanced codes with rate $\widetilde{\Omega}(\varepsilon^2)$ and efficient decoding, and moreover, by the known connection to small-biased distributions, also an explicit ε-biased distributions over $\mathbb{F}_2^n$ with support size $n \cdot \widetilde{O}(\varepsilon^{-2})$. See Section 6 for the details.

4 From Hypergraphs to Parity Samplers

Fix some $n, t, d \in \mathbb{N}$ and a d-regular 3-uniform hypergraph $H = (V, E_H)$ over n vertices. We will construct a parity sampler $\mathcal{W}_{H,t} \sim [n]^t$ from H and show that when H is a good spectral expander, $\mathcal{W}$ is a good parity sampler.

The construction

For each $v \in V$, there are d edges $(v_1, v_2, v_3) \in E_H$ with $v_1 = v$. For any $i \in [d]$, let $e_H(v, i)$ be the i^{th} such edge (i.e., $e_H(v, i)_1 = v$). Without loss of generality, we assume the vertices are each labeled with a unique integer between 1 and n (i.e $V = [n]$).

To sample $\boldsymbol{w} \sim \mathcal{W}_{H,t}$, independently sample a starting vertex $\boldsymbol{v}_0 \in [n]$ and edge labels $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_t \sim [d]$ uniformly. $\boldsymbol{w}$ will be a deterministic function of $\boldsymbol{v}_0$ and $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_t$ computed as follows. For each $j = 1, \ldots, t$,

1. Let $\boldsymbol{e}_j = e_H(\boldsymbol{v}_{j-1}, \boldsymbol{i}_j)$.
2. Let $\boldsymbol{v}_j = (\boldsymbol{e}_j)_3$.
3. Let $\boldsymbol{w}_j = (\boldsymbol{e}_j)_2$.

The sample is then $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$.

We present two simple claims about $\mathcal{W}_{H,t}$.

▷ Claim 23. For any $t \in \mathbb{N}$ and a d-regular hypergraph H over n vertices, $\mathcal{W}_{H,t}$ is (nd^t)-discretizable.

Proof. The sample $\boldsymbol{w} \sim \mathcal{W}_{H,t}$ is a deterministic function of $\boldsymbol{v}_0$ and $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_t$, and those variables are set to a uniform choice out of nd^t possibilities. The claim then follows from Definition 5. ◁

▷ Claim 24. For any $t \in \mathbb{N}$, d-regular hypergraph H, and $j \in [t]$, $\boldsymbol{e}_j$ is uniform over all nd edges of H. As a result, $\mathcal{W}_{H,t}$ is homogeneous.

Proof. We will first prove that each $\boldsymbol{e}_j$ is uniform over E_H by induction on j. $\boldsymbol{e}_1$ is uniform over the nd edges in E_H as it is sampled by independently choosing a starting vertex $\boldsymbol{v}_0 \sim V$ uniformly and then uniformly choosing one of its d-neighbors. Furthermore, for any $j \in [t-1]$, if $\boldsymbol{e}_j$ is uniform, then $\boldsymbol{v}_j$ is also uniform. Hence, $\boldsymbol{e}_{j+1}$ is sampled by selecting a uniform vertex and then (independently) one of its d-neighbors, so $\boldsymbol{e}_{j+1}$ is also uniform over E_H.

Next, we show $\mathcal{W}_{H,t}$ is homogeneous. Fix any $a \in [n]$ and $j \in [t]$. As $\boldsymbol{w}_j = a$ if and only if $\boldsymbol{e}_j$ is one of the d-edges in E_H whose second vertex is a and $\boldsymbol{e}_j$ is uniform over the nd edges in E_H, $\Pr[\boldsymbol{w}_j = a] = \frac{d}{nd} = \frac{1}{n}$. ◁

Finally, we show that whenever H is a good expander, $\mathcal{W}_{H,t}$ is a good parity sampler.

▶ **Theorem 25.** *For any $n, d, t \in \mathbb{N}$, $\lambda, \varepsilon_0 > 0$, and H a λ-spectral d-regular 3-uniform hypergraph on n vertices, $\mathcal{W}_{H,t}$ is an $(\varepsilon_0, \varepsilon \triangleq (\varepsilon_0 + \lambda)^t)$-parity sampler.*

As an immediate corollary of Theorem 25 and Proposition 15:

▶ **Corollary 26.** *There exists an absolute constant $c_{\mathsf{spec}} > 0$ for which the following holds. For any $n, d, t \in \mathbb{N}$, $\lambda, \varepsilon_0 > 0$, and H a λ-mixing symmetric d-regular 3-uniform hypergraph on n vertices, $\mathcal{W}_{H,t}$ is an $(\varepsilon_0, \varepsilon \triangleq (\varepsilon_0 + c_{\mathsf{spec}} \cdot \lambda \log(1/\lambda))^t)$-parity sampler.*

Throughout the remainder of this section, we will use the shorthand $\mathcal{W} \triangleq \mathcal{W}_{H,t}$. For proving Theorem 25, it will be convenient to consider $\sigma \in \{\pm 1\}^n$ rather than $z \in \mathbb{F}_2^n$ (as in Definition 9) using the mapping $\sigma_i = (-1)^{z_i}$. Equivalent to Definition 9, $\mathcal{W}$ is an $(\varepsilon_0, \varepsilon)$ parity sampler if $|\mathrm{bias}_{\mathcal{W}}(\sigma)| \leq \varepsilon$ for all $\sigma \in \{\pm 1\}^n$ satisfying $|\mathrm{bias}(\sigma)| = \left|\mathbb{E}_{\boldsymbol{i} \sim [n]}[\sigma_{\boldsymbol{i}}]\right| \leq \varepsilon_0$, where

$$\mathrm{bias}_{\mathcal{W}}(\sigma) \triangleq \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{W}} \left[\prod_{j=1}^{t} \sigma_{\boldsymbol{w}_j} \right].$$

Similarly to [34], we express that bias algebraically.

▶ **Lemma 27.** *Let $A^{(\sigma)} \in \mathbb{R}^{n \times n}$ be defined as*

$$A^{(\sigma)}_{i,k} \triangleq \frac{1}{d} \cdot \sum_{(i',j',k') \in E_H} \sigma_{j'} \cdot \mathbb{1}[i' = i \wedge k' = k].$$

Then, $\mathrm{bias}_{\mathcal{W}}(\sigma) = \frac{1}{n} \cdot \mathbf{1}^\dagger \left(A^{(\sigma)}\right)^t \mathbf{1}$.

Proof. Let $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_t$ and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$ be the random variables defined in the construction of $\mathcal{W}$. We claim that for each $j \in \{0, 1, \ldots, t\}$ and $v \in [n]$, that

$$\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v] \prod_{k=1}^{j} \sigma_{\boldsymbol{w}_k}\right] = \frac{1}{n} \cdot \left(\mathbf{1}^\dagger \left(A^{(\sigma)}\right)^j\right)_v \tag{6}$$

By induction on j. Clearly, for $j = 0$, Equation (6) holds as both sides are equal to $\frac{1}{n}$ for any $v \in [n]$. For $j \geq 1$,

$$\begin{aligned}
\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v]\prod_{k=1}^{j}\sigma_{\boldsymbol{w}_k}\right] &= \sum_{v'\in[n]}\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v \wedge \boldsymbol{v}_{j-1} = v']\prod_{k=1}^{j}\sigma_{\boldsymbol{w}_k}\right] \\
&= \sum_{v'\in[n]}\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_{j-1} = v']\prod_{k=1}^{j-1}\sigma_{\boldsymbol{w}_k}\right]\cdot\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v]\cdot\sigma_{\boldsymbol{w}_j}\,\middle|\,\boldsymbol{v}_{j-1} = v'\right] \\
&= \sum_{v'\in[n]}\frac{1}{n}\cdot\left(\mathbf{1}^\dagger\left(A^{(\sigma)}\right)^{j-1}\right)_{v'}\cdot\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v]\cdot\sigma_{\boldsymbol{w}_j}\,\middle|\,\boldsymbol{v}_{j-1} = v'\right] \\
&= \frac{1}{n}\sum_{(a,b,c)\in E_H}\left(\mathbf{1}^\dagger\left(A^{(\sigma)}\right)^{j-1}\right)_a\cdot\frac{1}{d}\cdot\sigma_b\cdot\mathbb{1}[v = c] \\
&= \frac{1}{n}\cdot\left(\mathbf{1}^\dagger\left(A^{(\sigma)}\right)^{j}\right)_v,
\end{aligned}$$

where the third equality is the inductive hypothesis. Then,

$$\mathrm{bias}_{\mathcal{W}}(\sigma) = \mathop{\mathbb{E}}_{\boldsymbol{w}\sim\mathcal{W}}\left[\prod_{j=1}^{t}\sigma_j\right] = \sum_{v\in[n]}\mathbb{E}\left[\mathbb{1}[\boldsymbol{v}_j = v]\prod_{k=1}^{j}\sigma_{\boldsymbol{w}_k}\right] = \frac{1}{n}\cdot\mathbf{1}^\dagger\left(A^{(\sigma)}\right)^t\mathbf{1}. \qquad \blacktriangleleft$$

Next, we use the fact that H is a λ-spectral expander to reason about $A^{(\sigma)}$.

▶ **Proposition 28.** *Let $J_n \in \mathbb{R}^{n\times n}$ be the matrix in which every element is $1/n$. For any $\sigma \in \{\pm1\}^n$, $\left\|A^{(\sigma)} - \mathrm{bias}(\sigma)J_n\right\|_{\mathrm{op}} \leq \lambda$.*

Proof. Fix any $x, z \in \mathbb{R}^n$. Then,

$$\begin{aligned}
z^\dagger\left(A^{(\sigma)} - \mathrm{bias}(\sigma)J_n\right)x &= \frac{1}{d}\cdot\sum_{(i,j,k)\in E_H}x_k\sigma_j z_i - \frac{\mathrm{bias}(\sigma)}{n}\cdot\sum_{i\in[n]}x_i\sum_{j\in[n]}z_j \\
&= \frac{1}{d}\cdot\sum_{(i,j,k)\in E_H}x_i\sigma_j z_k - \frac{1}{n^2}\cdot\sum_{k\in[n]}\sigma_k\sum_{i\in[n]}x_i\sum_{j\in[n]}z_j \\
&\leq \lambda\|x\|_2\|z\|_2. \qquad \text{(Definition 12)}
\end{aligned}$$

$\blacktriangleleft$

As an immediate consequence, we have:

▶ **Corollary 29.** *For any $\sigma \in \{\pm1\}^n$, $\left\|A^{(\sigma)}\right\|_{\mathrm{op}} \leq |\mathrm{bias}(\sigma)| + \lambda$.*

Proof. As $\|J_n\|_{\mathrm{op}} = 1$, the desired result follows from the reversed triangle inequality applied to the operator norm. $\blacktriangleleft$

Finally, we prove Theorem 25.

Proof of Theorem 25. For any $\sigma \in \{\pm1\}^n$ satisfying $|\mathrm{bias}(\sigma)| \leq \varepsilon_0$, we have:

$$\begin{aligned}
|\mathrm{bias}_{\mathcal{W}}(\sigma)| &= \frac{\mathbf{1}^\dagger}{\sqrt{n}}\left(A^{(\sigma)}\right)^t\frac{\mathbf{1}}{\sqrt{n}} \qquad \text{(Lemma 27)} \\
&\leq \left\|\left(A^{(\sigma)}\right)^t\right\|_{\mathrm{op}} \\
&\leq \left\|A^{(\sigma)}\right\|_{\mathrm{op}}^t \leq (|\mathrm{bias}(\sigma)| + \lambda)^t \leq (\varepsilon_0 + \lambda)^t. \qquad \text{(Corollary 29)}
\end{aligned}$$

$\blacktriangleleft$

5 Codes Closer to the GV Bound

5.1 The construction

We use our parity sampler to amplify the distance of a given base code via direct sum lifting. More formally, given $k \in \mathbb{N}$ and $\varepsilon > 0$, let $\varepsilon_0 = \varepsilon_0(\varepsilon)$ soon to be determined, and let

- $\mathcal{C}_0 \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ be an ε_0-balanced code, and,
- $\mathcal{W} \sim [n]^t$ be an $(\varepsilon_0, \varepsilon)$-parity sampler which is M-discretizable.

Denote $\bar{n} = |\mathrm{Supp}(\mathcal{W})| \leq M$. Thus, the lifted code $\mathcal{C} = \mathrm{dsum}_{\mathcal{W}}(\mathcal{C}_0) \subseteq \mathbb{F}_2^{\bar{n}}$ is clearly ε-balanced. For the base code, we use the codes by Guruswami and Indyk, that admit $O_{\varepsilon_0}(n)$-time encoding and decoding.

▶ **Theorem 30** ([15]). *For every integer n and any $\varepsilon_0 > 0$ there exists an ε_0-balanced code $\mathcal{C}_0 \subseteq \mathbb{F}_2^n$ of rate $\Omega(\varepsilon_0^3)$. Furthermore, $\mathcal{C}_0$ is encodable in time $\exp(\mathrm{poly}(1/\varepsilon_0))n$ and decodable from $\frac{1}{4} - \varepsilon_0$ fraction of errors in time $\exp(\mathrm{poly}(1/\varepsilon_0))n$.*[15]

Setting parameters

Given $\varepsilon > 0$, we henceforth set:

1. The initial bias of $\mathcal{C}_0$ to $\varepsilon_0 = \frac{1}{2} \cdot 2^{-\sqrt{\log(1/\varepsilon)}}$.
2. The number of steps over H to $t = \left\lceil \sqrt{\log(1/\varepsilon)} \right\rceil$.
3. The degree d of H to be the smallest even integer for which

$$c_{\mathsf{spec}} \cdot c_{\mathsf{rand}} \cdot \frac{1}{\sqrt{d}} \log\left(\frac{\sqrt{d}}{c_{\mathsf{rand}}}\right) \leq \varepsilon_0.$$

This is chosen so that Corollary 21 gives, with high probability, a symmetric hypergraph H such that, by Corollary 26, $\mathcal{W}_{H,t}$ is a $(\varepsilon_0, (2\varepsilon_0)^t)$-parity sampler.

Using the above parameters in Theorem 25, together with the random hypergraph of Corollary 21. we get the following (randomized) error correcting code $\mathcal{C}$.

▶ **Theorem 31.** *There exists an efficient randomized algorithm such that for every k and any $\varepsilon > 0$, outputs with probability $1 - 2^{-\Omega(k)}$ an ε-balanced linear code $\mathcal{C} \colon \mathbb{F}_2^k \to \mathbb{F}_2^{\bar{n}}$ of rate*

$$\frac{k}{\bar{n}} = 2^{-c_{\mathsf{r}} \log\log(1/\varepsilon)\sqrt{\log(1/\varepsilon)}} \cdot \varepsilon^2$$

for some universal constant c_{r}, that is encodable in deterministic time $\exp(\exp(\sqrt{\log(1/\varepsilon)})) \cdot k$.[16]

More precisely, there exists a randomized preprocessing step that runs in time

$$\exp\left(\sqrt{\log(1/\varepsilon)}\right) \cdot k$$

and succeeds with probability $1 - 2^{-\Omega(k)}$. Once it succeeds, it fixes a deterministic *mapping $\mathcal{C}$ such that for every $x \in \mathbb{F}_2^k$, $\mathcal{C}(x)$ can be computed in deterministic in the above $O_\varepsilon(\bar{n})$ time.*

If, moreover, for a large enough constant d we are given an explicit family of λ-spectral d-regular 3-uniform hypergraphs satisfying $\lambda = \frac{\mathrm{poly}(\log d)}{\sqrt{d}}$, there is no need for a preprocessing step, and $\mathcal{C}$ is explicit.

[15] One can get a better *randomized* encoding time of $\mathrm{poly}(1/\varepsilon_0)$.

[16] Following the previous footnote, one can get a randomized encoding in time $\mathrm{poly}(1/\varepsilon) \cdot k$ by using a randomized encoding of the based code.

Proof. Given $k \in \mathbb{N}$ and $\varepsilon > 0$, we set ε_0, t, and d as above. Theorem 30 gives us a code $\mathcal{C}_0 \colon \mathbb{F}_2^k \to \mathbb{F}_2^n$ which is ε_0-balanced, for $n = O(k/\varepsilon_0^3)$. By Corollary 21 we can output a $\lambda = \frac{c_{\mathsf{rand}}}{\sqrt{d}}$-mixing H with probability at least $1 - 2^{-n}$ in polynomial time. This is our preprocessing step. By Corollary 26 and our choice of parameters, indeed

$$\left(\varepsilon_0 + c_{\mathsf{spec}} \cdot \lambda \log \frac{1}{\lambda}\right)^t \leq (2\varepsilon_0)^t \leq \varepsilon,$$

so $\mathcal{W} = \mathcal{W}_{H,t}$ appropriately amplifies the distance. Moreover, $\mathcal{W}$ is M-discretizable for

$$M = nd^t = O\left(\frac{k}{\varepsilon_0^3}\right) \cdot d^{\left\lceil \sqrt{\log(1/\varepsilon)} \right\rceil} = k \cdot 2^{(3+\log d)\sqrt{\log(1/\varepsilon)} + \log d + O(1)}.$$

We proceed to bounding $\log d$ in terms of ε. Recalling how we set d, we see that $\frac{c}{\sqrt{d}} \log d = 2^{-\sqrt{\log(1/\varepsilon)}}$, where c is some universal positive constant. From this we can infer that

$$d \leq c^2 2^{2\sqrt{\log(1/\varepsilon)}} \cdot \log^3\left(c \cdot 2^{\sqrt{\log(1/\varepsilon)}}\right), \tag{7}$$

so $\log d = 2\sqrt{\log(1/\varepsilon)} + O(\log\log(1/\varepsilon))$. (We are assuming that ε is smaller than some small constant, which we can without loss of generality.) Hence,

$$M = \frac{k}{\varepsilon^2} \cdot 2^{O\left(\sqrt{\log(1/\varepsilon)} \cdot \log\log(1/\varepsilon)\right)}.$$

Since $\mathcal{W}$ is M-discretizable, $\mathcal{C} = \mathrm{dsum}_{\mathcal{W}}(\mathcal{C}_0)$ has block length $\bar{n} \leq M$, as required.

Drawing H at random, by Corollary 21 takes

$$O(dn) = \widetilde{O}\left(\frac{1}{\varepsilon_0^3}\right) \cdot k = \exp(\sqrt{\log(1/\varepsilon)}) \cdot k$$

time. Given the hypergraph H, the encoding amounts to computing $\mathcal{C}_0(x)$ and taking parities of its coordinates according to $\mathcal{W}$. This takes $\exp(\mathrm{poly}(1/\varepsilon_0)) \cdot k + M \cdot t$ time, following Theorem 30.

For the "moreover" part, one can verify that we can tolerate a $\mathrm{polylog}(d)$ multiplicative factor in λ with no substantial loss in parameters. ◀

Note that an explicit construction of H would also give an explicit ε-biased sample space over $\mathbb{F}_2^k$ with support size $\bar{n}$, improving upon the state-of-the-art $\frac{k}{\bar{n}} = 2^{-\widetilde{O}(\log(1/\varepsilon))^{2/3}} \cdot \varepsilon^2$ by Ta-Shma [34].

5.2 τ-sampling

Toward establishing efficient decoding, we will need the notion of τ-sampling.

▶ **Definition 32** (τ-sampling). *We say that a distribution $\mathcal{W}$ over $[n]^t$ is* τ-sampling *if for any $i \in [t-1]$, $S \subseteq [n]$, and $X \subseteq [n]^i$,*

$$\operatorname*{Cov}_{\boldsymbol{w} \sim \mathcal{W}}\Big[\mathbb{1}[\boldsymbol{w}_{i+1} \in S], \mathbb{1}[(\boldsymbol{w}_1, \ldots \boldsymbol{w}_i) \in X]\Big] \leq \tau.$$

To highlight the fact that it is indeed a (strong) sampling property, assume for simplicity that $\mathcal{W}$ is homogeneous. Fix any $i \in [t-1]$, $S \subseteq [n]$, and $X \subseteq [n]^i$. The property of τ-sampling thus tells us we can use $\mathcal{W}$ to sample S starting from any prefix. Namely, that

$$\left| \Pr_{\boldsymbol{w} \in \mathcal{W}} [\boldsymbol{w}_{i+1} \in S \mid (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_i) \in X] - \rho(S) \right| \leq \frac{\tau}{\rho(X)}.$$

Note that the $t = 2$ case corresponds to the setting of the expander mixing lemma.

A τ-sampling distribution $\mathcal{W}$ satisfies the property of the "Splittable Mixing Lemma" of Jeronimo, Srivastava, and Tulsiani [20, Lemma 4.6] for the family of "± 1 cut functions". This fact is crucial us, and for completeness we establish this in Appendix A.

Given a τ-sampling homogeneous $\mathcal{W} \sim [n]^t$, a standard hybrid argument shows the following.

▶ **Lemma 33.** *Let $\mathcal{W} \sim [n]^t$ be homogeneous and τ-sampling, and let $z \in \mathbb{F}_2^n$ be arbitrary. Then,*

$$\left| \mathop{\mathbb{E}}_{\boldsymbol{w} \in \mathcal{W}} \left[(-1)^{z_{\boldsymbol{w}_1} + \cdots z_{\boldsymbol{w}_t}} \right] - \mathrm{bias}(z)^t \right| \leq 2(t-1)\tau.$$

Thus, high-order mixing in particular implies that $\mathcal{W}$ is an $(\varepsilon_0, \varepsilon = \varepsilon_0^t + (t-1)\tau)$ parity sampler for all ε_0. However, for us, and also in [20], $t \cdot \tau$ is too large so we prove the parity sampling property separately.

We conclude our discussion about τ-sampling by showing that our $\mathcal{W}$ is indeed τ-sampling.

▶ **Lemma 34.** *For any $n, d, t \in \mathbb{N}$ and $\lambda > 0$, if $H = (V = [n], E)$ is a λ-spectral d-regular 3-uniform hypergraph, then $\mathcal{W}_{H,t}$ is $\tau = \frac{\lambda}{4}$-sampling.*

Proof. As in Definition 32, fix any $i \in [t-1]$, $S \subseteq [n]$ and $X \subseteq [n]^i$. Let $S_i(\boldsymbol{w})$ be the indicator that $\boldsymbol{w}_{i+1} \in S$, and $X_i(w)$ the indicator that $(w_1, \ldots, w_t) \in X$. Using the identity $\mathrm{Cov}[1 - 2\boldsymbol{a}, 1 - 2\boldsymbol{b}] = 4\,\mathrm{Cov}[\boldsymbol{a}, \boldsymbol{b}]$, it is sufficient to prove that

$$\mathop{\mathrm{Cov}}_{\boldsymbol{w} \sim \mathcal{W}_{H,t}} \left[(-1)^{S_i(\boldsymbol{w})}, (-1)^{X_i(\boldsymbol{w})} \right] \leq \lambda.$$

Let $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_t$, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$, and $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t$ be the random variables defined in the construction of $\mathcal{W}_{H,t}$. Furthermore, let $x, y, z \in \mathbb{R}^n$ be the vectors defined, for each $j \in [n]$, by

$$\begin{aligned} x_j &\triangleq \mathbb{E}\left[(-1)^{X_i(\boldsymbol{w})} \,\middle|\, \boldsymbol{v}_i = j \right], \\ y_j &\triangleq (-1)^{\mathbb{1}[j \in S]}, \\ z_j &\triangleq \frac{1}{n}. \end{aligned}$$

Note that $\|z\|_2 = 1/\sqrt{n}$, $\|y\|_\infty = 1$, and $\|x\|_2 \leq \sqrt{n}$ (which follows from $\|x\|_\infty \leq 1$). Our goal in this proof will be to show that the following equation holds:

$$\mathop{\mathrm{Cov}}_{\boldsymbol{w}} \left[(-1)^{S_i(\boldsymbol{w})}, (-1)^{X_i(\boldsymbol{w})} \right] = \frac{1}{d} \cdot \sum_{(a,b,c) \in E} x_a y_b z_c - \frac{1}{n^2} \cdot \sum_{a \in [n]} x_a \cdot \sum_{a \in [n]} y_a \cdot \sum_{a \in [n]} z_a \qquad (8)$$

Once we do, the desired result follows from Definition 12. In order to compute the covariance, we first expand:

$$\begin{aligned}\mathop{\mathbb{E}}_{\boldsymbol{w}}\left[(-1)^{S_i(\boldsymbol{w})}\cdot(-1)^{X_i(\boldsymbol{w})}\right] &= \frac{1}{nd}\cdot\sum_{e\in E}\mathop{\mathbb{E}}_{\boldsymbol{w}}\left[(-1)^{S_i(\boldsymbol{w})}\cdot(-1)^{X_i(\boldsymbol{w})}\;\middle|\;\boldsymbol{e}_{i+1}=e\right] && \text{(Claim 24)}\\ &= \frac{1}{nd}\cdot\sum_{(a,b,c)\in E}(-1)^{\mathbb{1}[b\in S]}\cdot\mathop{\mathbb{E}}_{\boldsymbol{w}}\left[(-1)^{X_i(\boldsymbol{w})}\,\middle|\,\boldsymbol{v}_i=a\right]\\ &= \frac{1}{d}\cdot\sum_{(a,b,c)\in E}x_a y_b z_c. && (9)\end{aligned}$$

Next, directly from the definition of y and Claim 24,

$$\mathop{\mathbb{E}}_{\boldsymbol{w}}\left[(-1)^{S_i(\boldsymbol{w})}\right]=\frac{1}{n}\sum_{a\in[n]}y_a. \tag{10}$$

Similarly,

$$\mathop{\mathbb{E}}_{\boldsymbol{w}}\left[(-1)^{X_i(\boldsymbol{w})}\right]=\frac{1}{n}\sum_{a\in[n]}x_a. \tag{11}$$

Equation (8) follows from Equations (9)–(11) and the fact that $\sum_{a\in[n]}z_a=1$. The desired result then follows Definition 12. ◀

5.3 Decoding $\mathcal{C}$

We follow the work of Jernoimo et al. who gave a near-linear time list- and unique-decoding algorithm for Ta-Shma's code via an efficient weak regularity lemma, and prove that their algorithm also applies to our code as well. In the language of τ-sampling distributions, they prove:[17]

▶ **Theorem 35** ([20]). *There exists a constant c_{JST} such that the following holds for any integers d,t,k,n and any $\tau,\varepsilon_0,\varepsilon>0$. Let $\mathcal{C}_0\colon\mathbb{F}_2^k\to\mathbb{F}_2^n$ be a code with bias at most ε_0 which is uniquely decodable to within distance $\frac{1-\varepsilon_0}{4}$ in time $T_0=T_0(n,\varepsilon_0)$. Let $\mathcal{W}\sim[n]^t$ be a homogeneous τ-sampling distribution, let $\mathcal{C}=\mathrm{dsum}_{\mathcal{W}}(\mathcal{C}_0)$ be the corresponding direct sum lifting, and assume that the bias of $\mathcal{C}$ is at most ε. Let β be such that*

$$\beta\geq\max\left\{\sqrt{\varepsilon},\sqrt{c_{\mathsf{JST}}\cdot t^3\tau},2\cdot\left(\frac{1}{2}+2\varepsilon_0\right)^t\right\}.$$

Then, there exists a randomized algorithm, which given $\tilde{y}\in\mathbb{F}_2^{|\mathcal{W}|}$, recovers the list $\mathcal{C}\cap B(\tilde{y},\frac{1}{2}-\beta)$ with probability at least $1-\frac{1}{\varepsilon}\cdot2^{-\Omega(\varepsilon_0^2 n)}$ in time $\widetilde{O}(c_{\beta,t,\varepsilon_0}\cdot(|\mathcal{W}|+T_0))$, for $c_{\beta,t,\varepsilon_0}=(6/\varepsilon_0)^{2^{O(t^3/\beta^2)}}$. Moreover, if we are able to set $\beta=\frac{1+\varepsilon}{4}$, we can uniquely decode $\mathcal{C}$ to within distance $\frac{1-\varepsilon}{4}$ with probability at least $1-2^{-\Omega(\varepsilon_0^2 n)}$.

Plugging-in our code $\mathcal{C}$ (using $\mathcal{W}_{H,t}$ and $\mathcal{C}_0$ as described in Section 5.1), we get our main result.

[17] Jernoimo et al. prove their result under stronger requirements, however one can verify that the mixing requirement suffices.

▶ **Theorem 36.** *Given $k \in \mathbb{N}$ and $\varepsilon > 0$, let $\mathcal{C} \colon \mathbb{F}_2^k \to \mathbb{F}_2^{\bar{n}}$ be the ε-balanced, linear-time encodable code guaranteed to us by Theorem 31 with high probability. Then, $\mathcal{C}$ also admits the following decoding capabilities.*

1. *$\mathcal{C}$ is list decodable up to radius $\frac{1}{2} - \beta$ for $\beta = 2^{-\frac{1}{2}\sqrt{\log(1/\varepsilon)}}$ by a randomized algorithm that runs in time $c_1(\varepsilon) \cdot \widetilde{O}(k)$ and succeeds with probability $1 - 2^{-\Omega(k)}$.*
2. *$\mathcal{C}$ is uniquely decodable to within distance $\frac{1-\varepsilon}{4}$ by a randomized algorithm that runs in time $c_2(\varepsilon) \cdot \widetilde{O}(k)$ and succeeds with probability $1 - 2^{-\Omega(k)}$.*

Above, $c_1(\varepsilon) = \exp(\exp(\exp(\sqrt{\log(1/\varepsilon)})))$ and $c_2(\varepsilon) = \exp(\exp(\sqrt{\log(1/\varepsilon)}))$.

Proof. By Lemma 34, our parity sampler $\mathcal{W}$ which we use for $\mathcal{C}$ is τ-sampling for

$$\tau = \frac{\lambda}{4} = O\left(\frac{\log d}{\sqrt{d}}\right) = \widetilde{O}\left(2^{-2\sqrt{\log(1/\varepsilon)}}\right),$$

where we used the fact that our randomized construction of H gives us $\lambda = O\left(\frac{1}{\sqrt{d}} \log d\right)$ and the bound on d from Equation (7). All that is left is to show how the two items follow from Theorem 35. For the list decoding result, we take β to be as small as possible. For us,

$$\left(\frac{1}{2} + 2\varepsilon_0\right)^t \leq 2^{-\frac{1}{2}\sqrt{\log(1/\varepsilon)}},$$

and $\sqrt{c_{\mathsf{JST}} \cdot t^3 \tau} = \widetilde{O}\left(2^{-\sqrt{\log(1/\varepsilon)}}\right)$. We can thus conclude that $\beta \leq 2^{-\frac{1}{2}\sqrt{\log(1/\varepsilon)}}$. (Again, we are assuming ε is smaller than some small constant.) For the running time, note that

$$T_0 = \exp(\mathrm{poly}(1/\varepsilon_0))n = \exp\left(2^{O\left(\sqrt{\log(1/\varepsilon)}\right)}\right) \cdot k,$$

and $c_{\beta,t,\varepsilon_0}$ is thus triply-exponential in $\sqrt{\log(1/\varepsilon)}$, and overall $\widetilde{O}\left(c_{\beta,t,\varepsilon_0} \cdot (|\mathcal{W}| + T_0)\right) = \widetilde{O}\left(c_{\beta,t,\varepsilon_0} \cdot k\right)$. For the unique decoding result, we take $\beta = \frac{1+\varepsilon}{4}$, and the running time becomes doubly-exponential in $\sqrt{\log(1/\varepsilon)}$. ◀

6 Assuming a Ramanujan Hypergraph

In Section 4 we showed how to construct a parity sampler given a mixing, or spectral, hypergraph. Applying that construction with a $\lambda = \frac{\mathrm{polylog}(d)}{\sqrt{d}}$-spectral hypergraph allows us to prove Theorem 3, giving a code that approaches the GV bound.

One can also hope for better spectral hypergraphs. For (non-hyper) graphs, the best λ possible is roughly $\frac{2\sqrt{d-1}}{d}$, and graphs with such good expansion are the celebrated *Ramanujan graphs*. In this section, we show how hypergraphs with similar expansion proprieties would give codes even closer to the GV bound.

▶ **Definition 37** (almost Ramanujan hypergraph)**.** *For any $\delta \geq 0$, we say that a d-regular 3-uniform hypergraph is* δ-almost Ramanujan *if it is λ-spectral for $\lambda = \frac{2(1+\delta)\sqrt{d-1}}{d}$.*

We will be interested in δ-almost Ramanujan hypergraphs with $\delta \leq d^c$ for any constant $c < 0$. The goal of this section is to prove the following theorem.

▶ **Theorem 38.** *For any absolute constants $c_1, c_2 > 0$, there is a deterministic algorithm that given any $n \in \mathbb{N}$ and $\varepsilon, \tau > 0$, and a d-regular 3-uniform ($\delta = d^{-c_1}$)-almost Ramanujan hypergraph on n vertices for any d in the range*

$$d_{\min} \leq d \leq d_{\min}^{c_2} \quad \text{where} \quad d_{\min} = \mathrm{poly}\left(\log \frac{1}{\varepsilon}, \frac{1}{\tau}\right),$$

constructs an $(\varepsilon_0, \varepsilon)$ *parity sampler* $\mathcal{W} \sim [n]^t$ *that is homogeneous,* M*-discretizable, and* τ*-sampling for*

$$\begin{aligned}\varepsilon_0 &= \frac{1}{\mathrm{poly}(\log(1/\varepsilon), 1/\tau)},\\ t &= O\left(\log(1/\varepsilon)\right),\\ M &= \frac{n}{\varepsilon^2} \cdot \mathrm{poly}(\log(1/\varepsilon), 1/\tau).\end{aligned}$$

Moreover, the algorithm runs in time $O(Mt)$*.*

We prove Theorem 38 in Section 6.5. As a corollary of the above theorem, assuming explicit almost Ramanujan hypergraphs, we get explicit codes with rate $\widetilde{\Omega}(\varepsilon^2)$ which are (probabilistically) decodable.

▶ **Corollary 39.** *Given an explicit family of almost Ramanujan expanders, as described in Theorem 38, for any* $k \in \mathbb{N}$ *and* $\varepsilon > 0$ *there exists an explicit*[18] ε*-balanced code* $\mathcal{C} \colon \mathbb{F}_2^k \to \mathbb{F}_2^{\bar{n}}$ *of rate* $\frac{k}{\bar{n}} = \varepsilon^2 \cdot \frac{1}{\mathrm{poly}(\log(1/\varepsilon))}$ *with the following decoding capabilities.*

1. $\mathcal{C}$ *is list decodable up to radius* $\frac{1}{2} - \beta$ *for* $\beta = \frac{1}{\mathrm{poly}(\log(1/\varepsilon))}$ *by a randomized algorithm that runs in time* $c(\varepsilon) \cdot k$ *and succeeds with probability* $1 - 2^{-\Omega(k)}$*.*
2. $\mathcal{C}$ *is uniquely decodable to within distance* $\frac{1-\varepsilon}{4}$ *by a randomized algorithm that runs in time* $c(\varepsilon) \cdot k$ *and succeeds with probability* $1 - 2^{-\Omega(k)}$*.*

Above, $c(\varepsilon) = \exp(\exp(\mathrm{poly}(\log(1/\varepsilon))))$*.*

By choosing an appropriate $\tau = \frac{1}{\mathrm{poly}(\log(1/\varepsilon))}$ in Theorem 38, the proof of Corollary 39 is essentially identical, up to parameters, to the proof of Theorem 36 so we omit it.

Overview of the remainder of this section

In Section 6.1, we describe how to construct the parity sampler, $\mathcal{W}$ of Theorem 38, by taking non-backtracking walks on a hypergraph. Then, in Section 6.2, we show that in order to prove $\mathcal{W}$ is a good parity sampler, it is sufficient to bound the operator norm of a certain non-symmetric matrix. In Section 6.3 we upper bound that operator norm. This is the most technically involved part of the proof of Theorem 38. Then, in Section 6.4, we show that $\mathcal{W}$ is τ-sampling for the τ of Theorem 38. Finally, in Section 6.5, we set all parameters, completing the proof of Theorem 38.

6.1 The non-backtracking parity sampler

In order to take advantage of an almost Ramanujan hypergraph $H = (V, E_H)$, we need a more efficient parity sampler than the one presented in Section 4. First, we recall that construction: For any edge $e \in E_H$, let the set of neighboring edges, $N_H(e)$, be all edges $e' \in E_H$ satisfying $e_3 = e'_1$. To sample from our original parity sampler $\boldsymbol{w} \sim \mathcal{W}_{H,t}$, we first sampled a starting hyperedge $\boldsymbol{e}_1$. Then, for each $j \in [2, n]$, we sampled $\boldsymbol{e}_j$ uniformly from the d edges in $N(\boldsymbol{e}_{j-1})$. The final sample $\boldsymbol{w}$ comprises $\boldsymbol{w}_j = (\boldsymbol{e}_j)_2$ for each $j \in [t]$.

When H is symmetric (as in Definition 14), the previous construction is wasteful, as there is a $\frac{1}{d}$ chance that $\boldsymbol{e}_{j+1}$ will just be $\boldsymbol{e}_j$ in reverse. To remedy that inefficiency, we define a non-backtracking parity sampler that avoids taking any reverse steps. Let

[18] Here we assume the explicitness of the base code $\mathcal{C}_0$. See Theorem 30 for the exact dependence of the encoding time on ε_0.

$H = (V, E_H)$ be a symmetric d-regular hypergraph. For any edge $e \in E_H$, let $N_H^{(\mathrm{nb})}(e)$ be the $(d-1)$-sized set consisting of all neighboring edges except for the reversed edge. Namely, $N_H^{(\mathrm{nb})}(e) = N_H(e) \setminus \{(e_3, e_2, e_1)\}$.

The parity sampler $\mathcal{W}_{H,t}^{(\mathrm{nb})}$

The non-backtracking parity sampler is identical to the original parity sampler, except $N_H^{(\mathrm{nb})}$ is used instead of N_H to sample $\boldsymbol{e}_{j+1}$ given $\boldsymbol{e}_j$. In more detail, to sample from it, $\boldsymbol{w} \sim \mathcal{W}_{H,t}^{(\mathrm{nb})}$, we first sample a starting edge $\boldsymbol{e}_1$ uniformly. Then, for each $j \in [2, n]$, we sample $\boldsymbol{e}_j$ uniformly and independently from $N_H^{(\mathrm{nb})}(\boldsymbol{e}_{j-1})$. The final sample $\boldsymbol{w}$ is once again the set $\boldsymbol{w}_j = (\boldsymbol{e}_j)_2$ for each $j \in [t]$.

▶ Remark 40 (unique edges). For convenience, we will assume that in the hypergraph $H = (V, E_H)$, for any $v_1, v_3 \in V$, there is at most one edge $e \in E_H$ satisfying $e_1 = v_1$ and $e_3 = v_3$. This assumption is not essential but simplifies notation. If H has multiple identical edges, then $N_H(e)$ is a multiset instead of a set, and in order to ensure $N_H^{(\mathrm{nb})}(e)$ has size exactly $d-1$, we only want to remove one copy of the reverse edge (e_3, e_2, e_1) from $N_H(e)$.

Much of the analysis of $\mathcal{W}_{H,t}^{(\mathrm{nb})}$ is similar to that of $\mathcal{W}_{H,t}$. We defer the more repetitive proofs to the appendix and will instead focus this section on novel machinery. For example, the proof of Claim 41 is given in Appendix B.

▷ Claim 41. For any $t \in \mathbb{N}$ and any d-regular symmetric hypergraph H over n vertices, the following holds.

- Proposition 59: $\mathcal{W}_{H,t}^{(\mathrm{nb})}$ is homogeneous, and,
- Proposition 60: $\mathcal{W}_{H,t}^{(\mathrm{nb})}$ is $nd(d-1)^{t-1}$-discretizable.

6.2 Expressing the bias algebraically

Fix some $\varepsilon_0 > 0$. In order to prove that $\mathcal{W}_{H,t}$ is an $(\varepsilon_0, \varepsilon \triangleq (\varepsilon_0 + \lambda)^t)$-parity sampler (Theorem 25), we considered any $\sigma \in \{\pm 1\}^n$ satisfying $\left|\mathbb{E}_{\boldsymbol{i} \sim [n]}[\sigma_{\boldsymbol{i}}]\right| \leq \varepsilon_0$ and proved that $\left|\mathrm{bias}_{\mathcal{W}_{H,t}}(\sigma)\right| \leq \varepsilon$, where

$$\mathrm{bias}_{\mathcal{W}_{H,t}}(\sigma) = \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{W}_{H,t}} \left[\prod_{j=1}^{t} \sigma_{\boldsymbol{w}_j} \right].$$

To do so, we expressed $\mathrm{bias}_{\mathcal{W}_{H,t}}(\sigma)$ algebraically in terms of the matrix[19] $A^{(\sigma)} \in \mathbb{R}^{V \times V}$, where

$$A_{i,k}^{(\sigma)} \triangleq \sum_{(i',j',k') \in E_H} \sigma_{j'} \cdot \mathbb{1}[i' = i, k' = k]. \tag{12}$$

Then, we showed that

$$\mathrm{bias}_{\mathcal{W}_{H,t}}(\sigma) = \frac{\mathbf{1}^\dagger \left(A^{(\sigma)}\right)^t \mathbf{1}}{nd^t} \leq \frac{\left\|(A^{(\sigma)})^t\right\|_{\mathrm{op}}}{d^t}.$$

[19] In this section ,we scale up $A^{(\sigma)}$ by a factor of d relative to in Section 4 so that all of its elements are integers. This simplifies notation.

Our approach to proving that $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ is a parity sampler will be similar. Rather than analyzing $A^{(\sigma)}$, we will analyze the non-backtracking operator $B^{(\sigma)} \in \mathbb{R}^{E_H \times E_H}$, where

$$B^{(\sigma)}_{e',e} \triangleq \mathbb{1}\left[e \in N^{(\mathrm{nb})}_H(e')\right] \cdot \sigma_{e_2}. \tag{13}$$

This is a slight extension, to hypergraphs, of the classical non-backtracking operator commonly used to analyze non-backtracking walks on graphs (a walk of vertices $v_0, v_1, \ldots, v_t$ is non-backtracking if $v_i \neq v_{i+2}$ for each $i \in [t-2]$). Just as in the proof of Theorem 25, we'll be able to bound $\mathrm{bias}_{\mathcal{W}^{(\mathrm{nb})}_{H,t}}(\sigma)$ in terms of an appropriate operator norm.

▶ **Lemma 42.** *For any d-regular symmetric hypergraph $H = (V, E_H)$, $\sigma \in \{\pm 1\}^n$, letting $B^{(\sigma)}$ be the non-backtracking operator defined in Equation* (13)*, we have that*

$$\left|\mathrm{bias}_{\mathcal{W}^{(\mathrm{nb})}_{H,t}}(\sigma)\right| = \left|\frac{\mathbf{1}^\dagger (B^{(\sigma)})^t \mathbf{1}}{nd(d-1)^t}\right| \leq \frac{\left\|(B^{(\sigma)})^t\right\|_{\mathrm{op}}}{(d-1)^t}.$$

As the proof of Lemma 42 is similar to that of Theorem 25, we defer it to Appendix B.

6.3 Bounding $\|B^t\|_{\mathrm{op}}$

Throughout this subsection, $\sigma \in \{\pm 1\}^n$ is fixed so we will use A and B as a shorthand for $A^{(\sigma)}$ and $B^{(\sigma)}$ respectively. In proving Theorem 25, we bounded the operator norm of A^t using $\|A^t\|_{\mathrm{op}} \leq \|A\|^t_{\mathrm{op}}$. As A is real and symmetric, that inequality is tight. The corresponding inequality for B, $\|B^t\|_{\mathrm{op}} \leq \|B\|^t_{\mathrm{op}}$, is *not* tight. We will later see that $\|B\|_{\mathrm{op}} = d-1$, and bounding $\|B^t\|_{\mathrm{op}} \leq (d-1)^t$ would be useless for Lemma 42 as it would only upper bound the bias at the trivial bound of 1.

In a different context, Lubetzky and Peres were able to analyze non-backtracking walks on Ramanujan *graphs* [25]. While we cannot use their results verbatim, by reasoning about A and B as operators on a graph (rather than the hypergraph H), we will be able to apply their ideas and techniques.

Let $G = (V, E_G)$ be the graph on the same vertices as H with the edge set $E_G \triangleq \{(e_1, e_3) \mid e \in E_H\}$. Hence, each edge $e \in E_H$ corresponds to an edge $(e_1, e_3) \in E_G$. With that edge, we associate the sign σ_{e_2}. The matrix A is then the *signed* adjacency matrix of G. If instead of viewing B as a matrix in $\mathbb{R}^{E_H \times E_H}$ (as in Equation (13)), we consider the equivalent matrix $B \in \mathbb{R}^{E_G \times E_G}$, then

$$B_{ab,cd} \triangleq \begin{cases} A_{cd} & \text{if } b = c \text{ and } a \neq d \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

The goal of this subsection is to prove the following bound on $\|B^t\|_{\mathrm{op}}$.

▶ **Lemma 43.** *For any $d \geq 2$ and a d-regular edge-signed graph $G = (V, E)$, let A be its (signed) adjacency matrix and B be defined as in Equation* (14)*. For*

$$\theta_{\max} \triangleq \begin{cases} \frac{\|A\|_{\mathrm{op}}}{2} + \sqrt{\frac{\|A\|^2_{\mathrm{op}}}{4} - (d-1)} & \text{if } \|A\|_{\mathrm{op}} \geq 2\sqrt{d-1} \\ \sqrt{d-1} & \text{otherwise} \end{cases}$$

and any $t \geq 1$,

$$\left\|B^t\right\|_{\mathrm{op}} \leq 2(d-1)t(\theta_{\max})^{t-1}. \tag{15}$$

To prove Lemma 43, we will show that B is *almost*-diagonalizable. In particular, that it is unitarily equivalent to a block-diagonal matrix in which each block has size at most 2×2. Lubetzky and Peres proved the below lemma in the case where G is not edge-signed (or equivalently, all edges have the sign $+1$) [25, Proposition 3.1]. Our proof follows theirs in spirit, but we aimed to provide additional details for some parts of the argument (see Remark 52 for a more detailed comparison).

▶ **Lemma 44.** *Let $G = (V, E)$ be a d-regular edge-signed graph on n vertices and $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of its (signed) adjacency matrix. For each $i \in [n]$, let*

$$R_i \triangleq \begin{cases} [d-1] & \text{if } \lambda_i = d \\ [-(d-1)] & \text{if } \lambda_i = -d \\ \begin{bmatrix} \theta_i & \alpha_i \\ 0 & \theta_i' \end{bmatrix} & \text{otherwise,} \end{cases}$$

for some $\alpha_i \in \mathbb{C}$ satisfying $|\alpha_i| \leq d-1$ and $\theta_i, \theta_i' \in \mathbb{C}$ being the two solutions of

$$\theta^2 - \lambda_i \theta + (d-1) = 0.$$

Then, for $k = nd - \sum_{i \in [n]} \dim(R_i)$ and some $b_i \in \{\pm 1\}$ for each $i \in [k]$, the operator B from Equation (14) is unitarily equivalent to $\Lambda \triangleq \operatorname{diag}(R_1, \ldots, R_n, b_1, \ldots, b_k)$.

First, we show how Lemma 43 follows from Lemma 44.

Proof of Lemma 43 assuming Lemma 44. By Lemma 44, we know that B is unitarily equivalent to $\Lambda \triangleq \operatorname{diag}(R_1, \ldots, R_n, b_1, \ldots, b_k)$, where $R_1, \ldots, R_n$ are as defined in Lemma 44 and $b_i \in \{\pm 1\}$ for all $i \in [k]$. Therefore, B^t is unitarily equivalent to

$$\Lambda^t = \operatorname{diag}(R_1^t, \ldots, R_n^t, b_1^t, \ldots, b_k^t).$$

As a result,

$$\left\|B^t\right\|_{\text{op}} = \left\|\Lambda^t\right\|_{\text{op}} = \max\left\{1, \max_{i \in [n]} \left\|R_i^t\right\|_{\text{op}}\right\}.$$

Our goal is to show that the above is bounded by the expression in Equation (15). Since that bound is larger than 1, it is enough to show that $\|R_i^t\|_{\text{op}} \leq 2(d-1)t(\theta_{\max})^{t-1}$ for each $i \in [n]$.

First, consider the case where $|\lambda_i| = d$. By Lemma 44, we have $R_i = [\pm(d-1)]$, and so $\|R_i^t\|_{\text{op}} = (d-1)^t$. In this case, we must have $\|A\|_{\text{op}} = d$ implying that $\theta_{\max} = d-1$. Hence, the right hand side of Equation (15) is at least $2t(d-1)^t$ which is at least as large as $\|R_i^t\|_{\text{op}}$, as desired.

In the other case, $|\lambda_i| < d$. By Lemma 44,

$$R_i = \begin{bmatrix} \theta_i & \alpha_i \\ 0 & \theta_i' \end{bmatrix}$$

for some $\alpha_i \in \mathbb{C}$ satisfying $|\alpha_i| \leq d-1$ and $\theta_i, \theta_i' \in \mathbb{C}$ the two solutions of $\theta^2 - \lambda_i \theta + (d-1) = 0$. As $|\lambda_i| \leq \|A\|_{\text{op}}$, the two solutions of the above equation have their magnitudes bounded by $\theta_{\max}$. We'll use that to bound $\|R_i^t\|_{\text{op}}$. By an easy inductive argument,

$$R_i^t = \begin{bmatrix} (\theta_i)^t & \alpha_i \sum_{j=0}^{t-1} (\theta_i)^j (\theta_i')^{t-1-j} \\ 0 & (\theta_i')^t \end{bmatrix}.$$

Therefore,

$$\begin{aligned}\|R_i^t\|_{\mathrm{op}} &\le \left\|\begin{bmatrix}(\theta_i)^t & 0\\ 0 & (\theta_i')^t\end{bmatrix}\right\|_{\mathrm{op}} + \left\|\begin{bmatrix}0 & \alpha_i\sum_{j=0}^{t-1}(\theta_i)^j(\theta_i')^{t-1-j}\\ 0 & 0\end{bmatrix}\right\|_{\mathrm{op}}\\ &\le \max(|\theta_i|,|\theta_i'|)^t + t\max(|\theta_i|,|\theta_i'|)^{t-1}|\alpha_i|\\ &\le (\theta_{\max})^t + t(\theta_{\max})^{t-1}(d-1)\\ &\le 2t(\theta_{\max})^{t-1}(d-1). && (\theta_{\max}\le(d-1) \text{ and } t\ge 1)\end{aligned}$$

We conclude that $\|R_i^t\|_{\mathrm{op}}$ is at most the bound of Equation (15) for every $i \in [t]$. ◀

In order to prove Lemma 44, we decompose $\mathbb{C}^E$ into subspaces on which B acts independently, namely they are orthogonal and B-invariant.

▶ **Fact 45.** *Let $B : \Omega \to \Omega$ a linear operator, and let $S_1, \ldots, S_m$ be disjoint subspaces for which $S_1 \oplus \cdots \oplus S_m = \Omega$. Assume that the following holds:*
1. *For any $i \neq j \in [m]$, $S_i \perp S_j$, and,*
2. *For any $i \in [m]$, S_i is an invariant subspace of B, meaning $BS_i \subseteq S_i$.*

For each $i \in [m]$, let $R_i \in \mathbb{C}^{(\dim S_i)\times(\dim S_i)}$ be a matrix computing the restriction $B|_{S_i} : S_i \to S_i$ with respect to some orthonormal basis of S_i. Then, B is unitarily equivalent to

$$\Lambda \triangleq \mathrm{diag}(R_1, \ldots, R_m).$$

Proof. For each $i \in [m]$, there is some orthonormal basis $v_{i,1}, \ldots, v_{i,k_i}$ in which the operator $B|_{S_i}$ corresponds to the matrix R_i. As $S_1 \oplus \cdots \oplus S_m = B$ and S_i are orthogonal subspaces, $v_{1,1}, \ldots, v_{1,k_1}, \ldots, v_{m,1}, \ldots, v_{m,k_m}$ is an orthonormal basis for all of Ω, and Λ computes B with respect to that basis. ◀

We break $\mathbb{C}^E$ into subspaces satisfying the requirements of Fact 45. For any $w \in \mathbb{C}^V$, we define $w^{(\mathrm{in})}, w^{(\mathrm{out})} \in \mathbb{C}^E$ as follows:[20]

$$\begin{aligned} w^{(\mathrm{in})}_{xy} &\triangleq w_y\\ w^{(\mathrm{out})}_{xy} &\triangleq A_{xy}w_x. \end{aligned} \tag{16}$$

For $A \in \mathbb{R}^{V\times V}$ being the adjacency matrix defined in Lemma 44, let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of A with corresponding eigenvectors $w_1, \ldots, w_n$. For each $i \in [n]$, we define

$$S_i \triangleq \mathrm{Span}\left\{w_i^{(\mathrm{in})}, w_i^{(\mathrm{out})}\right\}, \tag{17}$$

and define S_{rem} to be the orthogonal compliment of $S_1 \oplus \cdots \oplus S_n$ within $\mathbb{C}^E$. Our first goal is to show that $S_1, \ldots, S_n, S_{\mathrm{rem}}$ meet the requirements of Fact 45.

A large portion of this proof will require straightforward calculations. We collect all of these calculations into the following proposition, proved in Appendix B.

▶ **Proposition 46.**
1. *For any $w, w' \in \mathbb{C}^V$,*

$$\begin{aligned}\langle w^{(\mathrm{in})}, (w')^{(\mathrm{in})}\rangle &= d\cdot\langle w, w'\rangle\\ \langle w^{(\mathrm{out})}, (w')^{(\mathrm{out})}\rangle &= d\cdot\langle w, w'\rangle\\ \langle w^{(\mathrm{in})}, (w')^{(\mathrm{out})}\rangle &= \langle Aw, w'\rangle\end{aligned} \tag{18}$$

[20] We use the name $w^{(\mathrm{in})}$ as all edges going into a vertex y have weight w_y. Similarly, for $w^{(\mathrm{out})}$, all edges going out of the vertex x have weight w_x multiplied by that edge's sign.

2. *For any* $w \in \mathbb{C}^V$,

$$\begin{aligned} Bw^{(\text{in})} &= (Aw)^{(\text{in})} - w^{(\text{out})} \\ Bw^{(\text{out})} &= (d-1)w^{(\text{in})}. \end{aligned} \tag{19}$$

3. *For any* $v \in \mathbb{C}^E$ *satisfying* $v \perp w^{(\text{out})}$ *for all* $w \in \mathbb{C}^V$,

$$v_{xy} = -A_{yx}v_{yx} \qquad \text{for every edge } xy. \tag{20}$$

4. *The operator norm of* B *is*

$$\|B\|_{\text{op}} = d - 1. \tag{21}$$

We use Proposition 46 to prove Lemma 44 by showing that $S_1, \ldots, S_n, S_{\text{rem}}$ meet the requirements of Fact 45.

▶ **Proposition 47.** *For* B *defined in Lemma 44 and* $S_1, \ldots, S_n$ *defined in Equation* (17)*, for any* $i \neq j \in [n]$*,* $S_i \perp S_j$*.*

Proof. It is sufficient to prove that for any $i \neq j \in [n]$, $w_i^{(\text{in})} \perp w_j^{(\text{in})}$, $w_i^{(\text{out})} \perp w_j^{(\text{out})}$, and $w_i^{(\text{in})} \perp w_j^{(\text{out})}$. As A is real and symmetric, its eigenvectors are orthogonal, and so $w_i \perp w_j$ and $Aw_i \perp w_j$. The desired result follows from Equation (18). ◀

In order to apply Fact 45, we also need to prove that each S_i and S_{rem} are all invariant under B.

▶ **Proposition 48.** *For* B *defined in Lemma 44 and* $S_1, \ldots, S_n$ *defined in Equation* (17)*,* $BS_i \subseteq S_i$ *for each* $i \in [n]$*. Furthermore, for* S_{rem}*, the orthogonal compliment of* $S_1 \oplus \cdots \oplus S_n$ *within* $\mathbb{C}^E$*, we also have that* $BS_{\text{rem}} \subseteq S_{\text{rem}}$*.*

Proof. First we show that $BS_i \subseteq S_i$ for any $i \in [n]$. Recall that S_i is the span of $w_i^{(\text{in})}$ and $w_i^{(\text{out})}$, so it suffices to show that $Bw_i^{(\text{in})} \in S_i$ and $Bw_i^{(\text{out})} \in S_i$. This follows from Equation (19) and the fact that w_i is an eigenvector of A.

Secondly, we show that for any $v \in S_{\text{rem}}$, $Bv \in S_{\text{rem}}$. As $v \in S_{\text{rem}}$, we know that $v \perp w_i^{(\text{out})}$ and $v \perp w_i^{(\text{in})}$ for each eigenvector w_i. As the eigenvectors span all of $\mathbb{C}^V$, we further have that $v \perp w^{(\text{in})}$ and $v \perp w^{(\text{out})}$ for any $w \in \mathbb{C}^V$. In order to prove that $Bv \in S_{\text{rem}}$, we will need to prove that $Bv \perp w^{(\text{in})}$ and $Bv \perp w^{(\text{out})}$ for any $w \in \mathbb{C}^V$.

1. We show that $Bv \perp w^{(\text{in})}$:

$$\begin{aligned} \langle Bv, w^{(\text{in})} \rangle &= \sum_{xy \in E} (Bv)_{xy} w_y \\ &= \sum_{xy \in E} -A_{yx} v_{yx} w_y && \text{(Equation (20))} \\ &= -\sum_{xy \in E} (A_{xy} w_x) v_{xy} && \text{(switching names of } x \text{ and } y\text{)} \\ &= -\langle w^{(\text{out})}, v \rangle = 0. && (v \perp w^{(\text{out})}) \end{aligned}$$

2. Similarly, we show that $Bv \perp w^{(\text{out})}$:

$$\begin{aligned}
\langle Bv, w^{(\text{out})} \rangle &= \sum_{xy \in E} (Bv)_{xy} A_{xy} w_x \\
&= \sum_{xy \in E} -A_{yx} v_{yx} A_{xy} w_x && \text{(Equation (20))} \\
&= -\sum_{xy \in E} v_{yx} w_x && (A_{xy} = A_{yx} \in \{\pm 1\}) \\
&= -\sum_{xy \in E} v_{xy} w_y && \text{(switching names of } x \text{ and } y) \\
&= -\langle v, w^{(\text{in})} \rangle = 0. && (v \perp w^{(\text{in})})
\end{aligned}$$

Therefore, $Bv \in S_{\text{rem}}$, as desired. ◀

Using Propositions 47 and 48 we can apply Fact 45 with the decomposition $\mathbb{C}^E = S_1 \oplus \cdots \oplus S_n \oplus S_{\text{rem}}$. In order for Fact 45 to be useful, we need to understand the restricted operator $B|_{S_i}$. Toward this end, we recall the Schur decomposition.

▶ **Fact 49** (Schur decomposition). *For any linear operator $T : \Omega \to \Omega$, there is an orthonormal basis in which T can be written as an upper triangular matrix $U \in \mathbb{C}^{(\dim \Omega) \times (\dim \Omega)}$. In particular, the diagonal entries of U are the eigenvalues of T.*

▶ **Proposition 50.** *Let $A, B, \lambda_1, \ldots, \lambda_n$ be as defined in Lemma 44 and $S_1, \ldots, S_n$ as in Equation (17). For any $i \in [n]$, there is an orthonormal basis of S_i under which $B|_{S_i} : S_i \to S_i$ is computed by the following matrix.*

$$R_i \triangleq \begin{cases} [d-1] & \text{if } \lambda_i = d \\ [-(d-1)] & \text{if } \lambda_i = -d \\ \begin{bmatrix} \theta_i & \alpha_i \\ 0 & \theta_i' \end{bmatrix} & \text{otherwise,} \end{cases}$$

for some $\alpha_i \in \mathbb{C}$ satisfying $|\alpha_i| \leq d-1$, and $\theta_i, \theta_i' \in \mathbb{C}$ are the two solutions of $\theta^2 - \lambda_i \theta + (d-1) = 0$.

Proof. We first compute the dimension of $S_i = \text{Span}\left\{w_i^{(\text{in})}, w_i^{(\text{out})}\right\}$. This dimension is 1 if $w_i^{(\text{in})}$ and $w_i^{(\text{out})}$ are parallel, and 2 otherwise. They are parallel if and only if $\left|\langle w_i^{(\text{in})}, w_i^{(\text{out})} \rangle\right| = \left\|w_i^{(\text{in})}\right\|_2 \left\|w_i^{(\text{out})}\right\|_2$. Assuming that w_i is normalized to satisfy $\|w_i\|_2 = 1$, we have from Equation (18) that $\|w_i^{(\text{in})}\|_2 = \|w_i^{(\text{out})}\|_2 = \sqrt{d}$ and that $\langle w_i^{(\text{in})}, w_i^{(\text{out})} \rangle = \lambda_i$. There are three cases depending on the value of λ_i:

1. If $\lambda_i = d$, then $\langle w_i^{(\text{in})}, w_i^{(\text{out})} \rangle = \|w_i^{(\text{in})}\|_2 \|w_i^{(\text{out})}\|_2$, implying that $w_i^{(\text{in})} = w_i^{(\text{out})}$. In this case, the dimension of S_i is 1, and we can just set R_i to $[\|Bv\|_2 / \|v\|_2]$ for a single nonzero $v \in S_i$. Recall from Equation (19) that $Bw_i^{(\text{out})} = (d-1)w_i^{(\text{in})} = (d-1)w_i^{(\text{out})}$. Therefore, we set $R_i = [d-1]$.
2. If $\lambda_i = -d$, then $\langle w_i^{(\text{in})}, w_i^{(\text{out})} \rangle = -\|w_i^{(\text{in})}\|_2 \|w_i^{(\text{out})}\|_2$, implying that $w_i^{(\text{in})} = -w_i^{(\text{out})}$. Once again, the dimension of S_i is 1, and we set R_i to $[\|Bv\|_2 / \|v\|_2]$ for a single nonzero $v \in S_i$. Also from Equation (19), $Bw_i^{(\text{out})} = (d-1)w_i^{(\text{in})} = -(d-1)w_i^{(\text{out})}$. Therefore, we set $R_i = [-(d-1)]$.

3. Otherwise, $\dim(S_i) = 2$. Recall from Equation (19) that

$$\begin{aligned} Bw_i^{(\text{in})} &= \lambda_i w_i^{(\text{in})} - w_i^{(\text{out})}, \\ Bw_i^{(\text{out})} &= (d-1)w_i^{(\text{in})}. \end{aligned}$$

Therefore, the characteristic polynomial of $B|_{S_i}$ is $p(\theta) = \theta^2 - \theta\lambda_i + (d-1)$. Applying Fact 49, for θ_i, θ_i' being the two roots of $p(\theta)$ and some $\alpha_i \in \mathbb{C}$, we can set

$$R_i = \begin{bmatrix} \theta_i & \alpha_i \\ 0 & \theta_i' \end{bmatrix}.$$

Finally, we bound $|\alpha_i|$:

$$|\alpha_i| \leq \|R_i\|_{\text{op}} \leq \|B\|_{\text{op}} \leq d-1,$$

where $\|B\|_{\text{op}} \leq d-1$ is Equation (21). ◀

Similar to Proposition 50, we characterize $B|_{S_{\text{rem}}}$.

▶ **Proposition 51.** *Let A, B be as defined in Lemma 44, $S_1, \ldots, S_n$ as in Equation* (17)*, and S_{rem} be the orthogonal compliment of $S_1 \oplus \ldots \oplus S_n$ within $\mathbb{C}^E$. Then, there is an orthonormal basis for S_{rem} in which the restriction $B|_{S_{\text{rem}}}$ is expressed by $\text{diag}(b_1, \ldots, b_{\dim(S_{\text{rem}})})$ where $b_i \in \{\pm 1\}$ for each $i \in [\dim(S_{\text{rem}})]$.*

Proof. For any $v \in S_{\text{rem}}$ and any edge $xy \in E$, by Equation (20),

$$(Bv)_{xy} = -A_{yx}v_{yx}.$$

Hence, the operator $B|_{S_{\text{rem}}}$ preserves the ℓ_2 norm. By Fact 49, there is some orthonormal basis in which the matrix representing $B|_{S_{\text{rem}}}$ is an upper triangular matrix. The only upper triangular matrices that are ℓ_2 norm-preserving are diagonal matrices in which all diagonal entries are ± 1. ◀

Combining the above propositions completes the proof of Lemma 44.

Proof of Lemma 44. We apply Fact 45 with the subspaces $S_1, \ldots, S_n, S_{\text{rem}}$. By Propositions 47 and 48, those subspaces meet the requirement of Fact 45. Applying Propositions 50 and 51 gives the desired form for Λ. ◀

▶ Remark 52 (comparison with [25]). Our proof of Lemma 44 follows that of [25, Proposition 3.1] with a few (minor) technical differences. In order to handle edge signs, we must adjust the definitions in Equation (16) and carry that change throughout the computation. Additionally, to prove (the analogous statements of) Propositions 47 and 48, Lubetzky and Peres reason about the operator $C \triangleq (d-1)B^{-1} + B$, whereas we never need consider the inverse of B.

6.4 τ-sampling

In this section, we prove that $\mathcal{W}_{H,t}^{(\text{nb})}$ is τ-sampling for an appropriately chosen τ. Recall that this is needed for the decoding result.

▶ **Lemma 53.** *For any $n, d, t \in \mathbb{N}$ and $\lambda > 0$, if $H = (V = [n], E)$ is a λ-spectral d-regular 3-uniform hypergraph, then $\mathcal{W}_{H,t}^{(\text{nb})}$ is τ-sampling for $\tau = \frac{\lambda}{4} + \frac{2}{d}$.*

In order to prove Lemma 53, we will construct a distribution $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ that is close, in total variation distance, to $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ and show that $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ is $\tau = \frac{\lambda}{4}$-sampling. Once we do this, we will use the closeness of $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ and $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ to transfer that result to a bound on the τ-sampling of $\mathcal{W}^{(\mathrm{nb})}_{H,t}$. As the definition of τ-sampling involves computing a covariance, the following proposition will useful.

▶ **Proposition 54.** *Let $\mathcal{D}, \widehat{\mathcal{D}}$ be distributions each over some domain Ω. For any bounded functions $f, g\colon \Omega \to [0,1]$,*

$$\left|\operatorname*{Cov}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}), g(\boldsymbol{x})] - \operatorname*{Cov}_{\widehat{\boldsymbol{x}}\sim\widehat{\mathcal{D}}}[f(\widehat{\boldsymbol{x}}), g(\widehat{\boldsymbol{x}})]\right| \le 2d_{\mathrm{TV}}(\mathcal{D}, \widehat{\mathcal{D}}).$$

Proof. We use the following covariance identity:

$$\operatorname*{Cov}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}), g(\boldsymbol{x})] = \frac{1}{2}\cdot \mathop{\mathbb{E}}_{\boldsymbol{x}_1,\boldsymbol{x}_2\sim\mathcal{D}^2}[(f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2))(g(\boldsymbol{x}_1) - g(\boldsymbol{x}_2))].$$

Defining $h(x) \triangleq (f(x_1) - f(x_2))(g(x_1) - g(x_2))$, we have

$$\begin{aligned}\left|\operatorname*{Cov}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}), g(\boldsymbol{x})] - \operatorname*{Cov}_{\widehat{\boldsymbol{x}}\sim\widehat{\mathcal{D}}}[f(\widehat{\boldsymbol{x}}), g(\widehat{\boldsymbol{x}})]\right| &\le \frac{1}{2}\left|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}^2}[h(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\widehat{\boldsymbol{x}}\sim\widehat{\mathcal{D}}^2}[h(\widehat{\boldsymbol{x}})]\right| \\ &\le \frac{1}{2} d_{\mathrm{TV}}(\mathcal{D}^2, \widehat{\mathcal{D}}^2)\cdot\left(\sup_{x\in\Omega^2} h(x) - \inf_{x\in\Omega^2} h(x)\right) \\ &\le \frac{1}{2}(2d_{\mathrm{TV}}(\mathcal{D}, \widehat{\mathcal{D}}))\cdot(1-(-1)) = 2d_{\mathrm{TV}}(\mathcal{D}, \widehat{\mathcal{D}}).\end{aligned}$$

◀

In order to prove that $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ is close, in TV distance, to $\mathcal{W}^{(\mathrm{nb})}_{H,t}$, we'll use the following easy fact.

▶ **Fact 55.** *For any distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same domain, and any coupling of $\boldsymbol{x}\sim\mathcal{D}$ and $\boldsymbol{x}'\sim\mathcal{D}'$,*

$$d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') \le \Pr[\boldsymbol{x} \ne \boldsymbol{x}'].$$

The proof of Fact 55 follows readily from the definition of TV distance. Indeed, it is well known that the TV distance is equal to the infimum of $\Pr[\boldsymbol{x} \ne \boldsymbol{x}']$ over all couplings $\boldsymbol{x}\sim\mathcal{D}, \boldsymbol{x}'\sim\mathcal{D}'$.

Using Fact 55, it's not hard to show that the TV distance of $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ and $\mathcal{W}_{H,t}$ is at most $\frac{t}{d}$. Therefore, Lemma 34 and Proposition 54 are sufficient to show that $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ is $\frac{\lambda}{4} + \frac{2t}{d}$-sampling. In the following proof, we'll construct a distribution $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ that is closer to $\mathcal{W}^{(\mathrm{nb})}_{H,t}$, giving a better bound on τ.

Proof of Lemma 53. As in Definition 32, fix any $i \in [t]$, $S \subseteq [n]$ and $X \subseteq [n]^i$. Let $S_i(\boldsymbol{w})$ be the indicator that $\boldsymbol{w}_{i+1} \in S$, and $X_i(w)$ the indicator that $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t) \in X$. Our goal is to prove that

$$\operatorname*{Cov}_{\boldsymbol{w}\sim\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}}\Big[S_i(\boldsymbol{w}), X_i(\boldsymbol{w})\Big] \le \frac{\lambda}{4} + \frac{2}{d}.$$

To do so, we will define a distribution $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ satisfying $d_{\mathrm{TV}}(\mathcal{W}^{(\mathrm{nb})}_{H,t}, \widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}) \le \frac{1}{d}$ and prove that

$$\operatorname*{Cov}_{\widehat{\boldsymbol{w}}\sim\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}}\Big[S_i(\widehat{\boldsymbol{w}}), X_i(\widehat{\boldsymbol{w}})\Big] \le \frac{\lambda}{4}. \tag{22}$$

At a high level, $\widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$ will perform "non-backtracking" steps during the first i time steps and a normal ("backtracks" with probability $\frac{1}{d}$) step at time step $j = i+1$. In contrast, $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ performs "non-backtracking" steps at every time step. Formally, to sample $\widehat{\boldsymbol{w}} \sim \widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$, we first sample $\boldsymbol{w} \sim \mathcal{W}^{(\mathrm{nb})}_{H,t}$. Then, with probability $1 - \frac{1}{d}$, $\widehat{\boldsymbol{w}}$ is set to $\boldsymbol{w}$. Otherwise, $\widehat{\boldsymbol{w}}_j = \boldsymbol{w}_j$ for each $j \neq i+1$, and $\widehat{\boldsymbol{w}}_{i+1} = \boldsymbol{w}_i$. Through the remainder of this proof, we assume that $\boldsymbol{w}$ and $\widehat{\boldsymbol{w}}$ are coupled according to this generation process.

Clearly, $d_{\mathrm{TV}}(\mathcal{W}^{(\mathrm{nb})}_{H,t}, \widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}) \leq 1/d$. Hence, by Proposition 54 it suffices to prove Equation (22). The remainder of this proof is similar to that of Lemma 34. Instead of showing Equation (22), we prove the following (equivalent equation) holds:

$$\mathrm{Cov}\left[(-1)^{S_i(\widehat{\boldsymbol{w}})}, (-1)^{X_i(\widehat{\boldsymbol{w}})}\right] \leq \lambda.$$

Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t$ be the random variables defined in the construction of $\mathcal{W}^{(\mathrm{nb})}_{H,t}$ (which are coupled to the sample $\boldsymbol{w} \sim \mathcal{W}^{(\mathrm{nb})}_{H,t}$ and hence $\widehat{\boldsymbol{w}} \sim \widehat{\mathcal{W}}^{(\mathrm{nb})}_{H,t}$). If the probability-$\left(1 - \frac{1}{d}\right)$ event occurred where we set $\widehat{\boldsymbol{w}} = \boldsymbol{w}$, then we define $\widehat{\boldsymbol{e}}_{i+1} \triangleq \boldsymbol{e}_{i+1}$.

Recall that $\boldsymbol{e}_{i+1}$ is sampled uniformly from the $(d-1)$ hyperedges satisfying $(\boldsymbol{e}_i)_3 = (\boldsymbol{e}_{i+1})_1$ and $\boldsymbol{e}_{i+1} \neq ((\boldsymbol{e}_i)_3, (\boldsymbol{e}_i)_2, (\boldsymbol{e}_i)_1)$. Therefore, the generation process for $\widehat{\boldsymbol{e}}_{i+1}$ is equivalent to sampling uniformly from the d hyperedges satisfying $(\boldsymbol{e}_i)_3 = (\widehat{\boldsymbol{e}}_{i+1})_1$.

Let $x, y, z \in \mathbb{R}^n$ be the vectors defined, for each $j \in [n]$,

$$\begin{aligned}
x_j &\triangleq \mathbb{E}\left[(-1)^{X_i(\widehat{\boldsymbol{w}})} \,\middle|\, (\widehat{\boldsymbol{e}}_{i+1})_1 = j\right] \\
y_j &\triangleq (-1)^{\mathbb{1}[j \in S]} \\
z_j &\triangleq \frac{1}{n}.
\end{aligned}$$

Note that $\|z\|_2 = 1/\sqrt{n}$, $\|y\|_\infty = 1$, and $\|x\|_2 \leq \sqrt{n}$ (which follows from $\|x\|_\infty \leq 1$). Our goal will be to prove that the following equation holds.

$$\underset{\widehat{\boldsymbol{w}}}{\mathrm{Cov}}\left[(-1)^{S_i(\widehat{\boldsymbol{w}})}, (-1)^{X_i(\widehat{\boldsymbol{w}})}\right] = \frac{1}{d} \cdot \sum_{(a,b,c) \in E} x_a y_b z_c - \frac{1}{n^2} \cdot \sum_{a \in [n]} x_a \cdot \sum_{a \in [n]} y_a \cdot \sum_{a \in [n]} z_a. \tag{23}$$

Once we prove the above equation holds, the desired result follows from Definition 12. As we proved in Claim 41, the distribution of $\boldsymbol{e}_i$ is uniform over all nd hyperedges, and hence that of $\widehat{\boldsymbol{e}}_{i+1}$ is also uniform over all hyperedges. In order to compute the covariance, we first expand:

$$\begin{aligned}
\underset{\widehat{\boldsymbol{w}}}{\mathbb{E}}\left[(-1)^{S_i(\widehat{\boldsymbol{w}})} \cdot (-1)^{X_i(\widehat{\boldsymbol{w}})}\right] &= \frac{1}{nd} \cdot \sum_{e \in E} \underset{\widehat{\boldsymbol{w}}}{\mathbb{E}}\left[(-1)^{S_i(\widehat{\boldsymbol{w}})} \cdot (-1)^{X_i(\widehat{\boldsymbol{w}})} \mid \widehat{\boldsymbol{e}}_{i+1} = e\right] \\
&= \frac{1}{nd} \cdot \sum_{(a,b,c) \in E} (-1)^{\mathbb{1}[b \in S]} \cdot \mathbb{E}\left[(-1)^{X_i(\widehat{\boldsymbol{w}})} \,\middle|\, (\widehat{\boldsymbol{e}}_{i+1})_1 = a\right] \\
&= \frac{1}{d} \cdot \sum_{(a,b,c) \in E} x_a y_b z_c.
\end{aligned} \tag{24}$$

Next, directly from the definition of y and Claim 41,

$$\underset{\widehat{\boldsymbol{w}}}{\mathbb{E}}\left[(-1)^{S_i(\widehat{\boldsymbol{w}})}\right] = \frac{1}{n} \sum_{a \in [n]} y_a. \tag{25}$$

Similarly,

$$\mathop{\mathbb{E}}_{\widehat{\boldsymbol{w}}}\left[(-1)^{X_i(\widehat{\boldsymbol{w}})}\right] = \frac{1}{n}\sum_{a\in[n]} x_a. \tag{26}$$

Equation (23) follows from Equations (24)–(26) and the fact that $\sum_{a\in[n]} z_a = 1$. Lemma 53 follows from Definition 12, and the desired result from Proposition 54. ◀

6.5 Setting the parameters

In this subsection, we set the parameters needed to prove the following theorem, restated for convenience.

▶ **Theorem 38.** *For any absolute constants $c_1, c_2 > 0$, there is a deterministic algorithm that given any $n \in \mathbb{N}$ and $\varepsilon, \tau > 0$, and a d-regular 3-uniform ($\delta = d^{-c_1}$)-almost Ramanujan hypergraph on n vertices for any d in the range*

$$d_{\min} \le d \le d_{\min}^{c_2} \quad \text{where} \quad d_{\min} = \operatorname{poly}\left(\log\frac{1}{\varepsilon}, \frac{1}{\tau}\right),$$

constructs an $(\varepsilon_0, \varepsilon)$ parity sampler $\mathcal{W} \sim [n]^t$ that is homogeneous, M-discretizable, and τ-sampling for

$$\begin{aligned}\varepsilon_0 &= \frac{1}{\operatorname{poly}(\log(1/\varepsilon), 1/\tau)},\\ t &= O\left(\log(1/\varepsilon)\right),\\ M &= \frac{n}{\varepsilon^2}\cdot\operatorname{poly}(\log(1/\varepsilon), 1/\tau).\end{aligned}$$

Moreover, the algorithm runs in time $O(Mt)$.

Proof. We set

$$d_{\min} \triangleq \max\left(\frac{9}{\tau^2}, \log\left(\frac{1}{\varepsilon}\right)^{2/c_1}, 145\right).$$

Let H be the d-regular 3-uniform ($\delta \triangleq d^{-c_1}$)-almost Ramanujan hypergraph on n vertices for some $d_{\min} \le d \le (d_{\min})^{c_2}$ given to the algorithm. It will return $\mathcal{W} \triangleq \mathcal{W}_{H,t}^{(\mathrm{nb})}$ for some t to be later specified.

Regardless of what that t is, by Lemma 53, we have that $\mathcal{W}$ is τ'-sampling for

$$\tau' \le \frac{4\sqrt{d-1}}{4d} + \frac{2}{d} \le \frac{3}{\sqrt{d}} \le \tau.$$

To complete this proof, we need to set t large enough so that $\mathcal{W}$ is an $(\varepsilon_0, \varepsilon)$-parity sampler and small enough so that it is M-discretizable. Set

$$\varepsilon_0 = \frac{2\delta\sqrt{d-1}}{d}.$$

In order to set t, we first define,

$$\varepsilon(t') \triangleq \frac{2t'(1+5d^{-c_1/2})^{t'-1}}{(d-1)^{(t'-3)/2}}$$

and then set t to the minimum integer so that $\varepsilon(t) \le \varepsilon$. As $d \ge d_{\min} \ge 145$,

$$\varepsilon(t') \ge \frac{2t' \cdot 6^{t'-1}}{12^{t'-3}} = c \cdot t' \cdot 2^{-t'}$$

for an absolute constant c. Therefore, the minimum t for which $\varepsilon(t) \le \varepsilon$ is $O(\log(1/\varepsilon))$, as desired.

We proceed to bound M. For any $t' \in \mathbb{N}$, $\frac{\varepsilon(t')}{\varepsilon(t'+1)} \le \sqrt{d-1}$, so $\varepsilon(t) \ge \frac{\varepsilon}{\sqrt{d-1}}$. Therefore,

$$(d-1)^{t-1} = \frac{4t^2(d-1)^2(1+5d^{-c_1/2})^{2t-2}}{\varepsilon(t)^2} \le \frac{4t^2(d-1)^3(1+5d^{-c_1/2})^{2t-2}}{\varepsilon^2}.$$

By Claim 41, $\mathcal{W}$ is M-discretizable for $M \le \frac{n}{\varepsilon}^2 \cdot 4dt^2(d-1)^3(1+5d^{-c_1/2})^{2t-2}$. Using the bounds $(1+a)^b \le \exp(ab)$, $d \ge d_{\min} \ge \log(1/\varepsilon)^{2/c_1}$, and $t = O(\log(1/\varepsilon)$, we have $(1+5d^{-c_1/2})^{2t-2} = O(1)$. We've also already bounded d and t, each at most $\mathrm{poly}(\log(1/\varepsilon), 1/\tau)$, so we can give the desired bound on M, $M \le \frac{n}{\varepsilon}^2 \cdot \mathrm{poly}(\log(1/\varepsilon), 1/\tau)$.

Lastly, we need to show that $\mathcal{W}$ is an $(\varepsilon_0, \varepsilon)$-parity sampler. Choose any $\sigma \in \{\pm 1\}^n$ satisfying $|\mathbb{E}_{\boldsymbol{i} \sim [n]}[\sigma_{\boldsymbol{i}}]| \le \varepsilon_0$. By Lemma 42, it is sufficient to show, for $B^{(\sigma)}$ defined in Equation (13), that $\frac{\left\|(B^{(\sigma)})^t\right\|_{\mathrm{op}}}{(d-1)^t} \le \varepsilon$. Lemma 43 allows us to bound $\|(B^{(\sigma)})^t\|_{\mathrm{op}}$ in terms of $\|(A^{(\sigma)})^t\|_{\mathrm{op}}$ for $A^{(\sigma)}$ defined in Equation (12). Applying Corollary 29 (and noting that $A^{(\sigma)}$ of this section is scaled up by a factor of d relative to that in Corollary 29),

$$\begin{aligned}\left\|A^{(\sigma)}\right\|_{\mathrm{op}} &\le d\left(|\mathrm{bias}(\sigma)| + \lambda\right) \le d\varepsilon_0 + 2(1+\delta)\sqrt{d-1} \\ &= 2\delta\sqrt{d-1} + 2(1+\delta)\sqrt{d-1} = 2(1+2\delta)\sqrt{d-1}.\end{aligned}$$

The quantity $\theta_{\max}$ defined in Lemma 43 satisfies

$$\begin{aligned}\theta_{\max} &= \frac{\|A\|_{\mathrm{op}}}{2} + \sqrt{\frac{\|A\|_{\mathrm{op}}^2}{4} - (d-1)} \le (1+2\delta)\sqrt{d-1} + \sqrt{d-1}\sqrt{(1+2\delta)^2 - 1} \\ &= \sqrt{d-1} \cdot \left(1 + 2\delta + \sqrt{4\delta^2 + 4\delta}\right) \le \sqrt{d-1} \cdot \left(1 + 2\delta + \sqrt{8\delta}\right) \qquad (\delta \le 1) \\ &\le \sqrt{d-1} \cdot \left(1 + 2\delta + 3\sqrt{\delta}\right) \le \sqrt{d-1} \cdot \left(1 + 5\sqrt{\delta}\right) \le \sqrt{d-1} \cdot \left(1 + 5d^{-c_1/2}\right).\end{aligned}$$

By Lemma 43, we then have that

$$\begin{aligned}\frac{\left\|\left(B^{(\sigma)}\right)^t\right\|_{\mathrm{op}}}{(d-1)^t} &\le \frac{2(d-1)t(\theta_{\max})^{t-1}}{(d-1)^{t-1}} \\ &\le \frac{2(d-1)t(1+5d^{-c_1/2})^{t-1}}{(d-1)^{(t-1)/2}} = \varepsilon(t) \le \varepsilon.\end{aligned}$$

Therefore, $\mathcal{W}$ is an $(\varepsilon_0, \varepsilon)$ parity sampler. ◀

References

1. Noga Alon. Explicit expanders of every degree and size. *Combinatorica*, pages 1–17, 2021.
2. Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *Information Theory, IEEE Transactions on*, 38(2):509–516, 1992.
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 Avraham Ben-Aroya and Amnon Ta-Shma. A combinatorial construction of almost-Ramanujan graphs using the zig-zag product. *SIAM Journal on Computing*, 40(2):267–290, 2011.

5 Avraham Ben-Aroya and Amnon Ta-Shma. Constructing small-bias sets from algebraic-geometric codes. *Theory of Computing*, 9(5):253–272, 2013.

6 Yonatan Bilu and Shlomo Hoory. On codes from hypergraphs. *European Journal of Combinatorics*, 25(3):339–354, 2004.

7 Yonatan Bilu and Nathan Linial. Lifts, discrepancy and nearly optimal spectral gap. *Combinatorica*, 26(5):495–519, 2006.

8 Andrej Bogdanov. A different way to improve the bias via expanders. *Topics in (and out) the theory of computing, Lecture*, 2012.

9 Emma Cohen, Dhruv Mubayi, Peter Ralli, and Prasad Tetali. Inverse expander mixing for hypergraphs. *The Electronic Journal of Combinatorics*, 23(2):P2–20, 2016.

10 David Conlon, Jonathan Tidor, and Yufei Zhao. Hypergraph expanders of all uniformities from Cayley graphs. *Proceedings of the London Mathematical Society*, 121(5):1311–1336, 2020.

11 Joel Friedman and Avi Wigderson. On the second eigenvalue of hypergraphs. *Combinatorica*, 15(1):43–65, 1995.

12 Edgar N. Gilbert. A comparison of signalling alphabets. *The Bell system technical journal*, 31(3):504–522, 1952.

13 Oded Goldreich. On constructing expanders for any number of vertices, October 2019. Available at `https://www.wisdom.weizmann.ac.il/~oded/R1/ex4all.pdf`.

14 Konstantin Golubev and Ori Parzanchevski. Spectrum and combinatorics of two-dimensional Ramanujan complexes. *Israel Journal of Mathematics*, 230(2):583–612, 2019.

15 Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

16 Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential Coding Theory, 2015. URL: `http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book`.

17 Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *SIAM Journal on Computing*, pages FOCS17–157, 2019.

18 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

19 Fernando Granha Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. Unique decoding of explicit ε-balanced codes near the Gilbert–Varshamov bound. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 434–445. IEEE, 2020.

20 Fernando Granha Jeronimo, Shashank Srivastava, and Madhur Tulsiani. Near-linear time decoding of Ta-Shma's codes via splittable regularity. In *Proceedings of the 53rdth Annual Symposium on Theory of Computing (STOC 2021)*, pages 1527–1536. ACM, 2021.

21 Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *IEEE Transactions on Information Theory*, 67(1):296–316, 2020.

22 Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded Reed–Solomon and multiplicity codes. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 212–223. IEEE, 2018.

23 John Lenz and Dhruv Mubayi. Eigenvalues and linear quasirandom hypergraphs. In *Forum of Mathematics, Sigma*, volume 3. Cambridge University Press, 2015.

24 Ray Li and Mary Wootters. Improved list-decodability of random linear binary codes. In *APPROX–RANDOM*, volume 116 of *LIPIcs*, pages 50:1–50:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

25 Eyal Lubetzky and Yuval Peres. Cutoff on all Ramanujan graphs. *Geometric and Functional Analysis*, 26(4):1190–1216, 2016.

26 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

27 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type A_d. *European Journal of Combinatorics*, 26(6):965–993, 2005.

28 Grigorii Aleksandrovich Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problemy peredachi informatsii*, 24(1):51–60, 1988.

29 Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. Deterministic approximation of random walks in small space. *Theory of Computing*, 17(1):1–35, 2021.

30 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

31 Ori Parzanchevski. Mixing in high-dimensional expanders. *Combinatorics, Probability and Computing*, 26(5):746–761, 2017.

32 Ori Parzanchevski, Ron Rosenthal, and Ran J. Tessler. Isoperimetric inequalities in simplicial complexes. *Combinatorica*, 36(2):195–227, 2016.

33 Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE transactions on Information Theory*, 42(6):1710–1722, 1996.

34 Amnon Ta-Shma. Explicit, almost optimal, ε-balanced codes. In *Proceedings of the 49th Annual Symposium on Theory of Computing (STOC 2017)*, pages 238–251. ACM, 2017.

35 R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on information theory*, 27(5):533–547, 1981.

36 Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957.

37 Gillés Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001.

A Splittability and τ-Sampling

Our decoding result follows from the work of Jernoimo, Srivastava, and Tulsiani [20]. In [20], for the result of Theorem 35 to hold, they require $\mathcal{W}$ to be *τ-splittable*. τ-splittability is stronger than, and in fact implies, τ-sampling. We will not define splittability here, since for their result to hold, only a "Splittable Mixing Lemma" is required. The mixing property used in [20] goes as follows.

▶ **Definition 56** (splittable mixing for ± 1 cut functions)**.** *Given positive integers n, t, and $s < t-1$, we denote by $\mathcal{F}_s$ the set of all functions $f\colon [n]^t \to \{1,-1\}$ of the form*

$$f(x_1,\ldots,x_t) = b \cdot \chi_{A_1}(x_1)\cdot\ldots\cdot\chi_{A_s}(x_s)\cdot\chi_B(x_{s+1},\ldots,x_t), \tag{27}$$

where $b \in \{1,-1\}$, $A_1,\ldots,A_s \subseteq [n]$, $B \subseteq [n]^{t-s}$, and where $\chi_S(x) = -1$ if $x \in S$ and 1 otherwise.

For a distribution $\mathcal{W} \sim [n]^t$, we denote by $\mathcal{W}_{[i,j]}$ the distribution over $[n]^{j-i+1}$ that is obtained by sampling $\boldsymbol{x} \sim \mathcal{W}$ and outputting $\boldsymbol{x}_{[i,j]}$. For simplicity, we abbreviate $\mathcal{W}_i = \mathcal{W}_{[i,i]}$. Given $s \in [t-1]$, let ν_s be the probability measure over $[n]^t$ for which $\nu_s(x) = \prod_{i\in[s]} \Pr[\mathcal{W}_i = x_i]\cdot\Pr[\mathcal{W}_{[s+1,t]} = x_{[s+1,t]}]$. We say $\mathcal{W} \sim [n]^t$ satisfies splittable mixing with error τ*, if for every $s \in [t-1]$ and every $f, f' \in \mathcal{F}_s$ it holds that*

$$\left| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\nu_s}[f(\boldsymbol{x})f'(\boldsymbol{x})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\nu_{s-1}}[f(\boldsymbol{x})f'(\boldsymbol{x})] \right| \leq \tau. \tag{28}$$

In [20], they show that the above property follows from τ-splittablility. Here we show that the above property also follows from our weaker notion of τ-sampling. For the sake of being compatible with [20], we will use a slightly different definition of τ-sampling than what was given in Definition 32.

▶ **Definition 57.** *We say that a distribution $\mathcal{W}$ over $[n]^t$ is* τ-sampling *if for any $i \in [t-1]$, $S \subseteq [n]$, and $X \subseteq [n]^{t-i}$,*

$$\operatorname*{Cov}_{\boldsymbol{w}\sim\mathcal{W}}\Big[\mathbb{1}[\boldsymbol{w}_i \in S], \mathbb{1}[(\boldsymbol{w}_{i+1}, \dots \boldsymbol{w}_t) \in X]\Big] \leq \tau.$$

For decoding lifted sum codes, both definitions are essentially the same, since we can always work with $\mathcal{W}_{\mathrm{rev}}$, wherein we sample $\boldsymbol{w} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_t) \sim \mathcal{W}$ and output $(\boldsymbol{w}_t, \dots, \boldsymbol{w}_1)$. Moreover, our $\mathcal{W}$ satisfies $\mathcal{W} = \mathcal{W}_{\mathrm{rev}}$.

▷ Claim 58. Let $\mathcal{W} \subseteq [n]^t$ be homogeneous and τ-sampling. Then, $\mathcal{W}$ satisfies splittable mixing with error 4τ.

Proof. Let $s \in [t-1]$, $b, b' \in \{1, -1\}$ and sets $A_1, \dots, A_s, A'_1, \dots, A'_s \subseteq [n]$ and $B, B' \subseteq [n]^{t-s}$ that correspond to the functions f and f'. For $i \in [s]$ denote $C_i = A_i \triangle A'_i$ and $D = B \triangle B'$. We then have

$$f(x)f'(x) = \sigma \cdot \chi_{C_1}(x_1) \cdot \ldots \cdot \chi_{C_s}(x_s) \cdot \chi_D(x_{s+1}, \dots, x_t),$$

for $\sigma = bb' \in \{1, -1\}$. The subtraction on the left hand side of Equation (28) now reads

$$\sum_{x\in[n]^{s-1}} \prod_{i\in[s-1]} \Pr[\mathcal{W}_i = x_i] \cdot \sigma \cdot \chi_{C_1}(x_1) \cdot \ldots \cdot \chi_{C_{s-1}}(x_{s-1}) \cdot \sum_{y\in[n], z\in[n]^{t-s}} \chi_{C_s}(y) \cdot \chi_D(z) \cdot \left(\Pr[\mathcal{W}_s = y] \Pr[\mathcal{W}_{[s+1,t]} = z] - \Pr[\mathcal{W}_{[s,t]} = y \circ z]\right). \quad (29)$$

The second line of the above expression can be written as

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{W}_s}\left[(-1)^{\mathbb{1}[\boldsymbol{y}\in C_s]}\right] \cdot \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{W}_{[s+1,t]}}\left[(-1)^{\mathbb{1}[\boldsymbol{z}\in D]}\right] - \mathop{\mathbb{E}}_{(\boldsymbol{y},\boldsymbol{z})\sim(\mathcal{W}_s,\mathcal{W}_{[s+1,t]})}\left[(-1)^{\mathbb{1}[\boldsymbol{y}\in C_s]}(-1)^{\mathbb{1}[\boldsymbol{z}\in D]}\right], \quad (30)$$

which is simply the covariance between the random variables $(-1)^{\mathbb{1}[\boldsymbol{y}\in C_s]}$ and $(-1)^{\mathbb{1}[\boldsymbol{z}\in D]}$ where $\boldsymbol{y}$ and $\boldsymbol{z}$ are drawn appropriately. As $\mathcal{W}$ is τ-sampling, we know that

$$\left|\operatorname{Cov}\left[\mathbb{1}[\boldsymbol{y} \in C_s], \mathbb{1}[\boldsymbol{z} \in D]\right]\right| \leq \tau,$$

and so Equation (30), in absolute value, amounts to

$$\left|\operatorname{Cov}\left[(-1)^{\mathbb{1}[\boldsymbol{y}\in C_s]}, (-1)^{\mathbb{1}[\boldsymbol{z}\in D]}\right]\right| \leq 4\tau,$$

which readily follows from the fact that $(-1)^b = 1 - 2b$ for $b \in \{0, 1\}$, as we argued in Lemma 34. Taking absolute value, by the triangle inequality, Equation (29) can be bounded by

$$4\tau \cdot \sum_{x\in[n]^{s-1}} \prod_{i\in[s-1]} \Pr[\mathcal{W}_i = x_i].$$

As $\mathcal{W}$ is homogeneous, $\Pr[\mathcal{W}_i = x_i] = \frac{1}{n}$, and we are done. ◁

B Properties of the Non-Backtracking Parity Sampler

First, we prove the two propositions of Claim 41.

▶ **Proposition 59.** *For any $t \in \mathbb{N}$ and d-regular symmetric hypergraph H over n vertices, $\mathcal{W}_{H,t}^{(\mathrm{nb})}$ is homogeneous.*

Proof. $\boldsymbol{e}_1$ is uniform over E_H. Then, for every $j \in [t-1]$, if $\boldsymbol{e}_j$ is uniform, the $\boldsymbol{e}_{j+1}$ is uniform. By induction, $\boldsymbol{e}_j$ is uniform for every $j \in [t]$.

Next, fix any $a \in [n]$ and $j \in [t]$. As $\boldsymbol{w}_j = a$ if and only if $\boldsymbol{e}_j$ is one of the d-edges in E_H who's second vertex is a and $\boldsymbol{e}_j$ is uniform over the nd edges in E_H, $\Pr[\boldsymbol{w}_j = a] = \frac{d}{nd} = \frac{1}{n}$. ◀

▶ **Proposition 60.** *For any $t \in \mathbb{N}$ and d-regular symmetric hypergraph H over n vertices, $\mathcal{W}_{H,t}^{(\mathrm{nb})}$ is $(nd(d-1)^{t-1})$-discretizable.*

Proof. $\boldsymbol{e}_1$ is picked uniformly from nd options. Then, for each $j \in [2,t]$, $\boldsymbol{e}_{j+1}$ is picked uniformly (and independently from prior choices) from $N_H^{(\mathrm{nb})}(e)$, which has $(d-1)$ elements. Therefore, $\boldsymbol{w}$ is picked uniformly from $nd(d-1)^t$ possible items, potentially with duplicates. ◀

Next, we prove the following Lemma, restated for convenience.

▶ **Lemma 42.** *For any d-regular symmetric hypergraph $H = (V, E_H)$, $\sigma \in \{\pm 1\}^n$, letting $B^{(\sigma)}$ be the non-backtracking operator defined in Equation (13), we have that*

$$\left|\mathrm{bias}_{\mathcal{W}_{H,t}^{(\mathrm{nb})}}(\sigma)\right| = \left|\frac{\mathbf{1}^\dagger (B^{(\sigma)})^t \mathbf{1}}{nd(d-1)^t}\right| \leq \frac{\left\|(B^{(\sigma)})^t\right\|_{\mathrm{op}}}{(d-1)^t}.$$

Proof. Our goal is to prove that

$$\mathrm{bias}_{\mathcal{W}_{H,t}^{(\mathrm{nb})}}(\sigma) = \mathop{\mathbb{E}}_{\boldsymbol{w} \sim \mathcal{W}_{H,t}} \left[\prod_{j=1}^{t} \sigma_{\boldsymbol{w}_j}\right] = \frac{\mathbf{1}^\dagger (B^{(\sigma)})^t \mathbf{1}}{nd(d-1)^t}.$$

Draw some $\boldsymbol{w} \sim \mathcal{W}_{H,t}^{(\mathrm{nb})}$, and let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_t$ be the random variables (coupled to $\boldsymbol{w}$) defined in the construction of $\mathcal{W}_{H,t}^{(\mathrm{nb})}$. We claim that for each $j \in [t]$ and $e \in E_H$,

$$\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_j = e] \prod_{k=1}^{j} \sigma_{\boldsymbol{w}_k}\right] = \frac{(\mathbf{1}^\dagger (B^{(\sigma)})^j)_e}{nd(d-1)^j}. \tag{31}$$

For $j = 1$, $\boldsymbol{e}_1$ is initialized uniformly among the nd edges, so the above expression should be equal to $\frac{\sigma_{e_2}}{nd}$. Indeed,

$$\begin{aligned}\frac{(\mathbf{1}^\dagger B^{(\sigma)})_e}{nd(d-1)} &= \frac{1}{nd(d-1)} \sum_{e' \in E_H} B_{e',e} \\ &= \frac{1}{nd(d-1)} \sum_{e' \in E_H} \mathbb{1}\left[e \in N_H^{(\mathrm{nb})}(e')\right] \cdot \sigma_{e_2} = \frac{\sigma_{e_2}}{nd}.\end{aligned}$$

So Equation (31) holds for $j = 1$. For $j \geq 2$, we proceed by induction.

$$\begin{aligned}
\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_j = e]\prod_{k=1}^{j}\sigma_{\boldsymbol{w}_k}\right] &= \sum_{e'\in E_H}\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_j = e, \boldsymbol{e}_{j-1} = e']\prod_{k=1}^{j}\sigma_{\boldsymbol{w}_k}\right] \\
&= \sum_{e'\in E_H}\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_{j-1} = e']\prod_{k=1}^{j-1}\sigma_{\boldsymbol{w}_k}\right]\cdot\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_j = e]\cdot\sigma_{\boldsymbol{w}_j} \mid \boldsymbol{e}_{j-1} = e'\right] \\
&= \sum_{e'\in E_H}\frac{(\mathbf{1}^\dagger(B^{(\sigma)})^{j-1})_{e'}}{nd(d-1)^{j-1}}\cdot\mathbb{E}\left[\mathbb{1}[\boldsymbol{e}_j = e]\cdot\sigma_{\boldsymbol{w}_j} \mid \boldsymbol{e}_{j-1} = e'\right] \\
&= \sum_{e'\in E_H}\frac{(\mathbf{1}^\dagger(B^{(\sigma)})^{j-1})_{e'}}{nd(d-1)^{j-1}}\cdot\frac{\mathbb{1}\left[e\in N_H^{(\mathrm{nb})}(e')\right]\cdot\sigma_{e_2}}{d-1} \\
&= \frac{1}{nd(d-1)^j}\cdot\sum_{e'\in E_H}(\mathbf{1}^\dagger(B^{(\sigma)})^{j-1})_{e'}\cdot B^{(\sigma)}_{e',e} \\
&= \frac{(\mathbf{1}^\dagger(B^{(\sigma)})^{j})_{e}}{nd(d-1)^j}
\end{aligned}$$

Hence, Equation (31) holds by induction. The desired result follows by summing Equation (31) over all edges. ◀

Next, we do all the calculations necessary for Proposition 46, restated below for convenience.

▶ **Proposition 46.**

1. *For any $w, w' \in \mathbb{C}^V$,*

$$\begin{aligned}
\langle w^{(\mathrm{in})}, (w')^{(\mathrm{in})}\rangle &= d\cdot\langle w, w'\rangle \\
\langle w^{(\mathrm{out})}, (w')^{(\mathrm{out})}\rangle &= d\cdot\langle w, w'\rangle \\
\langle w^{(\mathrm{in})}, (w')^{(\mathrm{out})}\rangle &= \langle Aw, w'\rangle
\end{aligned} \tag{18}$$

2. *For any $w \in \mathbb{C}^V$,*

$$\begin{aligned}
Bw^{(\mathrm{in})} &= (Aw)^{(\mathrm{in})} - w^{(\mathrm{out})} \\
Bw^{(\mathrm{out})} &= (d-1)w^{(\mathrm{in})}.
\end{aligned} \tag{19}$$

3. *For any $v \in \mathbb{C}^E$ satisfying $v \perp w^{(\mathrm{out})}$ for all $w \in \mathbb{C}^V$,*

$$v_{xy} = -A_{yx}v_{yx} \qquad \text{for every edge } xy. \tag{20}$$

4. *The operator norm of B is*

$$\|B\|_{\mathrm{op}} = d - 1. \tag{21}$$

Proof. First, for Equation (18):

1. We compute $\langle w^{(\mathrm{in})}, (w')^{(\mathrm{in})}\rangle$:

$$\langle w^{(\mathrm{in})}, (w')^{(\mathrm{in})}\rangle = \sum_{xy\in E} w_y w'_y = d\cdot\sum_{y\in V} w_y w'_y = d\cdot\langle w, w'\rangle.$$

2. We compute $\langle w^{(\text{out})}, (w')^{(\text{out})} \rangle$:

$$\begin{aligned}\langle w^{(\text{out})}, (w')^{(\text{out})} \rangle &= \sum_{xy \in E} (A_{xy})^2 w_x w'_x \\ &= d \sum_{x \in V} w_x w'_x && (A_{xy} \in \{\pm 1\} \text{ for any edge } xy) \\ &= d \cdot \langle w, w' \rangle.\end{aligned}$$

3. We compute $\langle w^{(\text{in})}, (w')^{(\text{out})} \rangle$:

$$\begin{aligned}\langle w^{(\text{in})}, (w')^{(\text{out})} \rangle &= \sum_{xy \in E} A_{xy} w_y w'_x = \sum_{x \in V} w'_x \cdot \sum_{y : xy \in E} A_{xy} w_y \\ &= \sum_{x \in V} w'_x (Aw)_x = \langle Aw, w' \rangle.\end{aligned}$$

Next, we verify Equation (19). Consider any $w \in \mathbb{C}^V$.

1. We analyze $Bw^{(\text{in})}$. For any $xy \in E$,

$$\begin{aligned}(Bw^{(\text{in})})_{xy} &= \sum_{z : yz \in E, z \neq x} A_{yz} \cdot (w^{(\text{in})})_{yz} \\ &= \sum_{z : yz \in E, z \neq x} A_{yz} w_z \\ &= \sum_{z : yz \in E} A_{yz} w_z - A_{yx} w_x \\ &= (Aw)_y - A_{xy} \cdot w_x && (A_{yx} = A_{xy})\end{aligned}$$

Therefore, $Bw^{(\text{in})} = (Aw)^{(\text{in})} - w^{(\text{out})}$.

2. We analyze $Bw^{(\text{out})}$. For any $xy \in E$,

$$\begin{aligned}(Bw^{(\text{out})})_{xy} &= \sum_{z : yz \in E, z \neq x} A_{yz} \cdot (w^{(\text{out})})_{yz} \\ &= \sum_{z : yz \in E, z \neq x} (A_{yz})^2 (w)_y \\ &= (d-1) \cdot w_y && ((A_{yz})^2 = 1 \text{ for any } yz \in E.)\end{aligned}$$

Therefore, $Bw^{(\text{out})} = (d-1) w^{(\text{in})}$.

Next, we verify Equation (20). Choose any $v \in \mathbb{C}^E$ satisfying $v \perp w^{(\text{out})}$ for all $w \in \mathbb{C}^V$. In particular, v is orthogonal to $(e_x)^{(\text{out})}$ for every $x \in V$, where e_x is the vector that has weight 1 on vertex x and weight 0 everywhere else. This implies that, for all $x \in V$,

$$\sum_{y : xy \in E} v_{xy} A_{xy} = \langle v, (e_x)^{(\text{out})} \rangle = 0.$$

We compute $(Bv)_{xy}$ for any edge xy.

$$\begin{aligned}(Bv)_{xy} &= \sum_{z : yz \in E, z \neq x} A_{yz} v_{yz} = \sum_{z : yz \in E} A_{yz} v_{yz} - A_{yx} v_{yx} \\ &= -A_{yx} v_{yx}. && (v \perp (e_y)^{(\text{out})})\end{aligned}$$

Finally, we prove Equation (21). In the proof of Proposition 48, we showed that if $v' \in \mathbb{C}^E$ is orthogonal to $w^{(\text{out})}$ for all $w \in \mathbb{C}^V$, then Bv' is orthogonal to $w^{(\text{in})}$ for all $w \in \mathbb{C}^v$. Furthermore, in the proof of Proposition 51, we showed that under the same condition, $\|Bv'\|_2 = \|v'\|_2$.

Now, consider any $v \in \mathbb{C}^E$. We can decompose it into

$$v = w^{(\text{out})} + v'$$

for some $w \in \mathbb{C}^V$ and v' that is orthogonal to $(w')^{(\text{out})}$ for all $w' \in \mathbb{C}^V$. Then, we have that

$$\begin{aligned}
\|Bv\|_2 &= \|(d-1)w^{(\text{in})} + Bv'\|_2 \\
&= \sqrt{(d-1)^2\|w^{(\text{in})}\|_2^2 + \|Bv'\|_2^2} && (Bv' \text{ is orthogonal to } w^{(\text{in})}) \\
&= \sqrt{(d-1)^2\|w^{(\text{out})}\|_2^2 + \|v'\|_2^2} \\
&= \sqrt{(d-1)^2\|v\|_2^2 - ((d-1)^2-1)\|v'\|_2^2} && (\|v\|_2^2 = \|w^{(\text{out})}\|_2^2 + \|v'\|_2^2) \\
&\leq \sqrt{(d-1)^2\|v\|_2^2} \\
&= d-1.
\end{aligned}$$

Hence, $\|Bv\|_2 \leq (d-1)\|v\|_2$, and with equality whenever $v = w^{(\text{out})}$ for some $w \in \mathbb{C}^V$, proving that $\|B\|_{\text{op}} = d-1$. ◀

Certifying Solution Geometry in Random CSPs: Counts, Clusters and Balance

Jun-Ting Hsieh
Carnegie Mellon University, Pittsburgh, PA, USA

Sidhanth Mohanty
University of California at Berkeley, CA, USA

Jeff Xu
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

An active topic in the study of random constraint satisfaction problems (CSPs) is the geometry of the space of satisfying or almost satisfying assignments as the function of the density, for which a precise landscape of predictions has been made via statistical physics-based heuristics. In parallel, there has been a recent flurry of work on *refuting* random constraint satisfaction problems, via nailing refutation thresholds for spectral and semidefinite programming-based algorithms, and also on *counting* solutions to CSPs. Inspired by this, the starting point for our work is the following question: *What does the solution space for a random CSP look like to an efficient algorithm?*

In pursuit of this inquiry, we focus on the following problems about random Boolean CSPs at the densities where they are unsatisfiable but no refutation algorithm is known.

1. **Counts.** For every Boolean CSP we give algorithms that with high probability certify a subexponential upper bound on the number of solutions. We also give algorithms to certify a bound on the number of large cuts in a Gaussian-weighted graph, and the number of large independent sets in a random d-regular graph.
2. **Clusters.** For Boolean 3CSPs we give algorithms that with high probability certify an upper bound on the number of *clusters* of solutions.
3. **Balance.** We also give algorithms that with high probability certify that there are no "unbalanced" solutions, i.e., solutions where the fraction of +1s deviates significantly from 50%.

Finally, we also provide hardness evidence suggesting that our algorithms for counting are optimal.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases constraint satisfaction problems, certified counting, random graphs

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.11

Related Version *Full Version*: https://arxiv.org/abs/2106.12710

Funding *Jun-Ting Hsieh*: Supported by NSF CAREER Award #2047933.
Sidhanth Mohanty: Supported by Google PhD Fellowship.
Jeff Xu: Supported by NSF CAREER Award #2047933.

Acknowledgements We would like to thank Pravesh Kothari, Prasad Raghavendra, and Tselil Schramm for their encouragement and thorough feedback on an earlier draft. We would also like to thank Tselil Schramm for enlightening discussions on refuting random CSPs.

1 Introduction

Constraint satisfaction problems (CSPs) are fundamental in the study of algorithm design and complexity theory. They are simultaneously simple and also richly expressive in capturing a wide range of computational tasks, which has led to fruitful connections to other areas of theoretical computer science (see, for example, [33, 8] for connections to cryptography, [21] for

applications to hardness of learning, and [25] for applications to average-case hardness). Hence, understanding them has received intense attention in the past few decades, leading to several comprehensive theories of their complexity. Some of the highlights include: the Dichotomy Theorem, which characterizes the worst-case complexity of satisfiability of CSPs via their algebraic properties [69, 14, 78], inapproximability results via the PCP Theorem [35], and the theory of optimal inapproximability based on connections between semidefinite programming and the Unique Games conjecture [41, 42, 65].

In this work, we are interested in the algorithmic aspects of random instances of CSPs. There has been a diverse array of phenomena about random CSPs illustrated in recent work, of dramatically varying nature depending on the ratio of the number of constraints to the number of variables, known as the *density*. Of central importance is the *satisfiability threshold*, which marks a phase transition where a random CSP instance shifts from being likely satisfiable to being likely unsatisfiable. When the density is well below the satisfiability threshold, there are several algorithms for tasks such as counting and sampling assignments to a random CSP instance [55, 39, 30, 11], whereas well above this threshold there are efficient algorithms for *certifying* that random CSPs are unsatisfiable [5]. The densities in the interim hold mysteries that we don't yet fully understand, and this work is an effort to understand the algorithmic terrain there. To make matters concrete, for now we will specialize the discussion of the problem setup and our work to the canonical 3SAT predicate.

Consider a random 3SAT formula $\boldsymbol{\mathcal{I}}$ on n variables and Δn clauses where each clause is sampled uniformly, independently, and adorned with uniformly random negations. Once the density Δ is a large enough constant, this random instance is unsatisfiable with high probability.[1] On the other hand, the widely believed Feige's random 3SAT hypothesis [25] conjectures that when Δ is any constant, there is no algorithm to *certify* that a random instance is unsatisfiable. Further, the best known algorithms for efficiently certifying that it is unsatisfiable require $\Delta \gtrsim \sqrt{n}$ [32, 16, 27, 5]. Moreover, when $\Delta \lesssim \sqrt{n}$ there is a lower bound against the Sum-of-Squares hierarchy [34, 70] (known to capture many algorithmic techniques), which suggests an *information-computation gap* and earns $\sqrt{n}$ the name *refutation threshold*.

In this picture, at both densities $n^{.25}$ and $n^{.35}$, $\boldsymbol{\mathcal{I}}$ is likely unsatisfiable but "looks" satisfiable to an efficient algorithm. But is there a concrete sense in which a random formula at density $n^{.25}$ is "more satisfiable" than one at density $n^{.35}$ from the lens of a polynomial-time algorithm? A natural measure of a 3SAT formula's satisfiability is its number of satisfying assignments, which motivates the following question.

> *What is the best efficiently certifiable upper bound on the number of assignments satisfying $\boldsymbol{\mathcal{I}}$?*

Our work provides an extensive study in this open direction.

In the context of 3SAT, our work proves:

▶ **Theorem 1** (Informal). *There is an efficient algorithm to certify with high probability that a random* 3SAT *formula with density $\Delta = n^{1/2-\delta}$ has at most $\exp(\widetilde{O}(n^{3/4+\delta/2}))$ satisfying assignments.*

In addition to certifying the number of satisfying assignments, we can certify that the solutions form clusters and upper bound the number of clusters under the refutation threshold.

[1] In fact, it is conjectured that there is a sharp threshold for unsatisfiability once Δ crosses some constant $\alpha_{\text{SAT}} \approx 4.267$.

Clusters. Besides the satisfiability threshold, random kSAT is conjectured to go through other phase transitions too, as predicted in the work of [45]. In particular, the *clustering threshold* is the density where the solution space is predicted to change from having one giant component to roughly resembling a union of several small Hamming balls, known as *clusters*, that are pairwise far apart in Hamming distance.

Much like the refutation threshold that marks where efficient algorithms can witness unsatisfiability, it is natural to ask if there is some regime under the refutation threshold where an efficient algorithm can witness a bound on the number of clusters of solutions. The following more nuanced version of Theorem 1 gives an answer to this question.

▶ **Theorem 2** (Informal). *There is an efficient algorithm to certify with high probability that the satisfying assignments of a random* 3SAT *formula with density* $\Delta = n^{1/2-\delta}$ *are covered by at most* $\exp(\widetilde{O}(n^{1/2+\delta}))$ *diameter-*$\widetilde{O}(n^{3/4+\delta/2})$ *clusters.*

Balance in the solution space. Suppose at density Δ, a typical 3SAT formula has $\sim \exp(c_\Delta n)$ satisfying assignments, then due to the uniformly random negations in clauses, each string is satisfying with probability $\sim \exp((c_\Delta - 1)n)$. Then one can show via the first moment method that with high probability there are no satisfying assignments with Hamming weight outside $\left[\frac{1}{2} - f(c_\Delta), \frac{1}{2} + f(c_\Delta)\right]$.[2] In particular, the intersection of the solution space with the set of unbalanced strings empties out under the satisfiability threshold. This raises the question:

> *Is there an efficient algorithm to certify that a random CSP instance has no unbalanced assignments at density significantly under the refutation threshold?*

We affirmatively answer this question and in the special case of 3SAT prove:

▶ **Theorem 3** (Informal). *There is an efficient algorithm to certify with high probability that a random* 3SAT *formula with density* $\Delta = n^{1/2-\delta}$ *has no satisfying assignments with Hamming weight outside*

$$\left[\frac{1}{2} - \widetilde{\Theta}\left(\frac{1}{n^{1/4-\delta/2}}\right), \frac{1}{2} + \widetilde{\Theta}\left(\frac{1}{n^{1/4-\delta/2}}\right)\right].$$

We illustrate our upper bounds for counting satisfying assignments and clusters in Figure 1. We delve into the precise technical statements of our results and the techniques involved in proving them in Section 1.1. Then to put our work in context, we survey and discuss existing work on information-computation gaps, and algorithmic work on counting, sampling and estimating partition functions in Section 1.2.

1.1 Our Contributions

In this section, we give a more detailed technical description of our contributions. To set the stage for doing so, we first formally clarify the notion of *certification* and some preliminaries on constraint satisfaction problems.

Fix a sample space Ω, a probability distribution $\mathcal{D}$ over Ω, and a function $f : \Omega \to \mathbb{R}$. For example, Ω is the space of 3SAT instances, $\mathcal{D}$ is the distribution of instances given by the random 3SAT model, and f is the number of satisfying assignments.

[2] where f is chosen so that the number of strings outside that Hamming range is $\ll \exp((c_\Delta - 1)n)$.

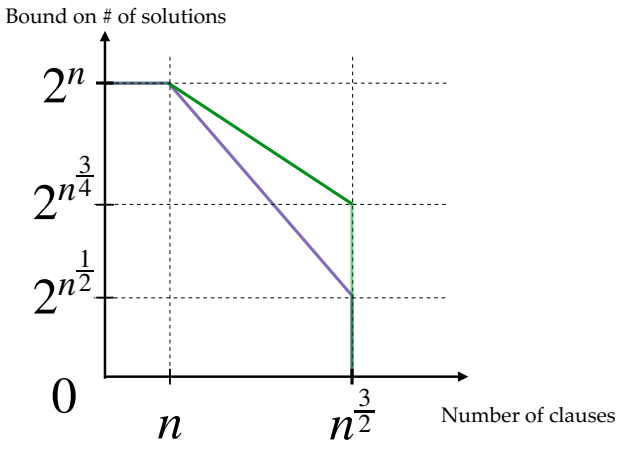


Figure 1 Our results for 3SAT. **Green**: certifiable upper bound on the number of satisfying assignments. **Purple**: upper bound on the number of clusters of satisfying assignments. In the case of kSAT, the green plot looks identical but with n replaced by $n^{(k-1)/2}$ and $n^{3/2}$ replaced by $n^{k/2}$.

▶ **Definition 4.** *We say that a deterministic algorithm $\mathcal{A}$ certifies that $f \leqslant C$ with probability over $1-p$ over $\mathcal{D}$ if $\mathcal{A}$ satisfies*

1. *For all $\omega \in \Omega$, $f(\omega) \leqslant \mathcal{A}(\omega)$.*
2. *For a random sample $\boldsymbol{\omega} \sim \mathcal{D}$, $\mathcal{A}(\boldsymbol{\omega}) \leqslant C$ with probability over $1-p$.*

We emphasize that an algorithm that always outputs the typical value of f is *not* a certification algorithm: it will satisfy the second condition but not the first. Thus, in several average-case problems, there are gaps between the typical value and the best known certifiable upper bound.

▶ Remark 5. Due to the guarantees of $\mathcal{A}$, one can think of the "transcript" of the algorithm on input ω as being a proof that $f(\omega) \leqslant \mathcal{A}(\omega)$.

▶ **Definition 6.** *A predicate $P : \{\pm 1\}^k \to \{0,1\}$ is any Boolean function that is not a constant function. An instance $\mathcal{I}$ of a constraint satisfaction problem on predicate P and vertex set $[n]$ is a collection of clauses, where a clause is a pair (c,S) with $c \in \{\pm 1\}^k$ and $S \in [n]^k$. Given $x \in \{\pm 1\}^n$, the value of $\mathcal{I}$ on x is:*

$$\mathcal{I}(x) \coloneqq \frac{1}{|\mathcal{I}|} \sum_{(c,S)\in\mathcal{I}} P(c_1 x_{S_1}, \ldots, c_k x_{S_k}).$$

We say x satisfies a clause (c,S) if $P(c_1 x_{S_1}, \ldots, c_k x_{S_k}) = 1$, and say x is $(1-\eta)$-satisfying if $\mathcal{I}(x) \geqslant 1-\eta$. If $\eta = 0$, we say x is exactly satisfying.

▶ Remark 7. An important predicate instrumental in all our results is the kXOR predicate, defined as follows:

$$k\mathsf{XOR}(x_1, \ldots, x_k) = \prod_{i=1}^{k} x_i.$$

In this work we are concerned with random CSPs. We defer an exact description of the random model to Section 2.3 of the full version (note however that the common random models used in the literature are all qualitatively similar; cf. [5, Appendix D]). Our first result is an algorithm certifying a subexponential upper bound on the number of $(1-\eta)$-satisfying assignments for random CSPs.

▶ **Theorem 8.** *Let $\boldsymbol{\mathcal{I}}$ be a random kCSP instance on any predicate P on n variables and Δn clauses. For every $\varepsilon > 0$ and $\eta \in [0, \eta_0]$ for some constant $\eta_0 > 0$, there is an algorithm that certifies with high probability that the number of $(1-\eta)$-satisfying assignments to $\boldsymbol{\mathcal{I}}$ is upper bounded by:*[3]

$$\exp\left(O\left(\eta \log \frac{1}{\eta}\right) n\right) \cdot \exp\left(\widetilde{O}\left(\sqrt{\frac{n^{(k+1)/2}}{\Delta}}\right)\right) \cdot \exp\left(O\left(\frac{n^{1+\varepsilon}}{\Delta^{1/(k-2)}}\right)\right).$$

To more easily parse the statement, let's plug in concrete parameters.

▶ Remark 9. Let's fix the predicate to be kSAT for any $k \geqslant 3$, $\eta = 0$, and $\Delta = n^{k/2-1.1}$. The quantity of interest is the number of exactly satisfying solutions to a random kSAT formula at a density strictly smaller than the refutation threshold of $\widetilde{\Omega}(n^{k/2-1})$. Then, we get an algorithm that with high probability certifies that the number of exactly satisfying assignments is at most: $\exp(\widetilde{O}(n^{0.8}))$, which is a subexponential bound. More generally, our algorithms certify a subexponential bound on the number of satisfying assignments for kSAT for $\Delta = n^{k/2-1.5+c}$ for any $c > 0$ and this bound improves as we increase c.

The proof of Theorem 8 relies on 3 ingredients of increasing complexity. The first is the simple observation that given a kCSP instance $\mathcal{I}$ on any predicate P, there is a transformation to a kSAT instance $\mathcal{I}'$ such that:

(i) For any $\eta > 0$, if x is $(1-\eta)$-satisfying for $\mathcal{I}$, then it is also $(1-\eta)$-satisfying for $\mathcal{I}'$.

(ii) If $\boldsymbol{\mathcal{I}}$ is a random instance of a CSP on P with density Δ, then $\boldsymbol{\mathcal{I}}'$ is a random instance of kSAT with density Δ.

This reduction is described in the proof of Corollary 4.8 of the full version.

The second ingredient is a generalization of the "3XOR-principle" of [25, 27], which we call the "kXOR-principle". The kXOR principle, which we state below, reduces count certification/refutation for a random kSAT formula to the same task on a random kXOR formula.

▶ **Lemma 10.** *Let $\boldsymbol{\mathcal{I}}$ be a random kSAT formula on $m = \Delta n$ clauses. There is an efficient algorithm that with high probability certifies that any $(1-\eta)$-satisfying assignment of $\boldsymbol{\mathcal{I}}$ must kXOR-satisfy at least $\left(1 - O(\eta) - \widetilde{O}\left(\sqrt{\frac{n^{(k-3)/2}}{\Delta}}\right)\right) m$ clauses.*

We detail the proof in Section 3 of the full version, which is close to the reduction from generic CSP refutation to kXOR refutation in [5] based on the Fourier expansion.

For the sake of a notationally simple sketch, let's restrict ourselves to the case $\eta = 0$. We can write $k\mathsf{SAT}(x_1, \ldots, x_k) = (1 - 2^{-k}) + 2^{-k} x_1 x_2 \cdots x_k + q(x_1, \ldots, x_k)$ where q is a degree-$(k-1)$ polynomial without a constant term. Thus, given a random kSAT instance $\boldsymbol{\mathcal{I}}$ and any satisfying assignment x:

$$1 = \boldsymbol{\mathcal{I}}(x) = 1 - 2^{-k} + 2^{-k} \frac{1}{|\boldsymbol{\mathcal{I}}|} \sum_{(c,S) \in \boldsymbol{\mathcal{I}}} \prod_{i=1}^{k} c_i x_{S_i} + \frac{1}{|\boldsymbol{\mathcal{I}}|} \sum_{c,S \in \boldsymbol{\mathcal{I}}} q(c_1 x_{S_1}, \ldots, c_k x_{S_k}).$$

Once $\Delta \gtrsim n^{(k-3)/2}$ the refutation algorithm of [5] can be employed to certify that the last term is insignificantly small by virtue of the last term being a degree-$(k-1)$ polynomial with no constant term. This would force $2^{-k} \frac{1}{|\boldsymbol{\mathcal{I}}|} \sum_{(c,S) \in \boldsymbol{\mathcal{I}}} \prod_{i=1}^{k} c_i x_{S_i}$ to be near 1, which is the same as saying x must kXOR-satisfy most clauses.

[3] We use the convention that $\eta \log \frac{1}{\eta} = 0$ when $\eta = 0$.

Our third ingredient for Theorem 8 is a count certification algorithm for kXOR, which we prove in Section 4 of the full version.

▶ **Theorem 11.** *For constant $k \geqslant 3$, consider a random kXOR instance with n variables and Δn clauses. For any constant $\varepsilon > 0$, there is a polynomial-time algorithm that certifies with high probability that the number of $(1-\eta)$-satisfying assignments is at most*

$$\exp\left(O\left(\eta \log \frac{1}{\eta}\right) n\right) \cdot \exp\left(O\left(\frac{n^{1+\varepsilon}}{\Delta^{1/(k-2)}}\right)\right).$$

In fact, the certification algorithm only depends on the hypergraph structure of the kXOR instance and not the signings of each clause. This is crucial since our algorithm recursively looks at $(k-1)$XOR subinstances with unknown signings. The stronger statement we prove is:

▶ **Theorem 12.** *For constant $k \geqslant 2$, consider a random k-uniform hypergraph $\boldsymbol{H}$ on n vertices and Δn hyperedges where $\Delta \gg \log n$. For $\varepsilon > 0$, there is a polynomial-time algorithm that certifies with high probability that the number of $(1-\eta)$-satisfying assignments to any kXOR instance on $\boldsymbol{H}$ is at most*

$$\exp\left(O\left(\eta \log \frac{1}{\eta}\right) n\right) \cdot \begin{cases} 1 & \text{if } k = 2 \\ \exp\left(\widetilde{O}\left(\frac{n^{1+\varepsilon}}{\Delta^{1/(k-2)}}\right)\right) & \text{if } k \geqslant 3. \end{cases}$$

Theorem 12 is of interest beyond algorithmic considerations as it gives a high-probability bound on the number of approximate solutions for *any* kXOR formula on a random hypergraph.

▶ Remark 13. Gaussian elimination is able to count exact solutions to an explicit kXOR instance but fails for counting $(1-\eta)$-satisfying assignments or when the signings are unknown.

We now give a brief sketch of our proof of Theorem 12. Given a random k-uniform hypergraph, we would like to certify that *any* kXOR instance on this hypergraph has no more than $\exp\left(O(\eta \log \frac{1}{\eta})n\right) \cdot \exp\left(\widetilde{O}\left(\frac{n^{1+\varepsilon}}{\Delta^{1/(k-2)}}\right)\right)$ approximate solutions. We will first present an overview in the context of 2XOR as the "base case", and then explain the algorithm for 3XOR to illustrate the "recursive step".

2XOR sketch. Let's consider a random graph $\boldsymbol{G}$ on n vertices and Δn edges where $\Delta \gg \log n$. Then, its degrees concentrate and its normalized Laplacian has a large spectral gap (more precisely, a spectral gap of $1 - O\left(\frac{1}{\sqrt{\Delta}}\right)$). As a consequence of Cheeger's inequality, any set S containing fewer than half the vertices has roughly half its edges leaving – which quantitatively would be around $\Delta|S|$. We prove that a large spectral gap and concentration of degrees is all that is necessary for any 2XOR instance to have an appropriately bounded number of satisfying assignments.

Now let $\mathcal{I}$ be any 2XOR instance on $\boldsymbol{G}$. The key observation is that if x and x' are two $(1-\eta)$-satisfying assignments for $\mathcal{I}$, then the pointwise product $y \coloneqq x \circ x'$ is $(1-2\eta)$-satisfying for $\mathcal{I}_+$, the 2XOR instance on $\boldsymbol{G}$ obtained by setting the sign on all constraints to $+1$. The constraints violated by y are the ones on the cut between S_+ and S_-, the positive and negative vertices in y respectively. There are roughly $\Delta \cdot \min\{|S_+|, |S_-|\}$, and consequently $\min\{|S_+|, |S_-|\} \leqslant 2\eta n$ since y is $(1-2\eta)$-satisfying. In particular, y either has at most $2\eta n$ positive entries or $2\eta n$ negative entries. The upshot is the number of $(1-\eta)$-satisfying assignments of $\mathcal{I}$ is at most $\exp(O(\eta \log \frac{1}{\eta})n))$. This sketched argument is carefully carried out in Section 4.1 of the full version.

3XOR sketch. Now, let $\boldsymbol{H}$ be a random hypergraph on n vertices and Δn hyperedges. The observation here is that for any 3XOR instance $\mathcal{I}$ on $\boldsymbol{H}$, any assignment x that $(1-\eta)$-satisfies $\mathcal{I}$ also approximately satisfies a particular *induced* 2XOR *instance* of a fixed subset of variables S. The induced 2XOR instance's underlying graph G is fixed and distributed like a random graph, and only the signings on the edges vary as we vary x. That lets us run the algorithm for 2XOR on G to obtain an upper bound F on all induced instances on G, which then yields a bound of $2^{|S|} \cdot F$. This is where we use that our algorithm depends only on the underlying graph, hence avoiding an enumeration of all assignments to variables in S.

We immediately see that for a fixed subset S, the above procedure throws away most of the clauses (keeping only clauses that have 1 variable in S). Thus, it is clearly suboptimal to look at just one subset S. To resolve this, we partition the variables into subsets $S_1, \ldots, S_\ell$, run the algorithm on each of them, and aggregate the results. This is explained in detail in the proofs of Lemma 4.6 and Theorem 4.4 of the full version.

Clustering. Our next result is an algorithm to upper bound the number of clusters formed by the solutions. Given $x \in \{\pm 1\}^n$, we call the Hamming ball $B(x, r)$ a *radius-r cluster* or *diameter-$2r$ cluster*. For 3CSPs we prove in Corollary 5.2 of the full version:

▶ **Theorem 14.** *Let P be any 3-ary predicate, and let $\boldsymbol{\mathcal{I}}$ be a random instance of P on n variables and Δn clauses. Let $\eta \in [0, \eta_0]$ where η_0 is a universal constant, and let $\theta \coloneqq 8\eta + O\left(\sqrt{\frac{\log^5 n}{\Delta}}\right)$. There is an algorithm that certifies with high probability that the $(1-\eta)$-satisfying assignments to $\boldsymbol{\mathcal{I}}$ as a P-CSP instance are covered by at most*

$$\exp(O(\theta^2 \log(1/\theta))n)$$

diameter-(θn) clusters.

Inspecting the proof of counting 2XOR (specifically the argument about $\mathcal{I}_+$), we see that it additionally certifies that the approximate solutions form clusters. In a similar fashion, we certify that any pair of $(1-\eta)$-satisfying assignments to a random 3SAT instance must have Hamming distance close to 0 or roughly $\frac{n}{2}$, i.e. the solutions form clusters where the clusters are roughly $\frac{n}{2}$ apart. The main ingredient is an efficient algorithm to certify an important structural result of random 3-uniform hypergraphs, allowing us to reason about the constraints violated in $\mathcal{I}_+$. In fact, this ingredient will also be a crucial step in refuting CSPs under global cardinality constraints in Section 6 of the full version. The upshot is that we will be able to certify that any pair of solutions is either ρ-close or $\frac{1-\rho}{2}$-far.

The second ingredient is a result in coding theory. Since the clusters are roughly $\frac{1\pm\rho}{2}n$ apart in Hamming distance, the number of clusters must be upper bounded by the cardinality of the largest ρ-balanced binary error-correcting code. The best known upper bound is $2^{O(\rho^2 \log(1/\rho))n}$ by [49] (see also [7]), which yields our final result. Complete details are in Section 5 of the full version.

Balance. We observe that the idea of hypergraph expansion can be applied to the problem of strongly refuting random CSPs with *global cardinality constraints*. This problem was first investigated by Kothari, O'Donnell, and Schramm [43], where they proved that under the refutation threshold $n^{k/2}$, the polynomial-time regime of the Sum-of-Squares hierarchy cannot refute the instance even with the global cardinality constraint $\sum_{i=1}^n x_i = B$ for any integer $B \in [-O(\sqrt{n}), O(\sqrt{n})]$ (here we assume $x \in \{\pm 1\}^n$). On the other hand, they proved once that $|B| > n^{3/4}$, Sum-of-Squares could indeed refute a random kXOR instance up to a factor of $\sqrt{n}$ under the refutation threshold.

We say an assignment x is ρ-biased if $\frac{1}{n}\left|\sum_{i\in[n]} x_i\right| \geqslant \rho$. We give a strong refutation algorithm for random instances of all Boolean CSPs under the constraint that the solution is "unbalanced".

▶ **Theorem 15.** *Let P be any k-variable predicate and let $\boldsymbol{\mathcal{I}}$ be a random CSP instance on $m \coloneqq n^{\frac{k-1}{2}+\beta}$ clauses where $\beta > 0$. For every constant $\rho > 0$, there is an efficient algorithm that certifies that $\boldsymbol{\mathcal{I}}$ has no 2ρ-biased assignment which $(1 - O(\rho^k))$-satisfies $\boldsymbol{\mathcal{I}}$ as a P-CSP instance.*

▶ Remark 16. Compared to [43], our result is a strong refutation algorithm for all CSPs, whereas their algorithm is specific for kXOR and only a weak refutation (refuting only exactly satisfying assignments). For $k = 3$, we match their cardinality constraint requirement. However, for $k \geqslant 4$, we require a slightly stronger cardinality assumption.

The formal statements and proofs are detailed in Section 6 of the full version. Akin to the case for counting solutions, we employ the reduction of every kCSP to kSAT and the kXOR principle to reduce the problem to strongly refuting kXOR under global cardinality constraints.

The first main insight is that given a graph G which is a sufficiently good spectral expander, we can efficiently certify that any 2XOR instance on G, where the number of positive constraints is roughly equal to the number of negative constraints, has no unbalanced approximately satisfying assignments. The proof of this is based on using the expander mixing lemma to show that any imbalanced assignment x must satisfy $x_u x_v = +1$ for $\gg \frac{1}{2}$ of the edges, which then lets us lower bound the number of negative constraints that are violated.

Then given a random kXOR instance $\boldsymbol{\mathcal{I}}$, we pick some set of ρn vertices S and consider all clauses with exactly $k-2$ vertices in S and 2 variables outside S. If we place an edge between the two variables outside S for every clause, we get some random graph $\boldsymbol{G}$. Now consider any assignment y to the variables in S. For this chosen set of clauses to be (nearly) satisfied, the assignment to variables outside S must nearly satisfy the induced 2XOR instance on the graph $\boldsymbol{G}$ whose signings are determined by y. The second insight is that we can efficiently certify that for any assignment y the induced 2XOR instance has a roughly equal number of positive and negative constraints. This is possible since the quantity $\#\text{positive constraints}(y) - \#\text{negative constraints}(y)$ is the objective value of a particular random $(k-2)$XOR instance on assignment y, which we can certify tight bounds on using the algorithm of [5].

Certified counting for subspace problems. So far, we have developed certification algorithms for CSPs mainly based on analyses of random hypergraphs. For other inherently different problems such as counting solutions to the SK model, we turn to a different technique. Our main insight is that for several problems, the approximate solutions must lie close to a small-dimensional linear subspace. Thus, we can reduce the problem to counting the number of (Boolean) vectors close to a subspace. We name this technique *dimension-based count certification* since the algorithms and their guarantees only depend on the dimension of the subspace.

▶ **Theorem 17.** *Let V be a linear subspace of dimension αn in $\mathbb{R}^n$. For any $\varepsilon \in (0, 1/4)$, the number of Boolean vectors in $\left\{\pm\frac{1}{\sqrt{n}}\right\}^n$ that are ε away from V is upper bounded by $2^{(H_2(4\varepsilon^2)+\alpha\log\frac{3}{\varepsilon})n}$.*

We note that the upper bound is almost tight.

We now give a brief overview of the proof of Theorem 17. First, we upper bound the maximum number of (normalized) Boolean vectors that can lie within any ε'-ball. Secondly, we take an ε-net of the unit ball in the subspace V (i.e. $B_1(0) \cap V$). We simply multiply the two quantities to get the upper bound, which only depends on the dimension of V.

Next, we apply this technique to two problems: the Sherrington-Kirkpatrick model and the independent sets in random d-regular graphs.

Sherrington-Kirkpatrick (SK). Given $\boldsymbol{M}$ sampled from $\mathsf{GOE}(n)$, the SK problem is to compute

$$\mathsf{OPT}(\boldsymbol{M}) = \max_{x \in \{\pm 1\}^n} x^\top \boldsymbol{M} x.$$

This problem can also be interpreted as finding the largest cut in a Gaussian-weighted graph. The SK model arises from the spin-glass model studied in statistical physics [71]. Talagrand [73] famously proved that $\mathsf{OPT}(\boldsymbol{M})$ concentrates around $2\mathsf{P}^* n^{3/2} \approx 1.526 n^{3/2}$, where P^* is the *Parisi constant*, first predicted by Parisi [61, 62].

Recently, the problem of certifying an upper bound for $\mathsf{OPT}(\boldsymbol{M})$ has received wide attention. A natural algorithm is the *spectral refutation*: $\mathsf{OPT}(\boldsymbol{M}) \leqslant n \cdot \lambda_{\max}(\boldsymbol{M})$. Since $\lambda_{\max}(\boldsymbol{M})$ concentrates around $2\sqrt{n}$, the algorithm certifies that $\mathsf{OPT}(\boldsymbol{M}) \leqslant (2 + o(1))n^{3/2}$, which we call the *spectral bound*. Clearly, there is a gap between the spectral bound and the true value, and it is natural to ask whether there is an algorithm that beats the spectral bound. Surprisingly, building on works by [58, 54, 46], Ghosh et. al. [31] showed that even the powerful Sum-of-Squares hierarchy cannot certify a bound better than $(2 - o(1))n^{3/2}$ in subexponential time. We also mention an intriguing work by Montanari [56] where he gave an efficient algorithm for the *search problem* – to find a solution with objective value close to $\mathsf{OPT}(\boldsymbol{M})$ with high probability (assuming a widely-believed conjecture from statistical physics). However, we emphasize that his algorithm is not a certification algorithm (recall Definition 4).

In the spirit of this work, a natural question is to certify an upper bound on the number of assignments $x \in \{\pm 1\}^n$ such that $x^\top \boldsymbol{M} x \geqslant 2(1-\eta)n^{3/2}$ for some $\eta > 0$.

▶ **Theorem 18.** *Let $\boldsymbol{M} \sim \mathsf{GOE}(n)$. Given $\eta \in (0, \eta_0)$ for some universal constant η_0, there is an algorithm certifying that at most $2^{O(\eta^{3/5} \log \frac{1}{\eta})n}$ assignments $x \in \{\pm 1\}^n$ satisfy $x^\top \boldsymbol{M} x \geqslant 2(1-\eta)n^{3/2}$.*

Our proof first looks at the eigenvalue distribution of $\boldsymbol{M}$, which follows the *semicircle law*. This shows that any x that achieves close to the spectral bound must be close to the top eigenspace of $\boldsymbol{M}$ (of dimension determined by the semicircle law). Then, we directly apply Theorem 17. See Section 7.1 of the full version for complete details.

Independent sets in d-regular graphs. The largest independent set size (the *independence number*) in a random d-regular graph has been studied extensively. It is well-known that with high probability, the independence number is $\leqslant \frac{2n \log d}{d}$ for a sufficiently large constant d (cf. [12, 76]). The current best known *certifiable* upper bound is via the smallest eigenvalue of the adjacency matrix (often referred to as Hoffman's bound, cf. [26, 13]): Let $\boldsymbol{A}$ be the adjacency matrix, and let $\lambda \coloneqq -\lambda_{\min}(\boldsymbol{A})$. Then, $|S| \leqslant \frac{\lambda}{d+\lambda} n$ for all independent sets S.

It is also well-established that $\lambda \leqslant 2\sqrt{d-1} + o(1)$ with high probability. Thus, we can certify that the independence number is at most $C_d n$ where $C_d \coloneqq \frac{2\sqrt{d-1}}{d+2\sqrt{d-1}}$.

The natural question for us is to certify an upper bound on the number of independent sets larger than $C_d(1-\eta)n$ for some $\eta > 0$.

▶ **Theorem 19.** *For a random d-regular graph on n vertices, given $\eta \in (0, \eta_0)$ for some universal constant η_0, there is an algorithm certifying that there are at most $2^{O(\eta^{3/5} \log \frac{1}{\eta})n}$ independent sets of size $C_d(1-\eta)n$.*

The proof is very similar to the SK model. We first map each independent set S to a vector $y_S \in \mathbb{R}^n$ such that if S is large, then y_S is close to the bottom eigenspace of $\boldsymbol{A}$. Then, using a variant of Theorem 17, we upper bound the number of such vectors that are close to the eigenspace. We carry out the proof in full detail in Section 7.2 of the full version.

Optimality for counting kCSP solutions. Finally, we give evidence suggesting that our algorithmic upper bounds are close to optimal. Our hardness results are built on the hypothesis that there is no efficient *strong* refutation algorithm for random kXOR under the refutation threshold (in the regime $n^{\varepsilon} \ll \Delta \ll n^{k/2-1}$). Although no NP-hardness results are known, this hypothesis is widely believed to be true. In particular, the problem was shown to be hard for the Sum-of-Squares semidefinite programming hierarchy [34, 70, 44], which is known to capture most algorithmic techniques for average-case problems. Thus, improving our results would imply a significant breakthrough.

We show that assuming this hypothesis is true, then we cannot certify an upper bound on the number of $(1-\eta)$-satisfying assignments better than $\exp(O(\eta n))$.

▶ **Theorem 20.** *If there is an efficient algorithm that with high probability can certify a bound of $\exp(\frac{\eta n}{10k})$ on the number of $(1-\eta)$-satisfying assignments to $\boldsymbol{\mathcal{I}}$, then there is an efficient algorithm that with high probability can certify that $\boldsymbol{\mathcal{I}}$ has no $(1-\eta/2)$-satisfying assignments.*

This shows that the term $\exp(O(\eta \log \frac{1}{\eta})n)$ in Theorem 8 and Theorem 11 is tight up to log factors. Our proof is simple: given a $(1-\eta/2)$-satisfying assignment and a small set S, we can flip the assignments to S arbitrarily and still be $(1-\eta)$-satisfying. Hence the number of $(1-\eta)$-satisfying assignments is at least $2^{|S|}$. Thus, an upper bound better than this would imply that there is no $(1-\eta/2)$-satisfying assignments. See Section 8.1 of the full version for complete details.

Surprisingly, the optimality of Theorem 8 suggests that there is a phase transition for certifiable counting at the refutation threshold. For concreteness, take random kSAT for example,

▶ Remark 21. At $m = \widetilde{\Omega}(n^{k/2})$, there is a strong refutation algorithm [5] which certifies that no $(1-\eta)$-satisfying assignment exists (even for constant $\eta < 1/2$). However, at $m = n^{k/2-\varepsilon}$ and take $\eta = n^{-\frac{1}{4}+\frac{\varepsilon}{2}}$, we can at best certify that the number of $(1-\eta)$-satisfying assignments is at most $\exp(O(n^{\frac{3}{4}+\frac{\varepsilon}{2}}))$. See also Figure 1 for illustration.

Optimality for counting independent sets. We also show barriers to improving Theorem 19, which can be viewed as a weak hardness evidence. Specifically, we show that improving the upper bound of Theorem 19 to $\exp(O(\eta \log(1/\eta)n))$ would imply beating Hoffman's bound by a factor of $1-\eta/2$ (for any small constant η), which would be an interesting algorithmic breakthrough.

▶ **Theorem 22.** *Let $\boldsymbol{G}$ be a random d-regular graph. Given constant $\eta \in (0, 1/2)$, if there is an efficient algorithm that with high probability certifies a bound of* $\exp\left(\frac{C_d}{4}\eta \log(1/\eta)n\right)$ *on the number of independent sets of size $C_d(1-\eta)n$, then there is an algorithm that with high probability certifies that $\boldsymbol{G}$ has no independent set of size $(1-\eta/2)C_d n$.*

The proof is a simple observation that for any independent set S, all subsets of S are also independent sets. Thus, if S is of size $(1-\eta/2)C_d n$, then we can lower bound the number of subsets of size $(1-\eta)C_d n$. We give a short proof in Section 8.2 of the full version. We note the interesting gap between $\eta^{3/5}$ and η in the exponent of the upper and lower bounds respectively, and we conjecture that there may be an algorithm matching the lower bound.

1.2 Context and related work

Information-computation gaps in CSPs. This work is very closely related to the line of work on information-computation gaps. In the context of certification in random CSPs, the most well-understood information-gaps are in that of refutation of random CSPs. Feige's random 3SAT hypothesis was one of the earliest conjectured gaps. As discussed earlier, while unsatisfiability for random 3SAT set in at constant density, it was conjectured by Feige that certifying this was hard at all constant densities. Further, integrality gaps for the Sum-of-Squares hierarchy of [34, 70] seem to point to hardness up to density $\sqrt{n}$. The wide information-computation gap is a main motivation for us to understand what an efficient algorithm can certify about the landscape of solutions in the regime between the satisfiability threshold and the refutation threshold. We refer the reader to the introduction of [5] for a comprehensive treatment of the literature on information-computation gaps for refuting random CSPs prior to their work, CSPs more broadly, as well as connections to other areas of theoretical computer science.

The situation for general constraint satisfaction problems beyond XOR and SAT was considered in the work of [5], which gave algorithms to refute all CSPs at density $n^{t/2-1}$ where t is the smallest integer such that there is no t-wise uniform distribution supported on the predicate's satisfying assignments. Then somewhat surprisingly, the work of [66] gave algorithms for refuting random CSPs between constant density and the $n^{t/2-1}$ threshold from [5], whose running time smoothly interpolated between exponential time at constant density to polynomial time at the [5] threshold, with a (steadily improving) subexponential running time in the intermediate regime. The algorithms of [5, 66] are spectral, and can be captured within the Sum-of-Squares hierarchy. Finally the work of [44] (presaged by [9]) established that the algorithm of [66] was tight for Sum-of-Squares in all regimes, thereby nailing a characterization for the exact gaps (up to logarithmic factors) for all random CSPs.

Solution geometry in random CSPs. One of the earlier predictions using nonrigorous physics techniques was the location of the 3SAT satisfiability threshold in the works of [53, 52]. In particular, they conjectured that there is a sharp threshold at a constant $\alpha_{\mathrm{SAT}} \approx 4.267$. These works put forth the "1-step replica symmetry breaking hypothesis" (a conjectured property of the solution space in random kSAT; we refer the reader to the introduction of [22] for a description), which was the starting point for several subsequent works. These techniques were used to precisely predict the kSAT satisfiability threshold for all values of k [50], proved for large k in a line of work culminating in [22] and building on [2, 3, 18, 19].

Eventually, the works of [45, 57] predicted that besides the satisfiability threshold, random kSAT goes through other phase transitions too, and gave conjectures for their locations. A notable one connected to this work is the *clustering threshold*, for which there has been

rigorous evidence given in the works of [51, 4, 1]. Above the clustering threshold, the solution space is predicted to break into exponentially many exponential-sized clusters far away from each other in Hamming distance. More precisely, there is some function Σ for which there are $\exp(\Sigma(s,\Delta)n)$ clusters of size approximately $\exp(sn)$ each. In particular, this leads to the prediction that the number of solutions at density Δ is roughly $\max_s\{\exp((s+\Sigma(s,\Delta))n)\}$. Another phase transition of interest is the *condensation threshold*, where the number of clusters of solutions drops to a constant.

Approximate Counting for CSPs. Approximate counting of solutions in CSPs has attracted much attention in recent years. There have been numerous positive algorithmic results for approximately counting solutions in (i) sparse CSPs in the worst case, (ii) sparse random CSPs well under the satisfiability threshold. The takeaway here is that even though the problems we consider get harder as we approach the satisfiability threshold, if one goes well under the threshold the algorithmic problems once again become tractable.

One exciting line of research for worst-case CSPs is the problem of approximately counting satisfying assignments of a $k\mathsf{SAT}$ formula under conditions similar to those of the Lovász Local Lemma (LLL) [24]. A direct application of the LLL shows that if the maximum degree D of the *dependency graph* is $\leqslant 2^k/e$, then the formula is satisfiable. Building on works of [55, 28, 29, 38], Jain, Pham, and Vuong [39] recently showed that there is an algorithm for approximate counting well under the LLL thresholds, i.e. when $D \lesssim 2^{k/5.741}$ (hiding factors polynomial in k), using techniques similar to an algorithmic version of the LLL. Further, the algorithms of [55, 38] are deterministic, which may suggest their techniques are amenable to obtaining certifiable counts. However, it was shown that the problem of approximately counting solutions to a $k\mathsf{SAT}$ formula is NP-hard when $D \gtrsim 2^{k/2}$ by [11], well in the sparse regime, which suggests a hard phase between the highly sparse setting and the dense setting we are concerned with.

For random $k\mathsf{SAT}$, the exact satisfiability threshold that was established by Ding, Sly, and Sun [22] takes on value $\alpha_{\mathrm{SAT}} = 2^k \ln 2 - \frac{1}{2}(1+\ln 2) + o_k(1)$. And similarly, well below the satisfiability threshold, Galanis, Goldberg, Guo, and Yang [30] adapted Moitra's techniques [55] to the random setting and developed a polynomial-time algorithm when the density $\Delta \leqslant 2^{k/301}$ and k sufficiently large.

Closely related to the counting problem is approximating the partition function of random $k\mathsf{SAT}$, for which there have also been positive algorithmic results. Specifically, given a random $k\mathsf{SAT}$ instance $\boldsymbol{\mathcal{I}}$, the partition function is defined as $Z(\boldsymbol{\mathcal{I}},\beta) \coloneqq \sum_\sigma e^{-\beta H(\sigma)}$, where $H(\sigma)$ is the number of unsatisfied clauses under assignment σ. The partition function can be viewed as a weighted (or "permissive") version of the counting problem. Montanari and Shah [59] first showed that the Belief Propagation algorithm approximately computes the partition function at $\Delta \sim \frac{2\log k}{k}$; their analysis is based on correlation decay (or the *Gibbs uniqueness property*). Recently, [17] further showed that Belief Propagation succeeds as long as the random $k\mathsf{SAT}$ model satisfies a *replica symmetry* condition, conjectured to hold up to $\Delta \sim 2^k \ln k/k$. See also the works of [45, 60, 15] for further details of this matter.

Counting independent sets and related problems. Another counting problem that has been the subject of active study is that of counting independent sets, especially in the statistical physics community. For a graph G with maximum degree d, let $\mathsf{IS}(G)$ be the set of independent sets in G. The task is to estimate the *independence polynomial* $Z_G(\lambda) = \sum_{I\in\mathsf{IS}(G)} \lambda^{|I|}$, also known as the partition function of the *hard-core model* with *fugacity* λ in the physics literature. Earlier works by [23, 74] developed randomized algorithms based on *Glauber dynamics* to

estimate $Z_G(\lambda)$ when $\lambda \leqslant \frac{2}{d-2}$. In a major breakthrough, Weitz [75] showed a deterministic algorithm, based on correlation decay, that approximates $Z_G(\lambda)$ when $0 \leqslant \lambda < \lambda_c$, where $\lambda_c \coloneqq \frac{(d-1)^{d-1}}{(d-2)^d}$. Sly and Sun [72] later proved that this is tight: no efficient approximate algorithm for $Z_G(\lambda)$ exists for $\lambda > \lambda_c$ unless $\mathsf{NP} = \mathsf{RP}$.

Recently, Barvinok initiated a line of research on estimating partition functions using the *interpolation method* (see Barvinok's recent book [10]). The main idea is to estimate the low-order Taylor approximation of $\log Z_G(\lambda)$ provided that the polynomial $Z_G(\lambda)$ does not vanish in some region in $\mathbb{C}$. This approach led to deterministic algorithms that match Weitz's result and work even for negative or complex λ's [63, 64]. These polynomial-based approaches were also used to obtain deterministic algorithms for counting colorings in bounded degree graphs [47], estimating the Ising model partition function [48], and algorithms for a counting version of the Unique Games problem [20].

There has also been works on worst-case upper bounds of $Z_G(\lambda)$ for d-regular graphs. Zhao proved that for any d-regular graph G and any $\lambda \geqslant 0$, $Z_G(\lambda) \leqslant (2(1+\lambda)^d - 1)^{n/2d}$ [77]. In particular, setting $\lambda = 1$, this shows that the total number of independent sets is bounded by $(2^{d+1} - 1)^{n/2d}$, settling a conjecture by Alon [6] and Kahn [40].

Certifying bounds on partition functions and free energy. A recent line of work [67, 68, 37] is focused on an approach based on a convex programming relaxation of entropy to certify upper bounds on the *free energy* of the Ising model (weighted 2XOR), both in the worst case and in the average case. While on the surface level, these approaches differ significantly from ours, an interesting direction is to investigate if these entropy-based convex programming relaxations can achieve our algorithmic results.

1.3 Open directions

In this section we suggest a couple of avenues for further investigation on the themes related to this work.

Worst-case complexity of certified counting. In this work, we deal mostly with random CSPs. Here we present a worst-case version of the problem, specialized to 3SAT. A classic result due to [35] is that it is NP-hard to distinguish between a $(7/8+\varepsilon)$-satisfiable 3SAT formula from a fully satisfiable 3SAT formula. However, it is unclear what the complexity of a version of this question is when there is a stronger promise on the satisfiable 3SAT formula.

▶ **Question 23.** Consider the following algorithmic task:

> Given a 3SAT formula $\mathcal{I}$ under the promise that it is either $(7/8+\varepsilon)$-satisfiable, or has at least T fully satisfying assignments, decide which of the two categories $\mathcal{I}$ falls into.

What is the complexity of the above problem?

We remark that this problem is similar to counting-3SAT, but subtly different.

Certifying optimal bounds on number of exactly satisfying kSAT solutions. In the context of kSAT, while our algorithms can certify subexponential bounds for both exactly satisfying assignments and approximately satisfying assignments, the matching evidence of hardness is only for the approximate version of the problem. Thus, it is still possible that there is an algorithm to certify an even tighter bound than ours for the problem of counting exactly satisfying assignments to a random kSAT formula. This motivates the following question:

▶ **Question 24.** What is the tightest bound an efficient algorithm can certify on the number of solutions to a random kSAT instance?

We conjecture that the algorithms presented in this paper are indeed optimal. An approach to providing hardness evidence for this is to construct a hard planted distribution, and prove it is hard within the *low-degree likelihood ratio* framework of [36]. We outline a possible approach in Section 8.3 of the full version to construct a planted distribution for readers interested in this problem.

Properties of arbitrary CSP instances on random hypergraphs. In the context of approximate kXOR, our certification algorithms for solution counts and cluster counts depend only on the hypergraph structure and not the random negations. Hence, they also prove nontrivial statements about the solution space of any XOR instance on a random hypergraph, which are potentially useful in the context of quiet planting or semi-random models of CSPs. However, our certification algorithms for other CSPs, such as kSAT, heavily make use of the random signings in the reduction to kXOR.

▶ **Question 25.** Can all the results related to certifying bounds on number of solutions/ clusters in this work for random kSAT instances be generalized to arbitrary kSAT instances on random hypergraphs?

References

1 Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 793–802. IEEE, 2008.

2 Dimitris Achlioptas and Cristopher Moore. The asymptotic order of the random k-SAT threshold. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 779–788. IEEE, 2002.

3 Dimitris Achlioptas and Yuval Peres. The threshold for random k-sat is $2^k(\ln 2 - o(k))$. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 223–231, 2003.

4 Dimitris Achlioptas and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 130–139, 2006.

5 Sarah R Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 689–708. IEEE, 2015.

6 Noga Alon. Independent sets in regular graphs and sum-free subsets of finite groups. *Israel journal of mathematics*, 73(2):247–256, 1991.

7 Noga Alon. Perturbed identity matrices have high rank: Proof and applications. *Combinatorics, Probability & Computing*, 18(1-2):3, 2009.

8 Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180, 2010.

9 Boaz Barak, Siu On Chan, and Pravesh K Kothari. Sum of squares lower bounds from pairwise independence. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 97–106, 2015.

10 Alexander Barvinok. *Combinatorics and complexity of partition functions*, volume 9. Springer, 2016.

11 Ivona Bezáková, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Daniel Stefankovic. Approximation via correlation decay when strong spatial mixing fails. *SIAM Journal on Computing*, 48(2):279–349, 2019.

12 Béla Bollobás. The independence ratio of regular graphs. *Proceedings of the American Mathematical Society*, pages 433–436, 1981.

13 Andries E Brouwer and Willem H Haemers. *Spectra of graphs*. Springer Science & Business Media, 2011.

14 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM journal on computing*, 34(3):720–742, 2005.

15 Amin Coja-Oghlan. Belief propagation guided decimation fails on random formulas. *Journal of the ACM (JACM)*, 63(6):1–55, 2017.

16 Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random k-SAT. *Combinatorics, Probability & Computing*, 16(1):5, 2007.

17 Amin Coja-Oghlan, Noëla Müller, and Jean B Ravelomanana. Belief Propagation on the random k-SAT model. *arXiv preprint*, 2020. `arXiv:2011.02303`.

18 Amin Coja-Oghlan and Konstantinos Panagiotou. Going after the k-SAT threshold. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 705–714, 2013.

19 Amin Coja-Oghlan and Konstantinos Panagiotou. The asymptotic k-SAT threshold. *Advances in Mathematics*, 288:985–1068, 2016.

20 Matthew Coulson, Ewan Davies, Alexandra Kolla, Viresh Patel, and Guus Regts. Statistical physics approaches to Unique Games. *arXiv preprint arXiv:1911.01504*, 2019.

21 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 441–448, 2014.

22 Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 59–68, 2015.

23 Martin Dyer and Catherine Greenhill. On Markov chains for independent sets. *Journal of Algorithms*, 35(1):17–49, 2000.

24 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Colloquia Mathematica Societatis Janos Bolyai 10. Infinite and Finite Sets, Keszthely (Hungary)*. Citeseer, 1973.

25 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543, 2002.

26 Uriel Feige and Eran Ofek. Spectral techniques applied to sparse random graphs. *Random Structures & Algorithms*, 27(2):251–275, 2005.

27 Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. *Theory of Computing*, 3(1):25–43, 2007.

28 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting k-SAT solutions in the local lemma regime. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 854–867, 2020.

29 Weiming Feng, Kun He, and Yitong Yin. Sampling Constraint Satisfaction Solutions in the Local Lemma Regime. *arXiv preprint*, 2020. `arXiv:2011.03915`.

30 Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Kuan Yang. Counting solutions to random CNF formulas. *arXiv preprint*, 2019. `arXiv:1911.07020`.

31 Mrinalkanti Ghosh, Fernando Granha Jeronimo, Chris Jones, Aaron Potechin, and Goutham Rajendran. Sum-of-squares lower bounds for Sherrington-Kirkpatrick via planted affine planes. *arXiv preprint*, 2020. `arXiv:2009.01874`.

32 Andreas Goerdt and André Lanka. Recognizing more random unsatisfiable 3-SAT instances efficiently. *Electronic Notes in Discrete Mathematics*, 16:21–46, 2003.

33 Oded Goldreich. Candidate one-way functions based on expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 76–87. Springer, 2011.

34 Dima Grigoriev. Complexity of Positivstellensatz proofs for the knapsack. *computational complexity*, 10(2):139–154, 2001.

35 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

36 Samuel B Hopkins and David Steurer. Efficient Bayesian estimation from few samples: community detection and related problems. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 379–390. IEEE, 2017.

37 Vishesh Jain, Frederic Koehler, and Andrej Risteski. Mean-field approximation, convex hierarchies, and the optimality of correlation rounding: a unified perspective. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1226–1236, 2019.

38 Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Towards the sampling Lovász Local Lemma. *arXiv preprint*, 2020. `arXiv:2011.12196`.

39 Vishesh Jain, Huy Tuan Pham, and Thuy-Duong Vuong. On the sampling Lovász Local Lemma for atomic constraint satisfaction problems. *arXiv preprint*, 2021. `arXiv:2102.08342`.

40 Jeff Kahn. An entropy approach to the hard-core model on bipartite graphs. *Combinatorics, Probability & Computing*, 10(3):219, 2001.

41 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

42 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.

43 Pravesh Kothari, Ryan O'Donnell, and Tselil Schramm. SOS lower bounds with hard constraints: think global, act local. *arXiv preprint*, 2018. `arXiv:1809.01207`.

44 Pravesh K Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 132–145, 2017.

45 Florent Krzakała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007.

46 Dmitriy Kunisky and Afonso S Bandeira. A tight degree 4 sum-of-squares lower bound for the Sherrington–Kirkpatrick Hamiltonian. *Mathematical Programming*, pages 1–39, 2020.

47 Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. A deterministic algorithm for counting colorings with 2Δ colors. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1380–1404. IEEE, 2019.

48 Jingcheng Liu, Alistair Sinclair, and Piyush Srivastava. The Ising partition function: Zeros and deterministic approximation. *Journal of Statistical Physics*, 174(2):287–315, 2019.

49 Robert McEliece, Eugene Rodemich, Howard Rumsey, and Lloyd Welch. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Transactions on Information Theory*, 23(2):157–166, 1977.

50 Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-sat from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

51 Marc Mézard, Thierry Mora, and Riccardo Zecchina. Clustering of solutions in the random satisfiability problem. *Physical Review Letters*, 94(19):197205, 2005.

52 Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

53 Marc Mézard and Riccardo Zecchina. Random k-satisfiability problem: From an analytic solution to an efficient algorithm. *Physical Review E*, 66(5):056126, 2002.

54 Sidhanth Mohanty, Prasad Raghavendra, and Jeff Xu. Lifting sum-of-squares lower bounds: degree-2 to degree-4. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 840–853, 2020.

55 Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *Journal of the ACM (JACM)*, 66(2):1–25, 2019.

56 Andrea Montanari. Optimization of the sherrington-kirkpatrick hamiltonian. In *Proceedings of the 60th IEEE Symposium on Foundations of Computer Science*, pages 1417–1433, 2019.

57 Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. Clusters of solutions and replica symmetry breaking in random k-satisfiability. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(04):P04004, 2008.

58 Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the 48th annual ACM Symposium on Theory of Computing*, pages 814–827, 2016.

59 Andrea Montanari and Devavrat Shah. Counting good truth assignments of random k-SAT formulae. *arXiv preprint*, 2006. `arXiv:cs/0607073`.

60 Dmitry Panchenko. Spin glass models from the point of view of spin distributions. *Annals of Probability*, 41(3A):1315–1361, 2013.

61 Giorgio Parisi. Infinite number of order parameters for spin-glasses. *Physical Review Letters*, 43(23):1754, 1979.

62 Giorgio Parisi. A sequence of approximated solutions to the SK model for spin glasses. *Journal of Physics A: Mathematical and General*, 13(4):L115, 1980.

63 Viresh Patel and Guus Regts. Deterministic polynomial-time approximation algorithms for partition functions and graph polynomials. *SIAM Journal on Computing*, 46(6):1893–1919, 2017.

64 Han Peters and Guus Regts. On a conjecture of Sokal concerning roots of the independence polynomial. *The Michigan Mathematical Journal*, 68(1):33–55, 2019.

65 Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254, 2008.

66 Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random CSPs below the spectral threshold. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 121–131, 2017.

67 Andrej Risteski. How to calculate partition functions using convex programming hierarchies: provable bounds for variational methods. In *Conference on Learning Theory*, pages 1402–1416. PMLR, 2016.

68 Andrej Risteski and Yuanzhi Li. Approximate maximum entropy principles via goemans-williamson with applications to provable variational methods. *Advances in Neural Information Processing Systems*, 29:4628–4636, 2016.

69 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

70 Grant Schoenebeck. Linear level Lasserre lower bounds for certain k-CSPs. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602. IEEE, 2008.

71 David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

72 Allan Sly and Nike Sun. The computational hardness of counting in two-spin models on d-regular graphs. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 361–369. IEEE, 2012.

73 Michel Talagrand. The Parisi formula. *Annals of mathematics*, pages 221–263, 2006.

74 Eric Vigoda. A note on the Glauber dynamics for sampling independent sets. *the electronic journal of combinatorics*, 8(1):R8, 2001.

75 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149, 2006.

76 Nicholas C Wormald. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.

77 Yufei Zhao. The number of independent sets in a regular graph. *Combinatorics, Probability and Computing*, 19(2):315–320, 2010.

78 Dmitriy Zhuk. A Proof of the CSP Dichotomy Conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020.

On Efficient Noncommutative Polynomial Factorization via Higman Linearization

Vikraman Arvind
Institute of Mathematical Sciences, Chennai, India

Pushkar S. Joglekar
Vishwakarma Institute of Technology, Pune, India

Abstract

In this paper we study the problem of efficiently factorizing polynomials in the free noncommutative ring $\mathbb{F}\langle x_1, x_2, \ldots, x_n\rangle$ of polynomials in noncommuting variables $x_1, x_2, \ldots, x_n$ over the field $\mathbb{F}$. We obtain the following result:

- We give a randomized algorithm that takes as input a noncommutative arithmetic formula of size s computing a noncommutative polynomial $f \in \mathbb{F}\langle x_1, x_2, \ldots, x_n\rangle$, where $\mathbb{F} = \mathbb{F}_q$ is a finite field, and in time polynomial in s, n and $\log_2 q$ computes a factorization of f as a product $f = f_1 f_2 \cdots f_r$, where each f_i is an irreducible polynomial that is output as a noncommutative algebraic branching program.
- The algorithm works by first transforming f into a linear matrix L using Higman's linearization of polynomials. We then factorize the linear matrix L and recover the factorization of f. We use basic elements from Cohn's theory of free ideals rings combined with Ronyai's randomized polynomial-time algorithm for computing invariant subspaces of a collection of matrices over finite fields.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Noncommutative Polynomials, Arithmetic Circuits, Factorization, Identity testing

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.12

Related Version *Full Version*: `https://arxiv.org/pdf/2202.09883.pdf`

Funding *Pushkar S. Joglekar*: Author would like to thank Science and Engineering Research Board for the funding through the MATRICS project, File no. MTR/2018/001214.

1 Introduction

Let $\mathbb{F}$ be any field and $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n free noncommuting variables. Let X^* denote the set of all free words (which are monomials) over the alphabet X with concatenation of words as the monoid operation and the empty word ϵ as identity element.

The *free noncommutative ring* $\mathbb{F}\langle X\rangle$ consists of all finite $\mathbb{F}$-linear combinations of monomials in X^*, where the ring addition $+$ is coefficient-wise addition and the ring multiplication $*$ is the usual convolution product. More precisely, let $f, g \in \mathbb{F}\langle X\rangle$ and let $f(m) \in \mathbb{F}$ denote the coefficient of monomial m in polynomial f. Then we can write $f = \sum_m f(m)m$ and $g = \sum_m g(m)m$, and in the product polynomial fg for each monomial m we have

$$fg(m) = \sum_{m_1 m_2 = m} f(m_1)g(m_2).$$

The *degree* of a monomial $m \in X^*$ is the length of the monomial m, and the degree $\deg f$ of a polynomial $f \in \mathbb{F}\langle X\rangle$ is the degree of a largest degree monomial in f with nonzero coefficient. For polynomials $f, g \in \mathbb{F}\langle X\rangle$ we clearly have $\deg(fg) = \deg f + \deg g$.

A *nontrivial factorization* of a polynomial $f \in \mathbb{F}\langle X\rangle$ is an expression of f as a product $f = gh$ of polynomials $g, h \in \mathbb{F}\langle X\rangle$ such that $\deg g > 0$ and $\deg h > 0$. A polynomial $f \in \mathbb{F}\langle X\rangle$ is *irreducible* if it has no nontrivial factorization and is *reducible* otherwise. For instance, all degree 1 polynomials in $\mathbb{F}\langle X\rangle$ are irreducible. Clearly, by repeated factorization every polynomial in $\mathbb{F}\langle X\rangle$ can be expressed as a product of irreducibles.

In this paper we study the algorithmic complexity of polynomial factorization in the free ring $\mathbb{F}\langle X\rangle$. The factorization algorithm is by an application of Higman's linearization process followed by factorization of a matrix with linear entries (under some technical conditions) using Cohn's factorization theory.

It is interesting to note that Higman's linearization process [15] has been used to obtain a deterministic polynomial-time algorithm for the RIT problem. That is, the problem of testing if a noncommutative rational formula (which computes an element of the free skew field $\mathbb{F}\langle\langle X\rangle\rangle$) is zero on its domain of definition [13, 17, 18, 16].

1.1 Overview of the results

The main result of the paper is the following.

▶ **Theorem** (Main Theorem). *Given a multivariate noncommutative polynomial $f \in \mathbb{F}_q\langle X\rangle$ for a finite field*[1] *$\mathbb{F}_q$ by a noncommutative arithmetic formula of size s as input, a factorization of f as a product $f = f_1 f_2 \cdots f_r$ can be computed in randomized time* $\mathrm{poly}(s, \log_2 q, |X|)$*, where each $f_i \in \mathbb{F}_q\langle X\rangle$ is an irreducible polynomial that is output as an algebraic branching program.*

The proof has three broad steps described below.

- **Higman linearization and Cohn's factorization theory.** Briefly, given a noncommutative polynomial $f \in \mathbb{F}\langle X\rangle$ by a formula, we can transform it into a linear matrix L such that $f \oplus I = PLQ$, where P is an upper triangular matrix with polynomial entries and all 1's diagonal and Q is a lower triangular matrix with polynomial entries and all 1's diagonal, P and Q are the matrices implementing the sequence of row and column operations required for the Higman linearization process. Cohn's theory of factorization of noncommutative linear matrices gives us sufficient information about the structure of irreducible linear matrices.

- **Ronyai's common invariant subspace algorithm.** Next, the most important tool algorithmically, is Ronyai's algorithm for computing common invariant subspaces of a collection of matrices over finite fields [24]. We show that Ronyai's common invariant subspace algorithm can be repeatedly applied to factorize a linear matrix $L = A_0 + \sum_{i=1}^{n} A_i x_i$, into a product of irreducible linear matrices provided A_0 is invertible and $[A_1 A_2 \cdots A_n]$ has full row rank or $[A_1^T A_2^T \cdots A_n^T]^T$ has full column rank. The later conditions are called as right and left monicity of the linear matrix L respectively. With some technical work we can ensure these conditions for a linear matrix L that is produced from a polynomial f by Higman linearization. Then Ronyai's algorithm yields the factorization of L into a product of irreducible linear matrices (upto multiplication by units).

[1] We present the detailed randomized algorithm over large finite fields. In the case of small finite fields we obtain a deterministic $\mathrm{poly}(s, q, |X|)$ time algorithm with minor modifications.

- **Recovering the factors of f.** Finally, we design a simple linear algebraic algorithm for trivializing a matrix product $AB = 0$, where A is a linear matrix and B is a column vector of polynomials from $\mathbb{F}\langle X\rangle$, using which we are able to extract the irreducible factors of f from the factors of L. An invertible matrix M with polynomial entries *trivializes* the relation $AB = 0$ if the modified relation $(AM)(M^{-1}B) = 0$ has the property that for every index i either the i^{th} column of AM is zero or the i^{th} row of $M^{-1}B$ is zero. While such matrices M exist for any matrix product $AB = 0$ with entries from $\mathbb{F}\langle X\rangle$, we obtain an efficient algorithm in the special case when A is linear and B's entries are polynomials computed by small arithmetic circuits. This special case is sufficient for our application.

There are some additional technical aspects we need to deal with. Let $L = A_0 + \sum_{i=1}^{n} A_i x_i$ be the linear matrix obtained from $f \in \mathbb{F}_q\langle X\rangle$ by Higman linearization, where $X = \{x_1, x_2, \ldots, x_n\}$ and $A_i \in \mathbb{F}_q^{d\times d}, 0 \le i \le n$. If A_0 is an invertible matrix then it turns out that the problem of factorizing L can be directly reduced to the problem of finding a common invariant subspace for the matrices $A_0^{-1}A_i, 1 \le i \le n$. In general, however, A_0 is not invertible. Two cases arise:

(a) The polynomial f is *commutatively nonzero*. That is, it is nonzero on $\mathbb{F}_q^n$ (or on $\mathbb{F}^n$ for a small extension field $\mathbb{F}$). In this case, by the DeMillo-Lipton-Schwartz-Zippel Lemma [9, 25, 27], we can do a linear shift of the variables $x_i \leftarrow x_i + \alpha_i$ in the polynomial f, for α_i randomly picked from $\mathbb{F}_q$ (or $\mathbb{F}$). Let the resulting polynomial be f' and let its Higman linearization be $L_{f'}$. In $L_{f'}$ the constant matrix term A_0' will be invertible with high probability, and the reduction steps outlined above will work for $L_{f'}$. Furthermore, from the factorization of f' we can efficiently recover the factorization of f. Section 4 deals with Case (a), with Theorem 32 summarizing the algorithm for factorizing f. Theorem 28 describes the algorithm for factorization of the linear matrix $L_{f'}$, and the factor extraction lemma (Lemma 31) allows us to efficiently recover the factorization of f' from the factorization of $L_{f'}$.

(b) In the second case, suppose f is zero on all scalars. Then, for example by Amitsur's theorem [1], for a random matrix substitution $x_i \leftarrow M_i \in \mathbb{F}^{2s\times 2s}$ the matrix $f(M_1, M_2, \ldots, M_n)$ is *invertible* with high probability, where s is the formula size of f.[2] [3] Accordingly, we can consider the factorization problem for shifted and dilated linear matrix $L' = A_0 \otimes I_\ell + \sum_{i=1}^{n} A_i \otimes (Y_i + M_i)$ which will have the constant matrix term invertible, where each Y_i is an $\ell \times \ell$ matrix of distinct noncommuting variables, where $\ell = 2s$. Recovering the factorization of L from the factorization of L' requires some additional algorithmic work based on linear algebra. A lemma from [14] (refer Section 5 and the Appendix for the details) turns out to be crucial here. The algorithm handling Case (b) is described in Section 5. Indeed, the new aspect of the algorithm is factorization of the dilated matrix L' from which we recover the factorization of the Higman linearization L_f of f. The remaining algorithm steps are exactly as in Section 4.

1.2 Small Finite fields

We now briefly explain the deterministic poly$(s, q, |X|)$ time factorization algorithm (when $\mathbb{F}_q$ is small). There are two places in the factorization algorithm outlined above where randomization is used: first, to obtain a matrix tuple $(M_1, M_2, \ldots, M_n)$ such that $f(M_1, M_2, \ldots, M_n)$

[2] Amitsur's theorem strengthens the Amitsur-Levitski theorem [2] often used in noncommutative PIT algorithms [5].

[3] In the actual algorithm we pick the matrices M_i using a result from [10]

is invertible, which ensures that the constant matrix term of the linear matrix L' is invertible. When $q = \Omega(d)$, where $d = \deg f$, it suffices to randomly pick $M_i \in \mathbb{F}_q^{2s\times 2s}$. However, if $q < d$ we can choose entries of the matrices M_i from a small extension field $\mathbb{F}_{q^k}$ such that $q^k = \Omega(d)$. Thereby, we will obtain factorization of L' and subsequently that of the polynomial f over the extension field $\mathbb{F}_{q^k}$. However, we can use the fact that the finite field $\mathbb{F}_{q^k}$ can be embedded using the regular representation of the elements of $\mathbb{F}_{q^k}$ in the matrix algebra $\mathbb{F}_q^{k\times k}$. Thus, we can obtain from $(M_1, M_2, \ldots, M_n)$ a matrix tuple $(M'_1, M'_2, \ldots, M'_n)$ with $M'_i \in \mathbb{F}_q^{2sk\times 2sk}$ such that $f(M'_1, M'_2, \ldots, M'_n)$ is invertible. This will ensure that the linear matrix L' can be factorized over the field $\mathbb{F}_q$ which will allow us to obtain a complete factorization of f into irreducible factors over $\mathbb{F}_q$.

In order to get a deterministic polynomial-time algorithm for finding such matrices $M'_i, 1 \le i \le n$ we will use the fact that the polynomial f is given by a small noncommutative formula and hence has a small algebraic branching program. Then, using ideas from [23, 11, 4] we can easily find such matrices M'_i in deterministic polynomial time.

Next, we notice that Ronyai's algorithm for finding common invariant subspaces of matrices over $\mathbb{F}_q$ is essentially a polynomial-time reduction to univariate polynomial factorization over $\mathbb{F}_q$. We can use Berlekamp's deterministic poly(q, D) algorithm for the factorization of univariate degree D polynomials over $\mathbb{F}_q$. Putting it together, we can obtain a deterministic poly$(s, q, |X|)$ time algorithm for factorization of $f \in \mathbb{F}_q\langle X\rangle$ as a product of irreducible factors over $\mathbb{F}_q$.

Unfortunately, the algorithm outlined above does not yield an efficient algorithm for noncommutative polynomial factorization over rationals. The bottleneck is the problem of computing common invariant subspaces for a collection of matrices over $\mathbb{Q}$. Ronyai's algorithm for the problem over finite fields [24] builds on the decomposition of finite-dimensional associative algebras over fields. Given an algebra $\mathcal{A}$ over a finite field $\mathbb{F}_q$ the algorithm decomposes $\mathcal{A}$ as a direct sum of minimal left ideals of $\mathcal{A}$ which is used to find nontrivial common invariant subspaces. However, as shown by Friedl and Ronyai [12], over rationals the problem of decomposing a *simple* algebra as a direct sum of minimal left ideals is at least as hard as factoring square-free integers.

1.3 Related research

The study of factorization in noncommutative rings is systematically investigated as part of Cohn's general theory of noncommutative free ideal rings [7, 8] which is based on the notion of the weak algorithm. In fact, there is a hierarchy of weak algorithms generalizing the division algorithm for commutative integral domains [7].

To the best of our knowledge, the complexity of noncommutative polynomial factorization has not been studied much, unlike the problem of commutative polynomial factorization [26, 19, 20]. Prior work on the complexity of noncommutative polynomial factorization we are aware of is [3] where efficient algorithms are described for the problem of factoring *homogeneous* noncommutative polynomials (which enjoy the unique factorization property, and indeed the algorithms in [3] crucially use the unique factorization property). When the input homogeneous noncommutative polynomial has a small noncommutative arithmetic circuit (even given by a black-box as in Kaltofen's algorithms [19, 20]) it turns out that the problem is efficiently reducible to commutative factorization by set-multilinearizing the given noncommutative polynomial with new commuting variables. This also works in the black-box setting and yields a randomized polynomial-time algorithm which will produce as output black-boxes for the irreducible factors (which will all be homogeneous). When the

input homogeneous polynomial is given by an algebraic branching program there is even a deterministic polynomial-time factorization algorithm. Indeed, the noncommutative factorization problem in for homogeneous polynomials efficiently reduces to the noncommutative PIT problem [3], analogous to the commutative case [21], modulo the randomness required for univariate polynomial factorization in the case of finite fields of large characteristic. The motivation of the present paper is to extend the above results to the inhomogeneous case.

Plan of the paper. In Section 2 we present basic definitions and the background results from Cohn's work on factorization. In Section 3 we further present some results from Cohn's work relevant to the paper. In Section 4 we present the factorization algorithm for polynomials f that does not vanish on scalars and in Section 5 we present the algorithm for the general case. Omitted proofs from the conference version can be found in full version.

2 Preliminaries

In this section we give some basic definitions and results relevant to the paper, mainly from Cohn's theory of factorization. Analogous to integral domains and unique factorization domains in commutative ring theory, P.M. Cohn [7, 8] has developed a theory for noncommutative rings based on the weak algorithm (a noncommutative generalization of the Euclidean division algorithm) and the notion of free ideal rings. We present the relevant basic definitions and results, specialized to the ring $\mathbb{F}\langle X\rangle$ of noncommutative polynomials with coefficients in a (commutative) field $\mathbb{F}$, and also for matrix rings with entries from $\mathbb{F}\langle X\rangle$.

The results about $\mathbb{F}\langle X\rangle$ in Cohn's text [7, Chapter 5] are stated uniformly for algebraically closed fields $\mathbb{F}$. However, those we discuss hold for any field $\mathbb{F}$ (in particular for $\mathbb{F}_q$ or a small degree extension of it). The proofs are essentially based on linear algebra.

Since we will be using Higman's linearization [15] to factorize noncommutative polynomials, we are naturally lead to studying the factorization of linear matrices in $\mathbb{F}\langle X\rangle^{d\times d}$ using Cohn's theory.

▶ **Definition 1** ([7]). *A matrix M in $\mathbb{F}\langle X\rangle^{d\times d}$ is called* full *if it has (noncommutative) rank d. That is, it cannot be decomposed as a matrix product $M = M_1 \cdot M_2$, for matrices $M_1 \in \mathbb{F}\langle\!\langle X\rangle\!\rangle^{d\times e}$ and $M_2 \in \mathbb{F}\langle\!\langle X\rangle\!\rangle^{e\times d}$ with $e < d$.*

▶ Remark 2. Based on the notion of noncommutative matrix rank [7], the square matrix $M \in \mathbb{F}\langle X\rangle^{d\times d}$ is full precisely when it is invertible in the skew field $\mathbb{F}\langle\!\langle X\rangle\!\rangle$. That is, M is full if and only if there is a matrix $N \in \mathbb{F}\langle\!\langle X\rangle\!\rangle^{d\times d}$ such that $MN = NM = I_d$, where I_d is $d \times d$ identity matrix. We refer to [13] and [18] for different equivalent formulations of noncommutative matrix rank.

We note the distinction between full matrices and units in the matrix ring $\mathbb{F}\langle X\rangle^{d\times d}$.

▶ **Definition 3.** *A matrix $U \in \mathbb{F}\langle X\rangle^{d\times d}$ is a* unit *if there is a matrix $V \in \mathbb{F}\langle X\rangle^{d\times d}$ such that $UV = VU = I_d$, where I_d is $d \times d$ identity matrix.*

Clearly, units in $\mathbb{F}\langle X\rangle^{d\times d}$ are full. Examples of units in $\mathbb{F}\langle X\rangle^{d\times d}$, which have an important role in our factorization algorithm, are upper (or lower) triangular matrices in $\mathbb{F}\langle X\rangle^{d\times d}$ whose diagonal entries are all *nonzero scalars*. Full matrices, in general, need not be units: for example, the 1×1 matrix x, where x is a variable, is full but it is not a unit in the ring $\mathbb{F}\langle X\rangle^{1\times 1} = \mathbb{F}\langle X\rangle$.

▶ Remark 4. Full non-unit matrices are essentially non-unit non-zero-divisors. For the factorization of elements in $\mathbb{F}\langle X\rangle^{d\times d}$, units are similar to scalars in the factorization of polynomials in polynomial rings. Cohn's theory [7] considers factorizations of full non-unit elements in $\mathbb{F}\langle X\rangle^{d\times d}$.

We next define *atoms* in $\mathbb{F}\langle X\rangle^{d\times d}$, which are essentially the irreducible elements in it.

▶ **Definition 5.** *A full non-unit element A in $\mathbb{F}\langle X\rangle^{d\times d}$ is an* atom *if A cannot be factorized as $A = A_1A_2$ for full non-unit matrices A_1, A_2 in $\mathbb{F}\langle X\rangle^{d\times d}$.*

Noncommutative polynomials do not have unique factorization in the usual sense of commutative polynomial factorization.[4] A classic example [7] is the polynomial $x + xyx$ with its two different factorizations

$$x + xyx = x(1 + yx) = (1 + xy)x,$$

where $1 + xy$ and $1 + yx$ are distinct irreducible polynomials.

▶ **Definition 6.** *Elements $A \in \mathbb{F}\langle X\rangle^{d\times d}$ and $B \in \mathbb{F}\langle X\rangle^{d'\times d'}$ are called* stable associates *if there are positive integers t and t' such that $d + t = d' + t'$ and units $P, Q \in \mathbb{F}\langle X\rangle^{(d+t)\times(d+t)}$ such that $A \oplus I_t = P(B \oplus I_{t'})Q$.*

It is easy to check that the polynomials $1 + xy$ and $1 + yx$ are stable associates.

Notice that if A and B are full non-unit matrices that are stable associates then A is atom if and only if B is atom. Furthermore, we note that stable associativity defines an equivalence relation between full matrices over the ring $\mathbb{F}\langle X\rangle$.

We observe that the problem of checking if two polynomials in $\mathbb{F}\langle X\rangle$ given as arithmetic formulas are stable associates or not has an efficient randomized algorithm (Lemma 17).

Now we turn to the problem of noncommutative polynomial factorization. By Higman's linearization [15, 7], given a polynomial $f \in \mathbb{F}\langle X\rangle$ there is a positive integer ℓ such that f is stably associated with a *linear matrix* $L \in \mathbb{F}\langle X\rangle^{\ell\times\ell}$, that is to say, the entries of L are affine linear forms.[5] Higman's linearization process is a simple algorithm obtaining the linear matrix L for a given f, and it plays a crucial role in our factorization algorithm. We describe it and state an effective version [13] which gives a simple polynomial-time algorithm to compute L when f is given as a noncommutative arithmetic formula.

Higman's linearization process

We describe a single step of the linearization process. Given an $m \times m$ matrix M over $\mathbb{F}\langle X\rangle$ such that $M[m, m] = f + g \times h$, apply the following:

- Expand M to an $(m + 1) \times (m + 1)$ matrix by adding a new last row and last column with diagonal entry 1 and remaining new entries zero:

$$\left[\begin{array}{c|c} M & 0 \\ \hline 0 & 1 \end{array}\right].$$

[4] However, as shown by Cohn, using the notion of stable associates there is a more general sense in which noncommutative polynomials have "unique" factorization [7].

[5] More generally, by Higman's linearization any matrix of polynomials M is stably associated with a linear matrix $L \in \mathbb{F}\langle X\rangle^{\ell\times\ell}$ for some ℓ.

- Then the bottom right 2×2 submatrix is transformed as follows by elementary row and column operations

$$\begin{pmatrix} f+gh & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} f+gh & g \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} f & g \\ -h & 1 \end{pmatrix}$$

Given a polynomial $f \in \mathbb{F}\langle X \rangle$ by repeated application of the above step we will finally obtain a *linear matrix* $L = A_0 + \sum_{i=1}^{n} A_i x_i$, where each $A_i, 0 \leq i \leq n$ is an $\ell \times \ell$ over $\mathbb{F}$, for some ℓ. The following theorem summarizes its properties.

▶ **Theorem 7** (Higman Linearization, [7]). *Given a polynomial $f \in \mathbb{F}\langle X \rangle$, there are matrices $P, Q \in \mathbb{F}\langle X \rangle^{\ell \times \ell}$ and a linear matrix $L \in \mathbb{F}\langle X \rangle^{\ell \times \ell}$ such that*

$$\left(\begin{array}{c|c} f & 0 \\ \hline 0 & I_{\ell-1} \end{array}\right) = PLQ \tag{1}$$

with P upper triangular, Q lower triangular, and the diagonal entries of both P and Q are all 1's (hence, P and Q are both units in $\mathbb{F}\langle X \rangle^{\ell \times \ell}$).

Instead of a single f, we can apply Higman linearization to a matrix of polynomials $M \in \mathbb{F}\langle X \rangle^{m \times m}$ to obtain a linear matrix L that is stably associated to M. We state the algorithmic version of Garg et al. [13] in this general form.

▶ **Theorem 8** ([13], Proposition A.2). *Let $M \in \mathbb{F}\langle X \rangle^{m \times m}$ such that $M_{i,j}$ is computed by a noncommutative arithmetic formula of size at most s and bit complexity at most b. Then, for some $k = O(s)$, in time $\mathrm{poly}(s, b)$ we can compute the matrices P, Q and L in $\mathbb{F}\langle X \rangle^{\ell \times \ell}$ of Higman's linearization such that*

$$\left(\begin{array}{c|c} M & 0 \\ \hline 0 & I_k \end{array}\right) = PLQ.$$

where $\ell = m + k$. Moreover, the entries of the matrices P and Q as well as P^{-1} and Q^{-1} are given by polynomial-size algebraic branching programs which can also be obtained in polynomial time.

We will sometimes denote the block diagonal matrix $\left(\begin{array}{c|c} M & 0 \\ \hline 0 & I_k \end{array}\right)$ by $M \oplus I_k$.

As P and Q are units with diagonal entries all 1's, the matrix M is full iff the linear matrix L is full. Also, the scalar matrix $M(\overline{0})$ (obtained by setting all variables to zero) is invertible iff the scalar matrix $L(\overline{0})$, similarly obtained, is invertible.

Invariant Subspaces and Ronyai's Algorithm

▶ **Definition 9.** *Let $A_1, \ldots, A_n \in \mathbb{F}^{d \times d}$. A subspace $V \subseteq \mathbb{F}^n$ is called as common invariant subspace of $A_1, \ldots, A_n$ if $A_i v \in V$ for all $i \in [n]$ and $v \in V$.*

Clearly 0 and $\mathbb{F}^n$ are, trivially, common invariant subspaces for any collection of matrices. The algorithmic problem is to find a *non-trivial* common invariant subspace if one exists. Ronyai [24] gives a randomized polynomial-time algorithm for this problem when $\mathbb{F}$ is finite field.

▶ **Theorem 10** ([24]). *Given $A_1, \ldots, A_n \in \mathbb{F}_q^{d \times d}$ there is a randomized algorithm running in time polynomial in $n, d, \log q$ that computes with high probability a non-trivial common invariant subspace of $A_1, \ldots, A_n$ if such a subspace exists, and outputs "no" otherwise.*

▶ Remark 11. We should note here, the classical Burnside's theorem [6] for matrix algebras over algebraically closed fields. It essentially shows that the algebra generated by $A_1, A_2, \ldots, A_n$ is the full matrix algebra iff there is no nontrivial common invariant subspace.

▶ Remark 12. As already mentioned in the introduction, Friedl and Ronyai [12] have shown that over rationals the problem is at least as hard as factoring square-free integers, and hence likely to be intractable.

For standard definitions of noncommutative formulas and noncommutative algebraic branching programs (ABPs) we refer to [22].

3 Some Basic Results

In this section we present some basic results required for our factorization algorithm.

Monic linear matrices

▶ **Definition 13** ([7]). *Let $L = A_0 + A_1x_1 + \ldots + A_nx_n \in \mathbb{F}\langle X\rangle^{d\times d}$ be a linear matrix, where each A_i is a $d\times d$ scalar matrix over $\mathbb{F}$. Then L is called* right monic *if the $d\times nd$ scalar matrix $[A_1\ A_2\ \ldots\ A_n]$ has full row rank. Equivalently, if there are matrices $B_1, \ldots, B_n \in \mathbb{F}^{d\times d}$ such that $\Sigma_{i=1}^{n} A_iB_i = I_d$ (i.e. the matrix $[A_1\ A_2\ \ldots\ A_n]$ has right inverse).*

Similarly, L is left monic *if the $nd \times d$ matrix $[A_1^T\ A_2^T\ \ldots\ A_n^T]^T$ has full column rank. L is called* monic *if it is both left and right monic.*

The next two results from Cohn [7] are important properties of monic linear matrices.

▶ **Lemma 14** ([7]). *A right (or left) monic linear matrix in $\mathbb{F}\langle X\rangle^{d\times d}$ is not a unit in $\mathbb{F}\langle X\rangle^{d\times d}$.*

Let $f \in \mathbb{F}\langle X\rangle$ be a nonzero polynomial and L be a linear matrix obtained from f by Higman linearization as in Equation 1. Clearly, L is a full linear matrix. We show that we can transform L to obtain a full and right (or left) monic linear matrix L' that is stably associated to f. Furthermore, we can efficiently compute L' and the related transformation matrices.

▶ **Theorem 15** ([7]). *Let $L = A_0 + \sum_{i=1}^{n} A_ix_i$ be a full linear matrix in $\mathbb{F}\langle X\rangle^{d\times d}$ obtained by Higman linearization from a non constant polynomial $f \in \mathbb{F}\langle X\rangle$. Then there are deterministic $\mathrm{poly}(n, d, \log_2 q)$ time algorithms that compute units $U, U' \in \mathbb{F}\langle X\rangle^{d\times d}$ and invertible scalar matrices $S, S' \in \mathbb{F}_q^{d\times d}$ such that:*

1. *$ULS = L' \oplus I_r$, and L' is right monic. Moreover, if L is not right monic then $r > 0$.*
2. *$S'LU' = L' \oplus I_{r'}$, and L' is left monic. Moreover, if L is not left monic then $r' > 0$.*

▶ Remark 16. By repeated application of the algorithm in Theorem 15 we can compute units $U_1, U_2 \in \mathbb{F}\langle X\rangle^{d\times d}$ such that $U_1LU_2 = L' \oplus I_r$, where L' is *both left and right monic*. Such a two-sided monic L' is called *monic* in [7].

For our factorization algorithm, it suffices to compute an L' that is either left or right monic that is associated to L as in Theorem 15. It turns out that either a left monic or a right monic L' suffices to use Ronyai's common invariant subspace algorithm to factorize L' (and hence also L) as we show in Theorem 28. More importantly, the fact that matrices S and S' in Theorem 15 are scalar is important for the factor extraction algorithm as discussed in Theorem 32.

▶ **Lemma 17.** *Given polynomials $f, g \in \mathbb{F}\langle X\rangle$ as input by noncommutative arithmetic formulas, we can check in randomized polynomial time if f and g are stable associates.*

The next result shows how irreducibility (more generally, the property of being an atom) is preserved by Higman linearization.

▶ **Theorem 18.** *Let $f \in \mathbb{F}\langle X\rangle$ be a nonconstant polynomial and L be a full linear matrix stably associated with f (obtained via Higman linearization). Then the polynomial f is irreducible iff L is an atom.*

We give a self-contained proof of the above theorem, using the following (suitably paraphrased) result of Cohn.

▶ **Lemma 19** (Matrix Product Trivialization, [8], pp. 198)**.** *Let $A \in \mathbb{F}\langle X\rangle^{m\times n}$ and $B \in \mathbb{F}\langle X\rangle^{n\times s}$ be polynomial matrices such that their product $AB = 0$. Then there exists a unit $P \in \mathbb{F}\langle X\rangle^{n\times n}$ such that for every index $i \in [n]$ either the i^{th} column of the matrix product AP is all zeros or the i^{th} row of the matrix product $P^{-1}B$ is all zeros.*

Let $L \in \mathbb{F}\langle X\rangle^{d\times d}$ be a full and right (or left) monic linear matrix. Let $L = A_0 + \sum_{i=1}^{n} A_i x_i$. For a positive integer ℓ let $M_i, i \in [n]$ be $\ell \times \ell$ scalar matrices with entries from $\mathbb{F}$ (or a small degree extension of $\mathbb{F}$). Let $Y_i, i \in [n]$ be $\ell \times \ell$ matrices whose entries are distinct noncommuting variables $y_{ijk}, 1 \le j, k \le \ell$. Then the evaluation of the linear matrix L at $x_i \leftarrow Y_i + M_i, 1 \le i \le n$ is the $d\ell \times d\ell$ linear matrix in the y_{ijk} variables:

$$L' = A_0 \otimes I_\ell + \sum_{i=1}^{n} A_i \otimes M_i + \sum_{i=1}^{n} \sum_{j,k=1}^{\ell} (A_i \otimes E_{jk}) \cdot y_{ijk}$$

▶ **Lemma 20.** *There is a positive integer $\ell \le 2d$ such that for randomly picked $\ell \times \ell$ matrices $M_i, i \in [n]$ (with entries from $\mathbb{F}$ or a small degree extension field) the matrix $A_0 \otimes I_\ell + \sum_{i=1}^{n} A_i \otimes M_i$ is an invertible matrix.*

Proof. Since $L \in \mathbb{F}\langle X\rangle^{d\times d}$ is a full linear matrix, it has noncommutative rank d. Hence, by the result of [10] for the generic $2d \times 2d$ matrix substitution $x_i \leftarrow X_i, i \in [n]$, where X_i is a matrix of distinct commuting variables, the commutative rank of $L(X_1, X_2, \ldots, X_n)$ is $2d^2$ (which means it is invertible). Hence there is a least $\ell \le 2d$ such that the commutative rank of $L(X_1, X_2, \ldots, X_n)$ is $d\ell$, where X_i are generic $\ell \times \ell$ matrices with commuting variables. Hence, by the DeMillo-Lipton-Schwartz-Zippel lemma [9, 25, 27] the rank of the scalar matrix $L(M_1, M_2, \ldots, M_n)$ is $d\ell$, where M_i is a random scalar matrix with entries from $\mathbb{F}$ or a small extension. ◀

Finally, we state and prove a *modified version* of a result due to Cohn that allows us to relate the factorization of a polynomial $f \in \mathbb{F}\langle X\rangle$ to the factorization of its Higman linearization L. The proof is given in the appendix.

▶ **Theorem 21** ([7], Theorem 5.8.8)**.** *Let $C \in \mathbb{F}\langle X\rangle^{d\times d}$ be a full and right monic (or left monic) linear matrix for $d > 1$. Then C is not an atom if and only if there are $d \times d$ invertible scalar matrices S and S' such that*

$$SCS' = \begin{pmatrix} A & 0 \\ D & B \end{pmatrix} \tag{2}$$

where A is an $r \times r$ full right (respec. left) monic linear matrix and B is an $s \times s$ full right (respec. left) monic linear matrix such that $r + s = d$.

▶ Remark 22. In [7] the theorem is proved under the stronger assumption that C is monic. However, as we show, it holds even for C that is right monic or left monic with minor changes to Cohn's proof. We require the above version for our factorization algorithm.

4 Polynomial factorization: commutatively non-zero case

Recall that $\mathbb{F}\langle X\rangle$ denotes the free noncommutative polynomial ring $\mathbb{F}\langle x_1, x_2, \ldots, x_n\rangle$ and our goal is to give a randomized polynomial-time factorization algorithm for input polynomials in $\mathbb{F}\langle X\rangle$ given as arithmetic formulas when $\mathbb{F} = \mathbb{F}_q$ is a finite field of size q.

A polynomial $f \in \mathbb{F}\langle X\rangle$ is *commutatively nonzero* if $f(\alpha_1, \alpha_2, \ldots, \alpha_n) \neq 0$ for scalars $\alpha_i \in \mathbb{F}$ (or a small extension field of $\mathbb{F}$).

In this section we will present the factorization algorithm for commutatively nonzero polynomials.[6] It has three broad steps:

(i) We transform the given polynomial f to a full and *right (or left) monic* linear matrix L by first the Higman linearization of f followed by the algorithm in the proof of Theorem 15.

(ii) Next, we factorize the full and right (or left) monic linear matrix L into atoms.

(iii) Finally, we recover the irreducible factors of f from the atomic factors of L.

We formally state the three problems of interest in this paper.

▶ **Problem 23** (FACT($\mathbb{F}$)).
Input *A noncommutative polynomial $f \in \mathbb{F}\langle X\rangle$ given by an arithmetic formula.*
Output *Compute a factorization $f = f_1 f_2 \cdots f_r$, where each f_i is irreducible, and each f_i is output as an algebraic branching program.*

▶ **Problem 24** (LIN-FACT($\mathbb{F}$)).
Input *A full and right (or left) monic linear matrix $L \in \mathbb{F}\langle X\rangle^{d\times d}$.*
Output *Compute a factorization $L = F_1 F_2 \cdots F_r$, where each F_i is a full linear matrix that is an atom.*

▶ **Problem 25** (INV($\mathbb{F}$)).
Input *A list of scalar matrices $A_1, A_2, \ldots, A_n \in \mathbb{F}^{d\times d}$.*
Output *Compute a nontrivial invariant subspace $V \subset \mathbb{F}^d$ or report that the only invariant subspaces are 0 and $\mathbb{F}^d$.*

In the three-step outline of the algorithm, for the second step we will show that factoring a full and right (or left) monic linear matrix is randomized polynomial-time reducible to the problem of computing a common invariant subspace for a collection of scalar matrices. For the third step, we will give a polynomial-time algorithm (based on Lemma 19) to recover the irreducible factors of f from the atomic factors of L.

▶ Remark 26. We use Ronyai's randomized polynomial-time algorithm [24] to solve the problem of computing a a common invariant subspace for a collection of matrices over $\mathbb{F}_q$. Over rational numbers $\mathbb{Q}$, even for a special case the problem of computing a common invariant subspace turns out to be at least as hard as factoring square-free integers [12]. Hence, our approach to noncommutative polynomial factorization does not yield an efficient algorithm over $\mathbb{Q}$.

Suppose $f \in \mathbb{F}\langle X\rangle$ is given by a noncommutative arithmetic formula. Since f has small degree we can check if it is commutatively nonzero in randomized polynomial-time by the DeMillo-Lipton-Schwartz-Zippel test [9, 25, 27] and, if so, find $\alpha_i \in \mathbb{F}, i \in [n]$ such that $f(\alpha_1, \alpha_2, \ldots, \alpha_n) \neq 0$ (if $\mathbb{F}$ is small, we pick α_i from a small extension field). Furthermore,

[6] In the next section we will deal with the general case. The algorithm is more technical in detail, although in essence the same.

by a linear shift of the variables $x_i \leftarrow x_i + \alpha_i, i \in [n]$ followed by scaling we can assume $f(\overline{0}) = 1$. Note that from the factorization of the linear shift of f we can recover the factors of f by shifting the variables back, and irreducibility is preserved by linear shift. For the rest of this section we will assume $f(\overline{0}) = 1$.

Let $L = A_0 + \sum_{i=1}^{n} A_i x_i$. As $f(\overline{0}) = 1$, we have $L(\overline{0}) = A_0$ is an invertible matrix. We now present an efficient algorithm for factoring L as a product of linear matrices $L_1 L_2 \cdots L_r$, where each L_i is an atom.

▶ Remark 27. The factorization algorithm for arbitrary full and right (or left) monic linear matrices (in which A_0 need not be invertible) is similar but more involved. It is based on Lemma 20 and is dealt with in the next section.

4.1 Algorithm for a special case of LIN-FACT($\mathbb{F}_q$)

▶ **Theorem 28.** *There is a randomized polynomial-time algorithm for the following two special cases of the LIN-FACT($\mathbb{F}_q$) problem:*

1. *Given a full right monic matrix L as input such that $L(\overline{0})$ is an invertible matrix, the algorithm outputs a factorization of L as a product of linear matrices that are atoms.*
2. *Given a full left monic matrix L as input such that $L(\overline{0})$ is an invertible matrix, the algorithm outputs a factorization of L as a product of linear matrices that are atoms.*

Proof. We present the algorithm only for the first part, as the second part has essentially the same solution.

Let $L = A_0 + \sum_{i=1}^{n} A_i x_i$ in $\mathbb{F}\langle X\rangle^{d\times d}$ be such an instance of LIN-FACT($\mathbb{F}_q$). We can write $L = A_0 \cdot L'$ where L' is the full and right monic linear matrix

$$L' = I_d + \sum_{i=1}^{n} A_0^{-1} A_i x_i.$$

Clearly, it suffices to factorize the linear matrix L' into atoms.

First we show that L' is not an atom iff matrices $A_0^{-1}A_i$, $1 \le i \le n$ have a nontrivial common invariant subspace. By Theorem 21, L' is not an atom if and only if we can write $S_1 L' S_2 = \begin{pmatrix} B & 0 \\ D & C \end{pmatrix}$ for invertible scalar matrices S_1 and S_2, where B and C are full and right monic linear matrices, and D is some linear matrix. Equating the constant terms on both sides of the above equation we have $S_1 S_2 = \begin{pmatrix} B_0 & 0 \\ D_0 & C_0 \end{pmatrix}$ as the constant term of L' is I_d. Thus the matrices $S_1 S_2$ and its inverse also has the same block form which implies that $S_1 L' S_1^{-1} = S_1 L' S_2 (S_1 S_2)^{-1}$ also has the same block form. It follows that the n matrices $A_0^{-1}A_i, 1 \le i \le n$ have a nontrivial common invariant subspace. Conversely, if the matrices $A_0^{-1}A_i, 1 \le i \le n$ have a nontrivial common invariant subspace then we have a basic change scalar matrix S such that $SL'S^{-1}$ has the block form $\begin{pmatrix} L_1 & 0 \\ * & L_2 \end{pmatrix}$, where L_1 and L_2 are full and right monic linear matrices. So by Theorem 21 L' is not an atom. So we have established, L' (and hence L) is not an atom iff matrices $A_0^{-1}A_i$, $1 \le i \le n$ have a nontrivial common invariant subspace. We will use Ronyai's randomized polynomial-time algorithm for finding a nontrivial common invariant subspace for matrices $A_0^{-1}A_i, 1 \le i \le n$ over finite field $\mathbb{F}_q$.

If there is no nontrivial invariant subspace then the linear matrix L' (and hence L) is an atom. Otherwise, by repeated application of Ronyai's algorithm we will obtain a basis change scalar matrix T which when applied to L' yields a linear matrix in the following *atomic block diagonal form*:

$$TL'T^{-1} = \begin{pmatrix} L_1 & 0 & 0 & \dots & 0 \\ * & L_2 & 0 & \dots & 0 \\ * & * & L_3 & \dots & 0 \\ & & & \ddots & \\ * & * & * & \dots & L_r \end{pmatrix}, \tag{3}$$

where for each $j \in [r]$, the full right monic linear matrix $L_j \in \mathbb{F}\langle X\rangle^{d_j \times d_j}$ is an atom, and each $*$ stands for some unspecified linear matrix. It is now easy to factorize $TL'T^{-1}$ as a product of atoms by noting one step of the factorization of $TL'T^{-1}$ from its form:

$$TL'T^{-1} = \begin{pmatrix} A & 0 \\ D & L_r \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ D & I \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ 0 & L_r \end{pmatrix}.$$

We note that $\begin{pmatrix} I & 0 \\ D & I \end{pmatrix}$ is a unit. Since L_r is an atom the product $\begin{pmatrix} I & 0 \\ D & I \end{pmatrix} \cdot \begin{pmatrix} I & 0 \\ 0 & L_r \end{pmatrix}$ is also an atom and a linear matrix, and it is the rightmost factor of $TL'T^{-1}$. Continuing thus with A now, we can factorize $TL'T^{-1}$ as a product $F_1'F_2' \cdots F_r'$ of r atoms, each of which is a linear matrix. It follows that $L = A_0T^{-1}F_1'F_2' \cdots F_r'T$ is a complete factorization of L as a product of atomic linear matrices (both A_0 and T are scalar invertible matrices). ◀

▶ Remark 29. We note that Ronyai's algorithm [24] for $\mathrm{INV}(\mathbb{F}_q)$ is actually a deterministic polynomial-time *reduction* from $\mathrm{INV}(\mathbb{F}_q)$ to univariate polynomial factorization over $\mathbb{F}_q$.

Based on whether we want to work with right monic or left monic case we will express $f \oplus I_s$ in an appropriate form using Higman linearization and Theorem 15 as described in the equation below:

$$f \oplus I_s = \begin{cases} PU(L' \oplus I_t)SQ, & \text{in the right monic case} \\ PS(L' \oplus I_t)UQ, & \text{in the left monic case} \end{cases} \tag{4}$$

where $d + t = s + 1$, $L' \in \mathbb{F}\langle X\rangle^{d\times d}$ is a full and right (or left) monic linear matrix, P is upper triangular with all 1's diagonal, Q is lower triangular with all 1's diagonal, $U \in \mathbb{F}\langle X\rangle^{(d+t)\times(d+t)}$ is a unit, and $S \in \mathbb{F}^{(d+t)\times(d+t)}$ is an invertible scalar matrix.

Algorithm for FACT($\mathbb{F}_q$)

We are now ready to describe the polynomial factorization algorithm for commutatively nonzero polynomials in $\mathbb{F}\langle X\rangle$. Starting with the Higman linearization of the input polynomial $f \in \mathbb{F}\langle X\rangle$ as in Equation 4, by an application of the first parts of Theorems 15 and 28 we obtain the factorization $f \oplus I_s = PUF_1'F_2' \cdots F_r'SQ$ using the structure in Equation 3.

Alternatively, by applying the second part of Theorem 15 we can compute a left monic linear matrix L' that is a stable associate of f and, applying the second part of Theorem 28 we can compute the factorization

$$f \oplus I_s = PS'F_1'F_2' \cdots F_r'U'Q. \tag{5}$$

where each linear matrix F'_i is an atom, P is upper triangular with all 1's diagonal, Q is lower triangular with all 1's diagonal, U' is a unit and S' is a scalar invertible matrix. Equation 5 is the form we will use for the algorithm (we could equally well use the other factorization).

From the structure of the atomic block diagonal matrix $TL'T^{-1}$ in Equation 3 notice that the product $S'F'_1F'_2\cdots F'_i$ is a linear matrix for each $1 \leq i < r$.

The next lemma presents an algorithm that is crucial for extracting the factors of f.

▶ **Lemma 30.** *Let $C \in \mathbb{F}\langle X\rangle^{u\times d}$ be a linear matrix and $v \in \mathbb{F}\langle X\rangle^{d\times 1}$ be a column of polynomials such that $Cv = 0$. Each entry v_i of v is given by an algebraic branching program as input. Then, in polynomial time we can compute a invertible matrix $N \in \mathbb{F}\langle X\rangle^{d\times d}$ such that*

- *For $1 \leq i \leq d$ either the i^{th} column of CN is all zeros or the i^{th} row of $N^{-1}v$ is zero.*
- *Each entry of N is a polynomial of degree at most d^2 and is computed by a polynomial size ABP, and also each entry of N^{-1} is computed by a polynomial size ABP.*

Proof. We will describe the algorithm as a recursive procedure Trivialize that takes matrix C and column vector v as parameters and returns a matrix N as claimed in the statement.

Procedure Trivialize($C \in \mathbb{F}\langle X\rangle^{u\times d}, v \in \mathbb{F}\langle X\rangle^{d\times 1}$).

1. If $d = 1$ then (since $Cv = 0$ iff either $C = 0$ or $v = 0$) **return** the identity matrix.
2. If $d > 1$ then
3. write $C = C_0 + C_1$, where C_0 is a scalar matrix and C_1 is the degree 1 homogeneous part of C. Let k be the degree of the highest degree nonzero monomials in the polynomial vector v, and let m be a nonzero degree k monomial. Let $v(m) \in \mathbb{F}_q^{d\times 1}$ denote its (nonzero) coefficient in v. Then $Cv = 0$ implies $C_1v(m) = 0$. Let $T_0 \in \mathbb{F}_q^{d\times d}$ be a scalar invertible matrix with first column $v(m)$ obtained by completing the basis.
   a. If $C_0v(m) = 0$ then the first column of CT_0 is zero.
   b. Otherwise, CT_0 has first column as the nonzero scalar vector $Cv(m) = C_0v(m)$. Suppose i^{th} entry of $Cv(m)$ is a nonzero scalar α. With column operations we can drive the i^{th} entry in all other columns of CT_0 to zero. Let the resulting matrix be CT_0T_1 (where the matrix T_1 is invertible as it is a product of elementary matrices corresponding to these column operations, each of which is of the form $\mathrm{Col}_i \leftarrow (\mathrm{Col}_i + \mathrm{Col}_1 \cdot \alpha_0 + \sum_i \alpha_i x_i)$). Notice that CT_0T_1 is still linear.
   c. As $Cv = (CT_0T_1)(T_1^{-1}T_0^{-1}v)$, and in the i^{th} row of CT_0T_1 the only nonzero entry is α which is in its first column, we have that the first entry of $T_1^{-1}T_0^{-1}v$ is zero.
4. Let $C' \in \mathbb{F}\langle X\rangle^{u\times(d-1)}$ obtained by dropping the first column of CT_0T_1. Let $v' \in \mathbb{F}\langle X\rangle^{(d-1)\times 1}$ be obtained by dropping the first entry of $T_1^{-1}T_0^{-1}v$. Note that C' is still linear.
5. Recursively call Trivialize($C' \in \mathbb{F}\langle X\rangle^{u\times(d-1)}, v' \in \mathbb{F}\langle X\rangle^{(d-1)\times 1}$). and let the matrix returned by the call be $T_2 \in \mathbb{F}\langle X\rangle^{(d-1)\times(d-1)}$.
6. Putting it together, return the matrix $T_0T_1(I_1 \oplus T_2)$.

To complete the proof, we note that a highest degree monomial m such that $v(m) \neq 0$ is easy to compute in deterministic polynomial time if each v_i is given by an algebraic branching program using the PIT algorithm of Raz and Shpilka [23]. Notice that for the recursive call we need C' to be also a linear matrix and each entry of v' to have a small ABP. C' is linear because CT_0T_1 is a linear matrix since CT_0 is linear, its first column is scalar, and each column operation performed by T_1 is scaling the first column of CT_0 by a linear form and subtracting from another column of CT_0. Each entry of v' has a small ABP because T_0^{-1} is scalar and it is easy to see that the entries of T_1^{-1} have ABPs of polynomial size. Finally, we

note that T_1 is a product of at most $d-1$ linear matrices (each corresponding to a column operation), and N is an iterated product of d such matrices. Hence, each entry of N as well as N^{-1} is a polynomial of degree at most d^2 and is computable by a small ABP. ◀

Turning back to our algorithm for FACT($\mathbb{F}_q$), in the next lemma we design an efficient algorithm that will allow us to extract all the irreducible factors of f (given Equation 5).

▶ **Lemma 31** (Factor Extraction). *Let $f \in \mathbb{F}\langle X\rangle$ be a polynomial and $G \in \mathbb{F}\langle X\rangle^{(d-1)\times(d-1)}$ be a unit such that*

$$\begin{pmatrix} f & u \\ 0 & G \end{pmatrix} = PCD, \tag{6}$$

such that

- *C is a full linear matrix that is a non-unit, P is upper triangular with all 1's diagonal, and $D \in \mathbb{F}\langle X\rangle^{d\times d}$ is a full non-unit matrix which is also an atom.*
- *The polynomial f, and the entries of u, G, P, D are all given as input by algebraic branching programs.*

Then we can compute in deterministic polynomial time a nontrivial factorization $f = g \cdot h$ of the polynomial f such that h is an irreducible polynomial.

Proof. Let

$$C = \begin{pmatrix} c_1 & c_3 \\ c_2 & c_4 \end{pmatrix} \text{ and } D = \begin{pmatrix} d_1 & d_3 \\ d_2 & d_4 \end{pmatrix},$$

written as 2×2 block matrices where c_1 and d_1 are 1×1 blocks. By dropping the first row of the matrix in the left hand side of Equation 6 and the first row of P we get

$$(0\ G) = (0\ P')CD,$$

where P' is also an upper triangular matrix with all 1's diagonal. Equating the first columns on both sides we have

$$\begin{aligned} 0 &= (0\ P')\begin{pmatrix} c_1 & c_3 \\ c_2 & c_4 \end{pmatrix}\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \text{ which implies that} \\ 0 &= P'(c_2\ c_4)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \text{ and hence} \\ 0 &= (c_2\ c_4)\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \text{ since } P' \text{ is invertible.} \end{aligned}$$

Since $(c_2\ c_4) \in \mathbb{F}\langle X\rangle^{(d-1)\times d}$ is a matrix with linear entries and $\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \in \mathbb{F}\langle X\rangle^{d\times 1}$ is a column vector of polynomials which are given by ABPs as input, we can apply the algorithm of Lemma 30 to compute a unit N such that its entries are all given by ABPs such that for $1 \le i \le d$, either the i^{th} column of $(c_2'\ c_4') = (c_2\ c_4)N$ is zero or the i^{th} row of $\begin{pmatrix} d_1' \\ d_2' \end{pmatrix} = N^{-1}\begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$ is zero.

Now the following argument is almost identical with the argument towards the end of the proof of the Theorem 18. We give it below for completeness. Since D is a full matrix, the matrix $N^{-1}D$ is also full which implies its first column $\begin{pmatrix} d_1' \\ d_2' \end{pmatrix}$ cannot be all zeros. So there

is at least one nonzero entry in $\begin{pmatrix} d'_1 \\ d'_2 \end{pmatrix}$ and the corresponding column in $(c'_2\ c'_4)$ is all zero. This implies there exist a permutation matrix Π such that the first column of $C(c'_2\ c'_4)\Pi$ is all zero and first entry of $\Pi^{-1}\begin{pmatrix} d'_1 \\ d'_2 \end{pmatrix}$ is non zero.

Consider the matrices $C'' = CN\Pi = \begin{pmatrix} c''_1 & c''_3 \\ c''_2 & c''_4 \end{pmatrix}$ and $D'' = \Pi^{-1}N^{-1}D = \begin{pmatrix} d''_1 & d''_3 \\ d''_2 & d''_4 \end{pmatrix}$. We have

$$\begin{pmatrix} f & * \\ 0 & G' \end{pmatrix} = P^{-1}\begin{pmatrix} f & u \\ 0 & G \end{pmatrix} = \begin{pmatrix} c''_1 & c''_3 \\ c''_2 & c''_4 \end{pmatrix}\begin{pmatrix} d''_1 & d''_3 \\ d''_2 & d''_4 \end{pmatrix},$$

where $G' = (P')^{-1}G$ is a unit, c''_2 is all zero column matrix and d''_1 is non-zero. Now observing $(2,1)^{th}$ matrix block in the above equation, we get d''_2 is all zero column. Hence, by looking at $(2,2)^{th}$ block in the above equation, we can see that c''_4 and d''_4 are units as G' is a unit. Clearly, we have $f = c''_1 \cdot d''_1$. Now, since C and D are non-units (by assumption), the matrices C'' and D'' are also non-units. Therefore, c''_1 is not a scalar for otherwise C'' would be a unit. Similarly, d''_1 is not a scalar. It follows that $f = c''_1 d''_1$ is a nontrivial factorization of f.

Furthermore, since D is an atom by assumption and D'' is a stable associate of D, D'' is an atom. As $D'' = \begin{pmatrix} d''_1 & d''_3 \\ 0 & d''_4 \end{pmatrix}$ and d''_4 is invertible, we get $\begin{pmatrix} 1 & 0 \\ 0 & (d''_4)^{-1} \end{pmatrix} \cdot D'' = \begin{pmatrix} d''_1 & d''_3 \\ 0 & I_s \end{pmatrix}$. Now applying suitable row operations to the matrix $(1 \oplus (d''_4)^{-1})D''$ we can drive d''_3 to zero. So we have $U(1 \oplus (d''_4)^{-1})D'' = (d''_1 \oplus I_s)$ for a unit U. Hence d''_1 is an associate of D'' and therefore d''_1 is irreducible as D'' is an atom. ◀

Finally, we describe the factorization algorithm for commutatively nonzero polynomials $f \in \mathbb{F}\langle X\rangle$ over finite fields $\mathbb{F}_q$.

▶ **Theorem 32.** *Let $\mathbb{F}\langle X\rangle = \mathbb{F}_q\langle X\rangle$ and $f \in \mathbb{F}\langle X\rangle$ be a commutatively nonzero polynomial given by an arithmetic formula of size s as input instance of FACT($\mathbb{F}_q$). Then there is a* $\mathrm{poly}(s, \log q)$ *time randomized algorithm that outputs a factorization $f = f_1 f_2 \cdots f_r$ such that each f_i is irreducible and is output as an algebraic branching program.*

Proof. Given f as input, we apply Higman linearization followed by the algorithm for LIN-FACT($\mathbb{F}_q$) described in Theorem 28. This will yield the factorization of $f \oplus I_s = PSS_1F_1F_2 \ldots F_rS_2UQ$ where each linear matrix F_i is an atom, P is upper triangular with all 1's diagonal, Q is lower triangular with all 1's diagonal, U is a unit and S is a scalar invertible matrix, as given in Equation 5. We can now apply Lemma 31 to extract irreducible factors of f (one by one from the right).

For the first step, let $C = SS_1F_1F_2 \cdots F_{r-1}$ and $D = F_rS_2UQ$ in Lemma 31. The proof of Lemma 31 yields the matrix $N_r = N\Pi$ such that both matrices $C'' = PSS_1F_1F_2 \cdots F_{r-1}N_r$ and $D'' = N_r^{-1}F_rS_2UQ$ has the first column all zeros except the $(1,1)^{th}$ entries c''_1 and d''_1 which yields the nontrivial factorization $f = c''_1 d''_1$, where $d''_1 = f_r$ is irreducible. Renaming c''_1 as g_r we have from the structure of C'':

$$\begin{pmatrix} g_r & * \\ 0 & G_r \end{pmatrix} = P(SS_1F_1F_2 \cdots F_{r-2})(F_{r-1}N_r).$$

Setting $C = SS_1F_1F_2 \cdots F_{r-2}$ and $D = F_{r-1}N_r$ in Lemma 31 we can compute the matrix N_{r-1} using which we will obtain the next factorization $g_r = g_{r-1}f_{r-1}$, where f_{r-1} is irreducible because the linear matrix F_{r-1} is an atom. Lemma 31 is applicable as all conditions are met by the matrices in the above equation (note that G_r will be a unit).

Continuing thus, at the i^{th} stage we will have $f = g_{r-i+1}f_{r-i+1}f_{r-i+2}\cdots f_r$ after obtaining the rightmost i irreducible factors by the above process. At this stage we will have

$$\begin{pmatrix} g_{r-i+1} & * \\ 0 & G_{r-i+1} \end{pmatrix} = P(SS_1F_1F_2\cdots F_{r-i-1})(F_{r-i}N_{r-i+1}),$$

where G_{r-i+1} is a unit and all other conditions are met to apply Lemma 31.

Thus, after r stages we will obtain the complete factorization $f = f_1f_2\cdots f_r$. For the running time, it suffices to note that the matrix N computed in Lemma 31 is a product of degree at most d^2 many linear matrices (corresponding to the column operations). Thus, at the i^{th} of the above iteration, the sizes of the ABPs for the entries of N_{r-i+1} are independent of the stages. Hence, the overall running time is easily seen to be polynomial in s and $\log q$. ◀

▶ **Corollary 33.** *If $f \in \mathbb{F}\langle X\rangle$ is commutatively nonzero polynomial given as input in sparse representation (as an $\mathbb{F}_q$-linear combination of its monomials) then in randomized polynomial time we can compute a factorization into irreducible factors in sparse representation.*

Proof. Let f be given as input in sparse representation. Suppose $\deg f = d$ and it is t-sparse. Then there are at most td^2 many monomials that can occur as a substring of the monomials of f. We can apply the randomized algorithm of Theorem 32 to obtain the factorization $f = f_1f_2\cdots f_r$, where each f_i is given by an ABP. Now, for each of the td^2 many candidate monomials of f_i we can find its coefficient in f_i in polynomial time (using the Raz-Shpilka algorithm [23]). Hence we can obtain the factorization $f = f_1'f_2'\cdots f_r'$, where each f_i' is a t-sparse polynomial. ◀

5 Factorization of Commutatively zero polynomials

In this section we will describe the general case of the factorization algorithm when the input polynomial $f \in \mathbb{F}\langle X\rangle$ is a commutatively zero polynomial. That is, f evaluates to zero on all scalar substitutions from $\mathbb{F}_q$ or any (commutative) extension field.

The factorization algorithm will follow the three broad steps described in Section 4 for the commutatively nonzero case: first, using Higman linearization and Theorem 15, transform the polynomial f to a stably associated linear matrix L that is full and left (or right) monic. Next, factorize the linear matrix L into atoms. Finally, recover the irreducible factors of f from the atomic factors of the linear matrix L using the factor extraction procedure described in Lemma 31.

The step that requires a new algorithm is factorizing a full and right (or left) monic linear matrix $L \in \mathbb{F}\langle X\rangle$ into atoms when f is commutatively zero, which means there is no scalar substitution $x_i \leftarrow \alpha_i, i \in [n]$ such that $L(\alpha_1, \alpha_2, \ldots, \alpha_n)$ is invertible. Note that in this case we cannot apply the algorithm for factorizing a linear matrix as discussed in the proof of Theorem 28).

5.1 Factorization of full and monic linear matrices

Let $f \in \mathbb{F}\langle X\rangle$ be the input polynomial given by a size s formula and let $L \in \mathbb{F}\langle X\rangle^{d\times d}$ be a full, right monic linear matrix stably associated with f obtained via Higman linearization and an application of Theorem 15.

Recall, by Equation 4 we have $f \oplus I_s = PU(L \oplus I_t)SQ$ where, P, Q are respectively upper triangular and lower triangular units with diagonal entries 1, U is a unit and S is scalar invertible.

Let $L = A_0 + \sum_{i=1}^{n} A_n x_i \in \mathbb{F}\langle X\rangle^{d\times d}$ be the given full and right monic linear matrix. First, by Lemma 20, we will find a suitable scalar matrix n-tuple $\bar{M} = (M_1, M_2, \ldots, M_n)$, each $M_i \in \mathbb{F}_q^{\ell\times\ell}$ for $\ell \leq 2d$, such that under the substitution $x_i \leftarrow M_i$ the matrix $L(\bar{M})$ is invertible.

For $1 \leq i \leq n$ let Y_i be an $\ell \times \ell$ matrix of distinct noncommuting variables y_{ijk}. We consider the dilated linear matrix

$$L' = A_0 \otimes I_\ell + \sum_{i=1}^{n} A_i \otimes (Y_i + M_i). \tag{7}$$

It is not hard to see that L' is full and L' is right monic as L is right monic. Additionally, its constant term is invertible. So, we can apply Theorem 28 to factorize L' as a product of two linear matrices, both non-units.

The following lemma [14] has an important role in our algorithm for recovering the factorization for L from a factorization of L'.

▶ **Lemma 34** ([14]). *Let $L \in \mathbb{F}\langle X\rangle^{d\times d}$ be a full linear matrix with $L = A_0 + A_1x_1 + \ldots + A_nx_n$ such that $A_i \neq 0$ for at least one i, $1 \leq i \leq n$ and $L' \in R^{d\ell\times d\ell}$ be a matrix obtained from L by substituting variable x_i by Y_i for $i \in [n]$, where Y_i is $\ell \times \ell$ matrix whose $(j,k)^{th}$ entry is a fresh noncommuting variable $y_{i,j,k}$ for $1 \leq j, k \leq \ell$. Then*

1. *If L' is of the form $GL'H = \left(\begin{array}{c|c} A' & 0 \\ \hline D' & B' \end{array}\right)$, where A' is $d' \times d'$ matrix and B' is $d'' \times d''$ matrix for $0 < d', d''$, with $d' + d'' = d\ell$ and G, H are $d\ell \times d\ell$ invertible scalar matrices then there exist $d \times d$ invertible scalar matrices U, V such that $ULV = \left(\begin{array}{c|c} A & 0 \\ \hline D & B \end{array}\right)$, where A is $e' \times e'$ matrix and B is $e'' \times e''$ matrix for $0 < e', e''$, with $e' + e'' = d$.*
2. *Moreover, given L' explicitly along with its representation mentioned above, we can find the matrices U, V in deterministic polynomial time (in n, ℓ, d).*

▶ Remark 35. We give a self-contained complete proof of the above linear-algebraic lemma in the appendix for $\mathbb{F}_q$, because the proof given in [14] is sketchy in parts with some details missing, and also their lemma is stated only for complex numbers and they are not concerned about computing the matrices U and V.

Now, we can apply Lemma 34 to transform the factorization of L' to a factorization of L as a product of two linear matrices, both non-units. Repeating the above on both the factors of L we will get a complete atomic factorization of L. Formally, we prove the following.

▶ **Theorem 36.** *On input a full and right (or left) monic linear matrix $L = A_0 + \sum_{i=1}^{n} A_i x_i$ where $A_i \in \mathbb{F}^{d\times d}$ for $i \in [n]$, there is a randomized polynomial time (poly(n, d)) algorithm to compute scalar invertible matrices S, S' such that SLS' has atomic block diagonal form.*

Proof. We present the algorithm only for right monic L; the left monic case has essentially the same solution.

If the input L is not full or right monic the algorithm can efficiently detect that and output "failure". If L is an atom the algorithm will output that L is an atom and set the matrices S and S' to I_d. Otherwise, the algorithm will compute invertible scalar matrices S and S' such that

$$SLS' = \begin{pmatrix} L_1 & 0 & 0 & \dots & 0 \\ * & L_2 & 0 & \dots & 0 \\ * & * & L_3 & \dots & 0 \\ & & & \ddots & \\ * & * & * & \dots & L_r \end{pmatrix}, \tag{8}$$

where the matrix on the right is in atomic block diagonal form, that is, each linear matrix L_i is an atom.

Procedure Factor(L).

1. Test if L has full noncommutative rank using the algorithm in [17] or [13]. Test if L is right monic by checking if the matrix $[A_1 A_2 \dots A_n]$ has full row rank (which is d). If L is not full and right monic the algorithm outputs "fail".
2. Assume L is full and right monic. Using Lemma 20, find smallest positive integer $\ell \leq 2d$ and $\ell \times \ell$ scalar matrices $M_i, i \in [n]$ with entries from $\mathbb{F}$ (or a small degree extension of $\mathbb{F}$) such that $W = L(\bar{M})$ is $d \cdot \ell \times d \cdot \ell$ invertible scalar matrix. Compute the dilated linear matrix L' in the y_{ijk} variables as in Equation 7 which can be rewritten as:

$$L' = A_0 \otimes I_\ell + \sum_{i=1}^{n} A_i \otimes M_i + \sum_{i=1}^{n} \sum_{j,k=1}^{\ell} (A_i \otimes E_{jk}) \cdot y_{ijk}.$$

   Let $L'' = W^{-1}L'$. Clearly $L''(\overline{0}) = I_{d\ell}$. Hence, by the algorithm of Theorem 28 we can either detect that L'' is an atom or factorize L''. If L'' is an atom then L is also an atom and the algorithm can output that and stop. Otherwise, L' is not an atom and by Theorem 28 we will obtain a basis change matrix T such that $TW^{-1}L'T^{-1} = TL''T^{-1} = \begin{pmatrix} C'' & 0 \\ * & D'' \end{pmatrix}$ where C'' and D'' are linear matrices of dimension $c'' \times c''$ and $d'' \times d''$ respectively, such that $c'' + d'' = d\ell$.
3. By linear shift of variables $y_{ijk} \leftarrow y_{ijk} - M_i(j,k)$ we obtain $\tilde{T}\tilde{L}\tilde{T}' = \begin{pmatrix} C' & 0 \\ * & D' \end{pmatrix}$ for some scalar invertible matrices $\tilde{T}, \tilde{T}'$ where $\tilde{L} = L(Y_1, \dots, Y_n)$.
4. Applying the algorithm of Lemma 34 to $\tilde{L}$, $\tilde{T}$, and $\tilde{T}'$, in deterministic polynomial time we obtain scalar invertible matrices $\tilde{S}, \tilde{S}'$ such that $\tilde{S}L\tilde{S}' = \begin{pmatrix} C & 0 \\ * & D \end{pmatrix}$ where C, D are square matrices of dimensions $e \times e$ and $g \times g$, respectively, such that $e + g = d$.
5. Recursively call Factor(C) and Factor(D). Let S_1, S_1' be the matrices returned by Factor(C) and S_2, S_2' be the matrices returned by Factor(D).
6. Let $S = (S_1 \oplus S_2)\tilde{S}$ and $S' = \tilde{S}'(S_1' \oplus S_2')$. Return the invertible scalar matrices S and S'. Note that at this stage SLS' has the desired atomic block diagonal form.

Next we give a brief argument for proving correctness of the above algorithm. Firstly, the algorithm declares L as an atom iff L is indeed an atom. To see this, we will prove L is not an atom iff L'' is not an atom. Forward direction is obvious. To prove the reverse direction of implication, let L'' is not an atom. Which implies $L' = WL''$ is not an atom. $\tilde{L}$ is a linear matrix obtained by substituting $M_i(j,k) = 0$ for all i, j, k in L'. Clearly, $\tilde{L}$ is not an atom as L' is not an atom. Using Lemma 34 it follows that L is not an atom. So we have established L is not a atom iff L'' is not an atom. So if input linear matrix L is an atom, the algorithm will correctly declare it to be an atom in step 2.

Now we argue that we will get correct atomic block diagonal form in the last step of the algorithm. Firstly, for giving recursive calls to the Factor procedure for the matrices C, D, we must have C, D to be right monic as stated in the claim below. This is proved by the same argument as in the proof of Theorem 21.

▷ Claim 37. Let $L \in \mathbb{F}\langle X\rangle^{d\times d}$ be a full and right monic linear matrix such that $P'LQ' = \begin{pmatrix} C & 0 \\ E & D \end{pmatrix}$ where C and D are linear matrices of dimensions $e \times e$, $g \times g$, respectively, such that $e + g = d$. Then both C, D are right monic.

By recursive calls Factor(C) and Factor(D) obtain matrices S_1, S_1', S_2, S_2' such that $S_1CS_1' = C'$ and $S_2DS_2' = D'$ are in atomic block diagonal form. We can write $\tilde{S}L\tilde{S}'$ as

$$
\begin{aligned}
&= \begin{pmatrix} C & 0 \\ E & D \end{pmatrix} \\
&= \begin{pmatrix} C & 0 \\ 0 & I_g \end{pmatrix} \begin{pmatrix} I_e & 0 \\ E & I_g \end{pmatrix} \begin{pmatrix} I_e & 0 \\ 0 & B \end{pmatrix} \\
&= (S_1^{-1} \oplus I_g)(C' \oplus I_g)({S'}_1^{-1} \oplus I_g) \begin{pmatrix} I_e & 0 \\ E & I_g \end{pmatrix} (I_e \oplus S_2^{-1})(I_e \oplus D')(I_e \oplus {S'}_2^{-1}) \\
&= (S_1^{-1} \oplus I_g)(C' \oplus I_g)(I_e \oplus {S'}_2^{-1}) \begin{pmatrix} I_e & 0 \\ S_2ES_1' & I_g \end{pmatrix} ({S'}_1^{-1} \oplus I_g)(I_e \oplus D')(I_e \oplus {S'}_2^{-1}) \\
&= (S_1^{-1} \oplus I_g)(I_e \oplus S_2^{-1})(C' \oplus I_g) \begin{pmatrix} I_e & 0 \\ S_2ES_1' & I_g \end{pmatrix} (I_e \oplus D')({S'}_1^{-1} \oplus I_g)(I_e \oplus {S'}_2^{-1}) \\
&= (S_1^{-1} \oplus I_g)(I_e \oplus S_2^{-1}) \begin{pmatrix} C' & 0 \\ S_2ES_1' & D' \end{pmatrix} ({S'}_1^{-1} \oplus I_g)(I_e \oplus {S'}_2^{-1}) \\
&= (S_1^{-1} \oplus S_2^{-1}) \begin{pmatrix} C' & 0 \\ S_2ES_1' & D' \end{pmatrix} ({S'}_1^{-1} \oplus {S'}_2^{-1}).
\end{aligned}
$$

Thus we have

$$(S_1 \oplus S_2)\tilde{S}L\tilde{S}'(S_1' \oplus S_2') = \begin{pmatrix} C' & 0 \\ S_2ES_1' & D' \end{pmatrix}.$$

As C' and D' are in atomic block diagonal form, it follows that $\begin{pmatrix} C' & 0 \\ S_2ES_1' & D' \end{pmatrix}$ is also in atomic block diagonal form. Letting $S = (S_1 \oplus S_2)\tilde{S}$ and $S' = \tilde{S}'(S_1' \oplus S_2')$, it follows that SLS' is in the desired atomic block diagonal form which proves the correctness of Factor procedure. In each call to the procedure (excluding the recursive calls) the algorithm takes $\text{poly}(n, d, \log_2 q)$ time. The total number of recursive calls overall is bounded by d. Hence, the overall running time is $\text{poly}(n, d, \log_2 q)$. This completes the proof of the theorem. ◀

For the factorization of f, we assume the stably associated full linear matrix L is left monic. After we obtain atomic block diagonal form as in Equation 8, we can factorize L into atomic factors by Theorem 28. Combined with Equation 5 we have

$$f \oplus I_s = PS'F_1'F_2' \cdots F_r'U'Q,$$

where each linear matrix F_i' is an atom, P is upper triangular with all 1's diagonal, Q is lower triangular with all 1's diagonal, and S' is scalar invertible and U' is a unit. Now, applying Lemma 31 and Theorem 32 we obtain the complete factorization of f into irreducible factors. This is summarized in the following.

▶ **Theorem 38.** *Let $f \in \mathbb{F}\langle X\rangle$ be a polynomial given by an arithmetic formula as input instance of FACT($\mathbb{F}_q$). Then there is a* $\mathrm{poly}(s, \log q, |X|)$ *time randomized algorithm that outputs a factorization $f = f_1 f_2 \cdots f_r$ such that each f_i is irreducible and is output as an algebraic branching program.*

Analogous to Corollary 33, when the polynomial is given in a sparse representation, we have

▶ **Corollary 39.** *If $f \in \mathbb{F}\langle X\rangle$ is a polynomial given as input in sparse representation (that is, an $\mathbb{F}_q$-linear combination of its monomials) then in randomized polynomial time we can compute a factorization into irreducible factors in sparse representation.*

5.2 Factorization over small finite fields

Finally, we briefly discuss the factorization problem over small finite fields. As explained in Section 1.2, the two steps in our factoring algorithm requiring randomization can be replaced with deterministic $\mathrm{poly}(s, q, |X|)$ time computation. Furthermore, as explained in Section 1.2, the matrix shift $(M_1, M_2, \ldots, M_n)$ required for the Theorem 36 can be obtained in deterministic polynomial time such that the entries of the matrices M_i are from $\mathbb{F}_q$ for each i. Putting it together, it gives us a deterministic factorization algorithm for noncommutative polynomials that are input as arithmetic formulas over $\mathbb{F}_q$. In summary, we have the following.

▶ **Theorem 40.** *Given as input a multivariate polynomial $f \in \mathbb{F}_q\langle X\rangle$ for a finite field $\mathbb{F}_q$ by a noncommutative arithmetic formula of size s, a factorization of f as a product $f = f_1 f_2 \cdots f_r$ can be computed in deterministic time* $\mathrm{poly}(s, q, |X|)$*, where each $f_i \in \mathbb{F}_q\langle X\rangle$ is an irreducible polynomial that is output as an algebraic branching program.*

6 Concluding Remarks

In this paper we present a randomized polynomial-time algorithm for the factorization of noncommutative polynomials *over finite fields* that are input as *arithmetic formulas*. The irreducible factors are output as algebraic branching programs.

Several open questions arise from our work. We mention two of them. The first question is the complexity of factorization over rationals of noncommutative polynomials given as arithmetic formulas. Our approach involves the crucial use of Ronyai's algorithm for invariant subspace computation which turns out to be a hard problem over rationals. We believe a different approach may be required for the rational case.

The use of Higman linearization prevents us from generalizing this approach to noncommutative polynomials given as arithmetic circuits. We do not know any non-trivial complexity upper bound for the factorization problem for noncommutative polynomials given as arithmetic circuits.

References

1 S.A Amitsur. Rational identities and applications to algebra and geometry. *Journal of Algebra*, 3(3):304–359, 1966. doi:10.1016/0021-8693(66)90004-4.

2 S.A. Amitsur and J. Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 4(2):449–463, 1950.

3 V. Arvind, Pushkar S. Joglekar, and Gaurav Rattan. On the complexity of noncommutative polynomial factorization. *Inf. Comput.*, 262:22–39, 2018. doi:10.1016/j.ic.2018.05.009.

4 Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. A special case of rational identity testing and the brešar-klep theorem. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 10:1–10:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.10.

5 Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 92–99, 2005. doi:10.1109/CCC.2005.13.

6 W. Burnside. On the condition of reducibility of any group of linear substitutions. *Proceedings of London Mathematical Society*, 3:430–434, 1905.

7 P. M. Cohn. *Free Ideal Rings and Localization in General Rings*. New Mathematical Monographs. Cambridge University Press, 2006. doi:10.1017/CBO9780511542794.

8 P. M. Cohn. *Introduction To Ring Theory*. Springer, 2011.

9 Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

10 Harm Derksen and Visu Makam. Polynomial degree bounds for matrix semi-invariants. *Advances in Mathematics*, 310:44–63, 2017. doi:10.1016/j.aim.2017.01.018.

11 Michael A. Forbes. *Polynomial identity testing of read-once oblivious algebraic branching programs*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2014. URL: http://hdl.handle.net/1721.1/89843.

12 Katalin Friedl and Lajos Rónyai. Polynomial time solutions of some problems in computational algebra. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 153–162. ACM, 1985. doi:10.1145/22145.22162.

13 Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. Operator scaling: Theory and applications. *Found. Comput. Math.*, 20(2):223–290, 2020. doi:10.1007/s10208-019-09417-z.

14 J Helton, Igor Klep, and Jurij Volčič. Factorization of Noncommutative Polynomials and Nullstellensätze for the Free Algebra. *International Mathematics Research Notices*, 2022(1):343–372, June 2020. doi:10.1093/imrn/rnaa122.

15 Graham Higman. The units of group-rings. *Proceedings of the London Mathematical Society*, s2-46(1):231–248, 1940. doi:10.1112/plms/s2-46.1.231.

16 Pavel Hrubes and Avi Wigderson. Non-commutative arithmetic circuits with division. *Theory Comput.*, 11:357–393, 2015. doi:10.4086/toc.2015.v011a014.

17 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Non-commutative edmonds' problem and matrix semi-invariants. *Comput. Complex.*, 26(3):717–763, 2017. doi:10.1007/s00037-016-0143-x.

18 Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Constructive non-commutative rank computation is in deterministic polynomial time. *Comput. Complex.*, 27(4):561–593, 2018. doi:10.1007/s00037-018-0165-7.

19 Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Adv. Comput. Res.*, 5:375–412, 1989.

20 Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Comput.*, 9(3):301–320, 1990. doi:10.1016/S0747-7171(08)80015-6.

21 Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *Comput. Complex.*, 24(2):295–331, 2015. doi:10.1007/s00037-015-0102-y.

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

23 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.

24 Lajos Rónyai. Computing the structure of finite algebras. *J. Symb. Comput.*, 9(3):355–373, 1990. doi:10.1016/S0747-7171(08)80017-X.

25 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

26 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra (3. ed.)*. Cambridge University Press, 2013.

27 Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.

A Better-Than-$3 \log n$ Depth Lower Bound for De Morgan Formulas with Restrictions on Top Gates

Ivan Mihajlin
Leonhard Euler International Mathematical Institute in Saint Petersburg, Russia

Anastasia Sofronova
Leonhard Euler International Mathematical Institute in Saint Petersburg, Russia
St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia

Abstract

We prove that a modification of Andreev's function is not computable by $(3 + \alpha - \varepsilon) \log n$ depth De Morgan formula with $(2\alpha - \varepsilon) \log n$ layers of AND gates at the top for any $0 < \alpha < \frac{1}{5}$ and any constant $\varepsilon > 0$. In order to do this, we prove a weak variant of Karchmer-Raz-Wigderson conjecture. To be more precise, we prove the existence of two functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^n \to \{0,1\}^n$ such that $f(g(x) \oplus y)$ is not computable by depth $(1 + \alpha - \varepsilon)n$ formulas with $(2\alpha - \varepsilon)n$ layers of AND gates at the top. We do this by a top-down approach, which was only used before for depth-3 model.

Our technical contribution includes combinatorial insights into structure of composition with random boolean function, which led us to introducing a notion of well-mixed sets. A set of functions is well-mixed if, when composed with a random function, it does not have subsets that agree on large fractions of inputs. We use probabilistic method to prove the existence of well-mixed sets.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases formula complexity, communication complexity, Karchmer–Raz–Wigderson conjecture, De Morgan formulas

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.13

Related Version *Previous Version*: https://eccc.weizmann.ac.il/report/2022/033/

Funding *Ivan Mihajlin*: The work is supported by Ministry of Science and Higher Education of the Russian Federation, agreement №075-15-2019-1620.
Anastasia Sofronova: The work is supported by Ministry of Science and Higher Education of the Russian Federation, agreement №075-15-2019-1620, as well as by the Theoretical Physics and Mathematics Advancement Foundation «BASIS», agreement №21-7-1-20-4.

Acknowledgements The authors would like to thank Kaave Hosseini for his invaluable insights on Lemma 27, Alexander Kulikov, Alexander Smal and Artur Riazanov for fruitful discussions and comments on the draft. The authors would also like to thank anonymous reviewers.

1 Introduction

Proving lower bounds on Boolean formulas remains one of the fundamental problems in complexity theory. Specifically, one of the major open question here is separating classes $\mathbf{P}$ and $\mathbf{NC}^1$ by proving a super-logarithmic depth lower bound for a function from $\mathbf{P}$. The long line of prior work includes [17, 12, 1, 15, 9] up to the currently best depth lower bound $(3 - o(1)) \log n$ from the celebrated paper by Håstad [6] which stands unbeaten for two decades up to lower order terms [18].

Karchmer, Raz and Wigderson [10] proposed an approach for attacking this problem, introducing *a block composition* of two Boolean functions:

▶ **Definition 1.** *The block composition* $f \diamond g \colon (\{0,1\}^m)^n \to \{0,1\}$ *of two Boolean functions* $f \colon \{0,1\}^n \to \{0,1\}$ *and* $g \colon \{0,1\}^m \to \{0,1\}$ *is defined as follows:*

$$(f \diamond g)(x_1, \ldots, x_n) = f(g(x_1), \ldots, g(x_n))$$

where $x_i \in \{0,1\}^m$.

Let $D(f)$ be the minimal depth of a formula computing f. It is easy to see that $f \diamond g$ can be computed by a formula of depth $D(f) + D(g)$. Karchmer, Raz and Wigderson conjectured that this bound is roughly optimal.

▶ **Conjecture 2** (KRW conjecture)**.** *For any non-constant functions* $f : \{0,1\}^n \to \{0,1\}$ *and* $g : \{0,1\}^m \to \{0,1\}$*:*

$$D(f \diamond g) \approx D(f) + D(g).$$

The symbol "approximately equal" could be interpreted in a number of ways, but pretty much all reasonable interpretations, should the conjecture be proven, imply $\mathbf{P} \not\subseteq \mathbf{NC}^1$. In fact, while in the original conjecture there is $\forall$ quantifier for both f and g, the existence of such g for every f would be quite enough.

As an example, let us formulate a weaker version of conjecture, from which $\mathbf{P} \not\subseteq \mathbf{NC}^1$ would still follow.

▶ **Conjecture 3** (KRW conjecture, weaker version)**.** *There exists a constant* ε *such that for any* n *and* m *and any non-constant* $f : \{0,1\}^n \to \{0,1\}$ *there exists* $g : \{0,1\}^m \to \{0,1\}$ *such that*

$$D(f \diamond g) \geq D(f) + \varepsilon m.$$

KRW conjecture was extensively studied in a series of work [4], [7], [5], [3], [13], [2], mostly from communication complexity point of view. To present a thorough overview, we include the necessary definitions as well.

For a function $f : \{0,1\}^n \to \{0,1\}$, let KW_f (Karchmer-Wigderson game for a function f) be a communication problem, where Alice gets $x \in f^{-1}(0)$, Bob gets $y \in f^{-1}(1)$, and they need to find i such that $x_i \neq y_i$. In [11] it was observed that $D(f) = CC(\mathrm{KW}_f)$, where $CC(R)$ denotes the minimal depth of a communication protocol solving relation R.

KRW conjecture can be reformulated in those terms as $CC(\mathrm{KW}_{f \diamond g}) \approx CC(\mathrm{KW}_f) + CC(\mathrm{KW}_g)$.

Karchmer-Wigderson games have been successfully applied to a monotone setting, separating monotone $\mathbf{NC}^1$ and $\mathbf{NC}^2$ [11]. There have been attempts to tackle monotone KRW conjecture [2], where the authors introduced also a semi-monotone setup. [4, 7] proved a lower bound for a block-composition of two *universal relations*.

▶ **Definition 4.** *The universal relation is defined as follows:*

$$U_n = \{(x, y, i) \mid x, y \in \{0,1\}^n, x_i \neq y_i\} \cup \{(x, x, \bot) \mid x \in \{0,1\}^n\}.$$

In other words, for non-equal x *and* y *the task is to determine the bit of difference, and for equal strings the answer to a problem is* $\bot$*.* [1]

[1] A more popular definition does not include $\bot$ answer, but rather a promise that $x \neq y$. It is not hard to see that difference of the communication complexities of these two versions of universal relation is at most $O(\log n)$.

In a sense, universal relation generalizes KW_f for any f, since a protocol for U_n can be used to solve KW_f as well. The difference is that in U_n, inputs for players do not come from two disjoint sets. The same way as universal relation generalizes KW games for functions, their composition generalizes KW games for composition of functions.

▶ **Definition 5.** *The composition of universal relations is defined as follows:*

$$U_n \diamond U_n = \{X, x, Y, y, (i,j) \mid X, Y \in \{0,1\}^{n\times n}, x, y \in \{0,1\}^n, X_{i,j} \neq Y_{i,j}\} \cup$$
$$\cup \{X, x, Y, y, \perp \mid X, Y \in \{0,1\}^{n\times n}, x, y \in \{0,1\}^n, x = y \text{ or } \exists i\colon x_i \neq y_i, X_i = Y_i\}.$$

In other words, given two matrices X and Y and two vectors x and y, the task is to determine a bit of difference in X and Y under promise $x_i \neq y_i \Rightarrow X_i \neq Y_i$ and $x \neq y$. If the promise is broken, the answer to a problem is $\perp$.

Universal relation can also be composed with an instance of Karchmer-Wigderson game, as well as the other way around, both versions produce different generalizations of compositions of functions. $KW_f \diamond U_n$ is the same as $U_n \diamond U_n$ but with an additional property that $x \in f^{-1}(0)$, $y \in f^{-1}(1)$. $U_n \diamond KW_f$ is the same as $U_n \diamond U_n$ but now $\forall i : x_i = f(X_i), y_i = f(Y_i)$.

There does not seem to be any formula lower bounds that follow from communication lower bounds involving universal relation, but it can be considered more of a stepping stone to hone our techniques before dealing with actual function.

Currently there exist non-trivial lower bounds in the following setups:

- a lower bound on $U_n \diamond U_n$ [4, 7];
- a lower bound on $\mathrm{KW}_f \diamond U_n$ for any f [5, 13];
- a lower bound on $U_n \diamond \mathrm{KW}_g$ and $U_n \boxplus \mathrm{KW}_g$ for some g [14].

Here the operation $\boxplus$ is defined in the following way:

▶ **Definition 6** ([14])**.** *XOR-composition $f \boxplus g\colon \{0,1\}^{2n} \to \{0,1\}$ of two function $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^n \to \{0,1\}^n$ is defined as follows:*

$$(f \boxplus g)(x, y) \coloneqq f(g(x) \oplus y).$$

In the same paper, the authors stated a variant of KRW conjecture using XOR-composition instead of block-composition. This variant of the conjecture also implies $\mathbf{P} \not\subseteq \mathbf{NC}^1$. Moreover, [14] also introduced a variant of XOR-KRW regarding size of the formulas. If proved, this would imply a supercubic lower bound on a modified Andreev's function.

To outline a general idea of how different variants of KRW work, for $\mathbf{P} \not\subseteq \mathbf{NC}^1$ we need to prove a variant of conjecture of the form "$\forall f \exists g$ such that the depth of a formula for $f \circ g$ (for a reasonable definition of $\circ$) noticeably increases in comparison to a depth of a formula for f". For beating cubic size lower bound for Andreev's function, we only need to prove that "$\exists f, g$ such that the formula size for $f \circ g$ is big enough".

The next step following the lower bounds mentioned above would be getting rid of universal relation as both inner and outer parts of the composition, since for formula lower bounds of any form we need functions instead of relations there. We are able to do that with restrictions on top gates of the formula:

▶ **Theorem 7** (Main lemma)**.** *For any $0 < \alpha < \frac{1}{5}$ and any constant $0 < \varepsilon < \alpha$, with probability $1 - o(1)$ for a random function $f\colon \{0,1\}^n \to \{0,1\}$, there exists a function $g\colon \{0,1\}^n \to \{0,1\}^n$, such that $f \boxplus g$ is not computable by an AND of $2^{(2\alpha-\varepsilon)n}$ formulas of size at most $2^{(1-\alpha-\varepsilon)n}$.*

AND in the statement could be replaced by OR, and α should be separated from $\frac{1}{5}$ by an arbitrary small constant. We also obtain a new lower bound on modified Andreev's function, again with the restriction on top gates of the formula.

Our approach equips a technique from [4] of tracking a suitable subadditive measure with new combinatorial insights.

The plan of the proof of Theorem 7 could be briefly summarized as follows:

- we sample a set of functions such that any big enough subset of its compositions with f have very few zeroes in common;
- we track a certain subadditive measure while we walk down the trees of formulas for the compositions;
- we consider different subformulas obtained in this way, and argue that they cannot represent too many functions g at once, or else the measure would be too small;
- then a counting argument gives a lower bound on maximal size of such subformulas.

The paper is organized in the following way. First we give the necessary definitions and some warm-up lemmas. Then we prove Theorem 7 and derive a lower bound on modified Andreev's function from it, assuming the existence of a set of functions with required combinatorial properties. Then we prove the existence of such set in Section 4.

While in our proof we rely on the fact that the top gate of a formula has a big fan-in (which, in a setting with fan-in 2, corresponds to having a top subtree of gates of the same type and big enough depth), this restriction seems somehow artificial. We believe that there is a possibility that this method could be adapted for the general case.

2 Preliminaries

2.1 Notation

Let us recall the definition of the XOR-composition.

▶ **Definition 6** ([14]). *XOR-composition* $f \boxplus g \colon \{0,1\}^{2n} \to \{0,1\}$ *of two function* $f \colon \{0,1\}^n \to \{0,1\}$ *and* $g \colon \{0,1\}^n \to \{0,1\}^n$ *is defined as follows:*

$$(f \boxplus g)(x,y) \coloneqq f(g(x) \oplus y).$$

Let us list some notation in regard to formula complexity.

▶ **Definition 8.** *Let* $L(f)$ *be the minimum number of leaves in a formula* F *over basis* $\{\wedge, \vee, \neg\}$ *such that it computes* f*.*

Let $h(x,y)$ be a function of two variables. We denote as h^x a function: $h^x(y) \coloneqq h(x,y)$.

We also introduce the following shortcut notation for dealing with matrices. Let M be a matrix with rows indexed by X and columns indexed by Y. We denote a submatrix with rows indexed by A for $A \subseteq X$ and columns indexed by Y as M^A. An element of a matrix, located in row indexed by $x \in X$ and column indexed by $y \in Y$, is denoted as $M[x,y]$. Analogously, we denote a row indexed by x as $M[x]$.

For the rest of the paper, we consider only boolean matrices whose rows and columns are indexed by $X \coloneqq \{0,1\}^n$ and $Y \coloneqq \{0,1\}^n$.

▶ **Definition 9.** *For a function* $h \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$*, we define a matrix* M_h*:*

$$M_h[x,y] \coloneqq h(x,y).$$

▶ **Definition 10.** *For a pair of functions* $f\colon \{0,1\}^n \to \{0,1\}$ *and* $g\colon \{0,1\}^n \to \{0,1\}^n$ *we define a matrix* $M_{f,g}$*:*

$$M_{f,g}[x,y] \coloneqq f(g(x) \oplus y).$$

▶ **Definition 11.** *For a function* $f\colon \{0,1\}^n \to \{0,1\}$ *and a set of functions* Z *from* $\{0,1\}^n \to \{0,1\}^n$ *we define a matrix* $M_{f,Z}$*:*

$$M_{f,Z}[x,y] \coloneqq \bigvee_{g \in Z} f(g(x) \oplus y).$$

For the rest of the paper, $N \coloneqq 2^n$.

As we use f and g solely to denote first and second arguments of XOR-composition, we always imply the following domains and ranges for them: $f : \{0,1\}^n \to \{0,1\}$, $g : \{0,1\}^n \to \{0,1\}^n$. Analogously, x and y are implied to be vectors from $\{0,1\}^n$.

2.2 Warm-up lemmas

▶ **Lemma 12.** *For a random* f *and arbitrarily fixed* g *and* x *we have:* $L((f \boxplus g)^x) \geq N^{1-o(1)}$ *with probability* $1 - o(1)$*.*

Proof. As we fix g and x, $f(g(x) \oplus y)$ depends only on y, so let $h(y) \coloneqq f(g(x) \oplus y)$.

Any formula for h can be transformed into a formula for f by adding negations to the variables y_i for all i where $g(x)_i = 1$. But, as a random function, f does not have a formula of size less than $N^{1-o(1)}$ with high probability [16]. ◀

The next lemma gives us connection between formula complexity of a function f and the size of $f^{-1}(0)$:

▶ **Lemma 13.** *For* $f\colon \{0,1\}^n \to \{0,1\}$*, it holds that* $L(f) \leq |f^{-1}(0)| \cdot n$*.*

Proof. Let us construct a CNF formula for f. For any $x \in f^{-1}(0)$, we write a clause of n variables which becomes violated iff we substitute x to those variables. It is easy to check that a CNF formula composed exactly of $|f^{-1}(0)|$ such clauses represents function f. This formula has $|f^{-1}(0)| \cdot n$ leaves, which proves the inequality. ◀

▶ **Lemma 14.** *Let* $h(x,y)$ *be a function of two variables. Then* $L(h) \geq L(h^x)$ *for any* x*.*

Proof. Taking a formula, computing h, we get a formula, computing h^x, by hardwiring the value of x into it. ◀

3 Proof of the main theorem

In this section, we prove Theorem 7 and, as a corollary, a lower bound on modified Andreev's function.

3.1 Modified Andreev's function

▶ **Definition 15.** *Modified Andreev's function Andr′ takes* $(3\log n + 1)n$ *bits and outputs* 1*. We will treat it's inputs as:*

- *first* $2n\log n$ *bits are* $2\log n$ *strings of size* n*;*
- *next* $n\log n$ *bits represent a description of a function from* $\{0,1\}^{\log n}$ *to* $\{0,1\}^{\log n}$*;*
- *last* n *bits represent a description of a function from* $\{0,1\}^{\log n}$ *to* $\{0,1\}$*.*

$$Andr'(x_1,\dots,x_{2\log n},g,f) = (f \boxplus g)\left(\bigoplus x_1,\dots,\bigoplus x_{2\log n}\right)$$

where $\bigoplus z$ *is parity of the vector* z*.*

▶ **Theorem 16.** *Modified Andreev's function is not computable by an AND of $n^{2\alpha-\varepsilon}$ formulas of size at most $n^{3-\alpha-\varepsilon}$ for any $0 < \alpha < \frac{1}{5}$ and any $0 < \varepsilon < \alpha$.*

Again, AND in the statement could be replaced by OR.

As size of the formula is at most exponential in its depth, it immediately follows that:

▶ **Theorem 17.** *Modified Andreev's function is not computable by an a $(3+\alpha-2\varepsilon)\log n$-depth formula with $(2\alpha-\varepsilon)\log n$ layers of AND gates at the top for any $0 < \alpha < \frac{1}{5}$ and any constant $0 < \varepsilon < \alpha$.*

Note that lower bounds for different values of α are incomparable, since for increasing depth we have to pay with increasing number of same-type layers.

This beats current lower bounds on the formula depth of an explicit function, but it is conditioned on the form of the formula. To the best of our knowledge, this is the first depth lower bound for formulas with restrictions on their top gates.

Let us first show how a lower bound for a modified Andreev's function follows from Theorem 7. We will follow the classical proof of hardness for Andreev's function. We will take a look at how modified version behaves under random restriction R_p, where $p \coloneqq \frac{2\ln\log n}{n}$ and show that it both shrinks well and remains hard with high probability. The only technical difficulty we need to overcome is that we need shrinkage to occur for many subformulas simultaneously. To do this we use concentration inequality on shrinkage proved in [8].

Let R_p be a distribution on partial assignments, such that for any variable x we independently assign:

- $x \coloneqq *$ with probability p;
- $x \coloneqq 1$ with probability $(1-p)/2$;
- $x \coloneqq 0$ with probability $(1-p)/2$;

▶ **Lemma 18** ([8])**.** *For any $p \geq \frac{1}{\sqrt{(L(f))}}$:*

$$\Pr\left[L(f \mid R_p)) \geq p^2 L(f)^{1+o(1)}\right] \leq \frac{1}{L(f)^{11}}.$$

As in [8] this lemma is proved somewhat implicitly, we refer the reader to the Appendix A.1 for a more detailed explanation for which families of random restrictions their result works.

Proof of Theorem 16 from Theorem 7. We can take any pair of functions f, g and hardwire it into $Andr'$. We pick those for which $f \boxplus g$ is not computable by AND of $n^{2\alpha-\delta}$ formulas of size at most $n^{1-\alpha-\delta}$. Such functions exist due to Theorem 7.

Let us take a look at how $Andr'$ with hardwired f and g behaves under random restriction R_p, where $p \coloneqq \frac{2\ln\log n}{n}$. We have $2\log n$ blocks of n variables that serve as an input to functions f and g.

$$\Pr[\text{all variables in a block are fixed by a } R_p] = (1-p)^n =$$

$$= \left(1 - \frac{2\ln\log n}{n}\right)^n \leq e^{-2\ln\log n} = (\log n)^{-2}.$$

As there are $2\log n$ input blocks, with probability $1 - o(1)$ we have at least one variable in each block that is not fixed. We pick exactly one such variable per block and fix other variables to arbitrary values. Now as there is exactly one variable in each block which is not fixed we end up with a function that is equal to $f \boxplus g$ up to possible negation of some variables. This means that with high probability $Andr'$ is not computable by AND of $n^{2\alpha-\delta}$ formulas of size at most $n^{1-\alpha-\delta}$ under random restriction R_p.

Now suppose that modified Andreev's function equals to $\bigwedge_{i=1}^{n^{2\alpha-\varepsilon}} a_i$, where each a_i is computable by $n^{3-\alpha-\varepsilon}$ size formula for some $\varepsilon > \delta > 0$.

We prove that under restriction R_p, all a_i shrink to a size less than $n^{1-\alpha-\delta}$.

For any i we have 3 different cases depending on $L(a_i)$:

- $L(a_i) < n^{1-\alpha-\varepsilon}$. In this case we are already done, as the formula size cannot increase under random restriction.
- $p \geq \frac{1}{\sqrt{L(a_i)}}$, $L(a_i) \geq n^{1-\alpha-\varepsilon}$. In this case, we apply Lemma 18. Since $L(a_i) \leq n^{3-\alpha-\varepsilon}$, $\Pr\left[L(a_i \mid R_p) \geq n^{1-\alpha-\varepsilon+o(1)}\right] \leq L(a_i)^{-11} \leq \frac{1}{n^5}$
- $p < \frac{1}{\sqrt{L(a_i)}}$, $L(a_i) \geq n^{1-\alpha-\varepsilon}$. In this case, we invoke monotonicity of $Pr[L(a_i \mid R_p) \geq k]$ on p with fixed k. Let $q := \frac{1}{\sqrt{L(a_i)}}$, then $\Pr\left[L(a_i \mid R_p) \geq q^{2+o(1)}L(a_i)\right] \leq \Pr\left[L(a_i \mid R_q) \geq q^{2+o(1)}L(a_i)\right] \leq L(a_i)^{-11} \leq \frac{1}{n^5}$, and $q^{2+o(1)}L(a_i) = L(a_i)^{o(1)}$.

Hence, with probability $1-o(1)$ there is no i such that $L(a_i \mid R_p) \geq n^{1-\alpha-\delta} \geq n^{1-\alpha-\varepsilon+o(1)}$ for $\delta < \varepsilon$. Then with probability very close to 1 function $Andr'$ under random restriction R_p is computable by AND of $n^{2\alpha-\delta}$ formulas of size at most $n^{1-\alpha-\delta}$, which is a contradiction.

3.2 Proof of Theorem 7

Now we prove Theorem 7.

▶ **Theorem 7** (Main lemma). *For any $0 < \alpha < \frac{1}{5}$ and any constant $0 < \varepsilon < \alpha$, with probability $1 - o(1)$ for a random function $f\colon \{0,1\}^n \to \{0,1\}$, there exists a function $g\colon \{0,1\}^n \to \{0,1\}^n$, such that $f \boxplus g$ is not computable by an AND of $2^{(2\alpha-\varepsilon)n}$ formulas of size at most $2^{(1-\alpha-\varepsilon)n}$.*

To achieve that, we need to define a notion of *well-mixed set of functions*.

▶ **Definition 19** (Well-mixed set of functions). *A set of functions G from $\{0,1\}^n \to \{0,1\}^n$ is (Q, D, P)-well-mixed for f if $\forall Z \subset G, |Z| = Q$, there exists a set $K \subset \{0,1\}^n$, $|K| \leq P$, such that $M_{f,Z}^{X\setminus K}$ has no more than D zeroes in total.*

We call the set K unlucky rows*, and we call the set $X \setminus K$* lucky rows*.*

Informally, we say that a set is well-mixed if all of its subsets behave close to how a random subset of functions would behave.

The key assumption for proving Theorem 7 is that there exists large enough well-mixed set of functions G with suitable parameters.

▶ **Theorem 20.** *For $0 < \alpha < \frac{1}{5}$, let G be a family of functions $\{0,1\}^n \to \{0,1\}^n$ of size $N^{\frac{1}{4}N^{1-\alpha}}$, each sampled uniformly at random. Then G is $(N^\alpha, 2N^{2-2\alpha}, 2N^{1-\alpha})$-well-mixed set for a random function f with probability at least $1 - \frac{1}{\exp(\exp(n))}$.*

We leave proving this theorem until next section, and for now let us prove Theorem 7 under this assumption.

Let us recall that N stands for 2^n, as this notation is used heavily below.

Proof. Let us randomly pick a function f. Then we take a set of functions G such that $|G| = N^{\frac{1}{4}N^{1-\alpha}}$ and G is $(N^\alpha, 2N^{2-2\alpha}, 2N^{1-\alpha})$-well-mixed. We aim to show that $f \boxplus g$ is hard for some $g \in G$.

We assume the contrary. Suppose for all $g \in G$ the XOR-composition $f \boxplus g$ can be represented as AND of small enough formulas. Formally, for any $g \in G$:

$$f \boxplus g = \bigwedge_{i=1}^{N^{2\alpha-\varepsilon}} h_{g,i}$$

where all $h_{g,i}$ are computable by formulas of size $N^{1-\alpha-\varepsilon}$. For any g we fix the smallest formula for $f \boxplus g$ of such form, and we denote it F_g.

We define a measure $C(h)$ of any function h of two arguments:

$$C(h) = \sum_{x \in X} L(h^x).$$

Let us note that $C(f \boxplus g) \geq N \cdot L(f) \geq N^{2-o(1)}$. We prove that this measure is subadditive in the following sense:

▶ **Lemma 21.** *Let $h(x,y) = \circ(g_1, g_2, \ldots, g_k)(x,y)$, where $\circ$ is either $\wedge$ or $\vee$. Then $C(h) \leq C(g_1) + \ldots + C(g_k)$.*

Proof. If we prove that the inequality holds for any specific x, namely, $L(h^x) \leq L(g_1^x) + \ldots + L(g_k^x)$, the lemma immediately follows.

Since $h(x,y) = \circ(g_1, g_2, \ldots, g_k)(x,y)$, we can construct a formula computing h^x, applying operation $\circ$ to formulas computing each g_i^x. The inequality follows. ◀

It means that for any $g \in G$ and $F_g = \bigwedge_{i=1}^{N^{2\alpha-\varepsilon}} h_{g,i}$ we can fix $1 \leq i_g \leq N^{2\alpha-\varepsilon}$ such that for h_{g,i_g} the measure is big enough, namely:

$$C(h_{g,i_g}) \geq N^{2-2\alpha+\varepsilon-o(1)}$$

We are going to arrive at a contradiction by showing that this measure is smaller than it should be. For that, we are going to use two different upper bounds on $L(h^x_{g,i_g})$ for any x. As we assumed that formulas for h_{g,i_g} are small enough, $L(h_{g,i_g}) \leq N^{1-\alpha}$. Then for any $x \in X$:

$$L(h^x_{g,i_g}) \leq L(h_{g,i_g}) \leq N^{1-\alpha}.$$

On the other hand,

$$L(h^x_{g,i_g}) \leq |(h^x_{g,i_g})^{-1}(0)| \cdot n$$

by Lemma 13.

Now let Z_h be a subset of G such that $\forall g \in Z_h : h_{g,i_g} = h$ for some fixed h.

We consider two cases depending on the size of Z_h. First, suppose that $|Z_h| \geq N^\alpha$. We are going to arrive at a contradiction with this assumption. Without losing generality, assume $|Z_h| = N^\alpha$, as we can take a subset of Z_h of exactly this size. Since G is $(N^\alpha, 2N^{2-2\alpha}, 2N^{1-\alpha})$-well-mixed, we can consider a matrix M_{f,Z_h} and distinguish set of unlucky rows K and set of lucky rows $X \setminus K$ in it.

From the properties of well-mixed set, we know that:

- $|K| \leq 2N^{1-\alpha}$;
- there is an upper bound on number of zeroes in $M_{f,Z_h}^{X \setminus K}$, which are exactly common zeroes of all $g \in Z_h$ on $X \setminus K$:

$$\sum_{x \notin K} \Big| \bigcap_{g \in Z_h} (g^x)^{-1}(0) \Big| \leq 2N^{2-2\alpha}.$$

Let us also note that $M_h \geq M_{f,Z_h}$, since $(f \boxplus g)(x,y) = 1$ for any $g \in Z_h$ implies $h(x,y) = 1$.

$$\begin{aligned} C(h) = \sum_x L(h^x) = \sum_{x \in K} L(h^x) + \sum_{x \notin K} L(h^x) \leq \\ \leq \sum_{x \in K} L(h^x) + \sum_{x \notin K} |(h^x)^{-1}(0)| \cdot n \leq \\ \leq 2N^{1-\alpha} \cdot N^{1-\alpha} + \sum_{x \notin K} \Big| \bigcap_{g \in Z_h} (g^x)^{-1}(0) \Big| \cdot n \leq \\ \leq 2N^{2-2\alpha} + 2N^{2-2\alpha} \cdot n = N^{2-2\alpha+o(1)} \end{aligned}$$

since we have no more than $2N^{2-2\alpha}$ zeroes in all lucky rows in M_{f,Z_h}, therefore, the number of zeroes in rows $x \notin K$ in M_h does not exceed this as well. We know that $C(h) \geq N^{2-2\alpha+\varepsilon-o(1)}$ though, so this is a contradiction.

So $|Z_h| < N^\alpha$, and then the number of different h's is at least $\frac{|G|}{N^\alpha} \geq N^{\frac{1}{4}N^{1-\alpha}(1-o(1))}$.

Now let $s \coloneqq \max_h L(h)$. The number of functions with formulas of size at most s is at most $s^{s(1+o(1))}$, so $s^{s(1+o(1))} \geq N^{\frac{1}{4}N^{1-\alpha}(1-o(1))}$ and $s \log s(1+o(1)) \geq \frac{1}{4}N^{1-\alpha} \log N(1-o(1))$. Then $s > N^{1-\alpha-\varepsilon}$, and this completes the proof. ◀

4 The existence of a well-mixed set of functions

Let us restate the definition of a well-mixed set:

▶ **Definition 19** (Well-mixed set of functions). *A set of functions G from $\{0,1\}^n \to \{0,1\}^n$ is (Q,D,P)-well-mixed for f if $\forall Z \subset G, |Z| = Q$, there exists a set $K \subset \{0,1\}^n$, $|K| \leq P$, such that $M_{f,Z}^{X \setminus K}$ has no more than D zeroes in total.*

We call the set K unlucky rows*, and we call the set $X \setminus K$* lucky rows*.*

The goal of this section is to prove the following theorem:

▶ **Theorem 20.** *For $0 < \alpha < \frac{1}{5}$, let G be a family of functions $\{0,1\}^n \to \{0,1\}^n$ of size $N^{\frac{1}{4}N^{1-\alpha}}$, each sampled uniformly at random. Then G is $(N^\alpha, 2N^{2-2\alpha}, 2N^{1-\alpha})$-well-mixed set for a random function f with probability at least $1 - \frac{1}{\exp(\exp(n))}$.*

Note that we sample G as a family of functions, with possible repetitions, which we use heavily in the proof. But with probability at least $\left(1 - \frac{N^{\frac{1}{4}N^{1-\alpha}}}{N^N}\right)^{N^{\frac{1}{4}N^{1-\alpha}}} \geq \left(1 - \frac{1}{N^{\frac{3}{4}N}}\right)^{N^{\frac{1}{4}N}} \geq e^{-\frac{1}{N^{N/2}}} \geq 1 - \frac{1}{N^{N/2}}$ all functions turn out to be different from each other.

The plan of the proof is as follows:

- We identify which rows are supposed to be unlucky, and there will be two kinds of those: good and bad.
- For each kind of unlucky rows, we do three steps:
  - we prove that for a fixed row the probability to be of this kind for a random family of functions is very low;
  - we amplify this probability further, calculating the probability that some number of rows fail us simultaneously;
  - we sum the error over subsets of the random family.

The first kind of unlucky rows would be *a bad row*.

▶ **Definition 22.** *Let Z be a family of functions $\{0,1\}^n \to \{0,1\}^n$, each function sampled uniformly at random. We call x a bad row for Z, if* $\dim\left(span\left(\bigcup_{g\in Z}\{g(x)\}\right)\right) \le 2\alpha n$, *and a good row otherwise. Here the vector space is defined over $\mathcal{F}_2$ in a natural way.*

We bound the probability that a given x is bad for a random Z:

▶ **Lemma 23.** *Let Z be a random family of functions $\{0,1\}^n \to \{0,1\}^n$ of size Q. Then for a fixed x:*

$$\Pr_Z\left[\dim\left(span\left(\bigcup_{g\in Z}\{g(x)\}\right)\right) \le 2\alpha n\right] \le N^{2\alpha n}\left(\frac{1}{N^{1-2\alpha}}\right)^Q.$$

Proof. The proof is a simple counting argument. To have a vector subspace of dimension $2\alpha n$, we need to pick $2\alpha n$ generating vectors. There are $2^{2\alpha n} = N^{2\alpha}$ vectors in such space, and for each of Q functions from Z we pick its value in point x from those possibilities, versus N^Q possibilities in general case. So we have:

$$\begin{aligned}
&\Pr_Z\left[\dim\left(\operatorname{span}\left(\bigcup_{g\in Z}\{g(x)\}\right)\right) \le 2\alpha n\right] \le \\
&\le \binom{N}{2\alpha n}\left(N^{2\alpha}\right)^Q N^{-Q} \le \\
&\le N^{2\alpha n}\left(\frac{1}{N^{1-2\alpha}}\right)^Q.
\end{aligned}$$

◀

Now let us prove that with high probability, we have no more than $N^{1-\alpha}$ bad rows for a random Z.

▶ **Lemma 24.** *Let Z be as in Lemma 23. Then*

$$\Pr_Z\left[\exists V \subset X, |V| = N^{1-\alpha} : \forall x \in V : \dim\left(span\left(\bigcup_{g\in Z}\{g(x)\}\right)\right) \le 2\alpha n\right] \le N^{-(1-2\alpha)QN^{1-\alpha}(1-o(1))}$$

if $Q = \omega(n)$.

Proof. Since each function in Z is sampled uniformly at random, for each x the event of "being bad" is independent of others. So for random Z the probability that a fixed set of $N^{1-\alpha}$ x's is bad can be bounded by $\left(N^{2\alpha n}\left(\frac{1}{N^{1-2\alpha}}\right)^Q\right)^{N^{1-\alpha}}$. After that we apply a union bound over all sets of x's.

We get:

$$\begin{aligned}
N^{N^{1-\alpha}} N^{2\alpha n N^{1-\alpha}}\left(\frac{1}{N^{1-2\alpha}}\right)^{QN^{1-\alpha}} &= \left(\frac{N^{2\alpha n+1}}{N^{(1-2\alpha)Q}}\right)^{N^{1-\alpha}} = \\
&= N^{-(1-2\alpha)Q(1-\frac{2\alpha n+1}{(1-2\alpha)Q})N^{1-\alpha}} \le \\
&\le N^{-(1-2\alpha)QN^{1-\alpha}(1-o(1))}
\end{aligned}$$

for $Q = \omega(n)$. ◀

At last, we apply union bound over all choices of Z from G.

▶ **Lemma 25.** *Let G be as in Theorem 20. Then:*

$$\Pr_G\left[\exists Z \subset G, |Z| = Q, \exists V \subset X, |V| = N^{1-\alpha} \colon \forall x \in V \colon x \text{ is bad for } Z\right] \leq N^{-\frac{Q}{4}N^{1-\alpha}(1-o(1))}$$

Proof. For any specific $Z \subset G$ we can apply Lemma 24, as each function in G is sampled uniformly at random. We sum the error over all possible choices of Z and get:

$$|G|^Q N^{-(1-2\alpha)QN^{1-\alpha}(1-o(1))} = \left(\frac{N^{\frac{1}{4}N^{1-\alpha}}}{N^{(1-2\alpha)N^{1-\alpha}(1-o(1))}}\right)^Q \leq N^{-\frac{Q}{4}N^{1-\alpha}(1-o(1))}. \quad \blacktriangleleft$$

Hence, for every $Z \subset G$ there are no more than $N^{1-\alpha}$ bad rows with high probability.

We consider bad rows unlucky. After we take those out of consideration, we prove that almost all good rows are lucky.

To do that, first we consider a technical Lemma:

▶ **Lemma 26.** *Let f be a function $\{0,1\}^n \to \{0,1\}$, chosen uniformly randomly, S be any set of functions from $\{0,1\}^n \to \{0,1\}^n$ such that for a fixed x values $g(x)$ for all $g \in S$ are linearly independent, $2 \leq |S| < \frac{n}{2}$. Then:*

$$\Pr_f\left[M_{f,S}[x] \text{ has } 2N2^{-|S|} \text{ zeroes}\right] \leq 2^{(n+1)|S|-2^{n-2|S|-2}}.$$

This is a corollary from the following statement by Kaave Hosseini (from personal communication):

▶ **Lemma 27.** *Let $B \subset \{0,1\}^n$ be a random set, where each point lies with probability $\frac{1}{2}$, $L = x_1, \ldots, x_k$ be linearly independent vectors, $k \geq 2$. Then:*

$$\Pr_B\left[|(B + x_1) \cap \ldots \cap (B + x_k)| \geq 2^{n-k+1}\right] \leq 2^{(n+1)k-2^{n-2k-2}}.$$

First we show how Lemma 26 follows from Lemma 27.

Proof of 26 from 27. The value of $M_{f,S}[x, y]$ is zero iff $\forall g \in S : f(g(x) \oplus y) = 0$. Plugging $B \coloneqq f^{-1}(0)$, $k \coloneqq |S|$, $L \coloneqq \{g(x)\}_{g \in S}$ in 27, we get 26, since zeroes of $f(g(x) \oplus y)$ are shifted by vector $g(x)$ in comparison with zeroes of f.

Now we prove Lemma 27.

Proof. To upper bound $|(B + x_1) \cap \ldots \cap (B + x_k)|$, let us consider the following sum:

$$A \coloneqq \sum_{y \in \{0,1\}^n} \chi(y + x_1)\chi(y + x_2) \ldots \chi(y + x_k)$$

where χ is a characteristic function of a set B. This sum equals $|(B + x_1) \cap \ldots \cap (B + x_k)|$ exactly.

Let us now extend set $L = \{x_1, \ldots, x_k\}$ to a basis and split $\{0,1\}^n$ onto 2^k parts, depending on whether a vector contains x_i in its decomposition onto basis vectors or not. Each part contains 2^{n-k} different points. Let us consider one of those parts, without losing generality we pick:

$$P = \{y \mid y \text{ does not contain any of } x_i \text{ in decomposition}\}.$$

In the sum $A_P \coloneqq \sum_{y \in P} \chi(y + x_1) \ldots \chi(y + x_k)$ every point in $\{0,1\}^n$ occurs as an argument of χ no more than once, so all summands and all multipliers in them are independent.

If we interpret every summand as a Bernoulli variable which equals 1 with probability $\frac{1}{2^k}$, we can apply Chernoff bounds.

The exact form that we use here is the following:

▶ **Theorem 28** (Chernoff bounds, multiplicative form). *For independent random $X_1, \ldots, X_n$ from $\{0,1\}$ and $0 \leq \delta \leq 1$:*

$$\Pr\left[\sum_i X_i \geq (1+\delta)\mu\right] \leq e^{-\frac{\delta^2\mu}{3}}$$

where μ is the expected value of $\sum_i X_i$.

Here the expected value of A_P equals 2^{n-2k}.

$$\Pr_B\left[A_P > (1+\delta)2^{n-2k}\right] < e^{(-\delta^2)2^{n-2k-2}}$$

This holds regardless of the choice of a part P. We apply union bound and sum the error over all possible parts:

$$\Pr_B\left[A > (1+\delta)2^{n-k}\right] \leq \Pr_B\left[\exists P\colon A_P > (1+\delta)2^{n-2k}\right] < 2^k \cdot e^{(-\delta^2)2^{n-2k-2}}.$$

At last, we apply union bound over all possible choices of L:

$$\Pr_B\left[\exists L\colon A > (1+\delta)2^{n-k}\right] < 2^{nk} \cdot 2^k \cdot e^{(-\delta^2)2^{n-2k-2}}.$$

We pick $\delta \coloneqq 1$ and get:

$$\Pr_B\left[A > 2^{n-k+1}\right] < 2^{(n+1)k-2^{n-2k-2}}.$$

This finishes the proof of the Lemma. ◀

We use the Lemma to bound the number of zeroes in good rows. First, let us remember that α is separated from $\frac{1}{5}$ by some small constant. Let that constant be γ, so $\alpha + \gamma = \frac{1}{5}$.

▶ **Lemma 29.** *Let x be a good row for a family of functions Z. Then:*

$$\Pr_f\left[M_{f,Z}[x] \text{ has } 2N^{1-2\alpha} \text{ zeroes}\right] \leq 2^{-N^{\alpha+5\gamma}(1-o(1))}.$$

Proof. Since x is good for Z, we can find a subset $S \subset Z$ such that $|S| = 2\alpha n$ and $\{g(x)\}_{g\in S}$ are linearly independent. By Lemma 26 and the fact that $n - 2 \cdot 2\alpha n \geq (\alpha + 5\gamma)n$, we immediately have an upper bound on number of zeroes in a row x:

$$\begin{aligned}
&\Pr_f\left[M_{f,Z}[x] \text{ has } 2N^{1-2\alpha} \text{ zeroes}\right] \leq \\
&\leq \Pr_f\left[M_{f,S}[x] \text{ has } 2N^{1-2\alpha} \text{ zeroes}\right] \leq \\
&\leq 2^{(n+1)2\alpha n - 2^{(\alpha+5\gamma)n-2}} = \\
&= 2^{-N^{\alpha+5\gamma}(1-o(1))}.
\end{aligned}$$ ◀

▶ **Lemma 30.** *Let Z be a random family of functions from $\{0,1\}^n \to \{0,1\}^n$. Then:*

$$\Pr_{f,Z}\left[\exists V, V \text{ is a set of good rows for } Z, |V| = N^{1-\alpha}\colon \forall x \in V\colon M_{f,Z}[x] \text{ has } 2N^{1-2\alpha} \text{ zeroes}\right] \leq$$
$$\leq 2^{-N^{1+5\gamma}(1-o(1))}.$$

Proof. Again, as Z is picked uniformly at random, the events regarding every x are independent for a fixed V. Now let us sum the error over all choices of V:

$$\Pr_{f,Z}\left[\exists V, V \text{ is a set of good rows for } Z, |V| = N^{1-\alpha} \colon \forall x \in V \colon M_{f,Z}[x] \text{ has } 2N^{1-2\alpha} \text{ zeroes}\right] \leq$$
$$\leq \binom{N}{N^{1-\alpha}} \left(2^{-N^{\alpha+5\gamma}(1-o(1))}\right)^{N^{1-\alpha}} \leq$$
$$\leq \binom{N}{N^{1-\alpha}} 2^{-N^{1+5\gamma}(1-o(1))} \leq$$
$$\leq 2^{N^{1-\alpha}\cdot \log N} \cdot 2^{-(1-o(1))N^{1+5\gamma}} \leq$$
$$\leq 2^{-(1-o(1))N^{1+5\gamma}}.$$

◀

Now, we are ready to sum over all possible choices of Z in a random G.

▶ **Lemma 31.** *Let G be as in Theorem 20. Then:*

$$\Pr_{f}\Big[\exists Z \subset G, |Z| = Q, \exists V, V \textit{ is a set of good rows for } Z, |V| = N^{1-\alpha} \colon$$
$$\forall x \in V \colon M_{f,Z}[x] \textit{ has } 2N^{1-2\alpha} \textit{ zeroes}\Big] \leq N^{Q\frac{1}{4}N^{1-\alpha}} 2^{-(1-o(1))N^{1+5\gamma}}.$$

Proof. This is a union bound over all choices of a subset Z of size Q from set G of size $N^{\frac{1}{4}N^{1-\alpha}}$. ◀

Now we are ready to prove Theorem 20.

Let us take a family G of functions from $\{0,1\}^n \to \{0,1\}^n$ of size $N^{\frac{1}{4}N^{1-\alpha}}$, where each function is sampled uniformly at random. With probability $1 - \frac{1}{\exp(\exp(n))}$ all functions turn out to be different from each other.

Let us take any $Z \subset G$ of size N^α. We plug $Q \coloneqq N^\alpha$ into Lemma 25 and get that with probability at least $1 - N^{-N^{1-\alpha}}$ we have no more than $N^{1-\alpha}$ bad rows for our Z.

Plugging this parameter in Lemma 31, we get that with probability $1 - N^{\frac{1}{4}N}2^{-N^{1+5\gamma}(1-o(1))} = 1 - 2^{-N^{1+5\gamma}(1-o(1))}$ no more than $N^{1-\alpha}$ good rows have $2N^{1-2\alpha}$ zeroes in them.

So, with very high probability, all properties hold. We say that both bad rows and good rows with at least $2N^{1-2\alpha}$ zeroes are our unlucky rows, and this is a set K from Theorem 20.

Now, as every lucky row has at most $2N^{1-2\alpha}$ zeroes, and there are no more than N lucky rows, there are no more than $2N^{2-2\alpha}$ zeroes in all lucky rows, which gives us the statement of Theorem 20.

5 Notes and open questions

Note that our result works for α that is separated from $\frac{1}{5}$ by some constant. With more accurate analysis in Section 4 it might be possible to push α up. So the first natural question is: can we prove variant of the main theorem for $\alpha < \frac{1}{2}$?

The second question concerns the balance of the parameters. In our current result, one can say that we are trading one arbitrary layer for two AND layers. Can this trade-off be made more favorable?

But the main question that arises from our work is whether this method could be adapted to work without restrictions on top gates of the formula, or with weaker restrictions. The first natural extension would be to to prove a lower bound for AC_0 formula on top of arbitrary De Morgan formulas.

▶ **Conjecture 32.** *For any positive integer d, there exists $\alpha > 0$ and $c > 1$ such that modified Andreev's function is not computable by an AC_0 formula of depth d and size $n^{c\alpha}$ on top of $(3-\alpha)\log n$-depth De Morgan formulas.*

6 Another important statement

The contents of this section had to be modified for the reasons of safety of the authors, compared to the original version, as the recently adopted Russian laws effectively establish censorship.

Nevertheless, we are deeply upset about the events that started in Ukraine on February 24th and wish for peace more than anything.

References

1 Alexander E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow University Mathematics Bulletin*, 42(1):24–29, 1987.

2 Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, and Robert Robere. KRW composition theorems via lifting. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 43–49. IEEE, 2020. doi:10.1109/FOCS46700.2020.00013.

3 Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 3:1–3:51. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CCC.2016.3.

4 Jeff Edmonds, Russell Impagliazzo, Steven Rudich, and Jirí Sgall. Communication complexity towards lower bounds on circuit depth. *Comput. Complex.*, 10(3):210–246, 2001. doi:10.1007/s00037-001-8195-x.

5 Dmitry Gavinsky, Or Meir, Omri Weinstein, and Avi Wigderson. Toward better formula lower bounds: The composition of a function and a universal relation. *SIAM J. Comput.*, 46(1):114–131, 2017. doi:10.1137/15M1018319.

6 Johan Håstad. The shrinkage exponent of de morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

7 Johan Håstad and Avi Wigderson. Composition of the universal relation. In *Advances In Computational Complexity Theory, Proceedings of a DIMACS Workshop, New Jersey, USA, December 3-7, 1990*, pages 119–134, 1990. doi:10.1090/dimacs/013/07.

8 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 111–119. IEEE, 2012.

9 Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4:121–134, 1993.

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. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 539–550, 1988. doi:10.1145/62212.62265.

12 Valeriy Mihailovich Khrapchenko. Complexity of the realization of a linear function in the class of II-circuits. *Mathematical Notes of the Academy of Sciences of the USSR*, 9(1):21–23, 1971.

13 Sajin Koroth and Or Meir. Improved composition theorems for functions and relations. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 48:1–48:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.48.

14 Ivan Mihajlin and Alexander Smal. Toward better depth lower bounds: The XOR-KRW conjecture. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 38:1–38:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CCC.2021.38.

15 Michael S. Paterson and Uri Zwick. Shrinkage of de morgan formulae under restriction. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 324–333, USA, 1991. IEEE Computer Society. doi:10.1109/SFCS.1991.185385.

16 J. Riordan and C. Shannon. The number of two-terminal series-parallel networks. *J Math Phys*, 21, January 1942. doi:10.1002/sapm194221183.

17 Bella Abramovna Subbotovskaya. Realization of linear functions by formulas using ∧, ∨, ¬. In *Doklady Akademii Nauk*, volume 136-3, pages 553–555. Russian Academy of Sciences, 1961.

18 Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 551–560, December 2014. doi:10.1109/FOCS.2014.65.

A Appendix

A.1 Notes on Shrinkage Lemma

As the formulation of Lemma 18 is proven in [8] somewhat implicitly, we give some notes on that. The following notation is consistent with [8].

Let *a p-regular distribution* on partial assignments to variables $x_1, \dots, x_n$ be the one for which for any variable x_i: $Pr[x_i = *] = p$.

For random restriction ρ, let $\mathrm{supp}(\rho) \coloneqq \{x_i \mid \rho(x_i) \neq *\}$.

[8] use a notion of an independent sequence of restrictions. While it is defined naturally on families of restrictions to which shrinkage is applicable, defining this in general can be quite technical. Below we explain the usage of this notion in [8].

The authors construct a random restriction ρ as a sequence of r k-wise independent restrictions $\rho_1, \dots, \rho_r$. First they sample ρ_1, and then ρ_2 is sampled independently on those variables x for which $\rho_1(x) = *$ and so on. Note that supports of such restrictions are not independent of each other, so formalizing this in general would require some accuracy with probability space. Nevertheless, this construction is truly independent for a sequence of uniform distributions with the same parameter.

Let us now present a general statement, which follows [8] almost to a letter.

▶ **Lemma 33** (Lemma 4.8 in [8]). *Let $\Gamma \coloneqq 2$. For a constant $c \geq 11$, $p \geq m^{-1/\Gamma}$, $r \geq 11$, a formula f on $\leq m$ variables with $L(f) = m$ and any p-regular random restriction ρ which is a sequence of r independent $(q = p^{\frac{1}{r}})$-regular k-wise independent restrictions:*

$$\Pr\left[L(f \mid \rho) \geq 2^{3c \log^{2/3} n} p^{\Gamma} m\right] \leq m^{-c}.$$

Note that in [8] authors argue the existence of such distribution and take great care of minimizing the number of random bits needed to generate it. Nevertheless, their proof works for any distribution with mentioned properties and we use it for uniform distribution.

The uniform distribution R_p can be broken up to a sequence of r distributions R_q, where $q = p^{\frac{1}{r}}$. All of them are k-wise independent for any k, so, plugging in the parameters along with $c = 11$, we get Lemma 18.

In terms of uniform distribution this statement was also mentioned in [18].

The Plane Test Is a Local Tester for Multiplicity Codes

Dan Karliner
Department of Computer Science, Tel Aviv University, Israel

Roie Salama
Department of Computer Science, Tel Aviv University, Israel

Amnon Ta-Shma
Department of Computer Science, Tel Aviv University, Israel

Abstract

Multiplicity codes are a generalization of RS and RM codes where for each evaluation point we output the evaluation of a low-degree polynomial and all of its directional derivatives up to order s. Multi-variate multiplicity codes are *locally decodable* with the natural local decoding algorithm that reads values on a random line and corrects to the closest uni-variate multiplicity code. However, it was not known whether multiplicity codes are *locally testable*, and this question has been posed since the introduction of these codes with no progress up to date. In fact, it has been also open whether multiplicity codes can be *characterized* by local constraints, i.e., if there exists a probabilistic algorithm that queries few symbols of a word c, accepts every c in the code with probability 1, and rejects every c not in the code with nonzero probability.

We begin by giving a simple example showing the line test *does not* give local characterization when $d > q$. Surprisingly, we then show the *plane test* is a *local characterization* when $s < q$ and $d < qs - 1$ for prime q. In addition, we show the s-dimensional test is a *local tester* for multiplicity codes, when $s < q$. [1] Combining the two results, we show our main result that the *plane test* is a local tester for multiplicity codes of degree $d < qs - 1$, with constant rejection probability for constant q, s.

Our technique is new. We represent the given input as a possibly very high-degree polynomial, and we show that for some choice of plane, the restriction of the polynomial to the plane is a high-degree bi-variate polynomial. The argument has to work modulo the appropriate kernels, and for that we use Grobner theory, the Combinatorial Nullstellensatz theorem and its generalization to multiplicities. Even given that, the argument is delicate and requires choosing a non-standard monomial order for the argument to work.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases local testing, multiplicity codes, Reed Muller codes

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.14

Related Version *Previous Version*: https://eccc.weizmann.ac.il/report/2022/028/

Funding *Dan Karliner*: The research leading to these results was supported by Len Blavatnik and the Blavatnik Family foundation and by the Israel Science Foundation grant number 952/18.
Roie Salama: The research leading to these results was supported by Len Blavatnik and the Blavatnik Family foundation and by the Israel Science Foundation grant number 952/18.
Amnon Ta-Shma: The research leading to these results was supported by the Israel Science Foundation grant number 952/18.

Acknowledgements We would like to thank Tali Kaufman and Noga Ron-Zewi for a stimulating discussion on the paper. In particular, we thank them for suggesting to utilize the approach for giving a new analysis of the RM characterization. Additionally, we would like to thank the referees to the submission of this paper for their insightful remarks and corrections.

[1] For a prime power $q = p^r$ the correct bound is $d < q(s - \frac{1}{p}) - 1$.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 14; pp. 14:1–14:33

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

1 Introduction

Multiplicity codes were defined in [17, 16, 9, 15] and are a generalization of RS and RM codes. The code $\mathrm{MRM}(q, m, d, s)$ has a codeword for each degree d m-variate polynomial p, and the codeword consists of the evaluation of p *and all of its directional derivatives up to order* s on $\mathbb{F}_q^m$. Thus, the length of the code is q^m (one coordinate per each evaluation point) and the alphabet size is $q^{\binom{m+s-1}{s-1}}$, consisting of one $\mathbb{F}_q$ value for each m-directional derivative of order up to s. Choosing $s = 1$ gives us the familiar RS code (when $m = 1$) and RM code (for general m). This work studies the *local* structure of multiplicity codes.

Let us begin with three (informal) definitions:

- A code $\mathcal{C}$ is *locally correctable* with t queries if there exists a randomized algorithm A that for any string w close to a codeword $c \in \mathcal{C}$, and any coordinate i, $A(w, i) = c_i$, while making at most t queries to w.[2]
- A code $\mathcal{C}$ is *locally testable* with t queries if there exists a randomized algorithm A that given a string w, decides whether w is a codeword of $\mathcal{C}$, or far away from any codeword of $\mathcal{C}$, while making at most t queries to w. Being a bit more precise, we require that the rejection probability of the algorithm on words w that are δ far from the code is at least $\min\{\alpha\delta, c\}$, for some constants $\alpha, c > 0$, and is zero on codewords. Note that α is bounded from above by t, the number of queries A makes, and may be larger than 1.
- With the same terminology as the last item, we say $\mathcal{C}$ has a local characterization if A rejects any word $w \notin \mathcal{C}$ with nonzero probability.

Local characterization is a necessary, but not sufficient, condition for local testability. In both, all codewords c pass all tests. Also, in both, any non-codeword w fails some test. However, in local testability there is an additional requirement that the rejection probability is linked to the distance from the code, and words far away from the code should have significant rejection probability.

We note that if $\mathcal{C} \subset \mathbb{F}_q^m$ is a linear code, having local characterization is equivalent to being *LDPC*, that is being defined by low weight linear constraints z_i of the form $z_i \cdot c = 0$. In this work we deal with $\mathbb{F}_q$-linear codes over a large alphabet $\mathcal{C} \subset \Sigma^m$ with $\Sigma \cong \mathbb{F}_q^r$, in which case the notions differ.

While at first it seems local correctability is stronger requirement than local testability, this is not the case because local correctability only imposes conditions on the behavior of A on words close to the code $\mathcal{C}$, while local testability also imposes conditions on the behavior of A on words w that are far away from the code $\mathcal{C}$. As a result, local correctability does not imply local testability.

Before we discuss how multiplicity codes fare with these local properties, let us first survey the extensive research done on local properties of RM codes.

Over large enough fields, RM codes have a natural and simple local characterization: A multi-variate polynomial is degree d iff for every line, its restriction to the line is a uni-variate degree d polynomial. We call this the line test characterization. The if direction is simple, while the only-if direction is more subtle and was proved in a sequence of works [8, 18, 19, 7]. Formally, when q is prime and $d < q - 1$, a multi-variate polynomial is degree d iff its restriction to all lines is a degree d polynomial. When $d = q - 1$ the assertion is clearly

[2] We say $A(w, i) = b$ if $A(w, i)$ is b with probability at least $2/3$ over the internal random coins of A.

false, as any function from $\mathbb{F}_q$ to $\mathbb{F}_q$ can be interpolated by a degree $q-1$ polynomial. For non-prime fields $\mathbb{F}_q$ with characteristic p, it was shown that when $d < q(1-\frac{1}{p})$ the line test is a characterization, whereas for $d = q(1-\frac{1}{p})$ it is not.

The small field case also attracted a lot of attention [2, 12, 11, 4, 10]. When q is prime and $q \leq d+1$ the line test is not a characterization, and the next natural candidate is the *plane* test, or more generally the k-dimensional test, where one chooses a random k dimensional affine space and tests whether the restriction of the function to it agrees with a degree d polynomial. Roughly speaking, the bottom line is that characterization by (affine) subspaces happens as soon as it makes sense. For example, when $d = q-1$ characterization by lines does not make sense, because every function on the line can be explained by a degree $q-1$ polynomials, whereas not all functions over $\mathbb{F}_q^m$ can be explained by a degree $q-1$ polynomial. Similarly, if $d \geq k(q-1)$ the k-dimensional test contains no information, because every function on a k dimensional space can be explained by a degree $k(q-1)$ polynomial. Consequently, we may define two quantities:

- The *naive characterization dimension* $\tilde{c}_{q,d} = \lceil \frac{d+1}{q-1} \rceil$, and,
- The *characterization dimension* $c_{q,d}$ which is the lowest k such that the k dimensional test locally characterizes $\mathrm{RM}(q,m,d)$.

By the reasoning above, clearly, $c_{q,d} \geq \tilde{c}_{q,d}$. The line of work cited above shows that in fact when q is prime characterization happens as soon as it is possible, namely that $c_{q,d} = \tilde{c}_{q,d}$.

When q is a prime p power, a similar phenomenon exits, but the characterization dimension should be adjusted to $c_{q,d} = \lceil \frac{d+1}{q-\frac{q}{p}} \rceil$. More precisely, the $c_{q,d}$ dimension test is a characterization, and,

▶ **Theorem 1** ([12], Theorem 4). *Let d be an integer and $q = p^n$ a prime power. If $k < \left\lceil \frac{d+1}{q-q/p} \right\rceil$ there exists a function f such that:*

- *f is not a degree d polynomial, but,*
- *The restriction of f to any k dimensional subspace can be explained by a degree d polynomial.*

We now move on testing. We may define the *RM testing dimension* $t_{q,d}$ to be the lowest k such that $\mathrm{RM}(q,m,d)$ is locally testable by the k-dimensional test. Roughly speaking, [12, 10] show that $t_{q,d} = c_{q,d}$, though here we need to mention the parameters that are associated with the rejection probability of the test. Being more precise:

▶ **Theorem 2** ([12]). *Let $k = \left\lceil \frac{d+1}{q-q/p} \right\rceil$. Given $f : \mathbb{F}_q^m \to \mathbb{F}_q$ let:*

- $\mathrm{REJ}_{k,d}(f)$ *denotes the rejection probability of the test, namely, the probability over a random k dimensional affine space, that f restricted to the space cannot be explained by a degree k polynomial, and,*
- $\delta(f, \mathrm{RM}(q,m,d))$ *be the distance of f from the code* $\mathrm{RM}(q,m,d)$.

Then

$$\mathrm{REJ}_{k,d}(f) \geq \min\left\{\alpha_0 \cdot \delta(f, \mathrm{RM}(q,m,d)), c_0\right\}.$$

where $\alpha_0 = \frac{q^k}{2}$ and $c_0 = \frac{1}{2(k+1)q^{k+1}}$.

RM codes over large enough fields are also locally decodable with q queries when $d < q$ [20]. A simple and natural local correction procedure is the following: Randomly choose a line in $\mathbb{F}_q^m$, read all the evaluations on points lying on the line, and answer according to the closest degree d univariate polynomial. Other variants exist, e.g., one may replace the line

with a low-degree curve to handle larger error, but all variants use the crucial observation that the restriction of a multi-variate degree d polynomial to a line is a degree d uni-variate polynomial.

Having said all that we turn our attention back to multiplicity codes, that are a natural generalization of RM codes. Which of the local properties of RM code are preserved in multiplicity codes?

Multi-variate multiplicity codes are locally decodable [15] when $d < qs$. The local correction procedure also involves reading the evaluations on points lying on a random line, and finding the closest degree d univariate polynomial. However, this procedure only gives partial information, and needs to be repeated several times in order to decode. Indeed, in [14] multiplicity codes previously served as a building block for the construction of the state of the art high-rate locally decodable codes.

The situation with regard to *local characterization* and *local testability* is different. The question whether multiplicity codes are locally testable is already mentioned in [13], and without local characterization there is no hope for local testability. To appreciate the problem let us try to imitate the successful line of thought attacking the RM case. Given q, m, d and s what is the trivial k for which there is no hope of characterizing the $\mathrm{MRM}(q, m, d, s)$ code by dimension k affine spaces? Stated differently, given k, for what d every table on $\mathbb{F}_q^k$ giving evaluations for the function and all directional derivatives, can be explained by a degree d polynomial? We will see that the answer to that is $d = (s-1)q + k(q-1)$. This allows for degrees d that are significantly larger than q. For example, for $k = 1$ (i.e., the line test) it allows d to go up to $sq - 2$. However, as we shall see soon, for $k = 1$ even $d = q + 1$ is too large.

In this paper we study local characterization and local testing of $\mathrm{MRM}(q, m, d, s)$ codes. The starting point is a simple example that local characterization by lines is not possible (except for extreme cases). We find what comes next very surprising. We show that local characterization by *planes* works as long as $s \leq q$ and $d < sq - \frac{q}{p}$. I.e., for a very large set of parameters (containing the parameters that are often used in multiplicity codes) the MRM characterization dimension is 2, regardless of q, m and d. Given this, the next natural goal is understanding the MRM testing dimension. Our first result here is that if k is above the RM testing dimension, then k-dimensional tests give a local MRM test. Said differently, the MRM testing dimension is no larger than the RM testing dimension. Having that it is natural to ask: Can it actually be smaller? Our second result shows that it indeed can be smaller. We show that if $d < q(s - \frac{1}{p})$ (and notice that d may get very close to qs) then the plane test is a local test (with parameters that depend on q and s). We devote the rest of the introduction to explaining our results, and discussing the new techniques developed to obtaining them. While we do not give applications of the new results, we believe our results give us new basic understandings on this important class of codes, extending the vast literature surveyed before on the corresponding question for RM codes (e.g. [2, 12, 11, 4, 10]).

1.1 Our results – I

1.1.1 The line test

We begin our journey by analyzing a natural candidate test for characterizing multiplicity codes: *the line test*. The line test adapts the standard, well known local testing algorithm for RM and checks whether a restriction to a line is a uni-variate $\mathrm{MRM}(q, 1, d, s)$ code. Specifically:

- We are given as input an evaluation table $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ where for every point $x \in \mathbb{F}_q^m$ and every directional derivative I of order up to s, we get a value in $\mathbb{F}_q$ and therefore $\Sigma_{m,s}$ is vector of $\binom{m+s-1}{s-1}$ values from $\mathbb{F}_q$, one value per each directional derivative.
- The test chooses a random line, i.e., we choose $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ uniformly at random and we define $\ell_{a,b} : \mathbb{F}_q \to \mathbb{F}_q^m$ by $\ell_{a,b}(t) = t \cdot \mathbf{a} + \mathbf{b}$.
- We then query the table T at the q points of $\mathbb{F}_q^m$ that lie on the line $\ell_{a,b}$, and we learn the function $T_{\mathbf{a},\mathbf{b}} : \mathbb{F}_q \to \Sigma_{m,s}$ defined by $T \circ \ell_{a,b}$.

Informally, we want to test whether $T_{\mathbf{a},\mathbf{b}}$ is the evaluation table of some degree $\leq d$ uni-variate polynomial and its derivatives up to order s. Formally, we should first apply a transformation $\phi_{\mathbf{a},\mathbf{b}}$ that converts multi-variate derivatives to uni-variate derivatives over the line (see Lemma 17). With this transformation at hand we test whether $\phi_{\mathbf{a},\mathbf{b}} \circ T \circ \ell_{\mathbf{a},\mathbf{b}}$ is a codeword of $\mathrm{MRM}(q, 1, d, s)$.

It is straight forward to check completeness, i.e., that the restriction to a line of an m-variate multiplicity codeword is indeed a uni-variate multiplicity codeword, and therefore an m-variate codeword passes all tests. However, it turns out soundness sometimes fails:

▶ **Theorem 3** (informal). *Fix a prime power $q = p^r$, m and $s \leq d$. Let $\mathcal{C} = \mathrm{MRM}(q, m, d, s)$.*
- *When $q \leqslant d$ the line test is NOT a local characterization for $\mathcal{C}$.*
- *When $q - \frac{q}{p} \geqslant d + 1$ the line test is a local characterization for $\mathcal{C}$.*

Formal statements appear in Theorem 49 and Corollary 52.

The first item states that the line test fails when $q \leq d$. To see that let us look at an example. Set $Q(x, y) = (x^q - x)y - x(y^q - y) = x^q y - xy^q$. Q is a degree $q+1$ homogeneous polynomial that vanishes on $\mathbb{F}_q^m$. When we restrict to the line $\ell_{\mathbf{a},\mathbf{b}}$ we get the polynomial

$$Q \circ \ell_{\mathbf{a},\mathbf{b}}(t) = Q(\mathbf{a}t + \mathbf{b}) = Q(a_1 t + b_1, a_2 t + b_2),$$

which is a degree q polynomial rather then a degree $q+1$ polynomial, because the coefficient of t^{q+1} is $Q(a_1, a_2) = 0$ because Q vanishes on $\mathbb{F}_q^2$. It therefore follows that the restriction of Q to lines behaves as a degree q polynomial, whereas Q itself is not degree q and the line test wrongly accepts Q.

We now turn to the second item. In the terminology of this paper, the case of $s = 1$ and $q \geq d + 2$ was proved in [7] and we generalize the $q \geq d + 2$ case to larger s. The second item is a special case of a more general claim, that we discuss next.

1.1.2 The k-dimensional test

We next consider the k-dimensional test, that tests whether the restriction to k-dimensional affine spaces reduces the the m-variate multiplicity code to a k-variate multiplicity code. More formally, we are given as input a table $T : \mathbb{F}_q^m \to \Sigma_{m,s}$. We choose $\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_k)$ with each $\mathbf{h}_i$ uniformly at random from $\mathbb{F}_q^m$ conditioned on $\mathbf{h}_1, \ldots, \mathbf{h}_k$ being independent and define the k-dimensional affine space $\ell_{\mathbf{h}} : \mathbb{F}_q \to \mathbb{F}_q^m$ by

$$\ell_{\mathbf{h}}(y_1, \ldots, y_k) = \mathbf{h}_0 + \sum_{i=1}^{k} y_i \mathbf{h}_i.$$

We check whether the restriction $T \circ \ell_{\mathbf{h}}$ is a k-dimensional multiplicity code, when applying the appropriate conversion $\phi_{\mathbf{h}}$ (see Lemma 19), i.e., we check whether $\phi_{\mathbf{h}} \circ T \circ \ell_{\mathbf{h}} : \mathbb{F}_q^k \to \Sigma_{k,s}$ is a codeword of $\mathrm{MRM}(q, k, d, s)$. As before completeness is easy, and the big question is whether soundness holds. In Section 5 we prove:

▶ **Theorem 4.** *Let $\mathbb{F}_q$ be a field of size q, and assume $s \leq \min\{d, q-1\}$. Suppose for $\mathrm{RM}(q,m,d)$ there exists $\alpha > 0$ and $c_0 \leq 1$ such that for every f*

$$\mathrm{REJ}^{\mathrm{RM}}_{k,d}(f) \geq \min\{\alpha \cdot \delta(f, \mathrm{RM}(q,m,d)), c_0\}. \tag{1}$$

Then, for every T we have

$$\mathrm{REJ}^{\mathrm{MRM}}_{k,d}(T) \geq \min\{\alpha' \cdot \delta(T, \mathrm{MRM}(q,m,d,s)), c_0\} \tag{2}$$

for

$$\alpha' = \frac{q-(s-1)}{q} \frac{1}{1+q^{d/(q-1)}\frac{1}{\alpha}}. \tag{3}$$

We have used $\mathrm{REJ}^{\mathrm{RM}}_{k,d}(f)$ to denote the rejection probability of the RM k-dimensional test on f, and $\mathrm{REJ}^{\mathrm{MRM}}_{k,d}(T)$ to denote the rejection probability of the MRM k-dimensional test on T.

A consequence of the theorem is that if k is above the RM testing dimension, then, automatically, the k dimensional MRM test gives local testing for MRM codes. For example, if q is prime and $d < 2(q-1)$, the RM testing dimension is 2 and by Theorem 2 the plane test satisfies Equation (1) with $\alpha = \frac{q^2}{2}$ and $c_0 = \frac{1}{6q^3}$. Hence, by the theorem, the plane test is a local testing procedure for $\mathrm{MRM}(q,m,d,s)$, for any $s < q$, with $\alpha' \geq \frac{1}{3q}$ in Equation (2), as $d/(q-1) < 2$. If $s \leq \frac{q}{2}$ we have $\alpha' \geq \frac{1}{6}$. In essence this means that if k is above the Reed Muller testing dimension, then the k-dimensional test is also a local testing algorithm for $\mathrm{MRM}(q,m,d,s)$ for any s as large as $q-1$, and if, say, $s < q/2$ we even get constant α'. Generally, if $d < k(q-1)$, the k-dimensional test is a local testing procedure for $\mathrm{MRM}(q,m,d,s)$ for s up to $q-1$. Item 2 in the previous subsection is the special case of this theorem when we pick $k = 1$ (i.e., we consider the line test) and we replace the testing property with the weaker characterization property.

The proof idea is as follows. Let us first consider characterization. When k is above the testing dimension, the evaluation of the function itself that are given in the input T, without the evaluations of the derivatives that are given in T, suffice to uniquely characterize the function. More precisely, if all the line tests pass the MRM test, then in particular the restriction of the function to all lines is a degree d polynomial. Then, by Theorem 2 the function itself is a degree d polynomial (because k is above the testing dimension). Let us call this polynomial P. This polynomial is the only possible candidate for a MRM explanation of the given input. What remains to show is that the given values of the derivatives in the input T are consistent with the derivatives of the global function P. To explain the problem, notice that every successful dimension k test gives us information about k-variate derivatives in P and T, while we need to claim about m-variate derivatives in P and T.

The crux of the solution uses the fact that given a specific k dimensional subspace H, the k-variate derivatives of $T|_H$ at the point x are a *linear* combination of the m-variate derivatives at the point x. We then show that the set of these linear combinations derived from observing the k-variate derivatives at x for different subspaces H form a good code. Thus, if a point $x \in \mathbb{F}_q^m$ is good in the sense that many of the tests passing through it are good, then we get a codeword which is close to the codeword obtained by taking the restrictions for P. When these two codewords are closer than the distance of this code, we know the m-variate derivatives given in P and T coincide. A similar (but technically more complicated) approach proves the local testing version, giving the theorem.

The theorem is satisfying in that it gives a local testing procedure for multiplicity codes. However, a natural question arises: Claim 44 shows every table $T : \mathbb{F}_q^k \to \Sigma_{k,s}$ can be explained by a degree $(s-1)q + k(q-1)$ polynomial P (meaning that $\mathrm{EVAL}(P) = T$).

Thus, if our strategy is to use the k-dimensional test to restrict a $\mathrm{MRM}(q,m,d,s)$ code to a $\mathrm{MRM}(q,k,d,s)$ code, then this approach breaks down when $d \geq (s-1)q + k(q-1)$. Thus, we can define the *naive MRM characterization dimension* to be $\lceil \frac{d+1-(s-1)q}{q-1} \rceil$, which is the minimal k needed for this approach to have chances to work. We have seen that the $\lceil \frac{d+1}{q-q/p} \rceil$-dimensional test is a local testing procedure for $\mathrm{MRM}(q,d,m,s)$, but considering the naive bound we must consider whether this is, perhaps, a gross overkill. After all, as far as the naive bound is concerned even the line test might have worked. True, we have already seen that the line test does not give a characterization. Yet, is it possible that the plane test already gives a characterization, or, perhaps, even a local test?

1.1.3 The plane test

So next we analyze the *plane* test, that tests whether the restriction to two-dimensional planes reduces the the m-variate multiplicity code to a two-dimensional multiplicity code. More formally, we are given as input a table $T : \mathbb{F}_q^m \to \Sigma_{2,s}$. We choose $\mathbf{a}, \mathbf{b}, \mathbf{c}$ uniformly at random from $\mathbb{F}_q^m$ conditioned on $\mathbf{a}, \mathbf{b}$ being linearly independent and define the plane $\ell_{\mathbf{a},\mathbf{b},\mathbf{c}} : \mathbb{F}_q \to \mathbb{F}_q^m$ by $\ell_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = t \cdot \mathbf{a} + r \cdot \mathbf{b} + \mathbf{c}$. We check whether the restriction $T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}}$ is a two-dimensional multiplicity code, when applying the appropriate conversion $\phi_{\mathbf{a},\mathbf{b},\mathbf{c}}$ (see Lemma 18), i.e., we check whether $\phi_{\mathbf{a},\mathbf{b},\mathbf{c}} \circ T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}} : \mathbb{F}_q^2 \to \Sigma_{2,s}$ is a codeword of $\mathrm{MRM}(q,2,d,s)$. As before completeness is easy, and the big question is whether soundness holds. Our main result is the surprising Theorem 9 which shows that the plane test is a good local tester with nontrivial soundness. The main ingredients in the proof are the reduction from RM local testing explained in the previous section, as well as the following local characterization result:

▶ **Theorem 5** (The plane test - informal). *Fix q, m, d, s such that q is a power of the prime p and $s \leq q$. Suppose $d < (s - \frac{1}{p})q$. Let $\mathcal{C} = \mathrm{MRM}(q,m,d,s)$. Then, the plane test is a local characterization for $\mathcal{C}$.*

See Section 6 for a formal statement. We mention that we do not know if the condition $s \leq q$ is redundant or not.

1.1.4 Why are planes better than lines?

In this subsection, we present an informal discussion on why the line test fails, while the plane test doesn't.

So far, we have seen Theorem 3 which shows that the degree up to which the line test is a local characterization is not much better than in the Reed-Muller case. In contrast, Theorem 5 shows that when $s < p$, the plane test is effective up to degree $(s - \frac{1}{p})q$, which is close to the highest degree we could have hoped for - $(s-1)q + 2(q-1)$. The failure of the line test is demonstrated by the polynomial

$$Q(x,y) = x^q y - y x^q.$$

This polynomial is a special case of the *Moore determinant*

▶ **Definition 6** (Moore determinant). *Let $n \geq 1$ be an integer. The order n Moore matrix M_n is a matrix over $\mathbb{F}_q[x_1, \ldots, x_n]$ given by*

$$(M_n)_{i,j} = x_i^{q^j}.$$

The order n Moore determinant D_n is $\det(M_n)$.

For example,

$$M_3 = \begin{pmatrix} x_1 & x_1^q & x_1^{q^2} \\ x_2 & x_2^q & x_2^{q^2} \\ x_3 & x_3^q & x_3^{q^2} \end{pmatrix}.$$

The determinant D_3 is a degree q^2+q+1 polynomial in 3 variables. It can be shown similarly to Theorem 3 that this polynomial is of degree at most q^2+q when restricted to any plane.

However, for this polynomial to not be equivalent (in the sense of having the same values and derivatives) to a lower degree polynomial it is required that $s \geq q+2$.

Therefore, D_3 demonstrates the plane test stops being effective, but only does so for quite large s. For the line test, this failure already happens at $s=2$ with the example $Q = D_2$. This means it's never significantly more useful than in the Reed-Muller case $s=1$.

We devote the next part of the introduction to an informal explanation of our approach and technique for proving the planet test characterization theorem, Theorem 5.

1.2 A warm-up proof for RM codes

As a warm-up towards the proof we first prove the line test is a characterization for RM codes (i.e., when $s=1$) and q is prime. This claim is well known. It appears as a well known claim in Rubinfeld Sudan [19] but only for the case $q \geq 2d+1$. In [7] another proof is given that holds for all fields $\mathbb{F}_q$ of characteristic p, as long as $(1-\frac{1}{p})q \geq d+1$. In particular, when q is prime, the proof works for all $q \geq d+2$. As mentioned above, [7] also show the bound is tight, i.e., that if d is such that $d+1 > (1-\frac{1}{p})q$, then the line test is not a characterization. Here, when q is prime we give yet another proof of the claim that is somewhat simpler than the one in [7] and will be easier to generalize to larger s. Other proofs exist, see, e.g., [11, Section 1.4].

The starting point is the same as in [7]. Suppose $T : \mathbb{F}_q^m \to \mathbb{F}_q$ is some function. There exists a polynomial $P : \mathbb{F}_q^m \to \mathbb{F}_q$ in $\mathbb{F}_q[X_1, \ldots, X_m]$ that agrees with T on $\mathbb{F}_q^m$. The polynomial P is not unique, but it is unique modulo $\mathcal{I}_{m,1}$ which is the ideal of all polynomials in $\mathbb{F}_q[X_1, \ldots, X_m]$ that vanish on $\mathbb{F}_q^m$. If we choose P to be the polynomial of minimal degree agreeing with T, then from the Combinatorial Nullstellensatz (see Theorem 32) we see that we can represent P as

$$P(x_1, \ldots, x_m) = \sum_{0 \leq i_1, \ldots, i_m < q} \alpha_{i_1, \ldots, i_m} x_1^{i_1} \ldots x_m^{i_m}.$$

For ease of notation let us denote $\mathbf{I} = (i_1, \ldots, i_m)$ and $\mathbf{X}^{\mathbf{I}} = x_1^{i_1} \ldots x_m^{i_m}$. Before we go on notice that while the individual degree of P in each of the m variables is smaller than q, the total degree of P,$\deg(P)$, may be as large as $m(q-1)$ and in particular much larger than q.

When we restrict P to the line $\ell_{\mathbf{a},\mathbf{b}}(t)$ we see that:

$$P_{\mathbf{a},\mathbf{b}}(t) \stackrel{\text{def}}{=} P \circ \ell_{\mathbf{a},\mathbf{b}}(t) \; = P(\mathbf{a}t + \mathbf{b}) \; = \; \sum_{\mathbf{I}} \alpha_{\mathbf{I}} (\mathbf{a}t + \mathbf{b})^{\mathbf{I}}$$

and we can express

$$P_{\mathbf{a},\mathbf{b}}(t) \; = \; \sum_{k=0}^{\deg(P)} A_k(\mathbf{a}, \mathbf{b}) t^k,$$

where $A_k \in \mathbb{F}_q[a_1, \ldots, a_m, b_1, \ldots, b_m]$.

At first it seems our task is to show that if $\deg(P) > d$, then $A_{\deg(P)}(\mathbf{a}, \mathbf{b})$ is non-zero, and therefore for some $\mathbf{a}, \mathbf{b} \in {\mathbb{F}_q}^m$ we have $P_{\deg(P)}(\mathbf{a}, \mathbf{b}) \neq 0$. Then, when we test the line $\ell_{\mathbf{a},\mathbf{b}}$, the restricted function $P \circ \ell_{\mathbf{a},\mathbf{b}}$ is a degree $\deg(P) > d$ uni-variate polynomial, and therefore fails the line test. This argument is, however, flawed in two essential points:

1. First, we should look not at $P_{\mathbf{a},\mathbf{b}}$ but rather at $P_{\mathbf{a},\mathbf{b}} \bmod \mathcal{I}_{1,1}$, i.e., modulo the ideal of functions that vanish on $\mathbb{F}_q$. This is because from our point of view a table can be associated with a degree d polynomial iff there exists a degree d polynomial with such an evaluation table, and two polynomials that differ by an element from $\mathcal{I}_{1,1}$ have the same valuation table. The ideal $\mathcal{I}_{1,1}$ is generated by $g(t) = t^q - t$ and so we need to look at $P_{\mathbf{a},\mathbf{b}}(t) \bmod (t^q - t)$.
2. It is not enough to show that A_k is non-zero in $\mathbb{F}_q[a_1, \ldots, a_m, b_1, \ldots, b_m]$ but rather that A_k has a non-zero evaluation point in $\mathbb{F}_q^{2m}$. By the Combinatorial Nullstellensatz this is equivalent to $A_k \bmod \mathcal{I}_{2m,1}$ being non-zero.

The way we fix these two issues is different than [7]. We say $\mathbf{I}$ is a *maximal monomial* of P if $\deg(\mathbf{X}^{\mathbf{I}}) = \deg(P)$. We say $(\mathbf{I}_0, \mathbf{I}_1)$ is a *partition* of $\mathbf{I}$ if $\mathbf{I}_0 + \mathbf{I}_1 = \mathbf{I}$ and $\mathbf{I}_0, \mathbf{I}_1 \geq 0$, where $\mathbf{I}_0, \mathbf{I}_1 \in \mathbb{Z}^m$ and the addition and inequality are in each of the m coordinates. We also let $w(\mathbf{I})$, the *weight* of $\mathbf{I}$, be $\sum_{j=1}^{m} i_j$. We claim:

▶ **Lemma 7.** *Assume q is prime. Let $\mathbf{I}$ be a monomial of P of weight at least $d+1$ and $\mathbf{I}_0, \mathbf{I}_1$ a partition of $\mathbf{I}$ with $w(\mathbf{I}_0) = d+1$. Then $A_{d+1}(\mathbf{a}, \mathbf{b})$ is a non-zero polynomial in $\mathbb{F}_q[a_1, \ldots, b_m]$ and furthermore $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$ appears in it as a non-zero monomial.*

To see why the lemma is true first notice that the coefficient of $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$ in $(\mathbf{a}t + \mathbf{b})^{\mathbf{I}}$ is $\binom{\mathbf{I}}{\mathbf{I}_0}$ which is non-zero if q is prime (see Section 2.1 for the notation $\binom{\mathbf{I}}{\mathbf{I}_0}$)[3] and it appears as a coefficient of $t^{w(\mathbf{I}_0)} \bmod (t^q - t) = t^{d+1} \bmod (t^q - t) = t^{d+1}$. In general, other terms may contribute to the coefficient of t^{d+1}, and we should make sure none of these terms cancel the monomial $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$. For this, we notice that from $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$ we can recover $\mathbf{I}_0, \mathbf{I}_1$, and therefore $\mathbf{I} = \mathbf{I}_0 + \mathbf{I}_1$. Hence, for all $\mathbf{J} \neq \mathbf{I}$, $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$ is not obtained in $(\mathbf{a}t + \mathbf{b})^{\mathbf{J}}$. Thus, there is a *unique* way to obtain $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$, and it appears with a non-zero coefficient and therefore $A_{d+1}(\mathbf{a}, \mathbf{b})$ has the monomial $\mathbf{a}^{\mathbf{I}_0}\mathbf{b}^{\mathbf{I}_1}$ with a non-zero coefficient, and the lemma follows.

For the second issue we notice that for every k, $A_k(\mathbf{a}, \mathbf{b})$ is a polynomial in $\mathbb{F}_q[a_1, \ldots, b_m]$ with individual degree at most $q-1$, and therefore it is already reduced modulo $\mathcal{I}_{2m,1}$. We can therefore conclude that for some $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ the polynomial $P_{\mathbf{a},\mathbf{b}}(t) \bmod (t^q - t)$ is a non-zero polynomial of degree at least $d+1$, and therefore the line test fails for this choice of $\mathbf{a}, \mathbf{b}$.

1.3 The general case

We now want to explore whether we can generalize the argument to show the plane test is a characterization for $s > 1$. Suppose we are given a table T of function and derivative evaluations, $T : \mathbb{F}_q^m \to \Sigma_{m,s}$. Every table T has some (possibly high degree) polynomial P such that T is the codeword of P. The functions whose table is identically zero are those polynomials that vanish on $\mathbb{F}_q^m$ with multiplicity s. Let $\mathcal{I}_{m,s}$ denote the set of m-variate polynomials that vanish on $\mathbb{F}_q^m$ with multiplicity s. $\mathcal{I}_{m,s}$ is an ideal of the ring $\mathbb{F}_q[X_1, \ldots, X_m]$. $\mathcal{I}_{1,1}$ is the ideal of all uni-variate functions that vanish on $\mathbb{F}_q$, and is

[3] One can give an analogous argument for a prime power q, and we indeed do that for later on, but we skip it here because this is just a warm-up exercise.

generated by $g(x) = \Pi_{\alpha\in\mathbb{F}_q}(x-\alpha) = x^q - x$. Similarly, $\mathcal{I}_{1,s}$ is generated by $g(x)^s$ (and $\mathcal{I}_{1,s} = \mathcal{I}_{1,1}^s$). From the combinatorial nullstellensatz [1] one can deduce that $\mathcal{I}_{m,1}$ is generated by $\{g(x_1),\ldots,g(x_m)\}$ and a further generalization [3] shows that $\mathcal{I}_{m,s} = \mathcal{I}_{m,1}^s$ and is therefore generated by

$$\mathcal{G}_{m,s} = \left\{ g(\mathbf{X})^{\mathbf{I}} \mid w(\mathbf{I}) = s \right\},$$

where we use the notation $g(\mathbf{X})^{\mathbf{I}} = g(x_1)^{i_1}\cdot\ldots\cdot g(x_m)^{i_m}$. It turns out that $\mathcal{G}_{m,s}$ is a Grobner basis for $\mathcal{I}_{m,s}$ (see Section 2.3). This implies that a basis for $\mathbb{F}_q[X_1,\ldots,X_m] \bmod \mathcal{I}_{m,s}$ (as a vector space) is

$$\mathcal{B}_{m,s} = \left\{ g(\mathbf{X})^{\mathbf{I}}\cdot\mathbf{X}^{\mathbf{J}} \mid (\mathbf{I},\mathbf{J}) \in \mathcal{M}_{s,q} \right\}.$$

Where $(\mathbf{I},\mathbf{J}) \in \mathcal{M}_{s,q}$ iff $w(\mathbf{I}) < s$ and $j_1,\ldots,j_m < q$. Notice that there are basis elements in $\mathcal{B}_{m,s}$ whose degree is as large as $m(q-1)+(s-1)q \gg sq$. For more details see Section 3.

We will occasionally abuse notation and refer to members of $\mathcal{B}_{m,s}$ as "monomials".

Going back to the plane test, we are given a table $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ and we want to check whether the polynomial P that represents T is a degree d polynomial or not. W.l.o.g., we can assume P is reduced modulo $\mathcal{I}_{m,s}$ and we express $P(X_1,\ldots,X_m) \in \mathbb{F}_q[X_1,\ldots,X_m] \bmod \mathcal{I}_{m,s}$ in the basis $\mathcal{B}_{m,s}$:

$$P(\mathbf{X}) = \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}}\cdot g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}.$$

We let $P_{\mathbf{a},\mathbf{b},\mathbf{c}}$ be P restricted to the plane $\ell_{\mathbf{a},\mathbf{b},\mathbf{c}}$, i.e., $P_{\mathbf{a},\mathbf{b},\mathbf{c}} \stackrel{\text{def}}{=} P\circ\ell_{\mathbf{a},\mathbf{b},\mathbf{c}} \in \mathbb{F}_q[t,r]$. We want to check whether $P_{\mathbf{a},\mathbf{b},\mathbf{c}}$ belongs to $\mathrm{MRM}(q,2,d,s)$ and we therefore take $P_{\mathbf{a},\mathbf{b},\mathbf{c}}$ modulo $\mathcal{I}_{2,s}$. We express

$$P_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) \bmod \mathcal{I}_{2,s} = \sum_{i+j<s,k,\ell<q} A_{i,j,k,\ell}(\mathbf{a},\mathbf{b},\mathbf{c})\cdot g(t)^i g(r)^j t^k r^\ell$$

Our plan is to show that if P has degree larger than d, then for some $\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0 \in \mathbb{F}_q^m$ it must be that $P_{\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0}(t,r) \bmod \mathcal{I}_{2,s}$ is a polynomial of degree larger than d, and therefore the test $\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0$ fails. Equivalently, we want to show that for some i_0,j_0,k_0,ℓ_0 with $i_0+j_0<s$ and $k_0,\ell_0<q$ it holds that:

- $A_{i_0,j_0,k_0,\ell_0}(\mathbf{a},\mathbf{b},\mathbf{c}) \bmod \mathcal{I}_{3m,1}$ is non-zero, and therefore for some $\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0 \in \mathbb{F}_q^m$ we have $A_{i_0,j_0,k_0,\ell_0}(\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0) \neq 0$ and the monomial $g(t)^{i_0}g(r)^{j_0}t^{k_0}r^{\ell_0}$ survives, and,
- $q\cdot(i_0+j_0)+k_0+\ell_0 > d$ and therefore $\deg(P_{\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0}) > d$.

The crux of the proof is finding an order on monomials under which the following lemma is true:

▶ **Lemma 8.** *If $(\mathbf{I}_{\max},\mathbf{J}_{\max})$ is such that $g(\mathbf{X})^{\mathbf{I}_{\max}}X^{\mathbf{J}_{\max}}$ is a maximal monomial in P in the monomial order, then for any partition of $\mathbf{J}_{\max}$ to $\mathbf{J}^b_{\max}+\mathbf{J}^c_{\max}$ such that $q\cdot w(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) < qs$ and $\binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \neq 0 \bmod p$, the monomial $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ appears with a non-zero coefficient at*

$$A_{w(\mathbf{I}_{\max}),\lfloor\frac{w(\mathbf{J}^b_{\max})}{q}\rfloor,0,w(\mathbf{J}^b_{\max}) \bmod q}(\mathbf{a},\mathbf{b},\mathbf{c}) \bmod \mathcal{I}_{3m,1}.$$

See Lemma 61 for a formal statement. There is no requirement for this order to be a monomial ordering in the sense usually used for Grobner bases. The proof is much more delicate than the one we presented before for the RM case (where $s=1$) and we give some essential ideas below. We omit some of the technical details, and, as a result, we do not see, e.g., why in the proof we also need the assumption $q \geq s$. The full proof appears in Section 6.

$P_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = p(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$ is a polynomial in $a_1, \ldots, a_m, b_1, \ldots, b_m, c_1, \ldots, c_m$, t and r. We first note that (recalling that $g(x) = x^q - x$)

$$g(at + br + c) = (at + br + c)^q - (at + br + c) = ag(t) + bg(r),$$

and so $g(at+br+c)$ behaves as a total degree q polynomial in t, r, and as a *linear* polynomial in $\mathbf{a} = a_1, \ldots, a_m$ and $\mathbf{b} = b_1, \ldots, b_m$. In particular $g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$ has total degree $w(\mathbf{I}) + w(\mathbf{J})$ in $\mathbf{a}, \mathbf{b}$, where by $P(\mathbf{X})(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$ we mean $P(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$.

One crucial difference between the $s = 1$ and $s > 1$ case is that now we may get monomials $\mathbf{a}^{\mathbf{I}_1}\mathbf{b}^{\mathbf{I}_2}\mathbf{c}^{\mathbf{I}_3}$ that are *not* reduced modulo $\mathcal{I}_{3m}$, e.g., $a_1^q = a_1$ is a monomial that appears in $g(X_1)X_1^{q-1}(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$. In general, if for some coordinate $j \in [m]$ we have $\mathbf{I}_j + \mathbf{J}_j \geq q$, we get (among other things) a monomial in $\mathbf{a}, \mathbf{b}, \mathbf{c}$ that gets reduced. This complicates things for us, because a monomial that has more then one "source" may cancel out.

To solve this problem we do two things:

- First, we choose a monomial order that first order monomials $g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}$ by $w(\mathbf{I}) + w(\mathbf{J})$, and then orders monomials by $w(\mathbf{I})$.
- Second, we focus on special monomials in $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and degrees of t, r. We fix $(\mathbf{I}_{\max}, \mathbf{J}_{\max})$ with maximal $w(\mathbf{I}_{\max}) + w(\mathbf{J}_{\max})$ in P, and we take some partition $\mathbf{J}^b_{\max} + \mathbf{J}^c_{\max}$ of $\mathbf{J}_{\max}$. We look at the monomial $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$.

We observe that the monomial $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ (which is reduced modulo $\mathcal{I}_{3m,1}$) is always obtained in a reduced $\mathcal{I}_{3m,1}$ form, i.e., it cannot appear as a reduction from $g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}$ for some $(\mathbf{I}, \mathbf{J}) \neq (\mathbf{I}_{\max}, \mathbf{J}_{\max})$. This is because it has maximal $w(\mathbf{I}_{\max}) + w(\mathbf{J}_{\max})$ weight, and if it was to appear as reduction from $(\mathbf{I}, \mathbf{J})$, then those $(\mathbf{I}, \mathbf{J})$ would have a higher $w(\mathbf{I}) + w(\mathbf{J})$ weight.

The monomial $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ is obtained from $g(\mathbf{X})^{\mathbf{I}_{\max}}\mathbf{X}^{\mathbf{J}_{\max}}(\mathbf{a}t+\mathbf{b}r+\mathbf{c})$ as a coefficient of $t^{qw(\mathbf{I}_{\max})}r^{w(\mathbf{J}^b_{\max})}$. We claim that this is the only way to obtain $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ as a coefficient of $t^{qw(\mathbf{I}_{\max})}r^{w(\mathbf{J}^b_{\max})}$. To see that suppose $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ is obtained as a coefficient of $t^{qw(\mathbf{I}_{\max})}r^{w(\mathbf{J}^b_{\max})}$ from some $g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$. Since the degree in t is $qw(\mathbf{I}_{\max})$, i.e., q times the total degree in $\mathbf{a}$, it must be that the $\mathbf{a}$ part is obtained from $g(\mathbf{X})^{\mathbf{I}}(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$, because only the g part behaves as a linear function in $a \in \mathbb{F}_q$ and a degree q polynomial in t. Furthermore $w(\mathbf{I}) \geq w(\mathbf{I}_{\max})$. Since $(\mathbf{I}_{\max}, \mathbf{J}_{\max})$ is maximal and monomials that have the maximal $w(\mathbf{I}) + w(\mathbf{J})$ are then ordered by $w(\mathbf{I})$, we must have $w(\mathbf{I}) = w(\mathbf{I}_{\max})$. From that it is easy to conclude that $(\mathbf{I}, \mathbf{J}) = (\mathbf{I}_{\max}, \mathbf{J}_{\max})$.

Next, we need to force the monomial in t, r to have degree $d + 1$ when taken modulo $\mathcal{I}_{2,s}$. We take advantage of the fact that our claims work for any partition of a maximal monomial, and therefore we can shift weight from $\mathbf{c}$ to $\mathbf{b}$ in the partition of the maximal monomial. Each time we shift weight one from $\mathbf{c}$ to $\mathbf{b}$ we change the degree in t, r by one, and this is true even when working modulo $\mathcal{I}_{2,s}$. If the degree of P is larger than d, then there is a way to put just the right weight on $\mathbf{b}$ such that the resulting polynomial in t, r is degree at least $d + 1$ even when taken modulo $\mathcal{I}_{2,s}$. In fact, one can calculate explicitly how the weight should be partitioned (see Lemma 62). This concludes the proof of Lemma 8.

Having the lemma, there exist some $\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0 \in \mathbb{F}_q^{3m}$ such that P restricted to the plane $\mathbf{a}_0 t + \mathbf{b}_0 r + \mathbf{c}_0$ is degree larger than d, even after doing the reduction modulo $\mathcal{I}_{2,s}$, because the corresponding coefficient polynomial in $\mathbf{a}, \mathbf{b}, \mathbf{c}$ is non-zero modulo $\mathcal{I}_{3m,1}$. Hence the test $\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0$ fails!

1.4 Our results – II

We have seen two results so far:

- The k-dimensional test, for k above the RM testing dimension gives a MRM local tester, and,
- The planes test gives a MRM local characterization,

where for both results we assume $s < q$. We now combine the two results to show that the plane test gives a MRM local tester, with constant parameters when s is constant. We prove:

▶ **Theorem 9.** *Suppose q is a prime power, $s \leq q$ and $d < q(s - \frac{1}{p})$. Let $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ be a table and let $\delta = \delta(T, \mathrm{MRM}(q, m, d, s))$. Then*

$$\mathrm{REJ}_{2,d}^{\mathrm{MRM}}(T) \geq \min\{\alpha\delta, c\}$$

with $\alpha = \Omega(q^{-6s+5})$ and $c = \Omega(q^{-8s+4})$.

The proof idea is follows. Suppose T is far from $\mathrm{MRM}(q, m, d, s)$. By the first result mentioned above, a random k dimensional affine space H (for k above the RM testing dimension) cannot be explained a $\mathrm{MRM}(q, k, d, s)$ polynomial. Then, by the second result, some plane in H cannot be explained by a $\mathrm{MRM}(q, 2, d, s)$ polynomial. Hence, that plane rejects. The number of planes in k dimensional space is about q^{3k}, and so, intuitively, the rejection probability should be about q^{-3s} times the rejection probability of the k-dimensional test, which we already saw is quite good. We give the details in Section 7.

1.5 Organization and open problems

In Section 2 we introduce notation, recall multiplicity codes and some basic results about ideals in polynomial rings (Grobner theory and combinatorial nullstellensatz). In Section 3, we develop an understanding of the relation between tables of valuations and polynomials which are consistent with them. This is done by relying on the theory of Grobner bases [5], and the combinatorial nullstellensatz [1] and a generalization of the combinatorial nullstellensatz for multiplicities higher than one [3]. In Section 4 we prove the line test is not a characterization when $d > q + 1$. In Section 5 we prove the k-dimensional test is a MRM local tester when k is above the RM testing dimension. In Section 6 we prove the plane test is a local characterization of the MRM code, and in Section 7 we combine the two results to show the plane test is a local tester for MRM codes.

Finally, we state some open problems:

- A self-evident open problem that arises from our work is whether the parameters of the plane test in Section 7 can be made better.
- Another intriguing question is whether the condition $s \leq q$ is necessary or not. We remark that in the usual setting of the parameters $q \gg s$. Nevertheless, we think it is interesting to know whether the condition is required, and this is likely to improve our understanding of the code.
- In Theorem 49 we show that for $d \geq q + 1$ the line test is not a local characterization for $\mathrm{MRM}(q, m, d, s)$ for any $s > 1$. When $s > 1$ and $d < q(1 - \frac{1}{p})$ it follows from Section 5 that the line test is a local characterization. An open problem is pinning down where in the range $q(1 - \frac{1}{p}) \leq d < q + 1$ the line test stops being a local characterization.
- Similarly, in Claim 44 we show that the plane test has no hope of being a local characterization for $\mathrm{MRM}(q, m, d, s)$ when $d \geq q(s-1) + 2(q-1) = qs + q - 2$. When $d < q(s - \frac{1}{p})$ Theorem 60 tells us that the plane test is a local characterization. The same question can be asked about where in the range $q(s - \frac{1}{p}) \leq d < qs + q - 2$ the plane test stops being a local characterization.

- Another natural open problem, pointed to us by Tali Kaufman, is understanding the $d \gg (s - \frac{1}{p})q$ case. As we mentioned before, For RM codes (where $s = 1$) the problem attracted a lot of attention, see, e.g., [2, 12, 11, 4, 10] and it is natural to ask what happens to multiplicity codes over such small fields.

2 Preliminaries

2.1 Notation

We denote vectors by bold letters. For $\mathbf{X} = X_1, \ldots, X_m$ we denote by $\mathbb{F}[\mathbf{X}]$ the set of multivariate polynomials in the variables $X_1, \ldots X_m$. We denote by $\mathbb{F}[\mathbf{X}]^{\leqslant d}$ the set of polynomials of individual degree at most d , and by $\mathbb{F}[\mathbf{X}]^{loc\leqslant d}$ the set of polynomials of individual degree at most d (i.e degree in each variable). Given a vector $\mathbf{I} = (i_1, \ldots, i_m) \in \mathbb{N}^m$, we use the notation

$$\mathbf{X}^{\mathbf{I}} \stackrel{\text{def}}{=} \prod_{k=1}^{m} X_k^{i_k}.$$

For a vector $\mathbf{I} \in \mathbb{N}^m$, and a set $S \subseteq [m]$, we define the vector $\mathbf{I}_S$ by $(\mathbf{I}_S)_j = \begin{cases} \mathbf{I}_j, & j \in S \\ 0, & j \notin S \end{cases}$.

Recall The definition of the binomial and multinomial coefficients for natural numbers:

$$\binom{a}{b} \stackrel{\text{def}}{=} \frac{a!}{b!(a-b)!}$$

$$\binom{a}{b_1, \ldots, b_\ell} = \frac{a!}{b_1! \cdots b_\ell!}$$

where $\sum b_i = a$. We extend this definition to $I, I_1, J, J_1, \ldots, J_\ell \in \mathbb{N}^m$ where $J = J_1 + \ldots + J_\ell$ by

$$\binom{I}{I_1} \stackrel{\text{def}}{=} \prod_{k=1}^{m} \binom{I_k}{(I_1)_k}$$

$$\binom{J}{J_1, \ldots, J_\ell} \stackrel{\text{def}}{=} \prod_{k=1}^{m} \binom{J_k}{(J_1)_k, \cdots, (J_\ell)_k}.$$

We mention Lucas theorem, that if p is prime, $q = p^w$ for $w \in \mathbb{N}$, and $a, b \in \mathbb{N}$ have base p representation $a = \sum_{\ell=0}^{w} a_\ell p^\ell$, $b = \sum_{\ell=0}^{w} b_\ell p^\ell$ with $0 \leq a_\ell, b_\ell < p$, then

$$\binom{a}{b} \bmod p = \Pi_{\ell=0}^{w} \binom{a_\ell}{b_\ell}, \tag{4}$$

where we use the convention that $\binom{c}{d} = 0$ when $d > c$. Thus, $\binom{a}{b} \bmod p \neq 0$ iff $a_\ell \geq b_\ell$ for all $\ell = 0, \ldots, w-1$.

Finally, we let $g \in \mathbb{F}_q[X]$ denote the polynomial $g(X) = X^q - X$. For $\mathbf{X} = (X_1, \ldots, X_m)$ and $\mathbf{I} = (i_1, \ldots, i_m) \in \mathbb{N}^m$ we let $g(\mathbf{X})^{\mathbf{I}}$ denote $\Pi_{k=1}^{m}(g(X_k))^{i_k}$.

2.2 Reed Muller and Multiplicity codes

▶ **Definition 10.** *Let d, m be non-negative integers, and q a prime power. The (m, d, q) Reed-Muller code is defined as the set of evaluation vectors of m-variate polynomials of degree $\leqslant d$ over $\mathbb{F}_q^m$, namely,*

$$RM(q, m, d) = \left\{ (f(\alpha))_{\alpha \in \mathbb{F}_q^m} \mid f \in \mathbb{F}_q[\mathbf{X}]^{\leqslant d} \right\}.$$

We will make use of the following lemma:

▶ **Lemma 11** ([10], Lemma 3.2). *Let $\delta^{\mathrm{RM}}_{q,m,d}$ be the relative distance of the code $\mathrm{RM}(q,m,d)$. Then $\delta^{\mathrm{RM}}_{q,m,d} \geq q^{-d/(q-1)}$.*

▶ **Definition 12** (Hasse derivative). *For a multivariate $P(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ where $\mathbf{X} = (X_1, \ldots X_m)$ for some $m \in \mathbb{N}$, and a non-negative vector $\mathbf{I} \in \mathbb{N}^m$, the I-th Hasse derivative of P, denoted $P^{(\mathbf{I})}(\mathbf{X})$, is the coefficient of Z^I in the polynomial $P(\mathbf{X}, \mathbf{Z}) = P(\mathbf{X} + \mathbf{Z})$. Thus*

$$P(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{I}} P^{(\mathbf{I})}(\mathbf{X}) \cdot \mathbf{Z}^i$$

Hasse derivatives are linear. I.e, for all $P, Q \in \mathbb{F}[\mathbf{X}]$ and $\lambda \in \mathbb{F}$, $(\lambda P)^{(\mathbf{I})}(\mathbf{X}) = \lambda P^{(\mathbf{I})}(\mathbf{X})$ and $P^{(\mathbf{I})}(\mathbf{X}) + Q^{(\mathbf{I})}(\mathbf{X}) = (P+Q)^{(\mathbf{I})}(\mathbf{X})$. The product rule shows $(PQ)^{(\mathbf{I})}(\mathbf{X}) = \sum_{\mathbf{I}_0+\mathbf{I}_1=\mathbf{I}} P^{(\mathbf{I}_0)}(\mathbf{X}) \cdot Q^{\mathbf{I}_1}(\mathbf{X})$.

▶ **Definition 13** (Weight). *If $\mathbf{I} = (i_1, \ldots, i_m) \in \mathbb{N}^m$ then $w(\mathbf{I}) = \sum_{j=1}^m i_j$.*

▶ **Definition 14** (Multiplicity). *For $P(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ and $\mathbf{a} \in \mathbb{F}^m$, the multiplicity of P at $\mathbf{a}$, denoted $mult(P, \mathbf{a})$, is the largest integer s such that for every non-negative vector $\mathbf{I}$ with $w(\mathbf{I}) < s$ we have $P^{(\mathbf{I})}(\mathbf{a}) = 0$. If s may be taken arbitrarily large, we set $mult(P, \mathbf{a}) = \infty$.*

Note that by definition $mult(P, \mathbf{a}) \geqslant 0$ for every $\mathbf{a}$. One important property about multiplicities is a generalization of the Schwartz-Zippel lemma for multivariate polynomials:

▶ **Lemma 15** ([6]). *Let $P \in \mathbb{F}[\mathbf{X}]$ be a non-zero polynomial of total degree at most d. Then for any finite $A \subseteq \mathbb{F}$,*

$$\sum_{\mathbf{a} \in A^m} mult(P, \mathbf{a}) \leqslant d \cdot |A|^{m-1}.$$

▶ **Definition 16** (Multiplicity code). *Let $m, d \geq s$ be non-negative integers, and let q be a prime power. Let*

$$\Sigma_{m,s} = \mathbb{F}_q{}^{\{\mathbf{I}: w(\mathbf{I}) < s\}} \simeq \mathbb{F}_q{}^{\binom{m+s-1}{m}}.$$

For $P(\mathbf{X}) \in \mathbb{F}_q[X_1, \ldots X_m]$ we define the order s evaluation of P at $\mathbf{a}$, denoted $P^{(<s)}(\mathbf{a})$, to be the vector $(P^{(\mathbf{I})}(\mathbf{a}))_{\mathbf{I}: w(\mathbf{I}) < s} \in \Sigma_{m,s}$. The multiplicity code $\mathrm{MRM}(q,m,d,s)$ is defined as follows. The alphabet of the code is $\Sigma_{m,s}$ and the length is q^m. Every polynomial $P(\mathbf{X}) \in \mathbb{F}_q[\mathbf{X}]$ of $deg(P) \leqslant d$ defines a codeword by $\left(P^{(<s)}(\mathbf{a})\right)_{\mathbf{a}: \mathbf{a} \in \mathbb{F}_q^m} \in (\Sigma_{m,s})^{q^m}$.

We also let $\mathrm{MRS}(q,d,s) \coloneqq Mult(q,1,d,s)$ stand for Reed-Solomon multiplicity code.

The following lemma states the relationship between the derivatives of a polynomial to the derivatives of its restriction to a line.

▶ **Lemma 17** ([15], Sec 4). *Let $P \in \mathbb{F}[\mathbf{X}]$ be a multivariate polynomial where $\mathbf{X} = (X_1, \ldots, X_m)$. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^m$ and define a univariate polynomial $PL_{\mathbf{a},\mathbf{b}}(t) = P(\mathbf{a}t + \mathbf{b})$. Then*

$$PL^{(j)}_{\mathbf{a},\mathbf{b}}(t) = \sum_{\mathbf{I}: w(\mathbf{I}) = j} P^{(\mathbf{I})}(\mathbf{a}t + \mathbf{b}) \cdot \mathbf{a}^{\mathbf{I}}.$$

We also derive a formula for the derivative of a restriction to a two dimensional plane.

▶ **Lemma 18.** *Let $P \in \mathbb{F}[\mathbf{X}]$ be a multivariate polynomial where $\mathbf{X} = (X_1, \ldots, X_m)$. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^m$ and define a bivariate polynomial by $PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = P(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$. Then for $(j_1, j_2) \in \mathbb{N}^2$:*

$$PP^{(j_1,j_2)}_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = \sum_{\mathbf{I}\in\mathbb{N}^m} P^{(\mathbf{I})}(\mathbf{a}t + \mathbf{b}r + \mathbf{c}) \cdot \sum_{\substack{\mathbf{I}_1+\mathbf{I}_2=\mathbf{I} \\ w(\mathbf{I}_1)=j_1, w(\mathbf{I}_2)=j_2}} \binom{\mathbf{I}}{\mathbf{I}_1} \mathbf{a}^{\mathbf{I}_1}\mathbf{b}^{\mathbf{I}_2}.$$

Proof. Given $R_1, R_2 \in \mathbb{F}$, we write the expression $P(\mathbf{a}(t+R_1) + \mathbf{b}(r+R_2) + \mathbf{c})$ in two different ways. On the one hand,

$$\begin{aligned} P(\mathbf{a}(t+R_1) + \mathbf{b}(r+R_2) + \mathbf{c}) &= PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t+R_1, r+R_2) = PP_{\mathbf{a},\mathbf{b},\mathbf{c}}((t,r) + (R_1,R_2)) \\ &= \sum_{j_1,j_2\in\mathbb{N}} PP^{(j_1,j_2)}_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) R_1^{j_1} R_2^{j_2}. \end{aligned}$$

On the other hand,

$$\begin{aligned} P(\mathbf{a}(t+R_1) + \mathbf{b}(r+R_2) + \mathbf{c}) &= P(\mathbf{a}t + \mathbf{b}r + c + R_1\mathbf{a} + R_2\mathbf{b}) \\ &= \sum_{\mathbf{I}\in\mathbb{N}^m} P^{(\mathbf{I})}(\mathbf{a}t + \mathbf{b}r + c) \cdot (R_1\mathbf{a} + R_2\mathbf{b})^{\mathbf{I}} \\ &= \sum_{\mathbf{I}\in\mathbb{N}^m} P^{(\mathbf{I})}(\mathbf{a}t + \mathbf{b}r + c) \cdot \prod_{k=1}^{m} (a_k R_1 + b_k R_2)^{i_k} \\ &= \sum_{\mathbf{I}\in\mathbb{N}^m} P^{(\mathbf{I})}(\mathbf{a}t + \mathbf{b}r + c) \sum_{\substack{\mathbf{I}_1+\mathbf{I}_2=\mathbf{I} \\ w(\mathbf{I}_1)=j_1 \\ w(\mathbf{I}_2)=j_2}} \binom{\mathbf{I}}{\mathbf{I}_1} \mathbf{a}^{\mathbf{I}_1}\mathbf{b}^{\mathbf{I}_2} R_1^{j_1} R_2^{j_2}. \end{aligned}$$

Comparing coefficients of $R_1^{j_1} R_2^{j_2}$ for every $\mathbf{J} = (j_1, j_2) \in \mathbb{N}^2$ we get the result. ◀

We will also be interested in the restrictions of polynomials to general k-dimensional subspaces. Let $PP_{\mathbf{h}_0,\mathbf{h}_1,\ldots,\mathbf{h}_k}(\mathbf{Y}) = P(\mathbf{h}_0 + \sum_{i=1}^{k} \mathbf{h}_i Y_i)$. Then, similarly to Lemma 18,

▶ **Lemma 19.**

$$PP^{\mathbf{J}}_{\mathbf{h}_0,\mathbf{h}_1,\ldots,\mathbf{h}_k}(\mathbf{Y}) = \sum_{\mathbf{I}\in\mathbb{N}^m} P^{(\mathbf{I})}(\mathbf{h}_0 + \sum_{i=1}^{k} \mathbf{h}_i Y_i) \cdot \sum_{\substack{\mathbf{I}_1+\cdots+\mathbf{I}_k=\mathbf{I} \\ w(\mathbf{I}_r)=j_r}} \binom{\mathbf{I}}{\mathbf{I}_1, \ldots, \mathbf{I}_k} \prod_{i=1}^{k} \mathbf{h}_k^{\mathbf{I}_k}.$$

The proof is identical to the proof of Lemma 18.

2.3 Grobner bases and Nullstellensatz

The theory of Grobner bases describes the structure of ideals in the ring $R = \mathbb{F}[\mathbf{X}]$ and we briefly explain some of the essential concepts of this theory. We refer to [5] for a thorough treatment of this theory.

▶ **Definition 20.** *A monomial order $\succ$ on R is a relation $\succ$ on $\mathbb{Z}^n_{\geqslant 0}$, or equivalently a relation on the set of monomials x^α, $\alpha \in \mathbb{Z}^n_{\geqslant 0}$ satisfying:*

1. *$\succ$ is a total ordering.*
2. *If $\alpha \succ \beta$ and $\gamma \in \mathbb{Z}^n_{\geqslant 0}$ then $\alpha + \gamma \succ \beta + \gamma$.*
3. *$\succ$ is a well-ordering. I.e, every non-empty $A \subset \mathbb{Z}^n_{\geqslant 0}$ has a minimal element.*

▶ **Example 21** (Lexicographic order). Let $\alpha, \beta \in \mathbb{Z}^m_{\geqslant 0}$. We say $\alpha \succ_{lex} \beta$ if the minimal i which satisfies $\alpha_i \neq \beta_i$, also satisfies $\alpha_i > \beta_i$.

▶ **Example 22** (Total degree lexicographic order). The total degree lexicographic order is defined as follows: A monomial m_1 is greater than m_2 if it has higher total degree, where ties are broken lexicographically (i.e $X_1 > X_2 > \cdots > X_m$). More formally , let $\alpha, \beta \in \mathbb{Z}^m_{\geqslant 0}$. Then $\alpha \succ_{tot} \beta$ if $w(\alpha) = \sum \alpha_i > w(\beta) = \sum \beta_i$ or, $w(\alpha) = w(\beta)$ and $\alpha \succ_{lex} \beta$.

▶ **Definition 23.** *Let $f(\mathbf{X}) = \sum_{\mathbf{I}} a_{\mathbf{I}} \mathbf{X}^{\mathbf{I}}$ and $\succ$ a monomial order.*

1. *The multi-degree of f is $multideg(f) = \max\{\mathbf{I} \mid a_{\mathbf{I}} \neq 0\}$ where the maximum is taken w.r.t $\succ$.*
2. *The leading coefficient of f is $LC(f) = a_{multideg(f)} \in \mathbb{F}$.*
3. *The leading monomial of f is $LM(f) = \mathbf{X}^{multideg(f)}$.*
4. *The leading term of f is $LT(f) = LC(f) \cdot LM(f)$.*

▶ **Definition 24** (Multivariate polynomial division). *Let $\succ$ be a monomial order on $\mathbb{Z}^m_{\geqslant 0}$, and let $F = \{f_1, \ldots, f_k\}$ be a set of k polynomials in $\mathbb{F}[\mathbf{X}]$. Then every $f \in \mathbb{F}[\mathbf{X}]$ can be written as*

$$f = q_1 f_1 + \cdots + q_k f_k + r,$$

where $q_i, r \in \mathbb{F}[\mathbf{X}]$, and either $r = 0$ or r is a linear combination, with coefficients in $\mathbb{F}$, of monomials, none of which is divisible by any of $LT(f_1), \ldots, LT(f_k)$. We call r a remainder of the division by F. Moreover, $multideg(q_i f_i) \leqslant multideg(f)$ for every $i \in [s]$. The remainder r is not necessarily unique, and might depend on the order of division.

▶ **Definition 25.** *Let $\{0\} \neq I \subseteq \mathbb{F}[\mathbf{X}]$ be an ideal. Fix a monomial ordering on $\mathbb{F}[\mathbf{X}]$. Then*

1. *We denote by $LT(I)$ the set of leading terms of non-zero elements in I.*

$$LT(I) = \{LT(f) \mid f \in I \setminus \{0\}\}$$

2. *We denote by $\langle LT(I) \rangle$ the ideal generated by the elements in $LT(I)$.*

▶ **Definition 26.** *Let $\{0\} \neq I \subseteq \mathbb{F}[\mathbf{X}]$ be an ideal. Fix a monomial order on $\mathbb{F}[\mathbf{X}]$. A subset $G = \{g_1, \ldots, g_t\} \subset I$ is said to be a **Grobner basis** for I, if*

$$\langle LT(g_1), \ldots, LT(g_t) \rangle = \langle LT(I) \rangle.$$

▶ Note 27. Every ideal $I \subset \mathbb{F}[X_1, \ldots, X_m]$ is finitely generated and has a Grobner basis.

The importance of a Grobner basis, is that it gives a natural way of choosing representatives for the quotient space $\mathbb{F}[X_1, \ldots, X_m]/I$.

▶ **Theorem 28** ([5], Section 2, Proposition 1). *Let $I \subset \mathbb{F}[X_1, \ldots, X_m]$ be an ideal and $G = \{g_1, \ldots, g_k\}$ a Grobner basis. Then given $f \in \mathbb{F}[\mathbf{X}]$ there is a **unique** $r \in \mathbb{F}[\mathbf{X}]$ such that there is a $g \in I$ such that $f = g + r$ and no term of r is divisible by any of $LT(g_1), \ldots, LT(g_k)$. We call this r,* the reduced form of *f (relative to I).*

Note that the reduced form of any polynomial is equivalent to this polynomial modulo I. Thus, the theorem gives us a natural way of choosing representatives modulo I.

▶ **Theorem 29.** *Let $R = \mathbb{F}[\mathbf{X}]$ be the ring of polynomials, and $I \subset R$ an ideal. Let G be a Grobner basis for I. Then the set*

$$\mathcal{B} = \{M(\mathbf{X}) \mid M \text{ is a monomial not divisible by any } LT(g) \text{ for } g \in G\},$$

is a basis for R/I.

The following criterion determines whether G is a Grobner basis.

▶ **Definition 30** (LCM and S polynomials)**.** *Let $f, g \in \mathbb{F}[\mathbf{X}]$ be non-zero polynomials. Let $\alpha = multideg(f)$ and $\beta = multideg(g)$.*

1. *The least common multiple of $LM(f)$ and $LM(g)$, denoted $LCM(LM(f), LM(g))$, is $\mathbf{X}^{\boldsymbol{\gamma}}$, where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m)$ and $\gamma_i = \max\{\alpha_i, \beta_i\}$ for each i.*
2. *The $S-$ polynomial of f and g is*

$$S(f,g) = \frac{LCM(LM(f), LM(g))}{LT(f)} \cdot f - \frac{LCM(LM(f), LM(g))}{LT(g)} \cdot g.$$

▶ **Theorem 31** (Buchberger's Criterion, [5], Sec 6)**.** *Let $I \subset \mathbb{F}[\mathbf{X}]$ be an ideal. Then a basis $G = \{g_1, \ldots, g_k\}$ of I is a Grobner basis of I if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G is zero .*

Note that we always have $S = S(g_i, g_j) \in I$ by the definition of S. When saying the remainder of the division by G is zero, we mean that there are $\{f_i\}$, such that $S = \sum f_i g_i$ and $multideg(f_i g_i) \leqslant multideg(S)$ for every i (as in Definition 24).

▶ **Theorem 32** (Combinatorial Nullstellensatz, [1])**.** *Let $\mathbb{F}$ be a field, and $A_1, \ldots, A_m \subseteq \mathbb{F}$. Let $g_i(X) = \prod_{\alpha \in A_i}(X - \alpha)$ for $i = 1, \ldots, m$. Assume a polynomial $f \in \mathbb{F}[\mathbf{X}]$ satisfies $f(\alpha) = 0$ for all $\alpha \in A_1 \times \cdots \times A_m$. Then there are $h_1, \ldots, h_t$ such that*

$$f = \sum h_i g_i,$$

and $deg(h_i) + deg(g_i) \leqslant deg(f)$ for all i.

When $A_i = \mathbb{F}_q$ denote

$$g(X) = \prod_{\alpha \in \mathbb{F}_q} (X - \alpha) = X^q - X.$$

Also, let $\mathcal{I}_m$ denote the ideal $\mathcal{I}_m = \left\{ f \in \mathbb{F}_q[\mathbf{X}] \mid \forall \alpha \in \mathbb{F}_q^m \;\; f(\alpha) = 0 \right\}$.

▶ **Corollary 33.** $\mathcal{I}_m = \langle g(X_1), \ldots, g(X_m) \rangle$.

Proof. Let $f \in \mathcal{I}_m$. By Theorem 32, taking $S_i = \mathbb{F}_q$ for every i, we get that $f = \sum h_i g_i$ for some $\{h_i\}$ and so $f \in \langle g_k \rangle_{k \in [m]}$. The other inclusion is trivial since $g(X_i) = X_i^q - X_i$ vanishes on $\mathbb{F}_q^m$ for every i. ◀

Using Theorem 31 it can be easily proved that:

▷ Claim 34. $G = \{g(X_k)\}_{k=1}^m$ is a Grobner basis for $\mathcal{I}_m$ relative to the total degree lexicographic order.

Let $s \in \mathbb{N}$ and let $\mathcal{I}_{m,s}$ denote the ideal

$$\mathcal{I}_{m,s} = \{ f \in \mathbb{F}_q[\mathbf{X}] \mid \forall \alpha \in {\mathbb{F}_q}^m \;\; Mult(f; \alpha) \geq s \}.$$

In this notation, $\mathcal{I}_{m,1} = \mathcal{I}_m$ defined before. For every $\mathbf{I} = (I_1, \ldots, I_m) \in \mathbb{N}^m$ define

$$g(\mathbf{X})^{\mathbf{I}} = \prod_{k=1}^{m} g(X_k)^{I_k}.$$

▶ **Theorem 35** (Combinatorial Nullstellensatz with multiplicity, [3], Sec 3)**.** *$\mathcal{I}_{m,s} = \langle g(\mathbf{X})^{\mathbf{I}} \rangle_{w(\mathbf{I})=s}$. Furthermore, the set $\mathcal{G}_{m,s} = \left\{ g(\mathbf{X})^{\mathbf{I}} \right\}_{w(\mathbf{I})=s}$ is a Grobner basis for $\mathcal{I}_{m,s}$.*

Proof. To see that $\mathcal{I}_{m,s}$ is indeed an ideal, fix $f \in \mathcal{I}_{m,s}$ and $h \in \mathbb{F}[\mathbf{X}]$. Then for $r < s$: $(hf)^{(r)} = \sum_{i=0}^{r} f^{(i)} \cdot h^{(r-i)} = 0$ and so $hf \in \mathcal{I}_{m,s}$. Also, clearly, $g(\mathbf{X})^{\mathbf{I}} \in \mathcal{I}_{m,s}$ for every $\mathbf{I}$ with $w(\mathbf{I}) = s$. We need to show $\mathcal{G}_{m,s}$ is a Grobner basis for $\mathcal{I}_{m,s}$,

[3, Section 3] show $\mathcal{G}_{m,s}$ generates $\mathcal{I}_{m,s}$ and, furthermore, $f = \sum_{\mathbf{b}:w(\mathbf{I})=s} g(\mathbf{X})^{\mathbf{I}} h_{\mathbf{I}}$ for some $h_{\mathbf{I}}$ with $deg(h_{\mathbf{I}}) \leqslant deg(f) - s\deg(g)$. In particular, this is true for the S polynomials in Theorem 31. I.e, every such S polynomial can be expressed as $S = \sum g(\mathbf{X})^{\mathbf{I}} h_{\mathbf{I}}$ where $deg(g(\mathbf{X})^{\mathbf{I}} h_{\mathbf{I}}) \leqslant deg(S)$. By Buchberger's criterion $\left\{ g(\mathbf{X})^{\mathbf{I}} \right\}$ is a Grobner basis. ◀

Finally, we look at equality modulo $\mathcal{I}_{m,1}$.

▶ **Definition 36.** *For $n_1, n_2 \in \mathbb{N}$ we say $n_1 =_{\mathbb{F}_q} n_2$ iff $x^{n_1} = x^{n_2} \bmod \mathcal{I}_{1,1}$. Equivalently, $n_1 =_{\mathbb{F}_q} n_2$ iff*

- *$n_1 = n_2$, or,*
- *$\min(n_1, n_2) > 0$ and $n_1 = n_2 \bmod (q-1)$.*

▶ **Definition 37.** *Let $A, B \in \mathbb{N}^m$. We say $A =_{\mathbb{F}_q} B$ for iff $A_k =_{\mathbb{F}_q} B_k$ for every $1 \leq k \leq m$.*

We also record:

▷ Claim 38. Let $a, b \in \mathbb{N}$. If $a =_{\mathbb{F}_q} b$ and $a < q$ then $a \leq b$.

Proof. If $b < q$ then $a =_{\mathbb{F}_q} b$ implies $a = b$. Otherwise $a < q \leq b$. ◁

3 Polynomials and tables

A *table* is an element of $\Sigma_{m,s}^{q^m}$, i.e., a function mapping every evaluation point in $\mathbb{F}_q^m$ to an element in $\Sigma_{m,s}$. EVAL takes a multi-variate polynomial and returns its table of evaluations. More precisely,

▶ **Definition 39.** *We define $\mathrm{EVAL}_{m,s} : \mathbb{F}_q[\mathbf{X}] \times \mathbb{F}_q^m \to \Sigma_{m,s}$ by*

$$\mathrm{EVAL}_{m,s}(P; \mathbf{a}) = \left(P^{(\mathbf{I})}(a) \right)_{w(\mathbf{I})<s}.$$

Similarly, we define $\mathrm{EVAL}_{m,s} : \mathbb{F}_q[\mathbf{X}] \to (\Sigma_{m,s})^{q^m}$ by

$$\mathrm{EVAL}_{m,s}(P) = \left(\mathrm{EVAL}(P; \mathbf{a}) \right)_{\mathbf{a} \in \mathbb{F}_q^m}.$$

We say $T \in (\Sigma_{m,s})^{q^m}$ is the table *of $P \in \mathbb{F}_q[\mathbf{X}]$ if $\mathrm{EVAL}(P) = T$.*

An element in $\Sigma_{m,s}$ contains information about all the derivatives of order up to s. Sometimes we would like to focus on a certain directional derivative. If $\sigma \in \Sigma_{m,s}$ and $\mathbf{I} \in \mathbb{N}^m$ with $w(\mathbf{I}) < s$, then $\sigma_{(\mathbf{I})} \in \mathbb{F}_q$ is the entry of σ that encodes the $\mathbf{I}$'th derivative. Similarly, if $f \in \Sigma_{m,s}^{q^m}$ is a table, i.e., $f : \mathbb{F}_q^m \to \Sigma_{m,s}$, then $f_{(\mathbf{I})} : \mathbb{F}_q^m \to \mathbb{F}_q$ is defined by letting $f_{(\mathbf{I})}(x)$ be the $\mathbf{I}$'th entry of $f(x) \in \Sigma_{m,s}$.

Note that every polynomial determines its table $\mathrm{EVAL}(P)$. However, two different polynomials might have the same table if their difference is the zero table on $\mathbb{F}_q^m$. From Theorem 35,

$$\mathrm{Ker}(\mathrm{EVAL}_{m,s}) = \mathcal{I}_{m,s} = \langle g(\mathbf{X})^{\mathbf{I}} \rangle_{w(\mathbf{I})=s}.$$

Since $\mathcal{I}_{m,s}$ does not contain any non-zero polynomial of total degree less than sq, we conclude that:

▶ **Corollary 40.** $\mathrm{EVAL}_{m,s}$ *is injective on polynomials of total degree less than* sq.

▷ Claim 41. $\mathrm{EVAL}_{m,s}$ is onto $\Sigma_{m,s}^{q^m}$.

Proof. We will show that

$$\dim\left(\mathbb{F}_q[X_1,\ldots,X_m]\Big/\mathcal{I}_{m,s}\right) \geq \dim\left((\Sigma_{m,s})^{q^m}\right) = q^m\cdot\binom{m+s-1}{m},$$

where the dimension is over $\mathbb{F}_q$. This implies that the image of $\mathrm{EVAL}_{m,s}$ is everything, and the mapping is injective.

To see that consider the set

$$\mathcal{B}_{m,s} = \left\{ g(\mathbf{X})^{\mathbf{I}}\cdot\prod_{k=1}^{m} X_k^{j_k} \mid w(\mathbf{I}) < s, 0 \leqslant j_k < q \right\}. \tag{5}$$

The elements in $\mathcal{B}_{m,s}$ have different multi degree and therefore are independent. Also elements in $\mathcal{B}_{m,s}$ are monomials of degree smaller than sq, and therefore are $\mathcal{I}_{m,s}$ reduced. Thus,

$$\begin{aligned}\dim\left(\mathbb{F}_q[\mathbf{X}]\Big/\mathcal{I}_{m,s}\right) &\geq \dim\ \mathrm{Span}(\mathcal{B}_{m,s})\\ &= \mid \{(\mathbf{I},\mathbf{J})\in\mathbb{N}^m\times\mathbb{N}^m \mid w(\mathbf{I})<s, 0\leqslant j_k<q\}\mid\\ &= \binom{m+s-1}{m}\cdot q^m,\end{aligned}$$

as desired. ◁

▶ **Corollary 42.** *The set* $\mathcal{B}_{m,s}$ *is a basis for* $\mathbb{F}[\mathbf{X}]\Big/\mathcal{I}_{m,s}$.

▶ **Example 43.** When $m=1$ the set $\left\{g(x)^i x^j \mid i\in\mathbb{N}, j<q\right\}$ is a basis of $\mathbb{F}[\mathbf{X}]$. Thus, for every $\ell\in\mathbb{N}$ there are $\beta_{\ell,i,j}$ such that

$$x^\ell = \sum_{i_1,i_2:i_1 q+i_2\leqslant\ell, i_2<q} \beta_{\ell,i_1,i_2} g(x)^{i_1}x^{i_2}.$$

By comparing coefficients of x^ℓ we see that $\beta_{\ell,\lfloor\frac{\ell}{q}\rfloor,\ell \bmod q}=1$.

▷ Claim 44. For every table $T:{\mathbb{F}_q}^m\to\Sigma_{m,s}$ there exists a degree $q(s-1)+(q-1)m$ polynomial such that $\mathrm{EVAL}(P)=T$.

Proof. By Claim 41, T may be written as $\mathrm{EVAL}(P)$, where P is an $\mathbb{F}_q$-combination of basis elements in $\mathcal{B}_{m,s}$. By the definition of $\mathcal{B}_{m,s}$, any basis element is of the form $g(\mathbf{X})^{\mathbf{I}}\cdot\prod_{k=1}^m X_k^{j_k}$ where $w(\mathbf{I})<s$ and $0\leqslant j_k<q$. As such, $\deg(P)\leq\deg(g(X)^{\mathbf{I}})+\deg(\prod_{k=1}^m X_k^{j_k})\leq q(s-1)+(q-1)m$. ◁

We will later need:

▶ **Lemma 45.** *Let* $A\in\mathbb{F}_q[t]$ *and* $B\in\mathbb{F}_q[r]$. *Then,*

$$\deg_t(A(t)B(r) \bmod \mathcal{I}_{2,s}) \leq \deg_t(A).$$

Proof. Express A in the basis $\left\{g(t)^i t^j\right\}_{i\in\mathbb{N},j<q}$,

$$A(t) = \sum_{(i,j)\in\mathcal{A}} \alpha_{i,j} g(t)^i t^j,$$

and similarly for B,

$$B(r) = \sum_{(k,\ell)\in\mathcal{B}} \beta_{k,\ell} g(r)^k r^\ell,$$

for some sets $\mathcal{A}, \mathcal{B}$, $\alpha_{i,j}, \beta_{k,\ell} \in \mathbb{F}_q$. Then,

$$A(t)B(r) \mod \mathcal{I}_{2,s} = \sum_{(i,j)\in\mathcal{A},(k,\ell)\in\mathcal{B}:i+k<s} \alpha_{i,j}\beta_{k,\ell} g(t)^i g(r)^k t^j r^\ell.$$

Suppose $\deg_t(A(t)B(r) \mod \mathcal{I}_{2,s})$ is achieved on the monomial of $(i,j) \in \mathcal{A}, (k,\ell) \in \mathcal{B}$. In particular, this degree in t is already achieved in A for the monomial (i,j). Thus, $\deg_t(AB) \leq \deg_t(A)$. ◀

Next, we would like to give a purely algebraic criteria, which states when exactly a table belongs to the code $\mathrm{MRM}(q,m,d,s)$.

▶ **Definition 46.** *Let $T \in \Sigma_{m,s}^{q^m}$ be a table. By Corollary 40 and Claim 41 there is a unique element $P_T \in \mathbb{F}[\mathbf{X}] \Big/ \mathcal{I}_{m,s}$ such that $\mathrm{EVAL}_{m,s}(P_T) = T$. We call P_T the* representing polynomial *of T.*

▶ **Lemma 47.** *Assume $d < sq$. Let $T \in \Sigma_{m,s}^{q^m}$ be a table, and P_T its representing polynomial. Then*

$$T \in \mathrm{MRM}(q,m,d,s) \iff deg(P_T) \leqslant d.$$

Proof. First, assume $deg(P_T) \leqslant d$. Then, by definition, since $\mathrm{EVAL}_{m,s}(P_T) = T$, we have $T \in \mathrm{MRM}(q,m,d,s)$. For the other direction, assume $T \in \mathrm{MRM}(q,m,d,s)$. Then there is some $Q \in \mathbb{F}[\mathbf{X}]$ of total degree $\leqslant d$ such that $\mathrm{EVAL}_{m,s}(Q) = T$. As $\deg(Q) \leq d < sq$, Q is $\mathcal{I}_{m,s}$ reduced and by Corollary 40 Q is the representing polynomial of T. ◀

To understand the low-dimensional tests on tables, we need to define the restriction of tables to subspaces. If T is a table $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ and we want to restrict to the plane $\mathbf{a}t + \mathbf{b}r + c$ the restriction $T_{\mathbf{a},\mathbf{b},\mathbf{c}}$ should be a table $\mathbb{F}_q^2 \to \Sigma_{2,s}$. To this end we define the alphabet reduction map $\phi_{(\mathbf{a},\mathbf{b})} : \Sigma_{m,s} \to \Sigma_{2,s}$ by

▶ **Definition 48.**

$$\left(\phi_{(\mathbf{a},\mathbf{b})}(z)\right)_{\mathbf{J}=(j_1,j_2)} = \sum_{\mathbf{I}\in\mathbb{N}^m} z_{\mathbf{I}} \cdot \sum_{\substack{\mathbf{I}_1+\mathbf{I}_2=\mathbf{I} \\ w(\mathbf{I}_1)=j_1, w(\mathbf{I}_2)=j_2}} \binom{\mathbf{I}}{\mathbf{I}_1} \mathbf{a}^{\mathbf{I}_1}\mathbf{b}^{\mathbf{I}_2}$$

This map applies the "chain rule" to an element in $\Sigma_{m,s}$, in accordance with Lemma 18. We may then define

$$T_{\mathbf{a},\mathbf{b},\mathbf{c}} = \phi_{(\mathbf{a},\mathbf{b})} \circ T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}}$$

.

Similarly, for $\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_k)$ we define $\phi_{\mathbf{h}}$ and $T_{\mathbf{h}}$ by:

$$(\phi_{\mathbf{h}}(z))_{\mathbf{J}} = \sum_{\mathbf{I}\in\mathbb{N}^m} z_{\mathbf{I}} \cdot \sum_{\substack{\mathbf{I}_1+\cdots+\mathbf{I}_k=\mathbf{I} \\ w(\mathbf{I}_r)=j_r}} \binom{\mathbf{I}}{\mathbf{I}_1,\ldots,\mathbf{I}_k} \prod_{i=1}^{k} \mathbf{h}_k^{\mathbf{I}_k}, \text{ and,}$$

$$T_{\mathbf{h}}(\mathbf{Y}) = \phi_{(\mathbf{h})} \circ T(\mathbf{h}_0 + \sum_{i=1}^{k} Y_i \mathbf{h}_i)$$

4 The line test is not a characterization when $d \geq q$

We now show that when the field size is smaller than the degree d, the line test fails.

▶ **Theorem 49.** *Assume q is a prime power, $q \leqslant d < sq - 1$ and $m \geqslant 2$. There exists a table $T \in \Sigma_{m,s}^{q}$ which passes all the tests of the line test, but there is no polynomial $P \in \mathbb{F}_q[X_1, \ldots, X_m]^{\leqslant d}$ that satisfies $\mathrm{EVAL}_{m,s}(P) = T$.*

Proof. Define

$$\begin{aligned} Q &= X_1^{d-q} \cdot (g(X_1)X_2 - g(X_2)X_1) \\ &= X_1^{d-q} \cdot (X_1^q X_2 - X_2^q X_1). \end{aligned}$$

Note that Q is homogeneous of degree $d+1$. Fix $\mathbf{a}, \mathbf{b} \in {\mathbb{F}_q}^m$ and let $QL_{\mathbf{a},\mathbf{b}}(t) = Q(\mathbf{a}t + \mathbf{b})$. Since Q is homogeneous of degree $d+1$, the coefficient of t^{d+1} in $QL_{\mathbf{a},\mathbf{b}}(t)$ is $Q(\mathbf{a})$. However, $Q \in \langle g(X_1), g(X_2) \rangle \subseteq \mathcal{I}_{m,1}$ and therefore $Q(\alpha) = 0$ for every $\alpha \in \mathbb{F}_q^m$. In particular $Q(\mathbf{a}) = 0$. Thus, $\deg(QL_{\mathbf{a},\mathbf{b}}) \leq d$ and Q passes the degree d line test for the line $\ell_{\mathbf{a},\mathbf{b}}$. Thus, Q passes the degree d line test for all lines.

Now take the table $T = \mathrm{EVAL}(Q)$. By Corollary 40, there cannot be a polynomial P with $deg(P) \leq d < deg(Q) = d + 1 < sq$ having the same table. However, we saw T passes all line tests. Thus, T wrongly passes the line test. ◀

5 Local testing above the RM testing dimension

In this section we look at the local characterization and local testing of $\mathrm{MRM}(q, m, d, s)$ by dimension k tests, when k is above the RM testing dimension $t_{q,d} = \lceil \frac{d+1}{q - \frac{q}{p}} \rceil$ of $\mathrm{RM}(q, m, d)$. By Theorem 2 dimension k subspaces give a local test (and hence also a local characterization) for $\mathrm{RM}(q, m, d)$. We show they also give a local test (and hence also a local characterization) for $\mathrm{MRM}(q, m, d, s)$ for $s < q$. There is, however, some parameter loss in the reduction as we next explain. To formally state the result we need some notation. For $x \in \mathbb{F}_q^m$ let

$$\begin{aligned} H_k &= \left\{ \mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_k) \mid \mathbf{h}_0, \ldots, \mathbf{h}_k \in \mathbb{F}_q^m, \dim(\mathrm{span}\{\mathbf{h}_1, \ldots, \mathbf{h}_k\}) = k \right\}, \text{and,} \\ H_{k,x} &= \left\{ \mathbf{h} \in H_k \mid x \in \mathbf{h}_0 + \mathrm{Span}\left\{\mathbf{h}_1, \ldots, \mathbf{h}_k\right\} \right\}. \end{aligned}$$

By $\mathbf{h} \sim_k H$ (resp. $H_{k,x}$) we mean a choosing $\mathbf{h}$ uniformly at random from H_k (resp. $H_{k,x}$). We let $A_{\mathbf{h}} = \mathbf{h}_0 + \mathrm{Span}\left\{\mathbf{h}_1, \ldots, \mathbf{h}_k\right\}$ be the k-dimensional affine space defined by $\mathbf{h}$. We also recall the distance and rejection function from the introduction:

▶ **Definition 50.** *Let $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ be a table. Then*

- *$\delta(T, \mathrm{MRM}(q, m, d, s))$ is the distance between T and the closest evaluation table of a degree d polynomial, i.e., $\delta(T, \mathrm{MRM}(q, m, d, s)) = \min_{G \in \mathrm{MRM}(q,m,d,s)}\{\delta(T, G)\}$, and,*
- *$\mathrm{REJ}_{k,d}^{\mathrm{MRM}}(T)$ is the probability a dimension-k test demonstrates that T is not a degree d polynomial.*

With this notation we prove:

▶ **Theorem 51.** *Let $\mathbb{F}_q$ be a field of size q, and assume $s \leq \min\{d, q-1\}$. Suppose for $\mathrm{RM}(q, m, d)$ there exists $\alpha > 0$ and $c_0 \leq 1$ such that for every f*

$$\mathrm{REJ}_{k,d}^{\mathrm{RM}}(f) \geq \min\left\{\alpha \cdot \delta(f, \mathrm{RM}(q, m, d)), c_0\right\}.$$

Then, for every $s < q$, for every T we have

$$\mathrm{REJ}^{\mathrm{MRM}}_{k,d}(T) \geq \min\left\{\alpha' \cdot \delta(T, \mathrm{MRM}(q,m,d,s)), c_0\right\} \tag{6}$$

for

$$\alpha' = \left(1 - \frac{s-1}{q}\right) \frac{1}{1 + q^{d/(q-1)\frac{1}{\alpha}}}. \tag{7}$$

For example, assume q is prime. If $d+1 \leq q-1$ the testing dimension is 1, meaning that lines are a good local test for $\mathrm{RM}(q,m,d)$. Then, Theorem 51 says that lines are also a good test for $\mathrm{MRM}(q,m,d,s)$, alas, with a larger coefficient. Similarly, if $q < d \leq 2(q-1)$ the RM testing dimension is 2 and therefore planes are a good local test for $\mathrm{RM}(q,m,d)$. Then, Theorem 51 says that planes are also a good test for $\mathrm{MRM}(q,m,d,s)$, with a larger coefficient. In the same vein 3-dimensional planes are a good local test for $\mathrm{MRM}(q,m,d,s)$ when $d \leq 3(q-1)$.

To explain the intuition behind the theorem we first consider the characterization aspect of it. Suppose k is above the RM testing dimension, and that the table T passes all $\mathrm{MRM}(q,m,d,s)$ k-dimensional tests $\mathbf{h}$. Specifically, this means that for every $\mathbf{h} \in H_k$, the table T restricted to the k-dimensional affine space $A_{\mathbf{h}}$ is consistent with a degree d polynomial $P_{\mathbf{h}}$. Now, let $T_{(0)}$ be the table T where at each entry we keep only the evaluation of the function itself and remove the evaluations that are associated with higher derivatives. Then, in particular, for every $\mathbf{h} \in H_k$, the table $T_{(0)}$ restricted to the k-dimensional space $A_{\mathbf{h}}$ is still consistent with the degree d polynomial $P_{\mathbf{h}}$. As k is above the RM testing dimension, there must be a unique degree d polynomial P that is consistent with $P_{\mathbf{h}}$ (and the table $T_{(0)}$) for every $\mathbf{h} \in H_k$. This P is the only possible candidate for a low-degree explanation of the table T. What we need to check, and is indeed correct, is that since T passes all dimension k tests, P is indeed consistent with T.

The testing case requires more technical details but is similar in spirit. We first, again, look at $T_{(0)}$ that contains only the function evaluations, and not the higher derivatives evaluations. If T passes the test with high enough probability, then so does $T_{(0)}$, and this ensures the existence of a global degree-d polynomial P that agrees with most values of $T_{(0)}$. Again, what remains to be shown is that P agrees with most values of T, which we indeed prove.

In short, one can informally say that Theorem 51 shows that the MRM testing dimension is not larger than the RM testing dimension, and that above the RM testing dimension one can get *local testing* for MRM codes. A natural question is whether the MRM testing dimension is equal to the RM testing dimension, or not. In Section 4 we saw that when the RM testing dimension is larger than 1, so is the MRM testing dimension, and lines do not characterize the MRM code. One might be drawn to the conjecture that the RM and MRM testing dimensions coincide. However, surprisingly, in Section 6 we show that no matter what the RM testing dimension is, when $d < sq$ the MRM testing dimension is at most two (for a precise statement see Theorem 60). For example, for $\mathrm{MRM}(q,d,m,s)$ with $2q < d < 3(q-1)$, the RM testing dimension is (roughly) 3 while the MRM testing dimension is still 2.

Combining the result with Theorem 2 we get:

▶ **Corollary 52.** *Let $d, q, s < q$ be positive integers and let $t = t_{q,d} = \lceil \frac{d+1}{q-\frac{q}{p}} \rceil$ be the RM testing dimension. Then,*

$$\mathrm{REJ}^{\mathrm{MRM}}_{t,d}(T) \geq \min\left\{\frac{1}{3} \cdot (1 - \frac{s-1}{q}) \cdot \delta(T, \mathrm{MRM}(q,m,d,s)), \frac{1}{2(t+1)q^{t+1}}\right\}.$$

Proof. Theorem 2 from [12] ensures that

$$\mathrm{REJ}_{t,d}(f) \geq \min\left\{\frac{q^t}{2}\cdot\delta(f,\mathrm{RM}(q,m,d)), \frac{1}{2(t+1)q^{t+1}}\right\}.$$

Using Theorem 51 we see that

$$\alpha' \geq \left(1-\frac{s-1}{q}\right)\frac{1}{1+q^{d/(q-1)}\frac{1}{\alpha}} \geq \frac{1}{3}(1-\frac{s-1}{q}).$$

using the fact that $\frac{d}{q-1} \leq t$ and $\frac{q^{d/(q-1)}}{\alpha} \leq 2$. ◀

In particular, for $d < (q - \frac{q}{p})$ the line test is a local characterization, as $t_{q,d} = 1$.

We note that the assumption $s \leq d$ is quite natural, as derivatives with order higher than the degree must be identically zero. In contrast, it is not clear whether the assumption $s < q$ is indeed required, and we leave it for future study.

5.1 The proof

Proof. Let $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ be a table. Let

$$\rho = \mathrm{REJ}_{k,d}^{\mathrm{MRM}}(T)$$

be the k-dimensional MRM test rejection probability. If $\rho \geq c_0$ we are done. Therefore, we assume $\rho < c_0$. We first utilize what we know at the zero level.

▷ Claim 53. Let $\delta^{\mathrm{RM}} = \delta^{\mathrm{RM}}_{q,m,d}$ be the distance of the $\mathrm{RM}(q,m,d)$ code. There exists a degree d polynomial P such that

$$\Pr_{\mathbf{h}\in H_k}\left[(\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}} \neq \mathrm{EVAL}(P|_{A_h})\right] \stackrel{\mathrm{def}}{=} \varepsilon_0 \leq \rho(1+\frac{1}{\alpha\cdot\delta^{\mathrm{RM}}}).$$

We call $\mathbf{h}$ *good* if $(\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}} = \mathrm{EVAL}(P|_{A_h})$ and *bad* otherwise. In this terminology, $\varepsilon_0 = \Pr_{\mathbf{h}\in H_k}[\mathbf{h} \text{ is bad}]$.

Proof. Let $T_{(\mathbf{0})} : \mathbb{F}_q^m \to \mathbb{F}_q$ be as in Section 3, i.e., the table where we keep only the entries of the function evaluations and ignore the evaluations of higher order derivatives. For a given affine space A we have that $(T|_A)_{(\mathbf{0})} = T_{(\mathbf{0})}\big|_A$, and so if the former agrees with a degree d polynomial, so does the latter. It follows that

$$\mathrm{REJ}_{k,d}^{\mathrm{RM}}(T_{(\mathbf{0})}) \leq \rho.$$

By the hypothesis regarding $\mathrm{RM}(q,m,d)$ and using $\rho < c_0$ there exists a unique degree d polynomial $P \in \mathbb{F}_q[\mathbf{X}]$ such that

$$\delta_{(0)} \stackrel{\mathrm{def}}{=} \delta(T_{(\mathbf{0})}, P) \leq \frac{\rho}{\alpha}.$$

We notice that

$$\mathbb{E}_{\mathbf{h}\sim H_k}\ \delta(T_{(\mathbf{0})}\big|_{A_{\mathbf{h}}}, P|_{A_{\mathbf{h}}}) = \Pr_{x\sim\mathbb{F}_q^m}(T_{(\mathbf{0})}(x) \neq P(x)) = \delta(T_{(\mathbf{0})}, P) = \delta_{(0)},$$

because the subspaces in $A_{\mathbf{h}}$ for $\mathbf{h} \in H_k$, cover every point an equal number of times. Therefore, by Markov's inequality,

$$\Pr_{\mathbf{h}\sim H_k}\left(\delta(T_{(\mathbf{0})}\big|_{A_{\mathbf{h}}}, P|_{A_{\mathbf{h}}}) \geq \delta^{\mathrm{RM}}\right) \leq \frac{\delta_{(0)}}{\delta^{\mathrm{RM}}}. \tag{8}$$

Also, by assumption,

$$\Pr_{\mathbf{h}\in H_k}\left((\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}}\notin \mathrm{MRM}(q,k,d,s)\right)\leq\rho. \tag{9}$$

This means that except for probability ρ over $\mathbf{h}$, $(\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}}$ agrees with $\mathrm{EVAL}(P_{\mathbf{h}})$ for some k-variate degree d polynomial $P_{\mathbf{h}}$. When this happens it also holds that $T_{(\mathbf{0})}$ agrees with $P_{\mathbf{h}}$ over $A_{\mathbf{h}}$ (because $\phi_{\mathbf{h}}$ does not change the zero level).

Thus, Equations (8) and (9) together imply that except for probability $\rho+\frac{\delta_0}{\delta^{\mathrm{RM}}}\leq\rho(1+\frac{1}{\alpha\delta^{\mathrm{RM}}})$ over $\mathbf{h}$, we simultaneously have that $T_{(\mathbf{0})}=P_{\mathbf{h}}$ over $A_{\mathbf{h}}$ and that $\delta(T_{(\mathbf{0})}\big|_{A_{\mathbf{h}}},P|_{A_{\mathbf{h}}})<\delta^{\mathrm{RM}}$. When both events happen we conclude that

$$\delta(P_{\mathbf{h}},P|_{A_{\mathbf{h}}})<\delta^{\mathrm{RM}}.$$

As $P_{\mathbf{h}}$ and $P|_{A_{\mathbf{h}}}$ are degree d polynomials on $A_{\mathbf{h}}$ and are closer than δ^{RM}, it must be the that in fact $P_{\mathbf{h}}=P|_{A_{\mathbf{h}}}$. Thus, $(\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}}$ is a valid $\mathrm{MRM}(q,k,d,s)$ table, and it is the table of the polynomial $P|_{A_{\mathbf{h}}}$, i.e., $(\phi_{\mathbf{h}}\circ T)|_{A_{\mathbf{h}}}=\mathrm{EVAL}(P|_{A_{\mathbf{h}}})$ as desired. ◁

Our goal is to bound $\delta(\mathrm{EVAL}(P),T)$. This means that at most $x\in\mathbb{F}_q^m$ we should have $T(x)=\mathrm{EVAL}(P;x)$. $T(x)$ and $\mathrm{EVAL}(P;x)$ contains values for all the m-variate directional derivatives of order up to s. Our handle on these values is Claim 53, that shows that most $\mathbf{h}$ are good, meaning that the k-variate derivatives of $P|_{A_{\mathbf{h}}}$ and $\phi_{\mathbf{h}}\circ T|_{A_{\mathbf{h}}}$ are the same. We notice that every such k-variate derivative is a linear combination (dependent on $\mathbf{h}$) of the m-variate derivatives. If x is such that for many $\mathbf{h}\in H_{k,x}$, $\mathbf{h}$ is good, then for that x we get many linear equations on the m-variate derivatives. Our task is to prove that for many x there are enough good $\mathbf{h}\in H_{k,x}$ to force the underlying m-variate derivatives of P and T to agree.

▷ Claim 54. We say $x\in\mathbb{F}_q^m$ is *bad* if $\Pr_{\mathbf{h}\in H_{k,x}}[\mathbf{h}\text{ is bad}]\geq 1-\frac{s-1}{q}$ and *good* otherwise. Then

$$\Pr_{x\in\mathbb{F}_q^m}[x\text{ is bad}]\leq\frac{q}{q-(s-1)}\cdot\Pr_{\mathbf{h}\in H_k}[\mathbf{h}\text{ is bad}].$$

Proof. We have

$$\mathbb{E}_{x\in\mathbb{F}_q^m}\left[\Pr_{\mathbf{h}\in H_{k,x}}(\mathbf{h}\text{ is bad})\right]=\Pr_{\mathbf{h}\in H_k}(\mathbf{h}\text{ is bad})=\varepsilon_0,$$

where ε_0 is as in Claim 53. This is because choosing a uniform $\mathbf{h}\in H_k$ is the same as first choosing a uniform $x\in\mathbb{F}_q^m$ and then choosing a uniform $\mathbf{h}\in H_{k,x}$. Therefore, for every $c>1$, by Markov's inequality,

$$\Pr_{x\in\mathbb{F}_q^m}\left[\Pr_{h\in H_{k,x}}(\mathbf{h}\text{ is bad})\geq c\cdot\varepsilon_0\right]\leq\frac{1}{c}.$$

Choosing $c=\frac{q-(s-1)}{q\cdot\varepsilon_0}$ gives the result. ◁

We note that if most k-dimensional subspaces are good, then most lines are:

▷ Claim 55. For every $x\in\mathbb{F}_q^m$ and $k\geq 1$,

$$\Pr_{\mathbf{u}\in H_{1,x}}[\mathbf{u}\text{ is bad}]\leq\Pr_{\mathbf{h}\in H_{k,x}}[\mathbf{h}\text{ is bad}]$$

Proof. Fix x. For every $\mathbf{h} \in H_{k,x}$, If $\mathbf{h}$ is good, then $(\phi_{\mathbf{h}} \circ T)|_{A_{\mathbf{h}}} = \mathrm{EVAL}(P|_{A_{\mathbf{h}}})$. This implies, in particular, that for every $\mathbf{u} \in H_{1,x}$ such that $A_{\mathbf{u}} \subseteq A_{\mathbf{h}}$, we also have that $(\phi_{\mathbf{u}} \circ T)|_{A_{\mathbf{u}}} = \mathrm{EVAL}(P|_{A_{\mathbf{u}}})$. The result then follows because we can sample $\mathbf{u} \in H_{1,x}$ by first sampling $\mathbf{h} \in A_{k,x}$ and then choosing a random $\mathbf{u} \in H_{1,x}$ such that $A_{\mathbf{u}} \subseteq A_{\mathbf{h}}$. ◁

We now fix any good x. We will prove:

▷ Claim 56. Suppose $x \in \mathbb{F}_q^m$ is good. Then for any m-variate direction $\mathbf{I}$ with $w(\mathbf{I}) < s$ we have

$$T_{(\mathbf{I})}(x) = P^{(\mathbf{I})}(x) \tag{10}$$

Once we prove the claim we can conclude the proof of the theorem because:

$$\begin{aligned}\delta(T, \mathrm{EVAL}(P)) \leq \Pr_{x \in \mathbb{F}_q^m}(x \text{ is bad}) &\leq \frac{q}{q-(s-1)} \cdot \varepsilon_0 \\ &\leq \frac{q}{q-(s-1)} \cdot \rho \cdot (1 + \frac{1}{\alpha \cdot \delta^{\mathrm{RM}}}) \leq \rho \left(1 - \frac{q}{s-1}\right)^{-1} \left(1 + \frac{q^{d/(q-1)}}{\alpha}\right).\end{aligned}$$

Proof of Claim 56. Fix a good $x \in \mathbb{F}_q^m$ and let $w_0 < s$. We will show Equation (10) simultaneously for all m-variate directions $\mathbf{I}$ with $w(\mathbf{I}) = w_0$. We know that

$$\Pr_{\mathbf{u} \in H_{1,x}}(\mathbf{u} \text{ is bad}) \leq \Pr_{\mathbf{h} \in H_{k,x}}(\mathbf{h} \text{ is bad}) < 1 - \frac{s-1}{q}, \tag{11}$$

where the first inequality is by Claim 55 and the second because x is good.

Suppose $\mathbf{u} = (\mathbf{b}, \mathbf{a})$ is good (where $\mathbf{b}, \mathbf{a} \in \mathbb{F}_q^m$). Then, by definition, $(\phi_{\mathbf{u}} \circ T)|_{A_{\mathbf{u}}} = \mathrm{EVAL}(P|_{A_{\mathbf{u}}})$. In particular $\phi_{\mathbf{u}}(T(x)) = \mathrm{EVAL}(P|_{A_{\mathbf{u}}}; x)$, because $x \in A_\ell$. Now:

- By Definition 48 (and plugging $k = 1$)

$$(\phi_{\mathbf{u}}(T(x)))_{w_0} = \sum_{\mathbf{I}, w(\mathbf{I}) = w_0} T(x)_{(\mathbf{I})} \cdot \mathbf{a}^{\mathbf{I}},$$

- Also, by Lemma 17 (plugging $t = 0$),

$$(\mathrm{EVAL}(P|_{A_{\mathbf{u}}}; x))_{w_0} = \sum_{\mathbf{I}, w(\mathbf{I}) = w_0} P^{\mathbf{I}}(x) \cdot \mathbf{a}^{\mathbf{I}}.$$

Thus, a good $\mathbf{u} = (\mathbf{b}, \mathbf{a})$ gives the liner equation

$$\sum_{\mathbf{I}, w(\mathbf{I}) = w_0} (P^{\mathbf{I}}(x) - T(x)_{(I)}) \cdot \mathbf{a}^{\mathbf{I}} = 0,$$

where the variables are $v_{\mathbf{I}} = P^{\mathbf{I}}(x) - T(x)_{(I)}$ for every m-variate direction $\mathbf{I}$ of total weight exactly w_0. Thus, Equation (11) implies that

$$\Pr_{\mathbf{a} \in \mathbb{F}_q^m \setminus \{0\}} \left(\sum_{\mathbf{I}, w(\mathbf{I}) = w_0} (T(x)_{(\mathbf{I})} - P^{(\mathbf{I})}(x)) \cdot \mathbf{a}^{\mathbf{I}} = 0 \right) > \frac{(s-1)}{q}.$$

Now, look at the polynomial $f_x \in \mathbb{F}_q[X_1, \ldots, X_m]$ defined by

$$f_x(\mathbf{a}) = \sum_{\mathbf{I}, w(\mathbf{I}) = w_0} (T(x)_{(\mathbf{I})} - P^{(\mathbf{I})}(x)) \cdot \mathbf{a}^{\mathbf{I}}.$$

f_x is an m-variate, degree w_0 homogeneous polynomial, and it is 0 with probability larger than $\frac{(s-1)}{q} \geq \frac{w_0}{q} = \frac{\deg(f_x)}{q}$. By the Schwartz-Zippel lemma, it must be the 0 polynomial. Therefore, $T(x)_{(\mathbf{I})} - P^{(\mathbf{I})}(x)$ for all $\mathbf{I}$ with $w(\mathbf{I}) = w_0$ as desired. ◁

◀

▶ Remark 57. Another way to view the argument, is that each $\mathbf{a} \in \mathbb{F}_q^m$ is an evaluation point of a homogeneous $\mathrm{RM}(q, m, w_0)$ codeword, and therefore each good $\mathbf{u} = (\mathbf{b}, \mathbf{a})$ gives a zero coordinate of the codeword. If the number of good $\mathbf{u}$ is too large, we get too many zeroes, and therefore the codeword must be the zero codeword, meaning that the values of the variables are zero as we wish.

▶ Remark 58. The argument we use has the information that many k-dimensional restrictions are good, but then chooses to reduce this knowledge to the weaker statement that for many x, for many lines passing through x, the *linear* restrictions are good. It seems that using the stronger statement might give a better code and improve the parameters, but we have not succeeded yet in analyzing this.

6 The plane test is a characterization

In this section we show that the multiplicity code $\mathrm{MRM}(q, m, d, s)$ can be characterized by restrictions to planes. Let $\mathbb{F}_q$ be a field and let $m, s \leq d$ be positive integers. For $\mathbf{a}, \mathbf{b}, \mathbf{c} \in {\mathbb{F}_q}^m$[4] define:

- $\ell_{\mathbf{a},\mathbf{b},\mathbf{c}} : \mathbb{F}_q^2 \to \mathbb{F}_q^m$ by:

$$\ell_{\mathbf{a},\mathbf{b},\mathbf{c}}(t, r) = \mathbf{a}t + \mathbf{b}r + \mathbf{c}.$$

From Lemma 18 and Definition 48 we see that:

▶ **Theorem 59** (Completeness). *Suppose $d < sq - 1$. If a table $T \in \Sigma_{m,s}^{q^m}$ satisfies $T \in \mathrm{MRM}(q, m, d, s)$ then for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_q^m$,*

$$\phi_{(\mathbf{a},\mathbf{b})} \circ T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}} \in \mathrm{MRM}(q, 2, d, s).$$

The main challenge is proving the converse:

▶ **Theorem 60** (Soundness). *Suppose q is a power of the prime p, $q \geq s$ and $d < q(s - \frac{1}{p})$. If a table $T \in \Sigma_{m,2}^{q^m}$ satisfies that for all $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_q^m$, $\phi_{(\mathbf{a},\mathbf{b})} \circ T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}} \in \mathrm{MRM}(q, 2, d, s)$ then $T \in \mathrm{MRM}(q, m, d, s)$.*

We define the vector space of tables which pass the test:

$$V_{m,d,s} = \left\{ T \in \Sigma_{m,s}^{q^m} \mid \forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_q^m \;\; \phi_{(\mathbf{a},\mathbf{b})} \circ T \circ \ell_{\mathbf{a},\mathbf{b},\mathbf{c}} \in \mathrm{MRM}(q, 2, d, s) \right\} \tag{12}$$

We denote by $C_{m,d,s} = V_{m,d,s} \setminus \mathrm{MRM}_{m,d,s}$, the set of tables which cheat the test. We would like to show that $C_{m,d,s} = \emptyset$. Assume towards a contradiction that there is a table $T \in C_{m,d,s}$. By Claim 41, T can be realized (uniquely) as an element P of the quotient space $\mathbb{F}_q[\mathbf{X}] \Big/ \mathcal{I}_{m,s}$. We use the basis $\mathcal{B}_{m,s}$ from Equation (5) to write P in the form

$$P(\mathbf{X}) = \sum_{(\mathbf{I},\mathbf{J}) \in \mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \cdot g(\mathbf{X})^{\mathbf{I}} \mathbf{X}^{\mathbf{J}},$$

[4] In this section, we omit the requirement that $\mathbf{a}, \mathbf{b}$ be independent. If some degenerate plane shows that a table is not a low degree polynomial, then some actual plane will too.

where $\alpha_{\mathbf{I},\mathbf{J}} \in \mathbb{F}_q$ and $(\mathbf{I},\mathbf{J}) \in \mathcal{M}_{s,q}$ iff $w(\mathbf{I}) < s$ and $J_k < q$ for every $1 \le k \le m$. Since $T \notin \mathrm{MRM}_{m,d,s}$ we have $deg(P) > d$. This means there must be some $\mathbf{I}$ and $\mathbf{J}$ such that $\alpha_{\mathbf{I},\mathbf{J}} \neq 0$ and

$$w(\mathbf{I})q + w(\mathbf{J}) > d. \tag{13}$$

We may assume that every $\mathbf{I},\mathbf{J}$ for which $\alpha_{\mathbf{I},\mathbf{J}} \neq 0$ satisfy Equation (13). This is since the test is linear, and any degree $\leqslant d$ terms have no effect on whether P passes the test or not.

We use the following monomial order $\succ_w$ on $\mathcal{B}_{m,s}$:

1. First order monomials according to $w(\mathbf{I}) + w(\mathbf{J})$,
2. Then order monomials according to $w(\mathbf{I})$,
3. Finally, order monomials according to the lexicographic order on $\mathbf{I},\mathbf{J}$.

We emphasize that $\succ_w$ is not a monomial order in the sense of Grobner bases, and we make no use of it in that sense.

Let $(\mathbf{I}_{\max},\mathbf{J}_{\max})$ be s.t. $g(\mathbf{X})^{\mathbf{I}_{\max}} \cdot X^{\mathbf{J}_{\max}}$ is a maximal monomial of P according to $\succ_w$.

For $\mathbf{a},\mathbf{b},\mathbf{c} \in \mathbb{F}^m$ the restriction of P to the plane defined by $\mathbf{a},\mathbf{b},\mathbf{c}$ is $PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = P(\mathbf{a}t + \mathbf{b}r + \mathbf{c})$. Expressing $PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r)$ in the basis $\mathcal{B}_{2,\infty}$:

$$PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = \sum_{i\in\mathbb{N},j\in\mathbb{N},k,\ell<q} A_{i,j,k,\ell}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot g(t)^i g(r)^j t^k r^\ell$$

$$PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) \bmod I_{2,s} = \sum_{i+j<s,k,\ell<q} A_{i,j,k,\ell}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot g(t)^i g(r)^j t^k r^\ell$$

We view $A_{i,j,k,\ell}(\mathbf{a},\mathbf{b},\mathbf{c})$ as a polynomial in the variables $\mathbf{a},\mathbf{b},\mathbf{c}$.

▶ **Lemma 61.** *For every partition $\mathbf{J}_{\max} = \mathbf{J}^b_{\max} + \mathbf{J}^c_{\max}$ such that:*

- $q \cdot w(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) \le qs - 1$, *and,*
- $\binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \neq 0 \bmod p$,

the monomial $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ appears with a non-zero coefficient at

$$A_{w(\mathbf{I}_{\max}),\lfloor\frac{w(\mathbf{J}^b_{\max})}{q}\rfloor,0,w(\mathbf{J}^b_{\max}) \bmod q}(\mathbf{a},\mathbf{b},\mathbf{c}) \bmod \mathcal{I}_{3m,1}.$$

Where the ideal $\mathcal{I}_{3m,1}$ above is in the variables $a_1,\ldots,a_m,b_1,\ldots,b_m,c_1,\ldots,c_m$.

Proof. We expand $PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r)$. First, $P(\mathbf{X}) = \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \cdot g(\mathbf{X})^{\mathbf{I}}\mathbf{X}^{\mathbf{J}}$. Thus

$$\begin{aligned}
PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) &= \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \cdot g(\mathbf{a}t + \mathbf{b}r + \mathbf{c})^{\mathbf{I}}(\mathbf{a}t + \mathbf{b}r + \mathbf{c})^{\mathbf{J}} \\
&= \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \prod_{k=1}^{m} (g(a_kt + b_kr + c_k))^{I_k} \cdot \Pi_{i=k}^{m}(a_kt + b_kr + c_k)^{J_k} \\
&= \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \prod_{k=1}^{m} (g(t)a_k + g(r)b_k)^{I_k} \cdot \Pi_{i=k}^{m}(a_kt + b_kr + c_k)^{J_k} \\
&= \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \sum_{\substack{\mathbf{I}_a+\mathbf{I}_b=\mathbf{I} \\ \mathbf{J}_a+\mathbf{J}_b+\mathbf{J}_c=\mathbf{J}}} \\
&\quad \binom{\mathbf{I}}{\mathbf{I}_a}\binom{\mathbf{J}}{\mathbf{J}_a,\mathbf{J}_b,\mathbf{J}_c}\mathbf{a}^{\mathbf{I}_a+\mathbf{J}_a}\mathbf{b}^{\mathbf{I}_b+\mathbf{J}_b}\mathbf{c}^{\mathbf{J}_c} \cdot g(t)^{w(\mathbf{I}_a)}g(r)^{w(\mathbf{I}_b)}t^{w(\mathbf{J}_a)}r^{w(\mathbf{J}_b)}.
\end{aligned} \tag{14}$$

We expand $t^{w(J_a)}$ and $r^{w(J_b)}$ in the basis $\mathcal{B}_{1,s}$ as in Example 43 to get:

$$PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(t,r) = \sum_{(\mathbf{I},\mathbf{J})\in\mathcal{M}_{s,q}} \alpha_{\mathbf{I},\mathbf{J}} \cdot \sum_{\substack{\mathbf{I}_a+\mathbf{I}_b=\mathbf{I}\\ \mathbf{J}_a+\mathbf{J}_b+\mathbf{J}_c=\mathbf{J}}} \binom{\mathbf{I}}{\mathbf{I}_a}\binom{\mathbf{J}}{\mathbf{J}_a\mathbf{J}_b\mathbf{J}_c} \sum_{\substack{i_1,i_2\\ i_1q+i_2\leqslant w(\mathbf{J}_a)\\ i_2<q}} \sum_{\substack{j_1,j_2\\ j_1q+j_2\leqslant w(\mathbf{J}_b)\\ j_2<q}} \beta_{w(\mathbf{J}_a),i_1,i_2}\beta_{w(\mathbf{J}_b),j_1,j_2} \mathbf{a}^{\mathbf{I}_a+\mathbf{J}_a}\mathbf{b}^{\mathbf{I}_b+\mathbf{J}_b}\mathbf{c}^{\mathbf{J}_c} \cdot g(t)^{w(\mathbf{I}_a)+i_1}t^{i_2} \cdot g(r)^{w(\mathbf{I}_b)+j_1}r^{j_2}. \tag{15}$$

We now have a representation in the basis $\mathcal{B}_{2,\infty}$.
Set:

$$\begin{aligned}\Delta_{rq} &= \left\lfloor \frac{w(\mathbf{J}^b_{\max})}{q} \right\rfloor\\ \Delta_t &= 0\\ \Delta_r &= w(\mathbf{J}^b_{\max}) \bmod q.\end{aligned} \tag{16}$$

We wish to see for which choice of values $\mathbf{I},\mathbf{J},\mathbf{I}_a,\mathbf{I}_b,\mathbf{J}_a,\mathbf{J}_b,\mathbf{J}_c,i_1,i_2,j_1,j_2$ in Equation (14), $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ appears as a coefficient of $A_{(w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r)}\bmod \mathcal{I}_{3m,1}$. We must have

$$\begin{aligned}\mathbf{I}_{\max} &=_{\mathbb{F}_q} \mathbf{I}_a+\mathbf{J}_a && \text{By comparing the powers of } \mathbf{a}\text{, remembering mod } \mathcal{I}_{3m,1}\ ,\\ \mathbf{J}^b_{\max} &=_{\mathbb{F}_q} \mathbf{I}_b+\mathbf{J}_b && \text{By comparing the powers of } \mathbf{b}\text{, remembering mod } \mathcal{I}_{3m,1}\ ,\\ \mathbf{J}^c_{\max} &=_{\mathbb{F}_q} \mathbf{J}_c && \text{By comparing the powers of } \mathbf{c}\text{, remembering mod } \mathcal{I}_{3m,1}\ ,\end{aligned}$$

where $=_{\mathbb{F}_q}$ was defined in Definition 36.

By Claim 38 together with $s<q$ we have

$$w(\mathbf{I}_{\max}) \leq w(\mathbf{I}_a)+w(\mathbf{J}_a) \tag{17}$$
$$w(\mathbf{J}^b_{\max}) \leq w(\mathbf{I}_b)+w(\mathbf{J}_b) \tag{18}$$
$$\mathbf{J}^c_{\max} = \mathbf{J}_c. \tag{19}$$

It follows that

$$\begin{aligned}w(\mathbf{I}_{\max})+w(\mathbf{J}_{\max}) &= w(\mathbf{I}_{\max})+w(\mathbf{J}^b_{\max})+w(\mathbf{J}^c_{\max})\\ &\leq w(\mathbf{I}_a)+w(\mathbf{J}_a)+w(\mathbf{I}_b)+w(\mathbf{J}_b)+w(\mathbf{J}_c)\\ &= w(\mathbf{I})+w(\mathbf{J}).\end{aligned}$$

As $(\mathbf{I}_{\max},\mathbf{J}_{\max})$ is maximal it follows that

$$w(\mathbf{I}_{\max})+w(\mathbf{J}_{\max}) = w(\mathbf{I})+w(\mathbf{J}). \tag{20}$$

This, in turn, implies that both inequalities in Equations (17) and (18) are in fact equalities, i.e.,

$$w(\mathbf{I}_{\max}) = w(\mathbf{I}_a)+w(\mathbf{J}_a) \tag{21}$$
$$w(\mathbf{J}^b_{\max}) = w(\mathbf{I}_b)+w(\mathbf{J}_b). \tag{22}$$

We now look at

$$degt \stackrel{\text{def}}{=} \deg_t(g(t\mathbf{a})^{\mathbf{I}_a}\cdot(t\mathbf{a})^{\mathbf{J}_a}\cdot g(r\mathbf{b})^{\mathbf{I}_b}\cdot(r\mathbf{b})^{\mathbf{J}_b}\cdot\mathbf{c}^{\mathbf{J}_c} \bmod \mathcal{I}_{2,s})$$

On the one hand, we look for the monomial

$$\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$$

in $A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r} \bmod \mathcal{I}_{3m,1}$, and so we should have $degt = q \cdot w(\mathbf{I}_{\max}) + \Delta_t$. On the other hand, by Lemma 45,

$$\begin{aligned} degt &= \deg_t(g(t\mathbf{a})^{\mathbf{I}_a}(t\mathbf{a})^{\mathbf{J}_a} g(r\mathbf{b})^{\mathbf{I}_b}(r\mathbf{b})^{\mathbf{J}_b}\mathbf{c}^{\mathbf{J}_c} \bmod \mathcal{I}_{2,s}) \\ &\leq \deg_t(g(t\mathbf{a})^{\mathbf{I}_a}(t\mathbf{a})^{\mathbf{J}_a}) \\ &= q \cdot w(\mathbf{I}_a) + w(\mathbf{J}_a). \end{aligned}$$

Thus,

$$q \cdot w(\mathbf{I}_{\max}) + \Delta_t \leq q \cdot w(\mathbf{I}_a) + w(\mathbf{J}_a).$$

Together with Equation (21) we see that $q \cdot w(\mathbf{I}_a) + q \cdot w(\mathbf{J}_a) + \Delta_t \leq q \cdot w(\mathbf{I}_a) + w(\mathbf{J}_a)$, and therefore $q \cdot w(\mathbf{J}_a) + \Delta_t \leqslant w(\mathbf{J}_a)$ which is possible iff $w(\mathbf{J}_a) = 0$ (and $\Delta_t = 0$, which is indeed true, see Equation (16)). Now, $w(\mathbf{J}_a) = 0$ implies:

$$\begin{aligned} &w(\mathbf{I}_a) = w(\mathbf{I}_{\max}), \text{ and,} \\ &\mathbf{J}_a = \emptyset. \end{aligned}$$

We saw in Equation (20) that $w(\mathbf{I}) + w(\mathbf{J}) = w(\mathbf{I}_{\max}) + w(\mathbf{J}_{\max})$. If $\mathbf{I}_b \neq \emptyset$ then

$$\begin{aligned} w(\mathbf{I}) &= w(\mathbf{I}_a) + w(\mathbf{I}_b) \\ &> w(\mathbf{I}_a) = w(\mathbf{I}_a) + w(\mathbf{J}_a) = w(\mathbf{I}_{\max}). \end{aligned}$$

Thus, $(\mathbf{I}, \mathbf{J}) \succ_w (\mathbf{I}_{\max}, \mathbf{J}_{\max})$ in contradiction to the maximality of $(\mathbf{I}_{\max}, \mathbf{J}_{\max})$. We conclude that

$$\mathbf{I}_b = \emptyset.$$

As $(\mathbf{I}_{\max}, \mathbf{J}_{\max}) \in \mathcal{M}_{s,q}$ we have $(\mathbf{J}_{\max})_k \leq q - 1$ for every $k \in [m]$. Thus, $\mathbf{J}^b_{\max} =_{\mathbb{F}_q} \mathbf{J}_b$, $w(\mathbf{J}^b_{\max}) = w(\mathbf{J}_b)$ and $\mathbf{J}^b_{\max}$ is already reduced. Together this implies that

$$\mathbf{J}^b_{\max} = \mathbf{J}_b.$$

Finally, we use the hypothesis that $q \geq s$. We have, $(\mathbf{I}_{\max})_k < s \leq q$ for all $k \in [m]$. Thus, $\mathbf{I}_{\max} =_{\mathbb{F}_q} \mathbf{I}_a$, $w(\mathbf{I}_{\max}) = w(\mathbf{I}_a)$ and $\mathbf{I}_{\max}$ is already reduced. Together this implies that that

$$\mathbf{I}_{\max} = \mathbf{I}_a.$$

Thus,

$$\begin{aligned} &\mathbf{I}_{\max} = \mathbf{I}_a = \mathbf{I}_a + \mathbf{I}_b = \mathbf{I}, \\ &\mathbf{J}^b_{\max} = \mathbf{I}_b + \mathbf{J}_b = \mathbf{J}_b. \end{aligned}$$

Altogether,the only term that may possibly contribute $\mathbf{a}^{\mathbf{I}_{\max}}\mathbf{b}^{\mathbf{J}^b_{\max}}\mathbf{c}^{\mathbf{J}^c_{\max}}$ to

$$A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r} \bmod \mathcal{I}_{3m,1}$$

is

$$(\mathbf{I}_a, \mathbf{I}_b, \mathbf{J}_a, \mathbf{J}_b, \mathbf{J}_c) = (\mathbf{I}_{\max}, \emptyset, \emptyset, \mathbf{J}^b_{\max}, \mathbf{J}^c_{\max}).$$

Also, the tuple $(\mathbf{I}_a, \mathbf{I}_b, \mathbf{J}_a, \mathbf{J}_b, \mathbf{J}_c) = (\mathbf{I}_{\max}, \emptyset, \emptyset, \mathbf{J}^b_{\max}, \mathbf{J}^c_{\max})$ contributes

$$\alpha_{\mathbf{I}_{\max},\mathbf{J}_{\max}} \cdot \binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \sum_{\substack{j_1,j_2 \\ j_1 q + j_2 \leqslant w(\mathbf{J}^b_{\max}) \\ j_2 < q}} \beta_{w(\mathbf{J}^b_{\max}),j_1,j_2} \mathbf{a}^{\mathbf{I}_{\max}} \mathbf{b}^{\mathbf{J}^b_{\max}} \mathbf{c}^{\mathbf{J}^c_{\max}} \cdot g(t)^{w(\mathbf{I}_{\max})} \cdot g(r)^{j_1} r^{j_2}.$$

to the term in Equation (14).

Notice that $w(\mathbf{I}_{\max}) + j_1 < s$, for otherwise $q \cdot w(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) \geq q(w(\mathbf{I}_{\max}) + j_1) \geq qs$ in contradiction to the hypothesis. Thus the term is already $\mathcal{I}_{2,s}$ reduced. The contribution to $A_{w(\mathbf{I}_{\max}),\Delta_{rq},0,\Delta_r} \bmod \mathcal{I}_{3m,1}$ occurs for (j_1, j_2) such that $j_1 = \Delta_{rq}$ and $j_2 = \Delta_r$. Thus, in the sum in Equation (14) there is exactly one possible way to contribute $\mathbf{a}^{\mathbf{I}_{\max}} \mathbf{b}^{\mathbf{J}^b_{\max}} \mathbf{c}^{\mathbf{J}^c_{\max}}$ to $A_{w(\mathbf{I}_{\max}),\Delta_{rq},0,\Delta_r} \bmod \mathcal{I}_{3m,1}$, and this is when

$$(\mathbf{I}_a, \mathbf{I}_b, \mathbf{J}_a, \mathbf{J}_b, \mathbf{J}_c) = (\mathbf{I}_{\max}, \emptyset, \emptyset, \mathbf{J}^b_{\max}, \mathbf{J}^c_{\max}), \text{ and}$$
$$(i_1, i_2, j_1, j_2) = (0, 0, \Delta_{rq}, \Delta_r).$$

The coefficient of this term is:

$$\alpha_{\mathbf{I}_{\max},\mathbf{J}_{\max}} \cdot \binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \cdot \beta_{w(\mathbf{J}^b_{\max}),\Delta_{rq},\Delta_r}.$$

$\alpha_{\mathbf{I}_{\max},\mathbf{J}_{\max}} \neq 0$. By assumption $\binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}}$ is non-zero. $\beta_{\ell,\lfloor \frac{\ell}{q} \rfloor,\ell \bmod q} = 1$ (see Example 43) and taking $\ell = w(\mathbf{J}^b_{\max})$ shows the coefficient is non-zero. As there is a unique term contributing the monomial with a non-zero coefficient, the monomial cannot cancel in $A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r} \bmod \mathcal{I}_{3m,1}$ and $A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r} \bmod \mathcal{I}_{3m,1}$ is non-zero. ◀

▶ **Lemma 62.** *There is a partition* $\mathbf{J}_{\max} = \mathbf{J}^b_{\max} + \mathbf{J}^c_{\max}$*, such that*

$$d < q \cdot w(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) \leq qs - 1, \text{ and,}$$
$$\binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \bmod p \neq 0.$$

Proof. Suppose $q = p^w$ where p is prime (if q is prime then $p = q$ and $w = 1$). We choose $\mathbf{J}^b_{\max}$ as follows. We go over $k = 1, \ldots, m$ and find the first $k_0 \geq 0$ such that $q \cdot w(\mathbf{I}_{\max}) + \sum_{k=0}^{k_0} (\mathbf{J}_{\max})_k > d$. There must be some $k_0 \leq m$ like that since $qw(\mathbf{I}_{\max}) + w(\mathbf{J}_{\max}) > d$. We set:

- $(\mathbf{J}^b_{\max})_k = (\mathbf{J}_{\max})_k$, for $k = 1, \ldots, k_0 - 1$, and,
- $(\mathbf{J}^b_{\max})_k = 0$, for $k = k_0 + 1, \ldots, m$.

Set $v = (\mathbf{J}_{\max})_{k_0}$. There is a first value $0 < v' \leq v$ such that

$$qw(\mathbf{I}_{\max}) + \sum_{k=1}^{k_0 - 1} (\mathbf{J}_{\max})_k + v' > d.$$

We express $v = (\mathbf{J}_{\max})_{k_0}$ in base p: $v = \sum_{\ell=0}^{w-1} v_\ell \cdot p^\ell$. We let v'' be the first integer such that:

- $v'' \geq v'$, and,
- If we express v'' in base p as $v'' = \sum_{\ell=0}^{w-1} v''_\ell \cdot p^\ell$ then $v''_\ell \leq v_\ell$ for every ℓ.

▷ Claim 63. $v' \leq v'' \leq \min\left\{v, v' + p^{w-1} - 1\right\}$.

Proof. Notice that v respects the conditions that we need, and so if $v \leq v' + p^{w-1} - 1$ the claim holds. Otherwise, $v \geq v' + p^{w-1} - 1$. Then, $v'' \leq v' + p^{w-1} - 1$ because we can increase v' by setting all the lower bits in the p-representation to 0, while increasing the most significant bit (that is multiplied by p^{w-1}) by 1. ◁

Thus, by Lucas theorem (see Equation (4))

$$\binom{v}{v''} \bmod p = \Pi_{\ell=0}^{w-1} \binom{v_\ell}{v''_\ell} \neq 0.$$

Having that, we let $(\mathbf{J}^b_{\max})_{k_0} = v''$. We have, $\mathbf{J}^b_{\max} \leq \mathbf{J}_{\max}$, $qw(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) > d$ and $\binom{\mathbf{J}_{\max}}{\mathbf{J}^b_{\max}} \bmod p \neq 0$. Also,

$$qw(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) \leq d + p^{w-1} \leq qs - 1,$$

because $d \leq qs - p^{w-1} - 1 = q(s - \frac{1}{p}) - 1$, completing the proof of the lemma. ◀

We are now ready to prove Theorem 60.

Proof. Fix a partition $\mathbf{J}_{\max} = \mathbf{J}^b_{\max} + \mathbf{J}^c_{\max}$ as in Lemma 62. Let

$$\begin{aligned} degr &= \deg_r(g(t)^{w(\mathbf{I}_{\max})} r^{w(\mathbf{J}^b_{\max})} \bmod \mathcal{I}_{2,s}) \\ &= \deg_r(g(t)^{w(\mathbf{I}_{\max})} r^{w(\mathbf{J}^b_{\max})}) = w(\mathbf{J}^b_{\max}). \end{aligned}$$

Define $\Delta_{rq} = \lfloor \frac{w(\mathbf{J}^b_{\max})}{q} \rfloor$,$\Delta_t = 0$ and $\Delta_r = w(\mathbf{J}^b_{\max}) \bmod q$. By Lemma 61 we know that

$$A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r}(\mathbf{a},\mathbf{b},\mathbf{c}) \bmod \mathcal{I}_{3m,1} \neq 0$$

Thus, there exist $\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0$ such that

$$A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r}(\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0) \neq 0$$

We look at the test $\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0$. We have

$$PP_{\mathbf{a},\mathbf{b},\mathbf{c}}(r,t) \bmod \mathcal{I}_{2,s} = \sum_{i+j<s,k,\ell<q} A_{i,j,k,\ell}(\mathbf{a},\mathbf{b},\mathbf{c}) g(t)^i g(r)^j t^k r^\ell.$$

As $A_{w(\mathbf{I}_{\max}),\Delta_{rq},\Delta_t,\Delta_r}(\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0) \neq 0$ we see that

$$\begin{aligned} \deg(PP_{\mathbf{a}_0,\mathbf{b}_0,\mathbf{c}_0} \bmod \mathcal{I}_{2,s}) &\geq q \cdot w(\mathbf{I}_{\max}) + q \cdot \Delta_{rq} + \Delta_t + \Delta_r \\ &= q \cdot w(\mathbf{I}_{\max}) + w(\mathbf{J}^b_{\max}) > d. \end{aligned}$$

Thus, by Lemma 47, the test $(\mathbf{a}_0, \mathbf{b}_0, \mathbf{c}_0)$ rejects. ◀

7 Planes give a local MRM tester

We restate Theorem 9:

▶ **Theorem 64.** *Suppose q is a prime power, $s \leq q$ and $d < q(s - \frac{1}{p})$. Let $T : \mathbb{F}_q^m \to \Sigma_{m,s}$ be a table and let $\delta = \delta(T, \mathrm{MRM}(q,m,d,s))$. Then*

$$\mathrm{REJ}^{\mathrm{MRM}}_{2,d}(T) \geq \min\{\alpha\delta, c\}$$

with $\alpha = \Omega(q^{-6s+5})$ and $c = \Omega(q^{-8s+4})$.

Proof. We remind the reader that

$$H_k = \left\{\mathbf{h} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_k) \mid \mathbf{h}_0, \ldots, \mathbf{h}_k \in \mathbb{F}_q^m, \dim(\mathrm{span}\{\mathbf{h}_1, \ldots, \mathbf{h}_k\}) = k\right\}$$

Let us say that $\mathbf{v} \in H_k$ is *bad*, if the k-dimensional test with $\mathbf{v}$ rejects, i.e., $(\phi_{\mathbf{v}} \circ T)|_{A_{\mathbf{v}}} \notin \mathrm{MRM}(q, k, d, s)$. We let $t = \lceil \frac{d+1}{q-\frac{q}{p}} \rceil$ be the $\mathrm{RM}(q, m, d, s)$ testing dimension. Selecting a uniform $\mathbf{u} \in H_2$ is the same as selecting $\mathbf{h} \sim H_t$ and then a uniform $\mathbf{u} \in H_2$ such that $A_{\mathbf{u}} \subset A_{\mathbf{h}}$. Thus,

$$\begin{aligned}\mathrm{REJ}_{2,d}^{\mathrm{MRM}}(T) &= \Pr_{\mathbf{u}\sim H_2}(\mathbf{u} \text{ is bad}) \\ &\geq \Pr_{\mathbf{h}\in H_t}(\mathbf{h} \text{ is bad}) \cdot \Pr_{\mathbf{u}\in H_2 : A_{\mathbf{u}} \subset A_{\mathbf{h}}}(\mathbf{u} \text{ is bad} \mid \mathbf{h} \text{ is bad}).\end{aligned}$$

- By Corollary 52, the probability of picking a bad $\mathbf{h} \in H_t$ is:

$$\mathrm{REJ}_{t,d}^{\mathrm{MRM}}(T) \geq \min\left\{\frac{\delta}{3q}, \frac{1}{2(t+1)q^{t+1}}\right\}.$$

- By Theorem 60 we know that for any bad $\mathbf{h} \in H_t$ there is at least one $\mathbf{u} \in H_2$ such that $A_{\mathbf{u}}$ is contained $A_{\mathbf{h}}$ and $\mathbf{u}$ is bad. Furthermore, if $A_{\mathbf{u}} = A_{\mathbf{u}'}$ then $\mathbf{u}$ is bad iff $\mathbf{u}'$ is bad. We look at $A_{\mathbf{u}}$. There are $(q^2-1)(q^2-q)q^2$ different $\mathbf{u}' \in H_2$ such that $A_{\mathbf{u}'} = A_{\mathbf{u}}$ (because there are q^2-1 choices for the first basis element, q^2-q choices for the second basis element and $|A_{\mathbf{u}}| = q^2$ choices for the offset).

  Altogether,

$$\mathrm{REJ}_{2,d}^{\mathrm{MRM}}(T) \geq \frac{\Omega(q^6)}{q^{3t}} \cdot \min\left\{\frac{\delta}{3q}, \frac{1}{2(t+1)q^{t+1}}\right\} = \Omega(\min\left\{q^{-3t+5}\delta, \frac{q^{-4t+5}}{t}\right\}).$$

We notice that $t \leq \lceil \frac{q(s-\frac{1}{p})}{q-\frac{q}{p}} \rceil \leq \lceil \frac{s}{1-\frac{1}{p}} \rceil \leq 2s$. Thus $\alpha = \Omega(q^{-6s+5})$ and $c = \Omega(q^{-8s+4})$ as promised. ◀

References

1. Noga Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8(1-2):7–29, 1999.
2. Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.
3. Simeon Ball and Oriol Serra. Punctured combinatorial nullstellensätze. *Combinatorica*, 29(5):511–522, 2009.
4. Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of reed-muller codes. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 488–497. IEEE, 2010.
5. David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.
6. 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.
7. Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pages 190–198. IEEE, 1995.

8 Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. 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 32–42. ACM, 1991. doi:10.1145/103418.103429.

9 Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed–solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.

10 Elad Haramaty, Amir Shpilka, and Madhu Sudan. Optimal testing of multivariate polynomials over small prime fields. *SIAM Journal on Computing*, 42(2):536–562, 2013.

11 Charanjit S Jutla, Anindya C Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 423–432. IEEE, 2004.

12 Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing*, 36(3):779–802, 2006.

13 Swastik Kopparty. Some remarks on multiplicity codes. *Discrete Geometry and Algebraic Combinatorics*, 625:155–176, 2013.

14 Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of the ACM (JACM)*, 64(2):1–42, 2017.

15 Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of the ACM (JACM)*, 61(5):1–20, 2014.

16 Rasmus Refslund Nielsen. *List decoding of linear block codes*. PhD thesis, DTU, 2001.

17 M Yu Rosenbloom and Michael Anatol'evich Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.

18 Ronitt Rubinfeld and Madhu Sudan. Self-testing polynomial functions efficiently and over rational domains. In Greg N. Frederickson, editor, *Proceedings of the Third Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 27-29 January 1992, Orlando, Florida, USA*, pages 23–32. ACM/SIAM, 1992. URL: http://dl.acm.org/citation.cfm?id=139404.139410.

19 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

20 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

Pseudorandom Generators, Resolution and Heavy Width

Dmitry Sokolov
St. Petersburg State University, Russia
St. Petersburg Department of Steklov Mathematical Institute of Russian Academy of Sciences, Russia

Abstract

Following the paper of Alekhnovich, Ben-Sasson, Razborov, Wigderson [2] we call a pseudorandom generator $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}^m$ hard for a propositional proof system P if P cannot efficiently prove the (properly encoded) statement $b \notin \mathrm{Im}(\mathcal{F})$ for any string $b \in \{0,1\}^m$.

In [2] the authors suggested the "functional encoding" of the considered statement for Nisan–Wigderson generator that allows the introduction of "local" extension variables. These extension variables may potentially significantly increase the power of the proof system. In [2] authors gave a lower bound of $\exp\left[\Omega\left(\frac{n^2}{m\cdot 2^{2^\Delta}}\right)\right]$ on the length of Resolution proofs where Δ is the degree of the dependency graph of the generator. This lower bound meets the barrier for the restriction technique.

In this paper, we introduce a "heavy width" measure for Resolution that allows us to show a lower bound of $\exp\left[\frac{n^2}{m2^{\mathcal{O}(\varepsilon\Delta)}}\right]$ on the length of Resolution proofs of the considered statement for the Nisan–Wigderson generator. This gives an exponential lower bound up to $\Delta \coloneqq \log^{2-\delta} n$ (the bigger degree the more extension variables we can use). In [2] authors left an open problem to get rid of scaling factor 2^{2^Δ}, it is a solution to this open problem.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases proof complexity, pseudorandom generators, resolution, lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.15

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2021/076/

Acknowledgements I would like to thank Anastasia Sofronova, Edward Hirsch for fruitful discussions and attempts to fix my writing; anonymous reviewers for improving the text; Alexander Razborov and anonymous reviewers for pointing out incorrect operations with "Canonical Form" in an earlier draft of the paper. The work was supported by the Theoretical Physics and Mathematics Advancement Foundation "BASIS".

1 Introduction

Pseudorandom generators [25] are one the most important notions in modern computer science. A pseudorandom generator can be considered as a function $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}^m$ such that for all small circuits $C \in \mathfrak{C}$:

$$\left|\Pr_{x\in\{0,1\}^n}[C(\mathcal{F}(x))] - \Pr_{y\in\{0,1\}^m}[C(y)]\right| \xrightarrow[n\to\infty]{} 0,$$

where $\mathfrak{C}$ is some circuit class, and x, y are taken from the uniform distribution.

The condition on a pseudorandom generator can be rephrased in the following more informal way: "For a class of circuits $\mathfrak{C}$ it is hard to distinguish points inside and outside of the image of $\mathcal{F}$". This fact was used by Alekhnovich, Ben-Sasson, Razborov, and Wigderson [2] who suggested a natural way of viewing pseudorandom generators from the proof complexity perspective. Following [2] we call a pseudorandom generator $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}^m$ hard for

a propositional proof system P if P cannot efficiently prove the (properly encoded as a CNF formula) statement $b \notin \mathrm{Im}(\mathcal{F})$ for any string $b \in \{0,1\}^m$. Similar constructions were also proposed by Krajíček [13].

The problem of proving lower and upper bounds on the considered formulas is natural and at the same time there are lots of motivations from different areas of computer science. We discuss some of them and also refer the reader to [2, 5, 22] for the detailed survey.

Candidate Hard Examples for Strong Proof Systems. Despite the success of proving lower bounds on weak proof systems like Resolution, Polynomial Calculus, etc. we are still far away from lower bounds on strong proof systems like Frege or Extended Frege. Moreover, at this moment, we have few candidates for hard examples for these systems. In [22] Razborov introduced explicit conjectures that formulas obtained from Nisan–Wigderson pseudorandom generators are hard for Frege and Extended Frege.

These conjectures of Razborov are based on the deep connection between pseudorandom generators and so-called $\mathsf{Circuit}$ formulas. This provides another important motivation in circuit complexity.

Circuit Lower Bound. In [18] Razborov introduced the principle $\mathsf{Circuit}_t(f_n)$ expressing that the circuit size of the Boolean function f_n in n variables, given as its truth-table, is lower bounded by $t = t(n)$. Razborov stated that to show that a proof system P does not have efficient proofs of the formula $\mathsf{Circuit}_t(f_n)$, it suffices to design a sufficiently constructive pseudorandom generator hard for P and such that the number of output bits, as a function of the number of input bits, is as large as possible. In other words, the pseudorandom generators in proof complexity capture the arguments that are required to prove the circuit lower bounds (see also [2, 19]).

In this paper, we focus on the Nisan–Wigderson generators that were mention above. This partial case already illustrates all considered applications and shows the limits of the current techniques for proving lower bounds in proof complexity that we discuss next section.

1.1 Nisan–Wigderson Generators

The Nisan–Wigderson pseudorandom generator [16] may be described by a family of functions $f \coloneqq \{f_1, \ldots, f_m\}$ and a bipartite dependency graph $G \coloneqq (L, R, E)$ where $|L| = m$, $|R| = n$ and each vertex in L has degree Δ. We identify the right part of this graph with a set of boolean variables $x_1, \ldots, x_n$ and the left part with the output bits. We define a function $\mathcal{F}_{G,f} \colon \{0,1\}^n \to \{0,1\}^m$ such that the j-th bit of the output is defined by $f_j(x_{i_1}, x_{i_2}, \ldots, x_{i_\Delta})$ where x_{i_k} are neighbours of the vertex $v_j \in L$ that are ordered in arbitrary but fixed way.

Pick some $b \in \{0,1\}^n \setminus \mathrm{Im}(\mathcal{F}_{G,f})$. We produce the unsatisfiable CNF formula $\mathsf{PRG}_{G,f,b}$ that states $b \in \mathrm{Im}(\mathcal{F})$ in the most natural way i.e. we encode the constraints $f_j(x) = b_j$ independently. If the function f_j is simple enough (or Δ is small enough) then we can encode it in CNF directly in terms of $x_{i_1}, x_{i_2}, \ldots, x_{i_\Delta}$. But if $\Delta \gg \log n$ this encoding will be superpolynomial even in the case of parity function. To solve this problem, in [2] the authors suggested to use "local" extension variables (so-called "functional encoding"). In other words we can introduce any variable y_g whose value corresponds to some function g that depends on some set of variables X_g and $X_g \subseteq \mathrm{N}(v)$ where $v \in L$ and $\mathrm{N}(v)$ is a set of neighbours of v.

Another important motivation for the considered functional encoding is that it naturally characterizes the spectrum of proof systems between Resolution and Extended Frege. To see this we remind a classical Theorem that Resolution with all extension variables is equivalent to Extended Frege [8, 12]. So if we omit the locality constraint $X_g \subseteq \mathrm{N}(v)$ and allow all

possible extension variables then any lower bound on the size of Resolution proofs can be transformed into lower bounds on the Extended Frege. Note that the bigger Δ the more extension variables we allow to use, and the behaviour of Resolution is closer to the behaviour of Extended Frege. So the question about proving the lower bounds for a bigger degree of the dependency graph is a necessary step for proving lower bound on stronger proof systems.

1.1.1 Technical Aspects of Proving Lower Bounds

The most popular technique for proving lower bounds in proof complexity is a restriction. The main idea of this technique that we can hit the small proof by some restriction and obtain a well-structured proof. For example, this trick was used to reduce the question about the size of the resolution proof to a question about the width of proof. It can be used explicitly [2] or implicitly [7, 11, 6].

The "quality" of the restriction trick depends on the number of variables in our formula. Hence the lower bounds on the functional encoding of Nisan–Wigderson generator are an important barrier. The lower bound presented in [2] shows the limits of the classical restriction approach.

1.1.2 Prior Results

Despite the importance of the problem, only a few lower bounds are known. Alekhnovich, Ben-Sasson, Razborov, and Wigderson [2] showed a lower bound $\exp\left[\Omega\left(\frac{n^2}{m\cdot 2^{2\Delta}}\right)\right]$ on the length of Resolution proofs on the functional encoding of the Nisan–Wigderson generator. Since this proof used the "pure restriction technique" it also works for the Polynomial Calculus, which was also done in the same paper. This is the only lower bound that deals with the full functional encoding. They formulated a list of open problems that included:

- to prove a lower bound that works for $m \gg n^2$;
- to get rid of the 2^{2^Δ} scaling factor in the lower bound.

The state of the art techniques say that if we want to prove a lower bound then we need an expansion property of the dependency graph of the NW generator. And without a solution to the second problem we cannot deal with large Δ and hence for large enough m we cannot deal with expanders. So it seems that a solution to the second problem is a necessary step for any solution to the first problem.

In [15] Krajíček showed a simplified proof of the lower bound from [2], but it works only for a certain choice of the small number of local extension variables. This lower bound is given via reduction from the Pigeonhole Principle and hence it works for the bigger class of proof systems. For another choice of local extension variables Razborov [22] showed a superpolynomial lower bound up to $m = \mathcal{O}\left(n^{\log n}\right)$. This lower bound works for the Resolution and k-DNF Resolution, and it is obtained via the so-called "Small Restriction Switching Lemma" [23, 22].

If we switch back from the Nisan–Wigderson generator to the general case then we must point out that in [22] Razborov showed a lower bound for subexponential parameter m. This lower bound is based on two ideas: a lower bound on the Nisan–Wigderson generator, and the composition Theorem [14, 22]. The generator used in this lower bound is a composition of several Nisan–Wigderson generators.

1.2 Our Results

We develop a new measure of resolution proofs that we call "heavy width". This measure gives us a way to deal with extension variables in a structured way. We modify the restriction technique and show that for "proper" formulas small resolution proof may be transformed into a proof of small heavy width. Also, we show a way for proving lower bounds on heavy width (even in cases when we cannot bound classical resolution width).

By using the considered measure and techniques we show the following result, that is a solution to an open problem [2]: to get rid of scaling factor $2^{2^{\Delta}}$ in lower bounds on resolution proofs of PRG formulas.

▶ **Theorem 1** (Informal). *Let $G := (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander, where $|L| = m, |R| = n$. If f_i is a family of good functions then for any $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$ any resolution proof of $\mathsf{PRG}_{G,f,b}$ requires size $\exp\left[2^{-\mathcal{O}(\varepsilon\Delta)} \cdot \frac{r^2}{m}\right]$.*

For good enough expander graphs $r = \Omega\left(\frac{n}{\mathsf{poly}(\Delta)}\right)$. We give a definition of a "good" function and expander graph later (see Definition 3 and section 2.1). Informally speaking function is good iff it remains balanced even if we fix some of its input variables (Parity is a good function). We may think about expander graphs as about bipartite random graphs with left degree Δ.

The parameter ε may depends on n and Δ which allows us to show lower bounds of the form $\exp\left[2^{-o(\Delta)} \cdot \frac{r^2}{m}\right]$ for the proper expander graphs. In particular, we may consider natural distribution over random graphs $\mathcal{G}(m, n, \Delta)$ (see section 2.2) for which, our Theorem implies the following result.

▶ **Theorem 2** (Informal). *Let n be large enough integer number, $\delta > 0$, $m := n^{2-\delta}$, $\Delta := \log^{2-\delta} n$ and $G \sim \mathcal{G}(m, n, \Delta)$. If f_i is a family of good functions then whp for any $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$ any resolution proof of $\mathsf{PRG}_{G,f,b}$ requires size $\exp\left[n^{\Omega(\delta)}\right]$.*

We believe that heavy width measure could be of independent interest.

1.3 Our Technique

Let $\mathcal{F}_{G,f} \colon \{0,1\}^n \to \{0,1\}^m$. Let us remind that we pick some $b \in \{0,1\}^n \setminus \mathrm{Im}(\mathcal{F}_{G,f})$ and produce the unsatisfiable CNF formula $\mathsf{PRG}_{G,f,b}$ that states $b \in \mathrm{Im}(\mathcal{F})$ by encoding the constraints $f_j(x) = b_j$ independently. To do this, for every function g that depends on some set of variables X_g and $X_g \subseteq \mathrm{N}(v)$ where $v \in L$, we introduce an extension variable y_g whose value corresponds to g. For formal construction see Section 3. Note, that since for all $v \in L$ the size of $\mathrm{N}(v)$ is Δ our CNF formula consists of $m \cdot 2^{2^{\Delta}}$ variables.

We start with the approach that gives a lower bound $\exp\left[\Omega\left(\frac{n^2}{m \cdot 2^{2^{\Delta}}}\right)\right]$ on the size of resolution proofs of $\mathsf{PRG}_{G,f}$. This strategy has the same flavor as a strategy from [2] but has some differences in details.

Let $\pi := (D_1, \ldots, D_\ell)$ be a Resolution proof of $\mathsf{PRG}_{G,f}$ and H is a set of clauses of width at least w_0. For the sake of contradiction assume that π has small size and apply the following algorithm.

1. If π is small then H is small.
2. Pick the most frequent literal y in H. Note that it is contained in at least $\frac{w_0}{m \cdot 2^{2^{\Delta}+1}}$ fraction of clauses (by a naive averaging argument).
3. Set y to 0 in π. This operation kills all clauses that contain y.
4. After this assignment $\pi \restriction (y = 0)$ is still a proof of a restricted formula.

5. We apply a "closure" trick [3, 2] to make sure that the remaining formula does not contain a "local contradiction" (see also an iterative version of this trick in [24]).
6. Repeat while we have clauses of large width.

If H is small we kill all clauses of large width in a few iterations. To achieve a contradiction it remains to show that if there is no "local contradiction" then any resolution proof requires width at least w_0 for the right choice of w_0.

This strategy is **semantic**, i.e. we do not care about the exact form of clauses in the proof; we need only two properties:

- clauses of large width can be killed with large probability by a "random assignment";
- clauses of small width are not so easy to satisfy (we need this property for the width lower bound).

The bottleneck of the considered strategy is the fraction of clauses that contain some specific literal. So if we want to improve the lower bound, we need to expand this bottleneck. First of all, we will count the number of output bits that are "touched" (in other words, i-th output bit is touched iff there is a variable (or extension variable) in a clause from $\mathrm{N}(i)$ (that value depends only on $\mathrm{N}(i)$)) by a clause rather than the number of input variables. To do so we define a "functional form" of a clause that helps to split all variables into m baskets. Unfortunately, our functional form of a clause is a **syntactic** representation, i.e., it heavily depends on the exact representation of a clause and not only the function defined by it. Moreover this representation is not unique and that affects our complexity measures.

On the one hand, the syntactic measure has already provided problems in the analysis. On the other hand, it is still not enough for our lower bound. To expand the bottleneck even more we introduce the new measure "heavy width". Informally speaking if we have clause D then we want to count only those output bits of the generator the value of which are heavily correlated to a value of the clause D.

Let us define the full strategy. Let $\pi \coloneqq (D_1, \ldots, D_\ell)$ be a Resolution proof of $\mathsf{PRG}_{G,f}$ and H be a set of clauses of heavy width at least w_0 (in other words there are at least w_0 output bits of the generator whose values are correlated with a value of a clause). For the sake of contradiction assume that π has small size and apply the following algorithm.

1. If π is small then H is small.
2. Pick an output bit v of the generator uniformly at random.
3. Set all neighbors of v in order to satisfy constraints to this output bit. In our case this operation kills $\frac{2^{-\varepsilon\Delta}}{m}$ fraction of clauses in H (this argument will follow from the definition of heavy width)
4. After this assignment the restricted proof is still a proof of a restricted formula.
5. We apply a "closure" trick to make sure that the remaining formula does not contain "local contradiction".
6. *Make sure that heavy width of alive clauses does not grow too much.* This is the new and one of the most problematic step.
7. Repeat while we have clauses of large width.

If H is small we kill all clauses of large width in a few iterations. To achieve a contradiction it remains to show that if there is no "local contradiction" then any resolution proof requires a clause of large "heavy width".

To show the lower bound on the heavy width we equip a game approach (that is similar to [17, 4]) with a new invariant. This is the place where the problem with the syntactic definition of a functional form arises. To avoid this problem we again will use the expansion properties of our dependency graph.

2 Preliminaries

Let x be a propositional variable, i.e., a variable that ranges over the set $\{0,1\}$. A literal of x is either x (denoted sometimes as x^1) or $\neg x$ (denoted sometimes as x^0). A **clause** $C \coloneqq x_1^{c_1} \vee x_2^{c_2} \cdots \vee x_k^{c_k}$ is a disjunction of literals where $c_1, c_2, \ldots, c_k \in \{0,1\}$. A **CNF formula** $\varphi \coloneqq C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We think of clauses and CNF formulas as sets: order is irrelevant and there are no repetitions.

A **Resolution proof** π of an unsatisfiable CNF formula φ is an ordered sequence of clauses $\pi \coloneqq C_1, \ldots, C_s$ such that $C_s = \emptyset$ is an empty clause and for each $i \in [s]$ either C_i is a clause in φ or there exist $j, k < i$ such that C_i is derived from C_j and C_k by the **resolution rule**

$$\frac{C \vee x \qquad D \vee \neg x}{C \vee D}$$

or by the **weakening** rule

$$\frac{C}{D}\,[C \subseteq D].$$

A **partial assignment** or a **restriction** on a function f (or a formula φ) is a mapping $\rho\colon \mathrm{Vars}(f) \to \{0,1,*\}$. We let $\mathrm{supp}(\rho) \coloneqq \rho^{-1}(\{0,1\})$ denote the set of assigned variables. The restriction of a function f (or a formula φ) by ρ, denoted $f|_\rho$ ($\varphi|_\rho$), is the Boolean function (propositional formula) obtained from f (from φ, respectively) by setting the value of each $x_i \in \mathrm{supp}(\rho)$ to $\rho(x_i)$ and leaving each $x_i \notin \mathrm{supp}(\rho)$ unassigned.

The **size** of a partial assignment ρ is the size of the $\mathrm{supp}(\rho)$. We denote it by $|\rho|$.

▶ **Definition 3.** *Let $f\colon \{0,1\}^n \to \{0,1\}$. We say that f is (δ, k)-**balanced** for some $0 < \delta \leq \frac{1}{2}$ and $k \geq 0$ iff for any $b \in \{0,1\}$ and any partial assignment ρ of size at most k the size of $(f|_\rho)^{-1}(b)$ is at least $\delta \cdot 2^{n-|\rho|}$.*

Some examples.

- $\mathsf{Parity}(x_1, \ldots, x_n)$ is $\left(\frac{1}{2}, n-1\right)$-balanced;
- $\mathsf{IP} \coloneqq \sum_{i=1}^{n/2} x_i y_i \bmod 2$ is $\left(\frac{1}{4}, \frac{n}{2}-1\right)$-balanced;
- a random function is $\left(\frac{1}{4}, n-\sqrt{n}\right)$-balanced (see Lemma 30 for the calculations).

2.1 Expanders and Closure

We use the following notation: $\mathrm{N}_G(S)$ is the set of neighbours of the set of vertices S in the graph G, $\partial_G(S)$ is the set of vertices u that are connected with S by exactly one edge. We omit the index G if the graph is evident from the context.

▶ **Definition 4.** *A bipartite graph $G \coloneqq (L, R, E)$ is an (r, Δ, c)-**expander** if all vertices $u \in L$ have degree at most Δ and for all sets $S \subseteq L$, $|S| \leq r$, it holds that $|\mathrm{N}(S)| \geq c \cdot |S|$. Similarly, $G \coloneqq (L, R, E)$ is an (r, Δ, c)-**boundary expander** if all vertices $u \in L$ have degree at most Δ and for all sets $S \subseteq L$, $|S| \leq r$, it holds that $|\partial(S)| \geq c \cdot |S|$.*

In this context, a simple but useful observation is that

$$|\mathrm{N}(S)| \leq |\partial(S)| + \frac{\Delta|S| - |\partial(S)|}{2} = \frac{\Delta|S| + |\partial(S)|}{2},$$

since all non-unique neighbours have at least two incident edges. This implies that for any $\varepsilon \leq \frac{1}{2}$ if a graph G is an $(r, \Delta, (1-\varepsilon)\Delta)$-expander then it is also an $(r, \Delta, (1-2\varepsilon)\Delta)$-boundary expander.

The next proposition is well known in the literature. In this form it was used in [10].

▶ **Proposition 5.** *If $G \coloneqq (L, R, E)$ is an (r, Δ, c)-boundary expander then for any set $S \subseteq L$ of size $k \leq r$ there is an enumeration $v_1, v_2, \ldots, v_k \in S$ and a sequence $R_1, \ldots, R_k \subseteq \mathrm{N}(S)$ such that:*

- $R_i = \mathrm{N}(v_i) \setminus \left(\bigcup_{j=1}^{i-1} \mathrm{N}(v_j) \right)$;
- $|R_i| \geq c$.

In particular, there is a matching on the set S.

Proof. We create this sequence in reversed order. Since $|S| \leq r$ it holds that $|\partial(S)| \geq c|S|$ and there is a vertex $v_k \in S$ such that $|\partial(S) \cap \mathrm{N}(v_k)| \geq c$. Let $R_k \coloneqq |\partial(S) \cap \mathrm{N}(v_k)|$, and repeat the process on $S \setminus \{v_k\}$. ◀

Let $G \coloneqq (L, R, E)$ denote a bipartite graph. Consider a **closure** operation that seems to have originated in [3, 2].

▶ **Definition 6.** *For a vertex set $U \subseteq R$ we say that a set $S \subseteq L$ is (U, r, ν)-**contained** if $|S| \leq r$ and $|\partial(S) \setminus U| < \nu|S|$. For any set $J \subseteq R$ let $\mathrm{Cl}^{r,\nu}(J)$ denote an arbitrary but fixed set of maximal size such that $\mathrm{Cl}^{r,\nu}(J)$ is (J, r, ν)-contained. We say that $\mathrm{Cl}^{r,\nu}(J)$ is a **closure** of J.*

Note that for any $J \subseteq R$ and any positive r, ν the empty set is (J, r, ν)-contained and closure is well-defined.

▶ **Lemma 7.** *Suppose that G is an (r, Δ, c)-boundary expander and that $J \subseteq R$ has size $|J| \leq \Delta r$. Then $|\mathrm{Cl}^{r,\nu}(J)| < \frac{|J|}{c-\nu}$.*

Proof. By definition we have that $|\partial(\mathrm{Cl}^{r,\nu}(J)) \setminus J| < \nu|\mathrm{Cl}^{r,\nu}(J)|$. Since $|\mathrm{Cl}^{r,\nu}(J)| \leq r$ by definition, the expansion property of the graph guarantees that $c|\mathrm{Cl}^{r,\nu}(J)| - |J| \leq |\partial(\mathrm{Cl}^{r,\nu}(J)) \setminus J|$. The conclusion follows. ◀

Suppose $J \subseteq R$ is not too large. Then Lemma 7 shows that the closure of J is not much larger. Thus, after removing the closure and its neighbourhood from the graph, we are still left with a decent expander. The following lemma makes this intuition precise.

▶ **Lemma 8.** *Let $J \subseteq R$ be such that $|J| \leq \Delta r$ and $|\mathrm{Cl}^{r,\nu}(J)| \leq \frac{r}{2}$ and let $G' \coloneqq G \setminus (\mathrm{Cl}^{r,\nu}(J) \cup J \cup \mathrm{N}(\mathrm{Cl}^{r,\nu}(J)))$. Then any set S of vertices from the left side of G', with size $|S| \leq \frac{r}{2}$, satisfies that $|\partial_{G'}(S)| \geq \nu|S|$.*

Proof. Suppose the set $S \subseteq L(G')$ violates the boundary expansion guarantee. Observe that $\mathrm{Cl}^{r,\nu}(J)$ and S are both sets of size at most $\frac{r}{2}$. Furthermore, the set $(\mathrm{Cl}^{r,\nu}(J) \cup S)$ is (J, r, ν)-contained in the graph G. As $\mathrm{Cl}^{r,\nu}(J)$ is a (J, r, ν)-contained set of maximal cardinality, this leads to a contradiction. ◀

2.2 Existence

For $n, m, \Delta \in \mathbb{N}$, we denote by $\mathcal{G}(m, n, \Delta)$ the distribution over bipartite graphs with disjoint vertex sets $L \coloneqq \{v_1, \ldots, v_m\}$ and $R \coloneqq \{u_1, \ldots, u_n\}$ where the neighbourhood of a vertex $v \in L$ is chosen by sampling a subset of size Δ uniformly at random from R.

The next claim follows from the standard calculation.

▶ **Lemma 9** (de Rezende et al. [9]). *Let n, m 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.*

3 Nisan–Wigderson PRG and Its Encoding

Let $G \coloneqq (L, R, E)$ be a bipartite graph such that $L \coloneqq \{v_1, v_2, \ldots, v_m\}$, $R \coloneqq \{u_1, u_2, \ldots, u_n\}$ and each vertex in L has degree Δ. Also for each vertex $v \in L$ we fix some arbitrary enumeration of its neighbours. We identify the right part of this graph with a set of boolean variables $\{x_1, x_2, \ldots, x_n\}$ and the left part with a set of output bits. Based on this identity we introduce a pseudorandom generator $\mathcal{F}_{G,f} \colon \{0,1\}^n \to \{0,1\}^m$ that is defined by the graph G and a family of the **base** functions $f_1, f_2, \ldots, f_m \colon \{0,1\}^\Delta \to \{0,1\}$ in the natural way: the j-th bit of output is defined by $f_j(u_{i_1}, u_{i_2}, \ldots, u_{i_\Delta})$ (here we use enumeration of neighbours of the vertex v_j) where $u_{i_k} \in \mathrm{N}(v_j)$ is a set of neighbours of the vertex $v_j \in L$. We also use a notation $\mathrm{Vars}_j \coloneqq \mathrm{N}(v_j)$.

We want to encode the question about inversion of the function $\mathcal{F}_G$ as a propositional formula. Following the [2] and [1] we allow to use "local" extension variables.

3.1 Functional Encoding

Let $\mathcal{F}_{G,f} \colon \{0,1\}^n \to \{0,1\}^m$ be a pseudorandom generator based on the graph G and base functions $f_1, f_2, \ldots, f_m \colon \{0,1\}^\Delta \to \{0,1\}$. Let $b \in \{0,1\}^m$ be an arbitrary point. We say that a boolean function g is **local** iff there is some $i \in [m]$ such that g depends only on Vars_i. Let $\mathfrak{G}$ be a collection of local functions.

For each local function $g \in \mathfrak{G}$ we introduce a variable y_g. And we write a CNF formula $\mathsf{PRG}_{G,f,b}$ on variables y_g which consists of the following disjunctions:

- $(y_{g_1}^{c_1} \vee y_{g_2}^{c_2} \vee y_{g_3}^{c_3} \cdots \vee y_{g_\ell}^{c_\ell})$, for all tuples $g_1, g_2, \ldots, g_\ell$ where $\ell \leq 2^{2^\Delta}$ and all $c_1, c_2, \ldots, c_\ell \in \{0,1\}$ such that there is $i \in [m]$:
  - g_j depends only on Vars_i for all $j \in [\ell]$;
  - any assignment $a \in \{0,1\}^\Delta$ that satisfy the equality $f_i(a) = b_i$ also satisfy the equality $g_k = c_k$ for at least one $k \in [\ell]$. In other words the equality $g_k = c_k$ **semantically follows** from the equality $f_i(a) = b_i$.

Note that, in particular, $\mathsf{PRG}_{G,f,b}$ contains the following constraints:

- $(y_{f_i}^{b_i})$, for all $i \in [m]$;
- $(\neg y_s \vee y_g)$, $(\neg y_s \vee y_h)$ and $(y_s \vee \neg y_g \vee \neg y_h)$ for all local functions s, g, h such that:
  - $s = g \wedge h$;
  - there is $i \in [m]$ such that s, g, h depends only on Vars_i.

We omit indices of $\mathsf{PRG}_{G,f,b}$ if it is clear from the context. Following [2] we say that it is the **functional encoding**. The following observation is a straightforward corollary from the definition.

▶ Remark 10 (Aleckhovich et al. [2]). Formula $\mathsf{PRG}_{G,f,b}$ is unsatisfiable iff $b \notin \mathrm{Im}(\mathcal{F}_G)$.

3.1.1 Assignments, Restrictions and Intuition

If we have some total assignment ρ to x variables of $\mathcal{F}$ (we call it **x-assignment**) it can define an assignment on y variables in the natural way:

- $y_{x_i} \leftarrow \rho(x_i)$;
- if $g(x)$ is a local function then $y_g \leftarrow g(x)|_\rho$.

We denote this assignment by ρ^y. The intuition behind the considered encoding and assignment is that a value of a variable y_g on ρ^y corresponds to a value of a function g on ρ.

We can also define this extension (of x-assignments to y variables) for partial assignments as well. If we have some partial assignment ρ to x variables, we define an assignment ρ^y in the following way: $y_g|_{\rho^y} \coloneqq y_{g|_\rho}$. We say that these assignments are **normal**. In this paper we consider only normal assignments.

Consider a clause $D \coloneqq (y_{g_1}^{c_1} \vee y_{g_2}^{c_2} \vee \cdots \vee y_{g_\ell}^{c_\ell})$ in y variables where $c_i \in \{0,1\}$. Note, that under normal assignments:

- $y_g^0 \equiv y_{1-g}$;
- $y_g \vee y_{g'} \equiv y_{g \vee g'}$.

We can use these equalities and group variables of the clause D. Let $B_1, B_2, \ldots, B_m$ be a sequence of subsets (or **bags**) of literals $\{y_{g_1}^{c_1}, y_{g_2}^{c_2}, \ldots, y_{g_\ell}^{c_\ell}\}$ such that:

- for all $i \in [m], j \in [\ell]$ if $y_{g_j}^{c_j} \in B_i$ then $\mathrm{Vars}(g_j) \subseteq \mathrm{Vars}_i$;
- for all $j \in [\ell]$ there is at least one $i \in [m]$ such that $y_{g_j}^{c_j} \in B_i$.

Note that we can rewrite a clause D in the equivalent form under normal assignments $D \equiv (y_{h_1} \vee y_{h_2} \vee \cdots \vee y_{h_m})$ where

$$h_i(x) \coloneqq \bigvee_{y_g^c \in B_i} (1 \oplus c \oplus g(x)).$$

We may think about a clause D as about the following disjunction:

$$F \coloneqq \bigvee_{i \in [m]} (h_i(x) = 1),$$

and we say that F is a **functional form** of the clause D. Denote by

$$F|_\rho \coloneqq \bigvee_{i \in [m]} (h_i(x)|_\rho = 1),$$

where ρ is an x-assignment.

▶ Remark 11. Functional form is not unique.

To illustrate Remark 11 consider the following situation for some function g such that $y_g \in D$: $\mathrm{Vars}(g) \subseteq (\mathrm{Vars}_i \cap \mathrm{Vars}_{i'})$. In this case we can put the literal y_g into a bag B_i, into a bag $B_{i'}$, or into both of these bags (also into none of them if we have an opportunity to put it into some third bag).

On the one hand Remark 11 provides additional problems if we want to work with functional forms of the clauses since we should care about all possible functional forms. On the other hand non-uniqueness allows to deal with assignments.

▶ **Lemma 12.** *Let $F \coloneqq \bigvee_{i \in [m]} (h_i(x) = 1)$ be a functional form of a clause D. If ρ is a partial x-assignment then $F|_\rho$ is a functional form of a clause $D|_{\rho^y}$.*

Proof. Follows from definition of the functional form. Let $D \coloneqq (y_{g_1}^{c_1} \vee y_{g_2}^{c_2} \vee \cdots \vee y_{g_\ell}^{c_\ell})$, and $B_1, B_2, \ldots, B_m$ be a collection of bags that generates the functional form $(y_{h_1} \vee y_{h_2} \vee \cdots \vee y_{h_m})$.

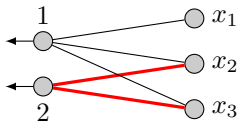


Figure 1 Dependency graph.

Note that $D|_{\rho^y} = (y^{c_1}_{g_1|_\rho} \vee y^{c_2}_{g_2|_\rho} \vee \cdots \vee y^{c_\ell}_{g_\ell|_\rho})$. We create a collection of bags B'_i for a clause $D|_{\rho^y}$ and we put $y^{c_j}_{g_j|_\rho} \in B'_i$ iff $y^{c_j}_{g_j} \in B_i$. By construction it satisfy all required properties for bags and

$$\bigvee_{y^c_g \in B'_i} (1 \oplus c \oplus g(x)) = \bigvee_{y^c_g \in B_i} (1 \oplus c \oplus g(x)|_\rho) = \left(\bigvee_{y^c_g \in B_i} (1 \oplus c \oplus g(x)) \right) |_\rho$$

that concludes the proof. ◀

▶ Remark 13. The definition of functional form is "syntactic", or in other words it heavily depends on variables that appear in the clause D and not only on the boolean function that is defined by it.

To illustrate the remark above let us consider an example. At first we have to define dependency graph (otherwise the notion of local function is meaningless). The graph is defined on fig. 1. Let us choose some collection of local functions:

$$\ell(x) \coloneqq x_1 \oplus x_2 \oplus x_3, \quad \ell'(x) \coloneqq x_1 \oplus x_2, \quad \ell''(x) \coloneqq x_3$$

and consider two clauses:

$$D \coloneqq y_\ell \vee y_{\ell'}, \quad D' \coloneqq y_\ell \vee y_{\ell''}.$$

To define a functional form of a clause D we have to define two bags B_1, B_2. We have to put literal y_ℓ into the bag B_1 (we have to put it somewhere, and we cannot put it into bag B_2 since $\mathrm{Vars}(\ell) \not\subseteq \mathrm{Vars}_2$). The same situation with the literal $y_{\ell'}$, so there is the only way to define bags: $B_1 \coloneqq \{y_\ell, y_{\ell'}\}, B_2 \coloneqq \emptyset$, and in this case the functional form of D is unique and is defined by the following functions:

$$h_1(x) \coloneqq (x_1 \oplus x_2 \oplus x_3) \vee (x_1 \oplus x_2), \quad h_2(x) \coloneqq 0.$$

To define a functional form of a clause D' we have to define two bags B'_1, B'_2. But the situation is different for this clause. Again we have to put literal y_ℓ into the bag B'_1, but the literal $y_{\ell''}$ we can put into B'_1 or B'_2 or into both of them, so there are three ways:

- $B'_1 \coloneqq \{y_\ell, y_{\ell''}\}, B'_2 \coloneqq \emptyset$ that give a functional form that is defined by the functions:

  $$h_1(x) \coloneqq (x_1 \oplus x_2 \oplus x_3) \vee x_3, \quad h_2(x) \coloneqq 0;$$

- $B'_1 \coloneqq \{y_\ell\}, B'_2 \coloneqq \{y_{\ell''}\}$ that give a functional form that is defined by the functions:

  $$h_1(x) \coloneqq (x_1 \oplus x_2 \oplus x_3), \quad h_2(x) \coloneqq x_3;$$

- $B'_1 \coloneqq \{y_\ell, y_{\ell''}\}, B'_2 \coloneqq \{y_{\ell''}\}$ that give a functional form that is defined by the functions:

  $$h_1(x) \coloneqq (x_1 \oplus x_2 \oplus x_3) \vee x_3, \quad h_2(x) \coloneqq x_3.$$

So functional forms of D and D' are different, but under normal assignments these clauses are equivalent. The observation 13 is a source of problems for the proof of the main Theorem, since we should always pay an attention to the exact form of the clauses.

4 Lower Bound

In this section we prove the main Theorem.

▶ **Theorem 14** (Formalization of Theorem 1). *Let $G := (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander, where $|L| = m, |R| = n$. If f_i is a family of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions then for any $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$ any resolution proof of $\mathsf{PRG}_{G,f,b}$ has size $\exp\left[\Omega\left(\frac{\varepsilon^5 r^2}{2^{6\varepsilon\Delta} m}\right)\right]$.*

We defer the proof of this Theorem to section 4.4 and start with a plan of our proof.

- We introduce an analog of the width measure on clauses with two important differences:
  - we want to count number of output bits that are touched by a clause rather than number of variables;
  - we want to count only those outputs that we cannot erase from a clause "for free".

  Let call this measure "heavy width".
- We hit our proof by a random restriction. We will do it step by step and at each step we create an x-assignment σ by choosing some output bit $v_i \in [m]$ and assign its neighbours in order to satisfy it. This assignment is equivalent to erasing some vertices from the right part of the graph. So after each step some output bit will not satisfy the expansion property of the graph. We say that these output bits are in danger and choose some x-assignment ν on its neighbours to satisfy them. We hit our proof by $(\sigma \cup \nu)^y$.
- We repeat this process while we do not kill all clauses of big heavy width. At this step it is important that we deal with heavy width rather than usual width.
- We prove a lower bound on the heavy width. For the sake of contradiction we assume that there is a proof of small heavy width. We trace a path in this resolution proof from the final clause to some axiom and maintain a partial assignment that:
  - does not satisfy the current clause (note that this clause may have a large classical width, hence our assignments will not set this clause to a constant);
  - does not violate any axiom.

  In the leaf these properties will give a contradiction.

We apply this Theorem for good enough graphs.

▶ **Theorem 15** (Formalization of Theorem 2). *Let n be large enough integer number, $\delta > 0$, $m := n^{2-\delta}$, $\Delta := \log^{2-\delta} n$ and $G \sim \mathcal{G}(m, n, \Delta)$. If f_i is a family of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions then whp for any $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$ any resolution proof of $\mathsf{PRG}_{G,f,b}$ has size $\exp\left[n^{\Omega(\delta)}\right]$.*

Proof. Fix $\chi := n^{\delta/10}$ and $\xi := \frac{100}{\delta \log n}$.

We use Lemma 9 and show that our graph G whp is an $\left(\frac{n^{1-\delta}}{\mathsf{polylog}(n)}, \log^{2-\delta} n, (1 - \frac{200}{\delta \log n})\Delta\right)$-expander. Indeed:

- $\xi < \frac{1}{2}$;
- $\xi \ln \chi = \frac{100}{\delta \log n} \frac{\delta}{10} \ln n > 2$;
- $\xi \ln \chi\Delta \geq 4 \ln m$.

Hence by Theorem 14 size of any resolution proof of $\mathsf{PRG}_{G,f,b}$ has size at least $\exp\left[\Omega\left(\frac{n^{2-\delta/5}}{\mathsf{polylog}(n) 2^{\Omega(\log^{1-\delta} n)} m}\right)\right] \geq \exp\left[\Omega\left(\frac{n^{2-\delta/5}}{n^{2-\delta/2}}\right)\right] = \exp\left[n^{\Omega(\delta)}\right]$. ◀

▶ Remark 16. Note that if f_i is a balanced function then $f_i(x) \oplus b_i$ is also a balanced function. Hence to simplify the notation wlog we assume that $b = 0^n$ and we omit an index b in the rest of the section. All the results holds for any $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$.

4.1 The "Heavy Width"

In the classical restriction technique the notion of the width of a clause C is used to estimate the probability that a random restriction will satisfy a clause. We give the next definition in order to save this property even if can deal with extension variables.

▶ **Definition 17.** *Fix a formula* $\mathsf{PRG}_{G,f}$*. Let* C *be a clause with functional form:* $F \coloneqq \bigvee_{i=1}^{m}(h_i(x)=1)$*. We say that* i*-th output bit is* η***-heavy*** *in* F *wrt* $\mathsf{PRG}_{G,f}$ *iff* $\Pr_{z\leftarrow f_i^{-1}(0)}[h_i(z)=1]\geq\eta$*. And the* η***-heavy width*** *or* $\mathrm{hw}^{\eta}_{\mathsf{PRG}_{G,f}}$ *of* F *is the number of* η*-heavy output bits in* F*. This definition of width depends on the formula.*

To justify this notion we may think about the "information" about i-th output bit in the clause C (despite on the fact that it is defined for functional form). We pick a point $z\in\{0,1\}^{\Delta}$ that satisfy the constraint $f_i(z)=0$ uniformly at random. If the probability that we satisfy C by this assignment is small then C "almost avoids" y variables that belongs to i-th output bit. In this case we pretend that the clause C is independent of i-th output bit, otherwise the value of C is heavily correlated with the value of f_i.

▶ Remark 18. The standard width measure can be considered as an η-heavy width measure. But in the different part of the proofs of classical resolution lower bounds we assume different parameters η.

- for the reduction from size to width: η is an absolute positive constant (usually $\frac{1}{2}$);
- for the width lower bound: $0<\eta<\frac{1}{2^n}$.

And it works since without extension variables for local functions we can state that: if an output bit is η-heavy for some $\eta>0$ then it also η'-heavy for some $\eta'\approx\frac{1}{2}$.

We define heavy width on functional form of the clause, that may give us potential problems due to the Remark 13.

▶ **Definition 19.** *Let* $\pi\coloneqq D_1,D_2,\ldots,D_s$ *be a resolution proof of* $\mathsf{PRG}_{G,f}$*. And the* η***-heavy width*** $\mathrm{hw}^{\eta}_{\mathsf{PRG}_{G,f}}$ *of the proof* π *is the minimal natural number* w *such that there is a sequence* $F_1,F_2,\ldots,F_s$ *where for all* $i\in[s]$*:*

- F_i *is a functional form of* D_i*;*
- $\mathrm{hw}^{\eta}_{\mathsf{PRG}_{G,f}}$ *of* F_i *is at most* w*.*

4.2 Size to Heavy Width Reduction

In this section we present a random restriction argument that helps to reduce the question about size of proof to a question about η-heavy width of the proof for carefully chosen parameter η. Fix some $\mathsf{PRG}_{G,f}$.

Let define the key object that we use in our main Theorem.

▶ **Definition 20.** *Let* $G\coloneqq(L,R,E)$ *be an* $(r,\Delta,(1-\varepsilon)\Delta)$*-expander. We say that an* x*-assignment* ρ *of* $\mathsf{PRG}_{G,f}$ *is* ***self-reduction*** *iff there is a set* $L_\rho\subseteq L$ *such that:*

- $|L_\rho|\leq\varepsilon^2\frac{r}{16}$*;*
- ρ *assigns all and only variables from* $\mathrm{N}(L_\rho)$*, moreover* ρ *satisfy constraints from the set* L_ρ*;*
- $G\setminus(L_\rho\cup\mathrm{N}(L_\rho))$ *is an* $(r,\Delta,(1-2\varepsilon)\Delta)$*-expander.*

The ***size*** *of self-reduction is the size of the set* L_ρ*.*

The next observation is trivial, but at the same it gives an opportunity to deal with heavy width measure since it is defined only for the PRG formulas.

▶ Remark 21. If ρ is a self-reduction of $\mathsf{PRG}_{G,f}$ then $\mathsf{PRG}_{G,f}|_{\rho^y}$ is equivalent to $\mathsf{PRG}_{G',f'}$ under normal assignments where:

- $G' \coloneqq G \setminus (L_\rho \cup \mathrm{N}(L_\rho))$;
- $f' \coloneqq \{f'_1, \ldots, f'_m\}$ and $f'_i \coloneqq f_i|_\rho$.

We use the following algorithm to generate self-reductions.

Algorithm 1 r, ε are parameters.

$O_1 \coloneqq \emptyset$ ▷ Set of **active** output bits
$G_1 \coloneqq G$ ▷ $G_i = (L_i, R_i, E_i)$
$i \coloneqq 1$
$\rho_1 \coloneqq \emptyset$
For all $j \in [m]$: $p_j^1 \coloneqq f_j$
while $i \leq \varepsilon^3 \frac{r}{32}$ **do**
    Pick a vertex $v^i \in L_i$ uniformly at random
    Pick an x-assignment $\sigma_i \leftarrow (p^i_{v^i})^{-1}(0)$ uniformly at random
    $O_{i+1} \coloneqq O_i \cup \{v^i\}$
    $G'_{i+1} \coloneqq G_i \setminus (\{v^i\} \cup \mathrm{N}_{G_i}(v^i))$
    $B_i \coloneqq \operatorname{argmax}\{|B| \mid B \subseteq L'_{i+1}, |B| \leq r, |\partial_{G'_{i+1}}(B)| \leq (1-2\varepsilon)|B|\}$
    Pick an x-assignment ν_i on $\mathrm{N}_{G'_{i+1}}(B_i)$ that satisfy all constraints from the set B_i
    $G_{i+1} \coloneqq G'_{i+1} \setminus (B_i \cup \mathrm{N}_{G'_{i+1}}(B_i))$
    $\rho_{i+1} \coloneqq \rho_i \cup \sigma_i \cup \nu_i$
    For all $j \in [m]$: $p_j^{i+1} \coloneqq f_j|_{\rho_{i+1}}$
    $i \coloneqq i + 1$
return ρ_i

Following the Remark 21 note that $(\mathsf{PRG}_{G,f})|_{\rho_i^y}$ is equivalent to PRG_{G_i,p^i}.

▶ **Lemma 22.** *Let $G \coloneqq (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander graph such that $|L| = m$, $|R| = n$ and $\frac{10}{\varepsilon} \leq r$. If $f \coloneqq \{f_1, \ldots, f_m\}$ is a collection of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions then Algorithm 1 generates a self-reduction of $\varphi \coloneqq \mathsf{PRG}_{G,f}$.*

Before the proof we present an intuition about parameters. We are given a family of expander graphs that fixes some ε (that we want to be as small as possible) and Δ. We choose a family of balanced functions with proper parameters. Parameter ε determines $\gamma \coloneqq 2^{-\varepsilon\Delta}$, on the one hand it is just abbreviation, on the other hand it corresponds to the scaling factor 2^ρ from the definition of balanced function. In the classical Resolution lower bounds γ is some constant (that is implicit and hidden inside the proof).

Proof. Let $\ell \coloneqq \varepsilon^3 \frac{r}{32}$ be the number of iterations of our algorithm.

By induction we show the following properties:

- G_i is an $(r, \Delta, (1-2\varepsilon)\Delta)$-expander;
- $|C_i| \leq \varepsilon^2 \frac{r}{32}$,

where $C_i \coloneqq \bigcup_{j=1}^{i} B_j$. For proof see Proposition 29 in appendix A.

We have to show that on each iteration we can find some x-assignment ν_i that satisfy the requirements. Since $|C_i| \leq \varepsilon^2 \frac{r}{32}$ that imply, in particular, that $|B_i| \leq \varepsilon^2 \frac{r}{32}$.

Fix some iteration i. Since G_i is an $(r, \Delta, (1-2\varepsilon)\Delta)$-expander then by Lemma 28 graph G'_{i+1} is an $(r, \Delta, (1-3\varepsilon)\Delta)$-expander. Hence by Proposition 5 there is an enumeration $v^1, v^2, \ldots, v^{|B_i|} \in B_i$ and a sequence $R_1, \ldots, R_k \subseteq \mathrm{N}_{G'_{i+1}}(S)$ such that for all $e \in [|B_i|]$:

- $R_e = \mathrm{N}_{G'_{i+1}}(v^e) \setminus \left(\bigcup_{j=1}^{e-1} \mathrm{N}_{G'_{i+1}}(v^j) \right)$;
- $|R_e| \geq (1-3\varepsilon)\Delta$.

We define the x-assignment ν_i step by step starting from v^1. Consider an auxiliary x-assignment $\kappa \coloneqq \rho_i \cup \sigma_i$. Since f_{v^1} is a $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced function and κ assigns at most $|\mathrm{N}_G(v^1)| - |\mathrm{N}_{G'_{i+1}}(v^1)| < 3\varepsilon\Delta$ its variables then $f_{v^1}|_\kappa$ is not a constant and we define an x-assignment $\nu_i^{v^1}$ to R_1 variables to satisfy the constraint $f_{v^1}(x) = 0$. We continue this process for vertices v^j and $\kappa \coloneqq \rho_i \cup \sigma_i \cup \bigcup_{b=1}^{j-1} \nu_i^{v^b}$. The x-assignment $\nu_i \coloneqq \bigcup_{b=1}^{|B_i|} \nu_i^{v^b}$ satisfy all constraints from the set B_i as desired.

At this moment we proved that we can realise all steps of our algorithm. The x-assignment ρ_ℓ assigns only variables from $\mathrm{N}(O_\ell \cup C_\ell)$, hence $L_\rho \coloneqq O_\ell \cup C_\ell$. The first property of self-reduction is satisfied: $|L_\rho| \leq \ell + \varepsilon^2 \frac{r}{32} \leq \varepsilon^2 \frac{r}{16}$. The second follow from the construction. The third property was proved above. Moreover all intermediate x-assignments ρ_i are also satisfy these properties. ◀

▶ **Theorem 23.** *Let $G \coloneqq (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander graph such that $|L| = m$, $|R| = n$ and $\frac{10}{\varepsilon} \leq r$. Fix $\gamma \coloneqq 2^{-\varepsilon\Delta}$. Let $f \coloneqq \{f_1, \ldots, f_m\}$ be a collection of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions, and $\pi \coloneqq D_1, \ldots, D_s$ be a resolution proof of $\varphi \coloneqq \mathsf{PRG}_{G,f}$.*

If $s < \exp\left(\frac{\varepsilon^3 r}{32} \cdot \frac{1}{3}\gamma^6 \frac{w}{m}\right)$ for some $w \in \mathbb{N}$ then there is a self-reduction ρ such that $\mathrm{hw}^{\gamma^3}_{\varphi|_{\rho^y}}$ of $\pi|_{\rho^y}$ is at most w.

Proof. Let $\theta \coloneqq F_1, F_2, \ldots, F_s$ where F_i is a functional form of D_i.

We show that under the current assumptions Algorithm 1 whp give such an x-assignment. Let $\ell \coloneqq \varepsilon^3 \frac{r}{32}$ be the number of iterations of our algorithm. By Lemma 22 it generates self-reduction ρ. We have to check that whp $\pi|_\rho$ is a proof of small heavy width. We do it for each line in the proof separately and moreover for all $i \in [s]$ we show that $(F_i)|_\rho$ has small heavy width.

One of the important difference of heavy width and a classical width that after application of a partial assignment it may increase since we also apply an assignment to the functions f_i and change our formula. To avoid this problem we analyse $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi}$ rather than $\mathrm{hw}^{\gamma^3}_{\varphi}$ of the clauses and show that for any $F \in \theta$:

- $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})$ is small for any $i \leq \ell$ than it cannot "grow" to much in the end (in terms of $\mathrm{hw}^{\gamma^3}_{\varphi}$);
- if $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})$ is big enough for some $i \leq \ell$ it will be killed with good enough probability on $i+1$-th iteration.

We start with the first part of the proof. Let $F \in \theta$ be a functional form of a clause from π and F is defined by the functions $h_1, h_2, \ldots, h_m$. Fix some $i \leq \ell$ and pick some alive output bit $v \in L \setminus L_\rho$. We remind a notation $p_v^i \coloneqq f_v|_{\rho_i}$. Output bit v is alive and graph G_ℓ is an $(r, \Delta, (1-3\varepsilon)\Delta)$-expander, hence x-assignments ρ_i and ρ_ℓ can assign at most $3\varepsilon\Delta$ variables from $\mathrm{N}(v)$. Thus for all $i \leq \ell$:

$$\begin{aligned}
\Pr_{z\leftarrow(p_v^\ell)^{-1}(0)}[h_v(z)=1] &\le \frac{|h_v^{-1}(1)\cap(p_v^\ell)^{-1}(0)|}{|(p_v^\ell)^{-1}(0)|}\\
&\le \frac{|h_v^{-1}(1)\cap(p_v^i)^{-1}(0)|}{|(p_v^\ell)^{-1}(0)|} && (\rho_i\subseteq\rho_\ell)\\
&= \frac{|h_v^{-1}(1)\cap(p_v^i)^{-1}(0)|}{|(p_v^i)^{-1}(0)|}\cdot\frac{|(p_v^i)^{-1}(0)|}{|(p_v^\ell)^{-1}(0)|}\\
&\le \frac{|h_v^{-1}(1)\cap(p_v^i)^{-1}(0)|}{|(p_v^i)^{-1}(0)|}\cdot\frac{|f_v^{-1}(0)|}{|(p_v^\ell)^{-1}(0)|} && (p_v^i=f_v|_{\rho_i})\\
&\le \frac{|h_v^{-1}(1)\cap(p_v^i)^{-1}(0)|}{|(p_v^i)^{-1}(0)|}\cdot\frac{\frac{3}{4}2^{\Delta}}{\frac{1}{4}2^{\Delta-3\varepsilon\Delta}} && (f \text{ is balanced})\\
&\le \Pr_{z\leftarrow(p_v^i)^{-1}(0)}[h_v(z)=1]\cdot 3\cdot 2^{3\varepsilon\Delta}.
\end{aligned}$$

Hence for all $F\in\theta$ and all $v\in L$ if v is γ^3-heavy in $F|_{\rho_\ell}$ wrt $\varphi|_{\rho^y}$ then v is $\frac{\gamma^6}{3}$-heavy in $F|_{\rho_i}$ wrt $\varphi|_{\rho_i^y}$ for all $i\le\ell$. And $\mathrm{hw}^{\gamma^3}_{\varphi|_{\rho_i^y}}(F|_{\rho_\ell})\ge w$ imply that $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})\ge w$ for all $i\in\ell$.

Consider a clause D in $\pi|_{\rho_{i-1}^y}$ and its functional form $F\in\theta$. Denote by H the event that v^i is $\frac{\gamma^6}{3}$-heavy output bit in F wrt $\mathsf{PRG}_{G,f}|_{\rho_{i-1}}$. Clause D is killed by ρ_i^y with probability at least:

$$\begin{aligned}
\Pr_{v^i,\sigma_i}[D|_{\sigma_i}=1] &\ge \Pr_{v_i,\sigma_i}[h_{v^i}(x)|_{\sigma_i^y}=1]\\
&\ge \Pr_{v^i,\sigma_i}[H]\cdot\Pr_{v^i,\sigma_i}[(h_{v^i}|_{\sigma_i^y}=1)\mid H]\\
&\ge \frac{|\{v^i\in[m]\mid H\}|}{m}\cdot\frac{\gamma^6}{3}.
\end{aligned}$$

Hence for the clause $D\in\pi$ there are two ways.

- At some moment $i\le\ell$ the $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})\le w$. In this case D is not interesting for us anymore, since as we proved above $\mathrm{hw}^{\gamma^3}_{\varphi|_{\rho^y}}(F|_{\rho})\le w$.
- If $\mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})\ge w$ then the probability that the clause $D|_{\rho_i}$ is survived on $i+1$-th iteration is at most:

$$\Pr[D|_{\rho_i}\text{ is survived on }i+1\text{-th iteration}]\le 1-\frac{w}{m}\frac{\gamma^6}{3}.$$

And hence:

$$\begin{aligned}
&\Pr[D\text{ is survived after }\ell\text{ iterations}]\le\\
&\prod_i\Pr[D|_{\rho_i}\text{ is survived on }i+1\text{-th iteration}]\le\\
&\qquad\left(1-\frac{w}{m}\frac{\gamma^6}{3}\right)^{\ell}. && \text{since } \mathrm{hw}^{\frac{\gamma^6}{3}}_{\varphi|_{\rho_i^y}}(F|_{\rho_i})\ge w
\end{aligned}$$

To conclude the proof note that

$$\Pr[\mathrm{hw}^{\gamma^3}_{\varphi|_{\rho_\ell^y}}(F|_{\rho_\ell})>w]\le\left(1-\frac{w}{m}\frac{\gamma^6}{3}\right)^{\ell}=\left(1-\frac{w}{m}\frac{\gamma^6}{3}\right)^{\varepsilon^3\frac{r}{32}}<\left(1-\frac{w}{m}\frac{\gamma^6}{3}\right)^{\frac{1}{\frac{w}{m}\frac{\gamma^6}{3}}\log s}<\frac{1}{s}.$$

By the union bound over all $F \in \theta$ we conclude that:

$$\Pr[\exists F \in \theta, \mathrm{hw}^{\gamma^3}_{\varphi|_{\rho^y_\ell}}(F|_{\rho_\ell}) > w] < 1.$$

Or in other words there is an x-assignment ρ that satisfy all required properties. ◀

4.3 Heavy Width Lower Bound

For the sake of contradiction assume that we have a proof $\pi \coloneqq (D_1, \ldots, D_s)$ of small heavy width. Starting from D_s we trace the path p in the dag of π to the initial clause. During this process we maintain a partial x-assignment σ such that in the clause $D \in p$ for any small set S of initial clauses the x-assignment σ can be extended for an x-assignment $\kappa \supseteq \sigma$ such that S is satisfied by κ, but D does not. That give us a contradiction in a leaf where D should be one of the initial clauses.

This assignment σ will assign neighbours of heavy output bits of the generator and some extension (closure) to make sure that the remaining graph (after removing assigned variables) is an expander. Since the remainder is an expander (in particular we assign not so many neighbours of any alive output bit) then the existence of the assignment κ will follow from the fact other output bit are not heavy that means that there are a lot of points that satisfy the constraint but not satisfy the clause.

In this section we assume that $G \coloneqq (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander and $\mathsf{PRG}_{G,f}$ is based on this graph an some functions f_i (again we assume that zero point not in $\mathrm{Im}(\mathcal{F}_{G,f})$ and state our result for this point, but it holds for all $b \notin \mathrm{Im}(\mathcal{F}_{G,f})$). All clauses deal with variables of $\mathsf{PRG}_{G,f}$. We also fix an abbreviation $\gamma \coloneqq 2^{-\varepsilon\Delta}$.

Let start with auxiliary objects and lemmas.

▶ **Definition 24.** *Let ρ be a self-reduction of $\mathsf{PRG}_{G,f}$. Let C be a clause and F its functional form, I_η be a set of η-heavy output bits wrt $\mathsf{PRG}_{G,f}|_{\rho^y}$. We say that an output bit v is (η,ν)-**dangerous** for F iff $v \in \mathrm{Cl}^{r,\nu}(\mathrm{N}_G(I_\eta \cup L_\rho))$. Denote this set by $\mathcal{D}^{\eta,\nu}_{F,\rho}$.*

Note that this definition make sense only if graph G is an expander. Also note that $I_\eta \cup L_\rho \subseteq \mathrm{Cl}^{r,\nu}(\mathrm{N}_G(I_\eta \cup L_\rho))$. For fixed parameters η and ν and a clause C with functional form F we also say that an x-assignment $\sigma \supseteq \rho$ is (η,ν)**-locally consistent** iff:

- $\sigma^{-1}(\{0,1\}) = \mathrm{N}\left(\mathcal{D}^{\eta,\nu}_{F,\rho}\right)$;
- $C|_{\sigma^y} \not\equiv 1$;
- σ satisfy all constraints that correspond to $\mathcal{D}^{\eta,\nu}_{F,\rho}$.

The following Lemma is the heart of the proof. It says that locally consistent assignments cannot *violate* any constraint from our formula.

▶ **Lemma 25.** *Let $f \coloneqq \{f_1, \ldots, f_m\}$ be a collection of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions, $\gamma < \frac{1}{8}$ and ρ be a self-reduction of $\mathsf{PRG}_{G,f}$. If C is a clause with functional form F such that $\mathrm{hw}^{\gamma^3}_{\mathsf{PRG}_{G,f}|_\rho}(F) \le \varepsilon\frac{r}{8}$ and σ is a $(\gamma^3, (1-2\varepsilon\Delta))$-locally consistent assignment then for any $J \subseteq L$ such that $|J| \le \varepsilon\frac{r}{4}$, there is an extension $\kappa \supseteq \sigma$ such that:*

- $\kappa^{-1}(\{0,1\}) \supseteq \mathrm{N}\left(\mathcal{D}^{\gamma^3,(1-2\varepsilon)\Delta}_{F,\rho}\right) \cup \mathrm{N}(J)$;
- $C|_{\kappa^y} \not\equiv 1$;
- $\forall v \in J, f_v(x)|_\kappa = 0$.

Proof. Let $\mathcal{D} \coloneqq \mathcal{D}^{\gamma^3,(1-2\varepsilon)\Delta}_{F,\rho}$ and $F \coloneqq \bigvee_i (h_i(x) = 1)$. Let $I \coloneqq \mathrm{Cl}^{r,(1-2\varepsilon)\Delta}(\mathrm{N}_G(J) \cup \mathrm{N}_G(\mathcal{D})) \setminus \mathcal{D}$. By Lemma 7 $|I| \le \frac{3}{8}r$. By the definition of closure $I \supseteq J \setminus \mathcal{D}$.

By Lemma 8 graph $G' \coloneqq G \setminus (\mathcal{D} \cup \mathrm{N}_G(\mathcal{D}))$ is an $\left(\frac{r}{2}, \Delta, (1-2\varepsilon)\Delta\right)$-expander. By Proposition 5 there is an enumeration $v^1, v^2, \ldots, v^{|I|} \in I$ and a sequence $R_1, \ldots, R_{|I|} \subseteq \mathrm{N}_{G'}(S)$ such that:

- $R_i = \mathrm{N}_{G'}\left(v^i\right) \setminus \left(\bigcup_{j=1}^{i-1} \mathrm{N}_{G'}\left(v^j\right)\right)$;
- $|R_i| \geq (1-2\varepsilon)\Delta$.

We define a family of x-assignments ν_i and $\kappa_i \coloneqq \bigcup_{j=1}^{i} \nu_i \cup \sigma$ step by step, starting from ν_1 in the following way:

- $\nu_i^{(-1)}(\{0,1\}) = R_i$;
- $f_{v^i}(x)|_{\kappa_i} = 0$;
- $C|_{\kappa_i^y} \not\equiv 1$.

We have to show the existence of such ν_i. Note that $|R_i| \geq (1-2\varepsilon)\Delta$ hence κ_{i-1} can assign at most $2\varepsilon\Delta$ variables in $\mathrm{N}_G\left(v^i\right)$. Since f_{v^i} is a balanced function:

$$|(f_{v^i}|_{\kappa_{i-1}})^{-1}(0)| \geq \frac{1}{4}\gamma^2 2^{\Delta} \geq \frac{1}{4}\gamma^2 |f_{v^i}^{-1}(0)|.$$

Output bit v^i is not γ^3-heavy hence there are at most $\gamma^3|f_{v^i}^{-1}(0)|$ different x-assignments to R_i that maps h_{v^i} to 1, assuming that $\gamma < \frac{1}{8}$ we can find an assignment that maps h_{v^i} to 0 and satisfy the constraint $f_{v^i}(x) = 0$. We define $\kappa \coloneqq \kappa_{|I|}$.

It remains to check that κ^y does not satisfy C, that does not immediately follow from the construction due to Remark 13. To show this fact we use an expansion of underlying graph. For the sake of contradiction assume that κ^y maps some literal $y_g^c \in C$ to 1 and this literal belongs to bag B_v. Consider three cases.

1. κ assigns all variables from $\mathrm{N}(v)$. In this case $h_v(x)|_\kappa$ is mapped to 1 since h_v is a disjunction of $(1 \oplus c \oplus g)$ with some function. that contradicts with the choice of κ.
2. κ assigns at most $2\varepsilon\Delta$ variables from $\mathrm{N}(v)$. Note that

$$\begin{aligned}
\Pr_{z \leftarrow f_v^{-1}(0)}[h_v(z) = 1] &\geq \Pr_{z \leftarrow f_v^{-1}(0)}[z \text{ cons. with } \kappa] && \kappa \text{ maps } y_g \text{ to } 1 \\
&\geq \Pr_{z \leftarrow \{0,1\}^\Delta}[z \text{ cons. with } \kappa \wedge f_v(z) = 0] \\
&\geq \Pr_{z \leftarrow \{0,1\}^\Delta}[z \text{ cons. with } \kappa] \cdot \Pr_{z \leftarrow \{0,1\}^\Delta}[f_v(z) = 0 \mid z \text{ cons. with } \kappa] \\
&\geq \gamma^2 \Pr_{z \leftarrow \{0,1\}^\Delta}[f_v(z) = 0 \mid z \text{ cons. with } \kappa] \\
&\geq \gamma^2 \Pr_{z \leftarrow \{0,1\}^\Delta}[f_v(z)|_\kappa = 0] \\
&\geq \frac{\gamma^2}{4} \geq \gamma^3. && f_v \text{ is balanced}
\end{aligned}$$

   But in this case $v \in \mathcal{D}$ by definition of $\mathcal{D}$. Hence κ should assign all variable in $\mathrm{N}(v)$.
3. κ assigns at least $2\varepsilon\Delta + 1$ variables from $\mathrm{N}(v)$ but not all of them. That contradicts with the fact that κ assigns variables from $\mathrm{N}(I \cap \mathcal{D}) = \mathrm{N}\left(\mathrm{Cl}^{r,(1-2\varepsilon)\Delta}(\mathrm{N}_G(J) \cup \mathrm{N}_G(\mathcal{D}))\right)$ and Lemma 8. ◀

▶ **Theorem 26.** *Let $\varepsilon < \frac{1}{3}$, $G \coloneqq (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-expander graph such that $|L| = m$, $|R| = n$. Fix $\gamma \coloneqq 2^{-\varepsilon\Delta} < \frac{1}{8}$. If $f \coloneqq \{f_1, \ldots, f_m\}$ is a collection of $\left(\frac{1}{4}, 3\varepsilon\Delta\right)$-balanced functions then $\mathrm{hw}^{\gamma^3}_{\mathsf{PRG}_{G,f}|_{\rho^y}}$ of any resolution proof of $\mathsf{PRG}_{G,f}|_{\rho^y}$ is at least $\varepsilon^2\frac{r}{16}$ where ρ is self-reduction.*

Proof. For the sake of contradiction assume that $\pi \coloneqq (D_1, \ldots, D_s)$ is a resolution proof of $\mathsf{PRG}_{G,f}|_{\rho^y}$ of $\mathrm{hw}^{\gamma^3}_{\mathsf{PRG}_{G,f}|_{\rho^y}}$ at most $\varepsilon^2\frac{r}{16}$. Let $\theta \coloneqq (F_1, F_2, \ldots, F_s)$ be a sequence of functional forms that is witnessing heavy width of π. For a disjunction $F_i \in \theta$ we denote $\mathcal{D}_i \coloneqq \mathcal{D}^{\gamma^3,(1-2\varepsilon)\Delta}_{F_i,\rho}$.

For the clause D_s with functional form F_s an x-assignment ρ is locally consistent, since graph $(G \setminus (L_\rho \cup \mathrm{N}(L_\rho))$ is an $(r, \Delta, (1-2\varepsilon)\Delta)$-expander. We want to show that the existence of a locally consistent assignment κ^D for some clause $D \in \pi$ with functional form $F \in \theta$ imply the existence of a locally consistent assignment for at least one of its predecessors. In this case the can trace the path from D_s to some initial clause $D_k \in \pi \cap \mathsf{PRG}_{G,f}|_{\rho^y}$ with functional form F_k and show the existence of a locally consistent assignment κ^{D_k} for this clause.

Suppose that $D_k \coloneqq (y^{c_1}_{g_1} \vee y^{c_2}_{g_2} \vee y^{c_3}_{g_3} \cdots \vee y^{c_\ell}_{g_\ell})$. By construction of $\mathsf{PRG}_{G,f}$ we can find an $i \in [m]$ such that for all $j \in [\ell]$:

- g_j depends only on Vars_i;
- the equality $g_j(x) = c_j$ semantically follows from the equality $f_i(x) = 0$.

But in this case i is 1-heavy for D_k and for any x-assignment σ the condition $f_i(x)|_\sigma = 0$ imply that $D_k|_{\rho^y} \equiv 1$. This fact contradicts with the definition of locally consistent assignment.

Suppose a locally consistent assignment κ exists for a clause $D_i \in \pi$ with functional form $F_i \in \theta$ and D_a, D_b are predecessors of D_i in π with functional forms F_a, F_b respectively. Note, that $\mathrm{hw}^{\gamma^3}_\varphi$ of these functional forms are at most $\varepsilon^2\frac{r}{16}$, hence Lemma 7 together with the upper bound $|L_\rho| \leq \frac{\varepsilon^2 r}{16}$ imply that the sizes of $\mathcal{D}_i, \mathcal{D}_a, \mathcal{D}_b$ are at most $\varepsilon\frac{r}{16} + \varepsilon\frac{r}{16} = \varepsilon\frac{r}{8}$. By Lemma 25 we have an extension $\sigma \supseteq \kappa$ on $\mathrm{N}_G(\mathcal{D}_a \cup \mathcal{D}_b)$ that satisfy constraints of $\mathsf{PRG}_{G,f}$ that correspond to $\mathcal{D}_a \cup \mathcal{D}_b$ but do not satisfy D_i. And since σ do not satisfy D_i it also do not satisfy at least one of its predecessor, wlog it is D_a. And the x-assignment $\sigma \cap \mathrm{N}_G(\mathcal{D}_a)$ is a locally consistent for D_a and F_a as desired. ◀

4.4 Proof of Theorem 14

For the sake of contradiction assume that $\pi \coloneqq D_1, D_2, \ldots, D_s$ is a resolution proof of $\mathsf{PRG}_{G,f}$ and $s \leq \exp\left[\delta\frac{\varepsilon^5 r^2}{2^{6\varepsilon\Delta}m}\right]$ for some $\delta \leq 10^{-4}$.

Fix $w \coloneqq \frac{\varepsilon^2 r}{20}$ and $\gamma \coloneqq 2^{-\varepsilon\Delta}$. Note that:

$$\exp\left(\frac{\varepsilon^3 r}{32}\cdot\frac{1}{3}\gamma^6\frac{w}{m}\right) = \exp\left(\frac{\varepsilon^3 r}{32}\cdot\frac{1}{3}\gamma^6\frac{\varepsilon^2 r}{20\cdot m}\right) \geq \exp\left(\frac{\varepsilon^5}{2000}\cdot\gamma^6\frac{r^2}{m}\right) > \exp\left(\delta\varepsilon^5\cdot\gamma^6\frac{r^2}{m}\right) \geq s,$$

hence we can apply Theorem 23 that gives a self-reduction ρ. We hit the proof π by ρ^y and the proof $\pi|_{\rho^y}$ is a proof of $\mathsf{PRG}_{G,f}|_{\rho^y}$. Moreover the $\mathrm{hw}^{\gamma^3}_{\mathsf{PRG}_{G,f}|_{\rho^y}}$ of $\pi|_{\rho^y}$ is at most w.

Since ρ is a self-reduction then by Theorem 26 any proof of $\mathsf{PRG}_{G,f}|_{\rho^y}$ requires $\mathrm{hw}^{\gamma^3}_{\mathsf{PRG}_{G,f}|_{\rho^y}}$ at least $\varepsilon^2\frac{r}{16} > w$. Contradiction.

5 Comments and Further Directions

The most important is the lower bounds on the Nisan–Wigderson generator with $m \gg n^2$. The technical barrier for doing it is the scaling factor $\frac{1}{m}$ that comes from the step 7 of the algorithm 1. And it is a fundamental problem of the general restriction technique that we use in proof complexity. The most promising approach for avoiding this problem is the "pseudowidth" that was created by Razborov in [20, 21] and equipped with a closure trick in [9].

The pseudowidth technique may be viewed as a replacement of the "self-reductions" and algorithm from Section 4.2. Instead of hitting the proof by a restriction we look at the small enough proof and try to add a carefully chosen set of axioms to our formula that allows to transform this formula into a proof of small "pseudowidth". The pseudowidth measure itself may be considered as an α-heavy width where parameter α can be different for different output bits. Unfortunately, to apply this strategy we have to deal with large enough parameters α, but all results from Section 4.3 used the fact that α is small enough. That leads to another technical, but the important open problem: can one prove that any resolution proof of $\mathsf{PRG}_{G,f}$ has $\frac{1}{100}$-heavy width at least $\Omega(n^{\delta})$?

We may also ask to generalize the lower bounds to stronger proof systems. It seems adaptation of this technique for Polynomial Calculus (or Sherali–Adams) may be a challenging problem if we want to go beyond the logarithmic threshold, i.e. $\Delta \gg \log n$.

The current result also does not seem tight. We believe that if the dependency graph of NW generator is an $(r, \Delta, 0.99 \cdot \Delta)$-boundary expander then the right bound should depend only on parameters m and r (usually $r \approx \frac{n}{\mathsf{poly}(\Delta)}$). As an explicit open problem we ask to show the following lower bound: $\exp\left[\frac{r^2}{m}\right]$.

References

1. Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002. doi:10.1137/S0097539700366735.
2. Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. doi:10.1137/S0097539701389944.
3. 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*.
4. 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.
5. Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present and future. *Bull. EATCS*, 65:66–89, 1998.
6. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.
7. Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 174–183, 1996. doi:10.1145/237814.237860.
8. Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979. doi:10.2307/2273702.
9. Susanna F. de Rezende, Jakob Nordström, Kilian Risse, and Dmitry Sokolov. Exponential resolution lower bounds for weak pigeonhole principle and perfect matching formulas over sparse graphs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 28:1–28:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.28.
10. Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. Optimal sherali-adams gaps from pairwise independence. In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 125–139, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

11 Russell Impagliazzo, Pavel Pudlák, and Jirí Sgall. Lower bounds for the polynomial calculus and the gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999. doi:10.1007/s000370050024.

12 Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 1995.

13 Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170:123–140, January 2001. doi:10.4064/fm170-1-8.

14 Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *J. Symb. Log.*, 69(1):265–286, 2004. doi:10.2178/jsl/1080938841.

15 Jan Krajíček. Proof complexity generators. In Hajo Broersma, Stefan S. Dantchev, Matthew Johnson, and Stefan Szeider, editors, *Algorithms and Complexity in Durham 2006 - Proceedings of the Second ACiD Workshop, 18-20 September 2006, Durham, UK*, volume 7 of *Texts in Algorithmics*, page 3. King's College, London, 2006.

16 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.

17 Pavel Pudlák. Proofs as games. *Am. Math. Mon.*, 107(6):541–550, 2000. URL: http://www.jstor.org/stable/2589349.

18 Alexander A. Razborov. Bounded arithmetic and lower bounds in boolean complexity. In Peter Clote and Jeffrey B. Remmel, editors, *Feasible Mathematics II*, pages 344–386, Boston, MA, 1995. Birkhäuser Boston.

19 Alexander A. Razborov. Lower bounds for propositional proofs and independence results in bounded arithmetic. In Friedhelm Meyer auf der Heide and Burkhard Monien, editors, *Automata, Languages and Programming, 23rd International Colloquium, ICALP96, Paderborn, Germany, 8-12 July 1996, Proceedings*, volume 1099 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 1996. doi:10.1007/3-540-61440-0_116.

20 Alexander A. Razborov. Improved resolution lower bounds for the weak pigeonhole principle. *Electron. Colloquium Comput. Complex.*, 8(55), 2001. URL: http://eccc.hpi-web.de/eccc-reports/2001/TR01-055/index.html.

21 Alexander A. Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theor. Comput. Sci.*, 303(1):233–243, 2003. doi:10.1016/S0304-3975(02)00453-X.

22 Alexander A. Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. *Ann. of Math.*, 181:415–472, 2015. doi:10.4007/annals.2015.181.2.1.

23 Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-dnf resolution. *SIAM J. Comput.*, 33(5):1171–1200, 2004. doi:10.1137/S0097539703428555.

24 Dmitry Sokolov. (Semi)Algebraic proofs over ±1 variables. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 78–90. ACM, 2020. doi:10.1145/3357713.3384288.

25 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.45.

A Missed Lemmas

At first we show a simple auxiliary statement.

▶ **Lemma 27.** *Suppose that* $G \coloneqq (L, R, E)$ *is an* (r, Δ, c)*-boundary expander and that* $J \subseteq R$ *has size* $|J| \leq \Delta r$*. Then if* $X \subseteq L$ *has size* $|X| \leq r$ *and* $|\partial(X) \setminus J| \leq \nu|X|$ *then* $X \leq \frac{|J|}{c-\nu}$.

Proof. The expansion property of the graph guarantees that $c|X| - |J| \leq |\partial(X) \setminus J|$. The conclusion follows. ◀

▶ **Lemma 28.** *Let $G \coloneqq (L, R, E)$ be an $(r, \Delta, (1-\varepsilon)\Delta)$-boundary expander. Then $G \setminus (\{v\} \cup \{N(v)\})$ is an $\left(r-1, \Delta, \left(1-\frac{3}{2}\varepsilon\right)\Delta\right)$ expander where $v \in L$ is an arbitrary vertex.*

Proof. Fix some $v \in L$ and denote $G' \coloneqq G \setminus (\{v\} \cup \{N(v)\})$.

Consider some set of $S \subseteq (L \setminus \{v\})$ of size at most $r-1$ and denote $H \coloneqq N(S) \cap N(v)$. Since G is an expander:

$$|\partial_G(S \cup v)| = |\partial_{G'}(S)| + \Delta - |H| \geq (1-\varepsilon)\Delta(|S|+1)$$

$$|\partial_{G'}(S)| \geq (1-\varepsilon)\Delta|S| - \varepsilon\Delta + |H|.$$

But from the other point of view:

$$|\partial_{G'}(S)| \geq |\partial_G(S)| - |H| \geq (1-\varepsilon)\Delta|S| - |H|.$$

Altogether:

$$|\partial_{G'}(S)| \geq (1-\varepsilon)\Delta|S| - \min(|H|, \varepsilon\Delta - |H|) \geq (1-\varepsilon)\Delta|S| - \frac{\varepsilon}{2}\Delta \geq \left(1-\frac{3}{2}\varepsilon\right)\Delta|S|. \quad \blacktriangleleft$$

▶ **Proposition 29** (Analog of [24]). *For all $i \leq \ell$:*

- *G_i is an $(r, \Delta, (1-2\varepsilon)\Delta)$-expander;*
- *$|C_i| \leq |\varepsilon^2 \frac{r}{32}|$.*

Proof. At first we prove the second claim $|C_i| \leq \varepsilon^2 \frac{r}{32}$ by induction. C_0 is an empty set. Suppose that $|C_{i-1}| \leq \varepsilon^2 \frac{r}{32}$. There are two steps in the proof:

- we show that $|B_i| \leq \frac{r}{3}$ that give us an opportunity to use expansion property for the set C_i;
- we give a lower bound on size $\partial_G(C_i)$ by using expansion property and the upper bound by the choice of B_i that together give us an upper bound on size of C_i.

Let start with the first step. $|\partial_G(B_i) \setminus \mathrm{N}_G(C_{i-1} \cup O_{i+1})| \leq |\partial_{G'_{i+1}}(B_i)| \leq (1-2\varepsilon)|B_i|$. By definition $|B_i| \leq r$ and hence by Lemma 27 $|B_i| \leq \frac{|\mathrm{N}_G(C_{i-1} \cup O_{i+1})|}{\varepsilon\Delta} \leq \frac{r}{4} + \frac{\varepsilon}{32}r \leq \frac{r}{3}$. That concludes the first step.

$$\begin{aligned}
(1-\varepsilon)\Delta|C_i| &\leq \\
|\partial_G(C_i)| &\leq && \text{by expansion} \\
\left|\bigcup_{j=1}^{i}(\partial_G(B_j) \setminus \mathrm{N}_G(C_{j-1}))\right| &\leq \\
\left|\bigcup_{j=1}^{i}(\partial_G(B_j) \setminus (\mathrm{N}_G(C_{j-1}) \cup \mathrm{N}(O_{j+1}))) \cup \mathrm{N}(O_{j+1})\right| &\leq \\
\left|\bigcup_{j=1}^{i}\partial_{G'_{j+1}}(B_j) \cup \mathrm{N}(O_{i+1})\right| &\leq && \text{by the choice of } B_j \\
(1-2\varepsilon)\Delta\sum_{j=1}^{i}|B_j| + |\mathrm{N}(O_{i+1})| &\leq \\
(1-2\varepsilon)\Delta|C_i| + |\mathrm{N}(O_{i+1})|.
\end{aligned}$$

And hence $|C_i| \leq \frac{|\mathrm{N}(O_{i+1})|}{\varepsilon\Delta} \leq \varepsilon^2 \frac{r}{32}$ as desired.

The first claim we prove by contradiction. Pick the minimal i such that $G \coloneqq G_i$ is not an $(r, \Delta, (1-2\varepsilon))$-boundary expander and $S \subseteq L$ be a witness of it, i.e. $|S| \leq r$ and $|\partial_G(S)| \leq (1-2\varepsilon)|S|$. As in previous case $|\partial_G(S) \setminus (\mathrm{N}_G(C_{i-1}) \cup O_\ell)| \leq |\partial_G(S)| \leq (1-2\varepsilon)|S|$ hence by Lemma 27 $|S| \leq \frac{|\mathrm{N}_G(C_{i-1}) \cup O_\ell|}{\varepsilon\Delta} \leq \frac{r}{2}$.

Consider a set $S \cup B_{i-1}$ and note that size of it at most r. $\partial_{G'_i}(S \cup B_{i-1}) \subseteq \partial_{G_i}(S) \cup \partial_{G'_i}(B_{i-1})$ by definition of G_i. This implies $|\partial_{G'_i}(S \cup B_{i-1})| \leq (1-2\varepsilon)\Delta|S| + (1-2\varepsilon)\Delta|B_{i-1}| = (1-2\varepsilon)\Delta|S \cup B_{i-1}|$. That contradicts with the choice of B_{i-1}. ◀

▶ **Lemma 30.** *There is a constant $n_0 \in \mathbb{N}$ such that for any $n > n_0$ if a function $f\colon \{0,1\}^n \to \{0,1\}$ is chosen uniformly at random then whp it is $\left(\frac{1}{4}, n - \sqrt{n}\right)$-balanced.*

Proof. There are at most

$$\sum_{i=0}^{n} \binom{n}{i} \cdot 2^i = 3^n$$

different partial assignments.

A fixed partial assignment ρ of size k corresponds to a boolean subcube $S \subseteq \{0,1\}$ of size 2^{n-k} for which we want to estimate number of ones and zeroes. Note that:

$$\Pr_f\left[|(f|_\rho)^{-1}(1)| \leq \frac{1}{4}2^{n-k}\right] \leq \sum_{i=0}^{2^{n-k}/4} \binom{2^{n-k}}{i} \cdot 2^{-2^{n-k}} \leq 2^{-(1-\mathrm{H}(1/4))2^{n-k}} \leq 2^{-0.1\cdot 2^{n-k}}.$$

Altogether:

$$\Pr_f\left[f \text{ is not } \left(\frac{1}{4}, n-\sqrt{n}\right)\text{-balanced}\right] \leq$$
$$\sum_{\rho, |\rho| \leq n-\sqrt{n}} \left(\Pr_f\left[|(f|_\rho)^{-1}(1)| \leq \frac{1}{4}2^{n-|\rho|}\right] + \Pr_f\left[|(f|_\rho)^{-1}(0)| \leq \frac{1}{4}2^{n-|\rho|}\right]\right) \leq$$
$$2 \cdot 3^n \cdot 2^{-0.1\cdot 2^{\sqrt{n}}} \leq 2^{-2^{\Omega(\sqrt{n})}}.$$

◀

Probabilistic Kolmogorov Complexity with Applications to Average-Case Complexity

Halley Goldberg
Simon Fraser University, Burnaby, Canada

Valentine Kabanets
Simon Fraser University, Burnaby, Canada

Zhenjian Lu
University of Warwick, Coventry, UK

Igor C. Oliveira
University of Warwick, Coventry, UK

Abstract

Understanding the relationship between the worst-case and average-case complexities of NP and of other subclasses of PH is a long-standing problem in complexity theory. Over the last few years, much progress has been achieved in this front through the investigation of *meta-complexity*: the complexity of problems that refer to the complexity of the input string x (e.g., given a string x, estimate its time-bounded Kolmogorov complexity). In particular, [11] employed techniques from meta-complexity to show that if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ then $\mathsf{UP} \subseteq \mathsf{DTIME}[2^{O(n/\log n)}]$. While this and related results [13, 6] offer exciting progress after a long gap, they do not survive in the setting of *randomized* computations: roughly speaking, "randomness" is the opposite of "structure", and upper bounding the amount of structure (time-bounded Kolmogorov complexity) of different objects is crucial in recent applications of meta-complexity. This limitation is significant, since randomized computations are ubiquitous in algorithm design and give rise to a more robust theory of average-case complexity [18].

In this work, we develop a *probabilistic* theory of meta-complexity, by incorporating randomness into the notion of complexity of a string x. This is achieved through a new probabilistic variant of time-bounded Kolmogorov complexity that we call pK^t complexity. Informally, $\mathsf{pK}^t(x)$ measures the complexity of x when shared randomness is available to all parties involved in a computation. By porting key results from meta-complexity to the probabilistic domain of pK^t complexity and its variants, we are able to establish new connections between worst-case and average-case complexity in the important setting of *probabilistic* computations:

- If $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then $\mathsf{UP} \subseteq \mathsf{RTIME}\left[2^{O(n/\log n)}\right]$.
- If $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, then $\mathsf{AM} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.
- In the fine-grained setting [6], we get $\mathsf{UTIME}[2^{O(\sqrt{n\log n})}] \subseteq \mathsf{RTIME}[2^{O(\sqrt{n\log n})}]$ and $\mathsf{AMTIME}[2^{O(\sqrt{n\log n})}] \subseteq \mathsf{BPTIME}[2^{O(\sqrt{n\log n})}]$ from stronger average-case assumptions.
- If $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$, then $\mathsf{PH} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$. Specifically, for any $\ell \geq 0$, if $\mathsf{Dist}\Sigma_{\ell+2}^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$ then $\Sigma_\ell^{\mathsf{P}} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.
- Strengthening a result from [13], we show that if $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ then polynomial size Boolean circuits can be agnostically PAC learned under any unknown $\mathsf{P/poly}$-samplable distribution in polynomial time.

In some cases, our framework allows us to significantly simplify existing proofs, or to extend results to the more challenging probabilistic setting with little to no extra effort.

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

Keywords and phrases average-case complexity, Kolmogorov complexity, meta-complexity, worst-case to average-case reductions, learning

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.16

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 16; pp. 16:1–16:60

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

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/072

Funding This work received support from the Royal Society University Research Fellowship URF\R1\191059 and from the EPSRC New Horizons Grant EP/V048201/1. This research was also partially supported by NSERC Discovery and NSERC CGS M programs.

Acknowledgements We thank Russell Impagliazzo for a clarification regarding the definition of average-case complexity classes. We are also grateful to the CCC reviewers for their comments and suggestions.

1 Introduction

1.1 Context and background

Basing the average-case hardness of NP on its worst-case hardness is one of the central questions in computational complexity theory. In the terminology of Impagliazzo's five possible complexity worlds [16], this corresponds to excluding Heuristica, a world where $\mathsf{P} \neq \mathsf{NP}$ but NP problems can be efficiently solved on average with respect to every samplable distribution. Eliminating this possibility can be seen as a first step toward basing the security of cryptography on the assumption that $\mathsf{P} \neq \mathsf{NP}$.

Despite the long history of this problem (see, e.g., [25]), we still have limited understanding of the relationships between the worst-case and average-case complexities of problems in NP and in other subclasses of the polynomial hierarchy. In order to understand the difficulty of making progress, a number of works have investigated limitations of known proof techniques (see, e.g., [45, 4, 17]). These results suggest that fundamentally new ideas are needed to show that if NP problems are easy to solve on average, then NP is easy to solve in the worst case. For a detailed discussion on this matter, see [11, Section 1.2] and references therein.

To formally discuss the problem, we briefly review standard definitions from average-case complexity theory [3]. Recall that a pair (L, D) is a *distributional problem* if $L \subseteq \{0,1\}^*$ and $D = \{D_n\}_{n\geq 1}$ is an ensemble of probability distributions, where each D_n is supported over $\{0,1\}^*$. Let $D = \{D_n\}_{n\geq 1}$ be an ensemble of this form. We say that $D \in \mathsf{PSamp}$ if there is a randomized polynomial time algorithm A such that, for every $n \geq 1$, $A(1^n)$ is distributed according to D_n. For a complexity class $\mathfrak{C}$ (e.g., $\mathfrak{C} = \mathsf{NP}$), we let $\mathsf{Dist}\mathfrak{C}$ denote the set of distributional problems (L, D) with $L \in \mathfrak{C}$ and $D \in \mathsf{PSamp}$.

We say that a distributional problem (L, D) is solvable in *polynomial time on average* if there is a (deterministic) algorithm B such that, for every n and for every x in the support of D_n, $B(x; n) = L(x)$, and there is a constant $\varepsilon > 0$ such that $\mathbf{E}_{x\sim D_n}[t_{B,n}(x)^\varepsilon] \leq O(n)$, where $t_{B,n}(x)$ denotes the running time of B on input $(x; n)$.[1] If this is the case, we write $(L, D) \in \mathsf{AvgP}$.

In a recent breakthrough, [11] established new connections between worst-case and average-complexity theory for subclasses of the polynomial hierarchy. Among other results, [11] proved that (i) if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, then $\mathsf{UP} \subseteq \mathsf{DTIME}[2^{O(n/\log n)}]$;[2] and (ii) if $\mathsf{Dist}\Sigma_2^\mathsf{P} \subseteq \mathsf{AvgP}$, then $\mathsf{NP} \subseteq \mathsf{DTIME}[2^{O(n/\log n)}]$. While these results constitute significant progress after a long gap, they come with an important caveat: *the new connections do not hold in the setting*

[1] It is possible to show that this is equivalent to saying that there is a constant $c > 0$ such that the probability (over D_n) that the algorithm runs for more than T steps is at most $\mathsf{poly}(n)/T^c$. We refer to [3] for more information about this definition and its motivation.

[2] Recall that UP is the class of languages in NP whose positive instances admit unique witnesses.

of randomized computations. In other words, under the assumption that NP (or $\mathsf{Dist}\Sigma_2^{\mathsf{P}}$) is easy on average for *probabilistic* algorithms (i.e., $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$), we can no longer conclude *probabilistic* worst-case upper bounds.[3]

This is an important issue for at least two reasons. On the one hand, the theory of average-case complexity is more robust when defined with respect to randomized algorithms. For instance, [18] proved that the average-case hardness of NP with respect to the *uniform distribution* is equivalent to its hardness with respect to the class of *samplable distributions* (see Section 2.2). On the other hand, randomized algorithms and computations are ubiquitous in both theory and practice. In particular, randomness is crucial for cryptography, which is only possible if there exist problems that are hard on average for probabilistic polynomial time algorithms (see, e.g., [8]).

To explain why [11] and related works do not extend to randomized computations, we need to elaborate on their approach. These papers explore, in a crucial way, techniques from *meta-complexity*. Meta-complexity investigates the complexity of problems that refer to the complexity of the input string (e.g., given a string x, estimate its time-bounded Kolmogorov complexity). Such meta-computational problems have been at the core of other exciting recent results in complexity theory, such as a new characterization of the existence of one-way functions given in [28] and its subsequent developments (e.g., [40, 30]).

A central topic in meta-complexity is the study of *time-bounded Kolmogorov complexity*. Here we consider the minimum description length of a string x with respect to time-bounded machines. We informally review this notion, referring the reader to Section 2.5 for details. For a Turing machine $\mathcal{M}$, we let $|\mathcal{M}|$ denote its description length. Then, for a function $t\colon \mathbb{N} \to \mathbb{N}$ and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t(x) \;=\; \min_{\text{TM } \mathcal{M}} \{|\mathcal{M}| \mid \mathcal{M}(\varepsilon) \text{ outputs } x \text{ in } t(|x|) \text{ steps}\}\,.$$

where ε denotes the empty string. Conditional K^t complexity $\mathsf{K}^t(x \mid z)$ is defined similarly, where now the machine $\mathcal{M}$ receives z as input instead of the empty string.

Following the simplified presentation from [7], the approach of [11] consists of showing, under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgP}$, that for every $L \in \mathsf{NP}$ and every NP-verifier V for L, if $x \in L$ then some witness y_x of $x \in L$ satisfies $\mathsf{K}^t(y_x \mid x) = O(n/\log n)$, where $t(n) = 2^{O(n/\log n)}$. Consequently, in order to solve L in the worst case, it is enough to exhaustively search for a witness of complexity at most $O(n/\log n)$, which can be done deterministically in time $2^{O(n/\log n)}$. This strategy is inherently *non-black-box*, in the sense that the *code* of the average-case algorithm obtained from the initial assumption is used in a crucial way to upper bound $\mathsf{K}^t(y_x \mid x)$: it is part of the description of a machine $\mathcal{M}$ that outputs y_x.

Intuitively, if we start with the initial assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, i.e., with a *probabilistic* average-case algorithm, the high-level approach described above simply does not work: a typical random string employed in the computation has nearly maximum Kolmogorov complexity and does not allow us to bound K^t complexity.[4] More generally, as mentioned in the abstract, "randomness" is the opposite of "structure", and upper bounding the amount of structure (time-bounded Kolmogorov complexity) of different objects is crucial in recent applications of meta-complexity.

[3] We informally discuss AvgBPP in Section 1.2.2. For a formal treatment, see Section 2.2.

[4] For the reader familiar with the arguments from [11], we mention that while it is possible to construct a pseudorandom generator under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ via [5], this is not clear under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$. See Section 1.3 for a discussion.

To address this fundamental issue, we develop a suitable *probabilistic* theory of meta-complexity. In other words, we consider probabilistic notions of Kolmogorov complexity and their corresponding meta-computational problems. We are inspired in part by recent extensions of K^t complexity to the randomized setting, such as rKt complexity [38, 32] and its variant rK^t [33]. Unfortunately, as explained in Section 1.3, these notions turn out to be insufficient to study relations between worst-case and average-case complexities in the setting of probabilistic computations, and a more delicate approach is necessary. To achieve this, we introduce and systematically investigate a new notion of probabilistic time-bounded Kolmogorov complexity: pK^t complexity.

1.2 Results

As alluded to above, while previous works have employed various techniques to *remove* randomness from their arguments in order to analyze K^t complexity, we *incorporate* randomness in our framework. This has several advantages compared with previous works:

- We establish connections that hold in the more natural and robust setting of probabilistic computations. Previous results for deterministic computations, such as the aforementioned connections from [11], can be easily derived from our statements.
- In some contexts, we obtain significantly simpler proofs, as in the case of fine-grained connections between worst-case and average-case complexity [6]. In particular, our approach does not require the design of new pseudorandom generators.
- Our probabilistic framework can improve some existing results with little to no extra effort. As a concrete example, we show how to derive agnostic learning algorithms from a weaker assumption about the average-case easiness of NP, strengthening a result from [13].

Next, we explain our contributions in detail. In Section 1.2.1 below, we describe the new notion of time-bounded Kolmogorov complexity that is key to our theory of probabilistic meta-complexity. Then, in Section 1.2.2, we discuss its applications to average-case complexity.

1.2.1 A new notion of probabilistic Kolmogorov complexity

Fix a function $t\colon \mathbb{N} \to \mathbb{N}$. For a string $x \in \{0,1\}^*$, the *probabilistic* t-bounded Kolmogorov complexity of x is defined as

$$\mathsf{pK}^t(x) = \min\left\{ k \;\middle|\; \mathop{\mathbf{Pr}}_{w\sim\{0,1\}^{t(|x|)}}\left[\exists\, \mathcal{M} \in \{0,1\}^k,\ \mathcal{M}(w) \text{ outputs } x \text{ within } t(|x|) \text{ steps}\right] \geq \frac{2}{3} \right\}.$$

In other words, if $\mathsf{pK}^t(x) \leq k$, then for a typical random string w, there is a (deterministic) machine $\mathcal{M}$ of length k that runs in at most $t(|x|)$ steps and prints x when given w.

Note that pK^t is conceptually different from rKt [38] and rK^t [33], where there is a *fixed* randomized machine $\mathcal{M}$ that outputs x with probability $\geq 2/3$ over its internal randomness. The definition of pK^t is more subtle, and its benefits are less evident. Our main conceptual discovery is that pK^t is a surprisingly useful measure of time-bounded Kolmogorov complexity.

In order to gain more intuition about this definition, consider a 2-party communication setting where a player A that knows x would like to communicate this string to a computationally bounded player B. If A and B share a typical random string w, then A can simply send to B the description of a machine $\mathcal{M}$ as above, and B will be able to run $\mathcal{M}(w)$ to recover x in at most $t(|x|)$ steps. In other words, $\mathsf{pK}^t(x)$ can be seen as $\mathsf{K}^t(x)$ in a setting where a public random string is available to all parties involved in a computation.

We elaborate now on some properties of pK^t that make it robust and particularly attractive in meta-complexity and its applications:

- *Short descriptions from bounded* pK^t *complexity and the complexity of a random string.* It is not immediately clear from the definition of pK^t that, in the absence of a shared random string, bounded pK^t complexity yields short descriptions (as in the case of K^t and rK^t). However, if $\mathsf{pK}^t(x) \leq k$, notice that we can *sample* x as follows: randomly generate a string w of length $t(|x|)$, randomly generate a program $\mathcal{M}$ of length k, then output whatever $\mathcal{M}(w)$ outputs after running for $t(|x|)$ steps. It is easy to see that this sampler outputs x with probability at least $\geq 2/3 \cdot 2^{-k}$. By the coding theorem for Kolmogorov complexity, it follows that x has a description of length about k. This also implies that a *random* string x of length n has $\mathsf{pK}^t(x)$ close to n, as one would expect of any reasonable and useful notion of resource-bounded Kolmogorov complexity.
- *Connection to the worst-case complexity of* NP. As mentioned in Section 1.1, previous results were obtained by showing, under an average-case easiness assumption, that every positive instance x admits a witness y_x such that $\mathsf{K}^t(y_x \mid x) = O(n/\log n)$, where $t(n) = 2^{O(n/\log n)}$. An important observation explored in our results is that a bound of the form $\mathsf{pK}^t(y_x \mid x) \leq k$ is also sufficient to show worst-case upper bounds. Roughly speaking, with probability $\geq 2/3$ over the choice of a random string, we can "pretend" that $\mathsf{K}^t(y_x \mid x) \leq k$, which allows us to exhaustively search for a witness in non-trivial time.
- pK^t *complexity, pseudorandom generators, and reconstruction procedures.* In a typical construction of a pseudorandom generator G based on a string x of high complexity, the correctness of G against a class of adversaries is established using a reconstruction procedure. The latter extracts from any candidate distinguisher for G an upper bound on the complexity of x. Typical reconstruction procedures are randomized, and for this reason do not yield bounds on deterministic notions of time-bounded Kolmogorov. Moreover, it is often important to fool randomized algorithms in addition to deterministic ones. As one of our key lemmas, we show that pK^t is an excellent complexity measure under these circumstances, in the sense that pK^t bounds can be obtained in a natural way from randomized distinguishers and reconstruction procedures. [5]
- *Symmetry of information for* pK^t *and average-case complexity.* We show, under the assumption $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, that pK^t satisfies the symmetry of information condition, one of the pillars of Kolmogorov complexity (see, e.g., [23]). In other words, we prove under this hypothesis that for every pair of strings (x, y), $\mathsf{pK}^t(x, y) \geq \mathsf{pK}^{\mathsf{poly}(t)}(x) + \mathsf{pK}^{\mathsf{poly}(t)}(y \mid x) - O(\log t)$. Consequently, symmetry of information is available in the probabilistic setting when investigating connections between worst-case and average-case complexity.
- *Optimal source coding theorem for* pK^t. As noticed by [34], pK^t admits an unconditional coding theorem with optimal parameters, the first result of this form in the time-bounded setting (see Section 3.3). This implies that if we can efficiently sample an n-bit string x with probability $\geq \delta$, then $\mathsf{pK}^{\mathsf{poly}}(x) \leq \log(1/\delta) + O(\log n)$.
- *The relationship between* pK^t, rK^t, *and* K^t. Finally, under standard derandomization assumptions, for every string $x \in \{0,1\}^n$ and constructive time bound $t(n) \geq n$, $\mathsf{K}^{\mathsf{poly}(t)}(x) = \mathsf{pK}^{\mathsf{poly}(t)}(x) = \mathsf{rK}^{\mathsf{poly}(t)}(x)$, up to an additive $O(\log t)$ term (see Section A.2

[5] Trevisan and Vadhan [43] observed that such reconstruction procedures are often randomized algorithms that take nonuniform advice dependent on the randomness used by the algorithm (and introduced the notation //). Our pK^t measure is a Kolmogorov complexity interpretation of the same phenomenon, with the randomness-dependent advice of [43] to reconstruct a string x becoming the probabilistic Kolmogorov description of x.

for details). In particular, results and insights about the probabilistic measure pK^t can often be translated to other previously investigated measures of time-bounded Kolmogorov complexity.

As a consequence of these and other desirable properties established in Section 3 (e.g., language compression for any $L \in \mathsf{AM}$ under randomized average-case easiness), pK^t is particularly well-suited for applications of meta-complexity in settings that involve probabilistic computations. In the next section, we discuss some applications of pK^t to average-case complexity.

1.2.2 Applications of pK^t to average-case complexity

A distributional problem (L, D) is in AvgBPP if it admits a randomized errorless heuristic scheme. Since the definition of a randomized heuristic scheme is somewhat technical, we refer to Section 2.2 for details, and remark that in our results the following weaker assumption suffices: there is a randomized polynomial-time algorithm B such that, for every $n \geq 1$,

(i) For every $x \in \mathsf{supp}(D_n)$, if $x \in L$, then $\mathbf{Pr}_{r_B}[B(x;n) = 1] \geq 1 - \frac{1}{n}$; and

(ii) $\mathbf{Pr}_{\substack{x \sim D_n \\ r_B}}[B(x;n) = L(x)] \geq 1 - \frac{1}{n}$,

where r_B denotes the randomness of B. In other words, our results also hold under the existence of one-sided error randomized algorithms that can make mistakes on negative instances.

First, we relate the worst-case and average-case complexities of subclasses of PH with respect to probabilistic computations.

▶ **Theorem 1** (Probabilistic Worst-Case to Average-Case Reductions). *The following results hold.*

1. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, *then* $\mathsf{UP} \subseteq \mathsf{RTIME}\big[2^{O(n/\log n)}\big]$.
2. *If* $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, *then* $\mathsf{AM} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.
3. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$, *then* $\mathsf{PH} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$. *More specifically, for any* $\ell \geq 0$, *if* $\mathsf{Dist}\Sigma_{\ell+2}^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, *then* $\Sigma_\ell^{\mathsf{P}} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.

In contrast, [11] established worst-case upper bounds in $\mathsf{DTIME}[2^{O(n/\log n)}]$ assuming a corresponding inclusion in AvgP. Theorem 1 provides the first connections of this form that hold with respect to probabilistic computations.[6]

In a recent work, [6] established *fine-grained* connections between worst-case and average-case complexity. For instance, they showed that if $\mathsf{NTIME}[n]$ can be (deterministically) solved in quasilinear time on average, then $\mathsf{UTIME}[2^{O(\sqrt{n \log n})}] \subseteq \mathsf{DTIME}[2^{O(\sqrt{n \log n})}]$. Next, we discuss implications of fine-grained average-case easiness assumptions in the probabilistic setting. Recall the quasilinear-time complexity classes $\mathsf{QL} = \mathsf{DTIME}[\widetilde{O}(n)]$, $\mathsf{NQL} = \mathsf{NTIME}[\widetilde{O}(n)]$, and the quasilinear-time analog $\mathsf{QLH} = \cup_{\ell \geq 0} \Sigma_\ell^{\mathsf{QL}}$ of the polynomial-time hierarchy PH; see, e.g., [36] for more details.[7] Define $\mathsf{BPQL} = \mathsf{BPTIME}[\widetilde{O}(n)]$, the quasilinear-time version of BPP. Also, define QLSamp to be the class of distribution families that are quasilinear-time samplable.

▶ **Theorem 2** (Probabilistic Fine-Grained Reductions). *The following results hold.*

[6] As in [11], we can obtain a stronger worst-case consequence under the additional assumption that the running time of the (randomized) average-case algorithm on a given instance can be efficiently estimated (without running the algorithm). We refer to Section 4.6 for this result.

[7] As usual, we use $\widetilde{O}(T(n))$ to denote a running time of the form $O(T(n) \cdot \mathsf{poly}(\log T(n)))$.

1. *If* $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$, *then* $\mathsf{UTIME}\left[2^{O(\sqrt{n \log n})}\right] \subseteq \mathsf{RTIME}\left[2^{O(\sqrt{n \log n})}\right]$.
2. *If* $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$, *then* $\mathsf{AMTIME}\left[2^{O(\sqrt{n \log n})}\right] \subseteq \mathsf{BPTIME}\left[2^{O(\sqrt{n \log n})}\right]$.

In contrast with the approach of [6], which requires the construction of highly efficient complexity-theoretic pseudorandom generators, our proofs are significantly simpler and do not require derandomization.

Theorem 1 and Theorem 2 are *formally stronger* than the corresponding results from [11, 6], which are restricted to deterministic algorithms. For instance, consider the implication from [11] that if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ then $\mathsf{UP} \subseteq \mathsf{DTIME}[2^{O(n/\log n)}]$. It immediately follows from $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, and therefore $\mathsf{UP} \subseteq \mathsf{RTIME}[2^{O(n/\log n)}]$ by Theorem 1. On the other hand, the assumption $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ yields $\mathsf{BPP} = \mathsf{P}$ [5]. By padding, we get that $\mathsf{RTIME}[2^{O(n/\log n)}] = \mathsf{DTIME}[2^{O(n/\log n)}]$. (For a similar example in the fine-grained case, see the short proof of Theorem 51 in Section 4.5.)

We also remark that, similarly to previous works, we are not aware of alternate proofs of the results stated above that do not rely on (probabilistic) meta-complexity.

Next, we establish a connection between learning algorithms and randomized average-case complexity. Recall that in the PAC learning model, a learner has access to examples $(x, f(x))$ labelled according to an unknown function $f \in \mathcal{C}$, where $\mathcal{C}$ is a fixed class of Boolean functions. The examples x are drawn according to an unknown probability distribution D_n, which we assume to be supported over $\{0,1\}^n$. The goal of the learning algorithm is to produce, with high probability over its internal randomness and draw of labelled examples, a hypothesis h such that $\Pr_{x \sim D_n}[h(x) \neq f(x)] \leq \varepsilon$.

For a distribution D_n supported over $\{0,1\}^n$, we say that $D_n \in \mathsf{Samp}[T(n)]/a(n)$ if it can be sampled by an algorithm that runs in time $T(n)$ and has advice complexity $a(n)$. We consider the learnability of the class $\mathcal{C} = \mathsf{SIZE}[s]$ of Boolean circuits of size at most $s(n)$, with respect to an unknown distribution D_n from $\mathsf{Samp}[T(n)]/a(n)$. Our result holds in the more challenging setting of agnostic learning (see Section 2.6 for a review of this learning model).

▶ **Theorem 3** (Agnostic Learning from Probabilistic Average-Case Easiness of NP)**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, *then for any time constructible functions* $s, T, a \colon \mathbb{N} \to \mathbb{N}$, *and* $\varepsilon \in [0,1]$, $\mathsf{SIZE}[s(n)]$ *is agnostic learnable on* $\mathsf{Samp}[T(n)]/a(n)$ *in time* $\mathsf{poly}\big(n, \varepsilon^{-1}, s(n), T(n), a(n)\big)$.

Theorem 3 strictly improves a result from [13], which established the same conclusion under the stronger assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.

We finish this section with a technical remark about relativization. The fact that pK^t allows us to avoid the use of a PRG implies that our proofs relativize, i.e., the results stated above hold in the presence of any oracle. In particular, we can show that for any oracle A,

$$\mathsf{DistNP}^A \subseteq \mathsf{AvgBPP}^A \;\Rightarrow\; \mathsf{UP}^A \subseteq \mathsf{RTIME}^A\left[2^{O(n/\log n)}\right].$$

It is not known whether the previous (deterministic) worst-case to average-case reduction for UP, which assumes $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, holds with respect to an arbitrary oracle. More precisely, its proof depends on a PRG from [5], whose proof relies on non-relativizing techniques. (In contrast, it was observed in [13] that one can obtain an alternate (relativized) PRG under the assumption that $\mathsf{DistP}^{\mathsf{NP}} \subseteq \mathsf{AvgP}$, so the worst-case to average-case reduction for NP, which assumes $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgP}$, relativizes.) The above implication is the best possible statement of such a theorem, as it matches the relativization barrier shown by [13], which says that there is an oracle $\mathcal{O}$ such that $\mathsf{DistPH}^{\mathcal{O}} \subseteq \mathsf{AvgP}^{\mathcal{O}}$ but $\mathsf{UP}^{\mathcal{O}} \cap \mathsf{coUP}^{\mathcal{O}} \not\subseteq \mathsf{BPTIME}^{\mathcal{O}}\left[2^{n/\omega(\log n)}\right]$.

1.3 Techniques

1.3.1 Hirahara's worst-case to average-case reduction

We first recall Hirahara's worst-case to average-case reduction from [11], following a presentation in [7]. For simplicity, we consider the case of NP only. Suppose we want an efficient worst-case algorithm for a given NP language L. The idea is to argue (under appropriate average-case easiness assumptions) that every $x \in L$ has an L-witness y_x of small conditional Kolmogorov complexity $\mathsf{K}^t(y_x \mid x)$, for a not too large time bound t, and then simply search for a good witness y by enumerating all possible short candidate Kolmogorov descriptions of y_x, decoding each in time t, and checking if it is a valid L-witness for $x \in L$. The overall time complexity of such a procedure is quasi-linear in t and exponential in $\mathsf{K}^t(y_x \mid x)$, and so we would like to minimize these two parameters.

1.3.1.1 Compression via Hadamard codes

How does one argue that a given binary string w has small K^t complexity under average-case easiness assumptions? The idea is to "encode" w into a distribution $\mathcal{D}(w)$ so that any algorithm distinguishing $\mathcal{D}(w)$ from the uniform distribution can be used to "reconstruct" w, possibly using some randomness and a few bits of advice. Several encoding methods with such properties are known in the literature on pseudorandomness. The one used by [11] is (the direct product of) the binary Hadamard error-correcting code encoding, which has an efficient list-decoding algorithm due to Goldreich and Levin [9]. This decoding algorithm is *randomized*, and it needs extra information (advice) about the string w in order to recover w from a distinguisher algorithm for $\mathcal{D}(w)$; moreover, the advice string a depends on the randomness r used by the decoding algorithm. To get a deterministic K^t complexity bound, Hirahara [11] fixes the random string r to be $G(\alpha)$, where G is an efficient PRG fooling polynomial-size Boolean circuits, and α is a short seed.[8] The seed α becomes part of the Kolmogorov description of w, with the other part being the advice string a corresponding to the random string $r = G(\alpha)$. Such a PRG G is known to exist under the average-case assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ [5]. So one gets to upper-bound $\mathsf{K}^t(w)$, *assuming* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *and* that one has an efficient *distinguisher* for the distribution $\mathcal{D}(w)$, where the time bound t is polynomial in the run time of the distinguisher and the length of w.

1.3.1.2 Getting a distinguisher

How does one argue the existence of an efficient distinguisher for $\mathcal{D}(w)$? Consider the case of $w = (x, y_x)$, for an n-bit string $x \in L$ (where $L \in \mathsf{NP}$) and y_x the lexicographically first L-witness for x. In the presence of the SAT oracle, one can easily compute y_x given x by a well-known search-to-decision reduction, which implies that $\mathsf{K}^{2t,\mathsf{SAT}}(x, y_x) \leq \mathsf{K}^t(x) + O(\log t)$, for any sufficiently large time bound t. On the other hand, for completely random pairs of strings (x, y), the Kolmogorov complexity of (x, y) (even with the SAT oracle) is typically larger than $\mathsf{K}^t(x) + O(\log t)$ (since y is unrelated to x). By carefully choosing the parameters of the Hadamard-based encoding of w, one gets a distinguisher for $\mathcal{D}(x, y_x)$ from uniform, where a

[8] The fact that a PRG G fools the Hadamard code decoding algorithm relies on the observation that the advice string a happens to be efficiently computable from w and randomness r.

distinguisher is a Σ_2^{P} algorithm. Since we just need a distinguisher that works well on average, the assumption $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgP}$ implies the existence of a sufficiently good deterministic polytime distinguisher for $\mathcal{D}(x, y_x)$. The latter implies that $\mathsf{K}^{\mathsf{poly}(t)}(x, y_x) \leq \mathsf{K}^t(x) + O(\log t)$.

1.3.1.3 Chain rule for K^t

But we are still not done. Recall that our goal is to upperbound the conditional Kolmogorov complexity $\mathsf{K}^t(y_x \mid x)$, and so far we only have an upper bound on $\mathsf{K}^{\mathsf{poly}(t)}(x, y_x)$. Intuitively, one might hope to have a "chain rule" for K^t, saying that $\mathsf{K}^t(x, y) \approx \mathsf{K}^{t'}(x) + \mathsf{K}^{t'}(y \mid x)$, for polynomially related time bounds t and t' (such a chain rule is known for the original, time-unbounded Kolmogorov complexity, and is often called "Symmetry of Information"). It is not known if such a chain rule holds in time-bounded settings (in fact, there is some negative evidence [31]), but it does hold under the average-case easiness assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ [12, 7]. Using this chain rule and the upper bound $\mathsf{K}^{\mathsf{poly}(t)}(x, y_x) \leq \mathsf{K}^t(x) + O(\log t)$, we get that $\mathsf{K}^{\mathsf{poly}(t)}(y_x \mid x) \leq \mathsf{K}^t(x) - \mathsf{K}^{\mathsf{poly}(t)}(x) + O(\log t)$.

1.3.1.4 Computational depth

Unfortunately, the two Kolmogorov complexity measures of x on the right-hand side are for different (polynomially related) time bounds, and so do not cancel out. It is still possible to get a nontrivial upper bound on such a difference (known as the computational depth of x) by making t large enough. In particular, a simple averaging argument implies that every $x \in \{0,1\}^n$ has the computational depth at most $O(n/\log n)$ for a time bound $t \leq 2^{O(n/\log n)}$. This concludes the argument bounding the conditional Kolmogorov complexity of the witness y_x.

1.3.2 Extending Hirahara's reduction to the randomized case

If Hirahara's worst-case to average-case reduction described above were *black-box*, then we could simply replace the assumed AvgP algorithm for $\mathsf{Dist}\Sigma_2^{\mathsf{P}}$ with an AvgBPP algorithm and obtain a randomized algorithm for solving every language in NP with the same running time. However, the reduction in Hirahara's argument is highly non-black-box: It crucially relies on the assumed average-case algorithm being *efficient* and *deterministic*. So we need to look inside each of the steps in the reduction and try to adapt it, using *randomized* average-case easiness assumptions.

For the Hadamard decoding step, we cannot derandomize the Goldreich-Levin algorithm as we no longer have a PRG (which is only known to exist under the *deterministic* average-case assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$). Leaving randomness in, we now get an upper bound on $\mathsf{pK}^t(w)$ from any (randomized) distinguisher for $\mathcal{D}(w)$. The fact that the advice in the Goldreich-Levin reconstruction algorithm depends on randomness forces us to use the stronger notion of randomized Kolmogorov complexity pK^t rather than rK^t. On the positive side, no average-case easiness assumptions are now needed for this step.[9]

Using a more complicated notion of pK^t creates new challenges in the next step, where we need to argue the existence of efficient distinguishers for distributions $\mathcal{D}(x, y_x)$, where y_x is the lexicographically smallest witness for $x \in L$, for some $L \in \mathsf{NP}$. Note that the pK^t

[9] Formally, the randomized reconstruction procedure only allows us to obtain a bound on $\mathsf{pK}^t_\delta(w)$ for $\delta = 1/\mathsf{poly}(|w|)$, where pK^t_δ is the natural generalization of pK^t with a relaxed success probability parameter δ instead of $2/3$. However, as another useful feature of pK^t complexity, we show that its success probability can be boosted with a small complexity overhead.

definition resembles the definition of AM, where Merlin provides a Kolmogorov description of a given string, based on Arthur's randomness.[10] A naive algorithm to distinguish $\mathcal{D}(x, y_x)$ from uniform (mimicking the algorithm for the deterministic case discussed above) would be in (the promise version of) $\mathsf{AM}^{\mathsf{NP}} \subseteq \Pi_3^{\mathsf{P}}$ (see Section A.1 for a related result), which would force us to use a stronger average-case assumption like $\mathsf{Dist}\Sigma_3^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$. We manage to get a good randomized distinguisher for $\mathcal{D}(x, y_x)$ under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, which is a generalization of the deterministic case from [11] discussed above. Roughly speaking, our idea is to give a random string used in the pK^t definition as an additional input to the distinguisher, reducing its complexity to Σ_2^{P}.[11]

The proof of Symmetry of Information for K^t under the assumption $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ from [7] is fortunately robust enough to generalize to the case of pK^t assuming $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. We extend it further by conditioning on random strings r, and using families of strings $\{y_r\}_r$, indexed by randomness r, instead of a single string y. This allows us to get a very simple proof of a natural generalization of the worst-case to average-case reduction above to the case of AM. Namely, we show that $\mathsf{AM} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$, assuming that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$. Intuitively, for $L \in \mathsf{AM}$, witnesses y depend on both an input $x \in L$ and the randomness r used by Arthur. So it's natural to try to upperbound $\mathsf{pK}^t(y_{x,r} \mid x, r)$. However, just using (x, r) as a new input x in the original Symmetry of Information statement would result in $\mathsf{pK}^t(y_{x,r} \mid x, r)$ being upperbounded by the pK^t version of the computational depth of (x, r), which is a string of length $\mathsf{poly}(n)$, for $|x| = n$. The averaging argument for the computational depth (which works for any natural notion of Kolmogorov complexity) would give us the bound $O(n'/\log n')$ for $n' = |(x, r)| = \mathsf{poly}(n)$, which is useless. On the other hand, with the new conditional Symmetry of Information statement, we get that $\mathsf{pK}^t(y_{x,r} \mid x, r)$ is at most the *conditional* computational depth $\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{\mathsf{poly}(t)}(x \mid r) + O(\log t)$. Now the input length for the averaging argument for the conditional computational depth is still $|x| = n$, yielding the upper bound $O(n/\log n)$, which allows us to prove Item 2 of Theorem 1.

While pK^t complexity is instrumental in the proof of Items 1 and 2 of Theorem 1, we show that one *can* use the simpler rK^t notion, *if* one assumes that $\mathsf{DistP}^{\Sigma_2^{\mathsf{P}}} \subseteq \mathsf{AvgBPP}$. This then allows us to get a worst-case to average-case reduction for the polytime hierarchy PH, assuming $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$, thereby proving Item 3 of Theorem 1. Another important ingredient in our PH proof is an idea of randomized *witness compression*, which allows us to perform a delicate induction on the level ℓ of PH; the randomized witness compression necessary for our inductive argument is achieved by using the rK^t complexity.

1.3.3 Fine-grained case

The use of pK^t also allows us to extend our results to the fine-grained case (Theorem 2) more easily. First of all, as observed in [6], an improved worst-case running time $2^{O(\sqrt{n \log n})}$ follows if one can refine the above-mentioned computational depth expression from $\mathsf{K}^t(x) - \mathsf{K}^{\mathsf{poly}(t)}(x) + O(\log t)$ to $\mathsf{K}^t(x) - \mathsf{K}^{\widetilde{O}(t)}(x) + O(\log t)$ (note that the "blow-up" of the time bound in K^t is quasilinear in the latter as opposed to polynomial in the former). A key to such a refinement is to optimize the running time of the reconstruction procedure for the Hadamard-based encoding described above (see [6, Section 2]). In particular, this running time depends on both the running time of the distinghuisher and the overhead incurred by

[10] Analogously, rK^t is similar to MA, and K^t to NP.

[11] In the case of Theorem 1 Item 1 (i.e., for UP), we further reduce the complexity of the distinguisher so we can rely on the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$.

running the PRG. It turns out that the running time of the distinghuisher, which can be obtained from a Σ_2^{QL} algorithm (in the NP case), can be optimized if one assumes that Σ_2^{QL} admits quasilinear-time average-case algorithms, which is exactly the fine-grained average-case easiness assumption. Then to minimize the overhead of the PRG, the authors of [6] construct a particular efficient PRG under the fine-grained average-case easiness assumption, which incurs just a quasilinear overhead in the running time (note that this overhead would be polynomial using the PRG from [5]).

Thanks to the use of pK^t, we do not need any PRG in our proof! Therefore we achieve a refined probabilistic computational depth expression of $\mathsf{pK}^t(x) - \mathsf{pK}^{\widetilde{O}(t)}(x) + O(\log t)$ directly from the fine-grained probabilistic average-case easiness assumption.

1.3.4 Learning under randomized average-case assumptions

Our improvement of the learning result from [13] under a weaker probabilistic average-case easiness assumption, Theorem 3, follows the high-level approach from [13] but also crucially relies on the use of pK^t. The key idea is to design something called random-right-hand-side-refuter (RRHS-refuter). Roughly speaking, this is an efficient algorithm that distinguishes the distribution $\left(x^{(1)}, \ldots, x^{(m)}, f(x^{(1)}), \ldots f(x^{(m)})\right)$ where each $x^{(i)}$ is picked from a distribution D and f is from the concept class $\mathcal{C}$, from the distribution $\left(x^{(1)}, \ldots, x^{(m)}, b^{(1)}, \ldots b^{(m)}\right)$ where each $b^{(i)}$ is uniform. It is known that such an algorithm implies a learner for $\mathcal{C}$ under the distribution D [44, 21]. The authors of [13] construct a deterministic RRHS-refuter, using an algorithm that estimates the K^t complexity of a given string for a given t, which exists under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ [11]. More specifically, [13] shows that if a string is picked from the former case, where D is samplable by a small circuit and f is also computable by a small circuit, then it is likely to have "small" K^t complexity (for carefully chosen m and t). On the other hand, using results such as symmetry of information and optimal coding for K^t under an average-case easiness assumption [11], it can be shown that a random string from the latter case is likely to have "large" K^t complexity.

In our case, we will use a *randomized* algorithm that estimates the more complicated pK^t complexity of a given string, which we show to exist under the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. Combining such an algorithm with the new symmetry of information for pK^t (which holds under the same probabilistic average-case easiness assumption) and an optimal coding result for pK^t from [34], we are able to construct a *randomized* RRHS-refuter, which suffices to yield the same learning result.

1.4 Directions and open problems

We have shown that several recent results proved under assumptions of the form $\mathcal{C} \subseteq \mathsf{AvgP}$ survive in the setting of randomized errorless heuristic schemes, i.e., under the weaker assumption that $\mathcal{C} \subseteq \mathsf{AvgBPP}$. Moreover, as in previous results, it is enough for us to assume the existence of *one-sided* error (randomized) algorithms that can be incorrect on negative inputs.[12] The next natural step would be to obtain results under an even weaker average-case easiness assumption such as $\mathcal{C} \subseteq \mathsf{HeurBPP}$ (i.e., under the existence of *two-sided* error randomized heuristic schemes). It would be interesting to investigate if our framework can be combined with recent results from [14, 15] to achieve progress on this front.

[12] Note that we also allow the average-case probabilistic algorithm to err on each fixed positive input with small probability over its internal randomness.

We are also interested in understanding the potential of pK^t complexity. We have seen how to employ it to design agnostic learning algorithms, obtain new relations between worst-case and average-case complexity, and simplify previous proofs. Additionally, in [34] this complexity measure is used to establish an unconditional version of the main result of [1]. Are there more applications of pK^t to algorithms and complexity theory?

Finally, is it possible to extend our techniques to quantum computations, and to show that if $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBQP}$ then $\mathsf{QMA} \subseteq \mathsf{BQTIME}[2^{O(n/\log n)}]$? Exploring this and related questions is likely to lead to interesting developments in quantum time-bounded Kolmogorov complexity.

2 Preliminaries

2.1 Notation

We say that a function $p\colon \mathbb{N} \to \mathbb{N}$ is *non-decreasing* if $p(\ell) \geq \ell$ for every integer $\ell \geq 0$. Throughout the paper, we will rely on constructions whose associated complexity parameters (e.g., running time) can be bounded by some polynomial p. We will implicitly assume that the polynomials provided in these results are non decreasing in order to avoid including this condition in every statement. This can be done without loss of generality by the monotonicity of our bounds.

For a probability distribution D_n, we use $\mathsf{supp}(D_n)$ to denote its support. Given an element x, we let $D_n(x)$ denote the probability of x under D_n. We denote by $\mathcal{U}_n$ the uniform distribution over n-bit strings.

Let D_1 and D_2 be probability distributions supported over a set X, $A\colon X \to \{0,1\}$, and $\varepsilon \geq 0$. We say that A *ε-distinguishes* D_1 and D_2 if

$$\left| \mathop{\mathbf{Pr}}_{x\sim D_1}[A(x)=1] - \mathop{\mathbf{Pr}}_{x\sim D_2}[A(x)=1] \right| \geq \varepsilon.$$

If A is a randomized algorithm, each probability in the expression above also takes into account the internal randomness of A, which will be denoted by r_A.

2.2 Average-case complexity

Recall that a pair (L, D) is a *distributional problem* if $L \subseteq \{0,1\}^*$ and $D = \{D_n\}_{n\geq 1}$ is an ensemble of probability distributions, where each D_n is supported over $\{0,1\}^*$.

Let $D = \{D_n\}_{n\geq 1}$ be an ensemble of distributions. We say that $D \in \mathsf{PSamp}$ if there is a randomized polytime algorithm A such that, for every $n \geq 1$, $A(1^n)$ is distributed according to D_n. More generally, we use $D_n \in \mathsf{Samp}[T(n)]/a(n)$ to denote that D_n can be sampled by an algorithm that runs in time $T(n)$ and has advice complexity $a(n)$.

We let DistNP denote the set of distributional problems (L, D) with $L \in \mathsf{NP}$ and $D \in \mathsf{PSamp}$.

▶ **Definition 4** ($\mathsf{Avg}_\delta\mathsf{BPP}$ [3]). *Let (L, D) be a distributional problem, and $\delta\colon \mathbb{N} \to [0,1]$. We say that $(L, D) \in \mathsf{Avg}_\delta\mathsf{BPP}$ if there is randomized polytime algorithm A such that*

1. *for every $n > 0$, and every $x \in \mathsf{supp}(D_n)$,*

$$\mathop{\mathbf{Pr}}_{r_A}[A(x;n) \notin \{L(x), \perp\}] \leq \frac{1}{4},$$

2. *for every $n > 0$,*

$$\mathop{\mathbf{Pr}}_{x\sim D_n}\left[\mathop{\mathbf{Pr}}_{r_A}[A(x;n)=\perp] \geq 1/4\right] \leq \delta(n),$$

where r_A denotes the internal randomness of A. Such an algorithm A is called an randomized errorless heuristic for (L, D) with failure probability at most δ.

▶ **Definition 5** (AvgBPP [3]). [13] *We say that* (L, D) *is in* AvgBPP *if there exist a randomized algorithm* A *and a polynomial* p *such that*

1. *for every* $n, \delta > 0$, *and every* $x \in \mathsf{supp}(D_n)$,

$$\Pr_{r_A}[A(x; n, \delta) \notin \{L(x), \perp\}] \leq \frac{1}{4},$$

2. *for every* $n, \delta > 0$,

$$\Pr_{x \sim D_n}\left[\Pr_{r_A}[A(x; n, \delta) = \perp] \geq 1/4\right] \leq \delta(n),$$

3. *for every* $n, \delta > 0$, *and every* $x \in \mathsf{supp}(D_n)$, $A(x; n, \delta)$ *runs in time at most* $p(n/\delta)$.

Such an algorithm A *is called a* randomized errorless heuristic scheme for (L, D).

▶ **Lemma 6.** *Let* (L, D) *be a distributional problem in* $\mathsf{Avg}_{n^{-1}}\mathsf{BPP}$. *There exists a randomized polynomial-time algorithm* B *such that*

1. *for every* $n > 0$, *and every* $x \in \mathsf{supp}(D_n)$, $B(x; n) \in \{0, 1\}$,
2. *for every* $n > 0$, *and every* $x \in \mathsf{supp}(D_n)$, *if* $x \in L$, *then* $\mathbf{Pr}_{r_B}[B(x; n) = 1] \geq 1 - \frac{1}{n}$, *and*
3. *for every* $n > 0$, $\mathbf{Pr}_{\substack{x \sim D_n \\ r_B}}[B(x; n) = L(x)] \geq 1 - \frac{3}{n}$.

Proof. Let A be an $\mathsf{Avg}_{n^{-1}}\mathsf{BPP}$ algorithm for (L, D) as in Definition 4. Let A' be the algorithm obtained, using standard amplification techniques,[14] such that for all $x \in \mathsf{supp}(D_n)$,

$$\Pr_{r_{A'}}[A'(x; n) \notin \{L(x), \perp\}] \leq \frac{1}{n} \tag{1}$$

and

$$\Pr_{x \sim D_n}\left[\Pr_{r_{A'}}[A'(x; n) = \perp] \geq 1/n\right] \leq \frac{1}{n}. \tag{2}$$

Let B be the randomized algorithm which, on input $x \in \mathsf{supp}(D_n)$, simulates $A'(x; n)$ and outputs 1 if $A'(x; n) \in \{1, \perp\}$ or 0 if $A'(x; n) = 0$. By Equation (1), if $x \in L$, then

$$\Pr_{r_B}[B(x; n) = 0] = \Pr_{r_{A'}}[A'(x; n) \notin \{L(x), \perp\}] \leq \frac{1}{n}.$$

Let $bad := \{x \in \mathsf{supp}(D_n) \mid \mathbf{Pr}_{r_{A'}}[A'(x; n) = \perp] \geq 1/n\}$. By Equation (2), at most a $(1/n)$-measure of $x \in \mathsf{supp}(D_n)$ are in bad. Thus,

$$\begin{aligned}
&\Pr_{\substack{x \sim D_n \\ r_B}}[B(x; n) \neq L(x)] \\
&\leq \Pr_{x, r_{A'}}[A'(x; n) = \perp] + \Pr_{x, r_{A'}}[A'(x; n) \notin \{L(x), \perp\}] \\
&\leq \Pr_{x}[x \in bad] + \Pr_{x, r_{A'}}[A'(x; n) = \perp \mid x \notin bad] + \Pr_{x, r_{A'}}[A'(x; n) \notin \{L(x), \perp\}] \\
&\leq \frac{1}{n} + \frac{1}{n} + \frac{1}{n} = \frac{3}{n}.
\end{aligned}$$

This completes the proof. ◀

[13] Some authors define AvgBPP as $\cap_{c>0}\mathsf{Avg}_{n^{-c}}\mathsf{BPP}$. All our results also hold with respect to that definition.

[14] In more detail, A' can be obtained from A by running it $O(n)$ times and selecting the most common output.

The following result of Impagliazzo and Levin [18] shows that, for every $\ell \geq 1$, the easiness of $\mathsf{Dist}\Sigma_\ell^\mathsf{P}$ is equivalent to that of $(\Sigma_\ell^\mathsf{P}, \mathcal{U})$. This implies that the average-case easiness under uniform distribution is sufficient for all our main results.

▶ **Theorem 7** ([18]; see also [3, Theorem 31]). [15] *For every $\ell \geq 1$,*

$$\left(\Sigma_\ell^\mathsf{P}, \mathsf{PSamp}\right) \subseteq \mathsf{AvgBPP} \iff \left(\Sigma_\ell^\mathsf{P}, \mathcal{U}\right) \subseteq \mathsf{AvgBPP}.$$

2.3 Pseudodeterministic PRGs

▶ **Definition 8** (Pseudodeterministic PRG [39, 33]). *Fix an error function $\varepsilon\colon \mathbb{N} \to \mathbb{R}$. A function family $G_n\colon \{0,1\}^\ell \to \{0,1\}^n$ is an* ε-error Pseudorandom Generator (PRG) *if, for all sufficiently large $n \in \mathbb{N}$, the distribution $G(\mathcal{U}_{\ell(n)})$ $\varepsilon(n)$-fools circuits of size n, using the seed length $\ell = \ell(n)$, i.e., for all circuits C of size n on n inputs,*

$$\left| \mathop{\mathbf{Pr}}_{r\sim\{0,1\}^n}[C(r)=1] - \mathop{\mathbf{Pr}}_{\alpha\sim\{0,1\}^{\ell(n)}}[C(G(\alpha))=1] \right| \leq \varepsilon.$$

Such a PRG G is called a pseudodeterministic PRG *if there is a randomized algorithm M_G such that, for every n and $\alpha \in \{0,1\}^{\ell(n)}$, $M_G(1^n, \alpha)$ outputs the string $G_n(\alpha) \in \{0,1\}^n$ with probability at least $1 - 2^{-n}$ over its internal randomness, and M_G runs in time $2^{O(\ell(n))}$.*

▶ **Lemma 9** ([19, 39]). *If BPE contains a language L of circuit complexity $2^{\Omega(n)}$ for almost all input lengths n, then there is an ε-error pseudodeterministic PRG of seed length $O(\log n/\varepsilon)$.*

▶ **Lemma 10** ([35, Lemma 2]). *$\mathsf{E}^{\Sigma_2^\mathsf{P}}$ contains a language of maximum circuit complexity (at least $2^n/n$) for almost all input lengths n.*

Using padding as in [2], we get the following.

▶ **Lemma 11.** *If $\mathsf{Dist}\mathsf{P}^{\Sigma_2^\mathsf{P}} \subseteq \mathsf{AvgBPP}$, then $\mathsf{E}^{\Sigma_2^\mathsf{P}} = \mathsf{BPE}$.*

Proof. The inclusion $\mathsf{BPE} \subseteq \mathsf{E}^{\Sigma_2^\mathsf{P}}$ follows from the inclusion $\mathsf{BPP} \subseteq \Sigma_2^\mathsf{P}$ [22] by padding. For the other direction, let $L \in \mathsf{E}^{\Sigma_2^\mathsf{P}}$ be arbitrary. Define its padded version $L^{pad} = \{(x, 1^{2^{2|x|}}) \mid x \in L\}$. Note that $L^{pad} \in \mathsf{P}^{\Sigma_2^\mathsf{P}}$. Define a family of distributions $D_n = (\mathcal{U}_n, 1^{2^{2n}})$, for the uniform distribution over n-bit strings $\mathcal{U}_n$. By assumption, $(L^{pad}, D_n) \in \mathsf{AvgBPP}$, and so, by Lemma 6, Item 3, there is a randomized polytime algorithm B such that $B(x, 1^{2^{2|x|}}) \neq L(x)$ with probability less than $3 \cdot 2^{-2n} < 1/(3 \cdot 2^n)$, where the probability is over a uniformly random $x \in \{0,1\}^n$ and the internal randomness of B. It follows that, for *every* $x \in \{0,1\}^n$, $B(x, 1^{2^{2|x|}}) \neq L(x)$ with probability less than $1/3$ over its internal randomness. Hence, the randomized algorithm $B'(x) := B(x, 1^{2^{2|x|}})$ decides if $x \in L$ with probability at least $2/3$, for all sufficiently large inputs $x \in \{0,1\}^n$, yielding $L \in \mathsf{BPE}$. ◀

▶ **Corollary 12.** *If $\mathsf{Dist}\mathsf{P}^{\Sigma_2^\mathsf{P}} \subseteq \mathsf{AvgBPP}$, then there is an ε-error pseudodeterministic PRG of seed length $O(\log n/\varepsilon)$.*

Proof. Immediate by combining Lemmas 9–11. ◀

[15] The published paper by Impagliazzo and Levin [18] contains a proof applicable only to the case of heuristics with errors (such as $\mathsf{HeurBPP}$). The case of errorless heuristics (such as AvgBPP) requires a different argument, which is given in Section 5.2 of [3], based on Impagliazzo's unpublished notes.

2.4 Direct Product Generator

For $x, z \in \{0,1\}^n$, we let $x \cdot z := \sum_{i=1}^{n} x_i z_i \pmod 2$ denote their inner product modulo 2.

▶ **Definition 13** (Direct Product Generator (DPG) [11, Definiton 3.10]). *For $k, n \in \mathbb{N}$, we define the k-*wise direct product generator *to be the function*

$$\mathsf{DP}_k \colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$$

such that

$$\mathsf{DP}_k(x; z^1, \ldots, z^n) := (z^1, \ldots, z^k, x \cdot z^1, \ldots, x \cdot z^k).$$

▶ **Lemma 14** (DPG Reconstruction [11, Lemma 3.14]). *For any $n, k \in \mathbb{N}$ with $k \leq 2n$, and $\varepsilon > 0$, there exists a pair of algorithms A and $\mathsf{Recon}^{(-)}$ such that*

- Recon^D *takes oracle access to an oracle $D \colon \{0,1\}^{nk+k} \to \{0,1\}$.*
- $A \colon \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^k$ *is called the advice function and is computable in time* $\mathsf{poly}(n/\varepsilon)$.
- $\mathsf{Recon}^{(-)} \colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^n$ *is called a reconstruction procedure and is computable in time* $\mathsf{poly}(n/\varepsilon)$.[16]
- *The randomness complexity r is at most* $\mathsf{poly}(n/\varepsilon)$.
- *For any string $x \in \{0,1\}^n$ and any function D that ε-distinguishes $\mathsf{DP}_k(x; \mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$, it holds that*

$$\Pr_{w \sim \{0,1\}^r} \left[\mathsf{Recon}^D(A(x,w), w) = x \right] \geq 1/\mathsf{poly}(n/\varepsilon).$$

2.5 Kolmogorov complexity measures

For a string $w \in \{0,1\}^*$, we use $|w| \in \mathbb{N}$ to denote its length. We let ϵ represent the empty string. Let U be a Turing machine. For a function $t \colon \mathbb{N} \to \mathbb{N}$ and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t_U(x) \stackrel{\text{def}}{=} \min_{p \in \{0,1\}^*} \left\{ |p| \mid U(p, \epsilon) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \right\}$$

be the *t-time-bounded Kolmogorov complexity* of x. The machine U is said to be *time-optimal* if for every machine M there exists a constant c_M such that for all $x \in \{0,1\}^n$ and $t \colon \mathbb{N} \to \mathbb{N}$ satisfying $t(n) \geq n$,

$$\mathsf{K}^{c_M \cdot t \log t}_U(x) \leq \mathsf{K}^t_M(x) + c_M,$$

where for simplicity we write $t = t(n)$. It is well known that there exist time-optimal machines (see, e.g., [26, Chapter 7]). We fix such a machine, and drop the index U when referring to time-bounded Kolmogorov complexity measures.

We remind the reader that the (time-unbounded) *Kolmogorov complexity* of a string x, denoted $\mathsf{K}(x)$, is defined in the same way but does not impose a fixed upper bound $t(|x|)$ on the running time of $U(p, \varepsilon)$.

Given strings $x, y \in \{0,1\}^*$, we can also consider the *conditional t-time-bounded Kolmogorov complexity of x given y*, defined as

$$\mathsf{K}^t(x \mid y) \stackrel{\text{def}}{=} \min_{p \in \{0,1\}^*} \left\{ |p| \mid U(p, y) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \right\}.$$

[16] We note that if the oracle is a uniform algorithm that runs in time $t(m)$ on inputs of length m, the reconstruction procedure can be computed in time $t(nk+k) \cdot \mathsf{poly}(n/\varepsilon)$.

From now on, we will not distinguish between a Turing machine M and its encoding p_M according to U. While the running time t of M on an input y and the running time of the universal machine U on (p_M, y) might differ by a multiplicative factor of $O(\log t)$, this will be inessential in all our results.[17] For simplicity, we will assume the encoded machines to U are paddable so that M and $M \circ 0^a$ denote the same machine for every natural number a.

We denote by $\mathsf{rK}^t_\delta(x)$ the minimum length of a randomized machine $\mathcal{M}$ that outputs x with probability at least δ when running for at most t steps (see [38, 33]). We simply write $\mathsf{rK}^t(x)$ when $\delta = 2/3$.

We formally introduce $\mathsf{pK}^t_\delta(x)$ and discuss its properties in Section 3.

2.6 Agnostic learning model

This section reviews standard notions from computational learning theory (see, e.g., [41]). Consider the problem of learning an unknown *concept* $f\colon X \to \{0,1\}$ over a finite *domain* X. We will always let $X = \{0,1\}^n$, for some $n \geq 1$. A *concept class* $\mathcal{C}$ is a collection of concepts of this form. For a fixed n, we let $\mathcal{C}_n$ denote $\mathcal{C} \cap \{f\colon \{0,1\}^n \to \{0,1\}\}$. For a size function $s\colon \mathbb{N} \to \mathbb{N}$, we let

$$\mathsf{SIZE}[s] = \{f\colon \{0,1\}^n \to \{0,1\} \mid n \in \mathbb{N} \text{ and } f \text{ admits a size-}s(n) \text{ circuit}\}$$

denote the concept class of Boolean circuits of size (number of gates) at most $s(n)$ over n input variables. For definiteness, we consider circuits over AND and OR gates of fan-in 2 and NOT gates.

A *randomized Boolean function* f maps an input x to a distribution D supported over $\{0,1\}$. Given a distribution D_n supported over $\{0,1\}^n$, a function $h\colon \{0,1\}^n \to \{0,1\}$, and a (possibly randomized) Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we let

$$\mathsf{err}_{D_n}(h, f) = \Pr_{f,\, x \sim D_n}[h(x) \neq f(x)].$$

We will consider the (agnostic) learnability of a concept class $\mathcal{C}$ with respect to a *class* $\mathcal{D}$ of ensembles $D = \{D_n\}_{n \geq 1}$ of distributions D_n. We use $\mathcal{D}_n$ to denote $\{D_n \mid D \in \mathcal{D}\}$. For simplicity, we refer to $\mathcal{D}$ as a class of distributions.

Given a randomized function $f\colon \{0,1\}^n \to \{0,1\}$ and a distribution D_n, the learning algorithm has access to an *example oracle* $\mathsf{EX}(f, D_n)$ that behaves as follows: each query to $\mathsf{EX}(f, D_n)$ returns an independent and identically distributed pair (x, b), where x is sampled according to D_n and $b = f(x)$. The *sample complexity* of the learning algorithm is the number of queries made to $\mathsf{EX}(f, D_n)$.

Before formalising the agnostic learning model, we provide an informal overview. The learning algorithm *knows* the class $\mathcal{D}$ of distributions and the concept class $\mathcal{C}$. It is given access to $\mathsf{EX}(f, D_n)$ for some *unknown* $D \in \mathcal{D}_n$ and an *arbitrary* (possibly randomized) function $f\colon \{0,1\}^n \to \{0,1\}$. The goal of the learning algorithm is to produce, with high probability over its internal randomness and queries to $\mathsf{EX}(f, D)$, a hypothesis h such that $\mathsf{err}_D(h, f) \leq \mathsf{opt}_{\mathcal{C}_n, D, f} + \varepsilon$, where

$$\mathsf{opt}_{\mathcal{C}_n, D, f} = \min_{c \in \mathcal{C}_n} \mathsf{err}_D(c, f).$$

[17] It is also possible to consider prefix-free notions of Kolmogorov complexity. Since our results hold up to additive $O(\log |x|)$ terms, we will not make an explicit distinction.

In other words, the learning algorithm is required to output an efficient representation of a hypothesis $h\colon \{0,1\}^n \to \{0,1\}$ that is almost as accurate as the best concept in $\mathcal{C}_n$ with respect to the pair (D, f). (Note that h is not required to be a function in $\mathcal{C}$.)

▶ **Definition 15** (Agnostic PAC learning [20]). *Let $\mathcal{C}$ be a concept class, and let $\mathcal{D}$ be a class of distributions. We say that a randomized algorithm A agnostically learns $\mathcal{C}$ on $\mathcal{D}$ if the following holds. For every $n \geq 1$, distribution $D \in \mathcal{D}_n$, and randomized function $f\colon \{0,1\}^n \to \{0,1\}$, when A is given oracle access to $\mathsf{EX}(f, D)$ and receives as input n, $\varepsilon > 0$, and $\delta > 0$,*

$$\Pr_{A,\,\mathsf{EX}(f,D)}\left[A^{\mathsf{EX}(f,D)} \text{ outputs a hypothesis } h \text{ such that } \mathsf{err}_D(h,f) \leq \mathsf{opt}_{\mathcal{C}_n,D,f} + \varepsilon\right] \geq 1-\delta.$$

We remark that an equivalent way of formulating the agnostic learning model is by considering distributions $\mathcal{D}$ supported over $\{0,1\}^{n+1}$. In this case, there is no need to refer to randomized functions, and one measures the error of a hypothesis h via $\mathsf{err}_{\mathcal{D}}(h) = \Pr_{(x,b)\sim\mathcal{D}}[h(x) \neq b]$, where $x \in \{0,1\}^n$ and $b \in \{0,1\}$.

2.7 Input encodings

In some situations, it will be useful to refer to collections of probability distributions that are indexed by a constant number of parameters. It is not hard to reduce this to the case of a single parameter 1^n. Moreover, it is possible to assume without loss of generality that distribution D_n is supported over $\{0,1\}^n$ instead of $\{0,1\}^{\leq \mathsf{poly}(n)}$. This can be done using standard techniques, and will be implicitly assumed in our arguments. In any case, for completeness, we include more details about input encodings in this section.

For $j \in \mathbb{N}$, we use $\mathsf{bin}(j) \in \{0,1\}^{\lfloor \log j \rfloor + 1}$ to denote its binary representation. We will often consider languages that view their inputs as a tuple of parameters. We describe how to encode such an input into a single binary string. For a tuple $(a_1, a_2, \ldots, a_k)$ where $k \in \mathbb{N}$, we encode it as

$$\ell_1 \circ 01 \circ \ell_2 \circ 01 \circ \ell_{k-1} \circ 01 \circ a_1 \circ a_2 \circ \cdots \circ a_k,$$

where for each $i \in [k]$, ℓ_i is obtained from $\mathsf{bin}(|a_i|)$ with every bit doubled. Note that in this case, $(a_1, a_2, \ldots, a_k)$ has an encoding length of

$$|a_k| + \sum_{i=1}^{k-1} \left(2 \cdot (\lfloor \log |a_i| \rfloor + 2) + |a_i|\right). \tag{3}$$

We will also need to consider distributions that generate input instances that are tuples, while the only known information to the distributions is the length of the instances. To do this, we will encode the format as the length of the instances. More specifically, we use the following way to encode tuples [3].

▶ **Proposition 16** ([3]; see also [6, Proposition 3.1]). *There is a pair of polytime encoding-decoding algorithms $\mathsf{Enc}\colon \mathbb{N}^* \to \mathbb{N}$ and $\mathsf{Dec}\colon \mathbb{N} \to \mathbb{N}^* \cup \{\perp\}$ such that*

1. *For $k \in \mathbb{N}$ and k integers $n_1, n_2, \ldots, n_k$, we write $\langle n_1, n_2, \ldots, n_k\rangle := \mathsf{Enc}(n_1, n_2, \ldots, n_k)$, and we have*
   a. $\sum_{i=1}^{k} (2 \cdot (\lfloor \log n_i \rfloor + 2) + n_i) \leq \langle n_1, n_2, \ldots, n_k\rangle \leq O_k\big(\Pi_{i=1}^{k}(n_i \cdot \log^2 n_i)\big)$.
   b. $\mathsf{Dec}(\langle n_1, n_2, \ldots, n_k\rangle) = n_1, \ldots, n_k$. *Here, Dec takes the binary representation of an integer and outputs k integers which are also represented in binary.*
2. $\mathsf{Dec}(u) = \perp$ *if $u \neq \langle \vec{n}\rangle$ for any $\vec{n} \in \mathbb{N}^*$.*

Proof. Given k integers $n_1, n_2, \ldots, n_k$, the encoding algorithm Enc first obtains the binary string

$$z := \alpha_1 \circ 01 \circ \alpha_2 \circ 01 \circ \alpha_k \circ 01 \circ \mathsf{bin}(n_1) \circ \mathsf{bin}(n_2) \circ \cdots \circ \mathsf{bin}(n_k),$$

where for each $i \in [k]$, α_i is obtained from $\mathsf{bin}(|\mathsf{bin}(n_i)|)$ with every bit doubled. Then Enc outputs the integer n whose binary representation is z. It is easy to see that given the binary representation of n, which is simply the string z, one can recover the $n_1, n_2, \ldots, n_k$ in time polynomial in $|z|$. Also, item (a) can be verified by a simple calculation. ◀

We give an example of how we will typically use Proposition 16. Suppose we have some language L that takes $k = 3$ parameters, then we may define a family of distributions $\{D_n\}_n$, where each D_n does the following.

1. On input 1^n, if $\mathsf{Dec}(n) \neq n_1, n_2, n_3$ for any $n_1, n_2, n_3 \in \mathbb{N}$, then output 0^n.
2. Otherwise, sample $x \sim \{0,1\}^{n_1}$, $y \sim \{0,1\}^{n_2}$, and let $z := 1^{n_3}$.
3. Output $\left(x, y, z, 0^{n - \sum_{i=1}^{3}(2\cdot(\lfloor \log n_i \rfloor + 2) + n_i)}\right)$.

Note that D_n runs in time $\mathsf{poly}(n)$, and the output length (using Equation (3)) is

$$\left(n - \sum_{i=1}^{3}(2 \cdot (\lfloor \log n_i \rfloor + 2) + n_i)\right) + \sum_{i=1}^{3}(2 \cdot (\lfloor \log n_i \rfloor + 2) + n_i) = n.$$

Also, note that the above distribution D_n outputs instances with $k+1$ parameters, where the last parameter is just some padding so that the output is of length n. To cope with this, instead of the language L, we will then work with a padded language L' which takes $k+1$ parameters and L' simulates L by ignoring the last parameter. In particular, for some $\ell_1, \ell_2, \ell_3 \in \mathbb{N}$ of interest, we can use $D_{\langle \ell_1, \ell_2, \ell_3 \rangle}$ to generate instances of the form (x, y, z, g) where $x \in \{0,1\}^{\ell_1}, y \in \{0,1\}^{\ell_1}, z \in \{0,1\}^{\ell_3}$, and the length of such instances is $O\big(\Pi_{i=1}^{3}\ell_i \cdot \mathsf{polylog}(\ell_i)\big) = \widetilde{O}(\ell_1 \cdot \ell_2 \cdot \ell_3)$.

3 Probabilistic Time-Bounded Kolmogorov Complexity

In this section, we formally define probabilistic Kolmogorov complexity and prove some useful properties of this notion, which will be used later in proving our main results.

▶ **Definition 17** (pK^t). *For strings $x, y \in \{0,1\}^*$, a time bound $t \in \mathbb{N}$, an oracle A, and $\delta \in [0,1]$, the δ-probabilistic A-oracle t-bounded Kolmogorov complexity of x given y is defined as*

$$\mathsf{pK}^{t,A}_{\delta}(x \mid y) := \min\left\{k \;\middle|\; \Pr_{w \sim \{0,1\}^t}\left[\exists\, \mathcal{M} \in \{0,1\}^k,\ \mathcal{M}^A(w, y) \text{ outputs } x \text{ within } t \text{ steps}\right] \geq \delta\right\},$$

where $\mathcal{M}$ is a RAM-machine.[18] *We omit the subscript δ if $\delta = 2/3$, omit the superscript A if $A = \varnothing$, omit "$\mid y$" if y is the empty string, and omit the superscript t if $t = \infty$.*

For simplicity, in the rest of this section we only consider pK^t without oracles. It is easy to see that all the results also hold with any oracle.

[18] We use the RAM model in this definition for convenience: it simplifies the time bound estimates for a machine $\mathcal{M}$ that takes as input both randomness w and an "auxiliary" string y, and allows both w and y to be as long as the running time of the RAM-machine $\mathcal{M}$; cf. [29] for a similar definition in the case of deterministic time-bounded conditional Kolmogorov complexity. Alternatively, we could use a TM model, with inputs w and y presented on two different tapes. However, we prefer the RAM model as it's also useful in the context of fine-grained worst-case to average-case reductions [6].

3.1 Basic properties of pK^t complexity

▶ **Lemma 18.** *There is a universal constant $c > 0$ such that the following holds. For every time bound $t \in \mathbb{N}$ and $x \in \{0,1\}^n$,*

$$\mathsf{K}(x \mid t) \leq \mathsf{pK}^t(x) + c \log n.$$

Proof. Let $\ell := \mathsf{pK}^t(x)$. Since x is of length n, we can represent ℓ using $O(\log n)$ bits. Given t, ℓ and n, it is possible to *sample* x with probability $\mu \geq 2/3 \cdot 2^{-\ell}$ by randomly guessing $w \sim \{0,1\}^t$ and $\mathcal{M} \sim \{0,1\}^\ell$ then simulating $\mathcal{M}(w)$ for t steps. Consequently, by the coding theorem for (time-unbounded) Kolmogorov complexity [24], we get that $\mathsf{K}(x \mid t) \leq \log(1/\mu) + O(\log n) \leq \mathsf{pK}^t(x) + O(\log n)$. ◀

▶ **Proposition 19.** *For any string $x \in \{0,1\}^*$, time bound $t \in \mathbb{N}$, and $\delta \in [0,1]$, we have*

$$\mathsf{pK}^t_\delta(x) \leq \mathsf{rK}^t_\delta(x) \leq \mathsf{K}^t(x).$$

Proof. Immediate from the definition of the involved Kolmogorov complexity measures. ◀

▶ **Lemma 20** (Probabilistic Incompressibility)**.** *For any string $y \in \{0,1\}^*$, time bound $t \in \mathbb{N}$ (including $t = \infty$), $\delta \in (0,1]$, and positive integer α, we have*

$$\Pr_{x \sim \{0,1\}^n}\left[\mathsf{pK}^t_\delta(x \mid y) \leq n - \alpha\right] \leq \frac{1}{2^{\alpha-1} \cdot \delta}.$$

Proof. For the sake of contradiction, suppose the statement of the lemma does not hold. Then by the definition of pK^t_δ, we have

$$\begin{aligned}&\mathop{\mathbf{E}}_{\substack{x \sim \{0,1\}^n \\ w \sim \{0,1\}^t}}\left[\exists\, \mathcal{M}_{(x,w)} \in \{0,1\}^{\leq n-\alpha} \text{ such that } \mathcal{M}_{(x,w)}(w,y) \text{ outputs } x \text{ within } t \text{ steps}\right] \\ &> \frac{1}{2^{\alpha-1}}.\end{aligned}$$

By averaging, there is a way to fix $w \in \{0,1\}^t$ that at least preserves this expectation, yielding

$$\Pr_{x \sim \{0,1\}^n}\left[\exists\, \mathcal{M}_{(x,w)} \in \{0,1\}^{\leq n-\alpha} \text{ such that } \mathcal{M}_{(x,w)}(w,y) \text{ outputs } x \text{ within } t \text{ steps}\right] > \frac{1}{2^{\alpha-1}}.$$

However, by counting, the above probability is at most $2^{n-\alpha+1}/2^n = 1/2^{\alpha-1}$. A contradiction. ◀

▶ **Lemma 21** (Success Amplification)**.** *For any string $x \in \{0,1\}^n$, time bound $t \in \mathbb{N}$, and $0 \leq \alpha < \beta \leq 1$, we have*

$$\mathsf{pK}^{O(qt/\alpha)}_\beta(x) \leq \mathsf{pK}^t_\alpha + O(\log(q/\alpha)),$$

where $q = \ln(1/(1-\beta))$.

Proof. Suppose $\mathsf{pK}^t_\alpha(x) = k$. Then by definition, we have

$$\Pr_{w \sim \{0,1\}^t}\left[\exists\, \mathcal{M}_w \in \{0,1\}^k \text{ such that } \mathcal{M}_w(w) \text{ outputs } x \text{ within } t \text{ steps}\right] \geq \alpha.$$

Call such a w *good*. After sampling ℓ independent $w^{(1)}, w^{(2)}, \ldots, w^{(\ell)} \in \{0,1\}^t$, the probability that no $w^{(i)}$ is good is at most $(1-\alpha)^\ell \leq e^{-\alpha\ell}$, which can be made at most $1-\beta$ by choosing $\ell = q/\alpha$. It follows that, with probability at least β over a random $w = \left(w^{(1)}, w^{(2)}, \ldots, w^{(q/\alpha)}\right) \in \{0,1\}^{qt/\alpha}$, there exists an index $1 \leq i \leq q/\alpha$ (described with at most $\lceil \log(q/\alpha) \rceil$ bits) and a program $\mathcal{M}_{w^{(i)}} \in \{0,1\}^k$ such that $\mathcal{M}_{w^{(i)}}\left(w^{(i)}\right)$ outputs x within $O(qt/\alpha)$ steps. Hence, $\mathsf{pK}^{O(qt/\alpha)}_\beta(x) \leq k + O(\log(q/\alpha))$. ◀

Note that the parameter α affects the time complexity in the resulting $\mathsf{pK}^{O(qt/\alpha)}_{\beta}$ bound. Lemma 21 is sufficient in applications where $\alpha \geq 1/\mathsf{poly}(n)$.

3.2 Bounding pK^t and rK^t via DPG reconstruction

The following is a key lemma for pK^t that we will use in proving our main results related to randomized average-case complexity.

▶ **Lemma 22** (pK^t Reconstruction Lemma). *For $\varepsilon > 0$, $x \in \{0,1\}^n$, $s \in \mathbb{N}$, and $k \in \mathbb{N}$ satisfying $k \leq 2n$, let D be a randomized algorithm that takes an advice string β and runs in time t_D such that D ε-distinguishes $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Then there is a polynomial p_{DP} such that*

$$\mathsf{pK}^{\widetilde{O}(t_D)\cdot p_{\mathsf{DP}}(n/\varepsilon)}(x \mid \beta) \leq k + \log p_{\mathsf{DP}}(nt_D/\varepsilon).$$

Proof. We will view the distinguisher D as a deterministic algorithm that takes a string of t_D bits, denoted by r_D, as its internal randomness. By our assumption (and dropping the absolute value sign without loss of generality), we have

$$\mathop{\mathbf{E}}_{\substack{y\sim\{0,1\}^{nk}\\ r_D\sim\{0,1\}^{t_D}}}[D(\mathsf{DP}_k(x,y);r_D)] - \mathop{\mathbf{E}}_{\substack{z\sim\{0,1\}^{nk+k}\\ r_D\sim\{0,1\}^{t_D}}}[D(z;r_D)] \geq \varepsilon.$$

Then by an averaging argument, we have

$$\mathop{\mathbf{Pr}}_{r_D}\left[\mathop{\mathbf{E}}_{y}[D(\mathsf{DP}_k(x,y);r_D)] - \mathop{\mathbf{E}}_{z}[D(z;r_D)] \geq \varepsilon/2\right] \geq \varepsilon/2. \tag{4}$$

In other words, with probability at least $\varepsilon/2$ over the internal randomness r_D, $D(-,r_D)$ is an $(\varepsilon/2)$-distinguisher for $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ and $\mathcal{U}_{nk+k}$. Let us say that r_D is *good* if this is true.

Let $\mathsf{Recon}^{(-)}\colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^n$ be the reconstruction procedure in Lemma 14 that works for distinguishing parameter $\varepsilon/2$, where the randomness complexity r is $\mathsf{poly}(n/\varepsilon)$. We have

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{\substack{w\sim\{0,1\}^r\\ r_D\sim\{0,1\}^{t_D}}}\left[\exists\,\alpha\in\{0,1\}^k \text{ such that } \mathsf{Recon}^{D(-,r_D)}(\alpha,w)=x\right]\\
&\geq \mathop{\mathbf{Pr}}_{w,r_D}\left[\exists\,\alpha\in\{0,1\}^k \text{ such that } \mathsf{Recon}^{D(-,r_D)}(\alpha,w)=x \,\middle|\, r_D \text{ is good}\right]\cdot \mathop{\mathbf{Pr}}_{r_D}[r_D \text{ is good}]\\
&\geq \frac{1}{\mathsf{poly}(n/\varepsilon)}\cdot(\varepsilon/2). \qquad \text{(Lemma 14 and Equation (4))}
\end{aligned}$$

The above implies that for at least $1/\mathsf{poly}(n/\varepsilon)$ fraction of the randomness (w,r_D), there exists a program $\mathcal{M}$, which takes k bits for some α (that can depend on the randomness) and the advice of β used by D, such that $\mathcal{M}$ simulates $\mathsf{Recon}^{D(-,r_D)}$ on (α,w) and outputs x. Note that since Recon^D runs in time $t_D\cdot\mathsf{poly}(n/\varepsilon)$, our program $\mathcal{M}$ can be made to run in time $T := \widetilde{O}(t_D)\cdot\mathsf{poly}(n/\varepsilon)$. Therefore, we have

$$\mathsf{pK}^T_{1/\mathsf{poly}(n/\varepsilon)}(x \mid \beta) \leq k + O(\log(nt_D/\varepsilon)).$$

Then the lemma follows easily from Lemma 21. ◀

The following is an analog of Lemma 22 for the case of rK^t. In this case, we additionally assume the existence of a pseudodeterministic PRG (see Definition 8); later we will use Corollary 12 to argue the existence of a pseudodeterministic PRG from the assumption that $\mathsf{DistP}^{\Sigma_2^{\mathsf{P}}} \subseteq \mathsf{AvgBPP}$.

▶ **Lemma 23** (rK^t Reconstruction Lemma). *Assume the existence of a pseudodeterministic PRG. For $\varepsilon > 0$, $x \in \{0,1\}^n$, $s \in \mathbb{N}$, and $k \in \mathbb{N}$ satisfying $k \leq 2n$, let D be a randomized algorithm that takes an advice β and runs in time t_D such that D ε-distinguishes $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Then there is a polynomial p'_{DP} such that*

$$\mathsf{rK}^{p'_{\mathsf{DP}}(nt_D/\varepsilon)}(x \mid \beta) \leq k + \log p'_{\mathsf{DP}}(nt_D/\varepsilon).$$

Proof. By averaging, arguing as in Lemma 22 up to Equation (4),

$$\Pr_{r_D \sim \{0,1\}^s}\left[\mathbf{E}_y[D(\mathsf{DP}_k(x,y),r_D)] - \mathbf{E}_z[D(z,r_D)] \geq \varepsilon/2\right] \geq \varepsilon/2. \tag{5}$$

That is, with probability at least $\varepsilon/2$ over its internal randomness r_D, $D(-,r_D)$ is an $(\varepsilon/2)$-distinguisher for $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ and $\mathcal{U}_{nk+k}$. Let us say that r_D is *good* if this is true.

Let $\mathsf{Recon}^{(-)}\colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^n$ be the reconstruction procedure from Lemma 14 that works for distinguishing parameter $\varepsilon/2$. By definition of Recon, if r_D is good, then

$$\Pr_{w \sim \{0,1\}^r}\left[\mathsf{Recon}^{D(-,r_D)}(A(x,w),w) = x\right] \geq 1/\mathsf{poly}(n/\varepsilon),$$

and so

$$\Pr_{w,r_D}\left[\mathsf{Recon}^{D(-,r_D)}(A(x,w),w) = x\right] \geq \frac{\varepsilon}{2} \cdot \frac{1}{\mathsf{poly}(n/\varepsilon)} \stackrel{\text{def}}{=} \varepsilon'.$$

Observe that by Lemma 14, the condition $\mathsf{Recon}^{D(-,r_D)}(A(x,w),w) = x$ can be checked by a circuit of size $s = \mathsf{poly}(nt_D/\varepsilon)$ given (r_D, w) as input. Consider an assumed pseudodeterministic PRG

$$G : \{0,1\}^\ell \to \{0,1\}^{|r_D|+|w|}$$

that $(\varepsilon'/2)$-fools all circuits of size s, where $\ell \in O(\log(s/\varepsilon')) \subseteq O(\log(ns/\varepsilon))$. Let M_G be an algorithm as described in Definition 8, running in time $\mathsf{poly}(s/\varepsilon') \leq \mathsf{poly}(nt_D/\varepsilon)$, such that for all $\sigma \in \{0,1\}^\ell$,

$$\Pr_{r_M \sim \{0,1\}^{\mathsf{poly}(nt_D/\varepsilon)}}[M_G(\sigma, r_M) = G(\sigma)] \geq 1 - 2^{-s},$$

where r_M denotes the internal randomness of M_G. By definition of G,

$$\Pr_{\sigma}\left[\mathsf{Recon}^{D(-,G_0(\sigma))}(A(x,G_1(\sigma)),G_1(\sigma)) = x\right] \geq \varepsilon'/2,$$

where $G_0(\sigma) = G(\sigma)_{1\ldots|r_D|}$ (the $|r_D|$-length prefix) and $G_1(\sigma) = G(\sigma)_{|r_D|+1\ldots|r_D|+|w|}$ (the remaining suffix). So there must exist a seed $\sigma \in \{0,1\}^\ell$ such that

$$\mathsf{Recon}^{D(-,G_0(\sigma))}(A(x,G_1(\sigma)),G_1(\sigma)) = x.$$

Let σ be a seed with this property, and let $\alpha = A(x,G_1(\sigma)) \in \{0,1\}^k$. The definition of M_G implies that α and σ are such that

$$\Pr_{r_M}\left[\mathsf{Recon}^{D(-,M_{G_0}(\sigma,r_M))}(\alpha, M_{G_1}(\sigma,r_M)) = x\right] \geq 1 - 2^{-s} > 2/3,$$

where $M_{G_0}(\sigma,r_M)$ denotes the $|r_D|$-length prefix of $M_G(\sigma,r_M)$, and $M_{G_1}(\sigma,r_M)$ the remaining suffix of $M_G(\sigma,r_M)$. It is easy to verify that the reconstruction procedure $\mathsf{Recon}^{D(-,M_{G_0}(\sigma,r_M))}(\alpha, M_G(\sigma,-))$ runs in time $\mathsf{poly}(nt_D/\varepsilon)$ overall. By definition of rK^t, for some polynomial p'_{DP}, we get

$$\begin{aligned}\mathsf{rK}^{p'_{\mathsf{DP}}(nt_D/\varepsilon)}(x \mid \beta) &\leq |\alpha| + |\sigma| + O(\log(ns/\varepsilon)) \\ &\leq k + \log p'_{\mathsf{DP}}(nt_D/\varepsilon),\end{aligned}$$

as required. ◀

3.3 Optimal source coding for pK^t

We will need the following result, which is an easy extension of an unconditional source coding theorem for pK^t described in [34].

▶ **Lemma 24** (Unconditional Source Coding for pK^t [34]). *There exists a polynomial p such that for any $T, a\colon \mathbb{N} \to \mathbb{N}$, $n \in \mathbb{N}$, $D_n \in \mathsf{Samp}[T(n)]/a(n)$, and $x \in \mathsf{Supp}(D_n)$,*

$$\mathsf{pK}^{p(T(n))}(x) \leq \log(1/D_n(x)) + O(\log(T(n))) + a(n).$$

For a distribution D_n supported over X and an integer $m \geq 1$, we let $D_n^m := D_n \times \ldots \times D_n$ be the probability distribution supported over X^m obtained by taking the product of m independent copies of D_n.

As a consequence of Lemma 24, we obtain the following result.

▶ **Lemma 25** (Source Coding for m copies of D_n). *There exists a polynomial p such that for any $T, a\colon \mathbb{N} \to \mathbb{N}$, $n, m \in \mathbb{N}$, $D_n \in \mathsf{Samp}[T(n)]/a(n)$, and $x \in \mathsf{Supp}(D_n^m)$,*

$$\mathsf{pK}^{p(T(n),m)}(x) \leq \log(1/D_n^m(x)) + O(\log(T(n))) + a(n) + O(\log(m)).$$

Proof. Since D_n can be sampled in time $T(n)$ using $a(n)$ bits of advice, it is possible to sample from D_n^m in time $\mathsf{poly}(T(n), m)$ using $a(n) + O(\log m)$ bits of advice, where the extra advice is used to encode m. The result now immediately follows from Lemma 24. ◀

3.4 Symmetry of information under randomized average-case easiness

Recently, [7], extending a technique from [11], have proved a symmetry of information result for the deterministic version of time-bounded Kolmogorov complexity K^t under the average-case easiness assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. We adapt and generalize their argument to prove an analogous result for conditional pK^t (and rK^t) under the randomized average-case easiness assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ (respectively, $\mathsf{DistP}^{\Sigma_2^{\mathrm{p}}} \subseteq \mathsf{AvgBPP}$).

▶ **Lemma 26** (Symmetry of information for pK^t and rK^t under the average-case easiness of NP).

1. *If $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then there exist polynomials p and p_0 such that for all sufficiently large $n, m \in \mathbb{N}$, all $t \geq p_0(n, m)$, and all $0 \leq \tau \leq t$ the following holds: for every $x \in \{0,1\}^n$ and every family $\{y_r \in \{0,1\}^m \mid r \in \{0,1\}^\tau\}$, with probability at least $9/10$ over $r \in \{0,1\}^\tau$,*

$$\mathsf{pK}^t(x, y_r \mid r) > \mathsf{pK}^{p(t)}(x \mid r) + \mathsf{pK}^{p(t)}(y_r \mid x, r) - \log p(t).$$

2. *If $\mathsf{DistP}^{\Sigma_2^{\mathrm{p}}} \subseteq \mathsf{AvgBPP}$, then there exist polynomials p' and p'_0 such that for all sufficiently large $n, m \in \mathbb{N}$, all $t \geq p'_0(n, m)$, and all $0 \leq \tau \leq t$ the following holds: for every $x \in \{0,1\}^n$ and every family $\{y_r \in \{0,1\}^m \mid r \in \{0,1\}^\tau\}$, with probability at least $9/10$ over $r \in \{0,1\}^\tau$,*

$$\mathsf{rK}^t(x, y_r \mid r) > \mathsf{rK}^{p'(t)}(x \mid r) + \mathsf{rK}^{p'(t)}(y_r \mid x, r) - \log p'(t).$$

Moreover, the special case without conditioning on r also holds for both items above, i.e., when r is the empty string (for $\tau = 0$) and y is an arbitrary single string.

Proof of Item 1. Define a language

$$L := \left\{(u, v, w, w', 1^s) \mid \exists\, \mathcal{M} \in \{0,1\}^s, \mathcal{M}(w, w') \text{ prints } uv \text{ in } |w| \text{ steps, and } s = |u| + |v| - 10\right\}.$$

Note that $L \in \mathsf{NP}$. Define a distribution family $D = \{D_{\langle nk+k, mk'+k', 2t, \tau, s\rangle}\}$ as follows: sample $u \sim \mathcal{U}_{nk+k}$, $v \sim \mathcal{U}_{mk'+k'}$, $w \sim \mathcal{U}_{2t}$, and $w' \sim \mathcal{U}_{\tau}$, and then output $(u, v, w, w', 1^s)$. Note that the ensemble D is in PSamp under an appropriate encoding of its defining tuple (for simplicity, we omit the padding in the output of D; see Section 2.1). Using the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, it follows that $(L, D) \in \mathsf{AvgBPP}$. Let B be a randomized algorithm for (L, D) as in Lemma 6.

Let $x \in \{0,1\}^n$ and $\{y_r\}_r \subseteq \{0,1\}^m$ be given (for sufficiently large n and m). Let $k, k' \in \mathbb{N}$ be arbitrary parameters such that $k \leq n$ and $k' \leq m$. Observe that there exists a polynomial p_0 such that for any $t \geq p_0(n, m)$, some constant d, and all $z \in \{0,1\}^{nk}$, $z' \in \{0,1\}^{mk'}$, and $r \in \{0,1\}^{\tau}$,

$$\mathsf{pK}^{2t}(\mathsf{DP}_k(x; z), \mathsf{DP}_{k'}(y_r; z') \mid r) \leq \mathsf{pK}^{t}(x, y_r \mid r) + |z| + |z'| + d \log t. \tag{6}$$

Here $p_0(n, m)$ reflects the time required to deterministically compute $(\mathsf{DP}_k(x; z), \mathsf{DP}_{k'}(y_r; z'))$ given x, y_r, z, z', and $d \log t$ bits of information to delineate x from y_r.

Let $t \geq p_0(n, m)$. By a counting argument, for independently random u, v, w, and w',

$$\mathop{\mathbf{Pr}}_{u,v,w,w'}[(u, v, w, w', 1^s) \in L] \leq \frac{2^s \cdot 2^{|w|+|w'|}}{2^{|u|+|v|+|w|+|w'|}} = 2^{s-|u|-|v|} = 2^{-10},$$

where the last equality is by the definition of L requiring that $s = |u| + |v| - 10$. Then

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{u,v,w,w',r_B}[B(u, v, w, w', 1^s) = 1] \\
&\leq \mathop{\mathbf{Pr}}_{u,v,w,w'}[(u, v, w, w', 1^s) \in L] + \mathop{\mathbf{Pr}}_{u,v,w,w',r_B}[B(u, v, w, w', 1^s) \neq L(u, v, w, w', 1^s)] \\
&\leq 2^{-10} + (3/n).
\end{aligned} \tag{7}$$

By Markov's inequality, for at least $9/10$ fraction of random strings $r \in \{0,1\}^{\tau}$, we get

$$\mathop{\mathbf{Pr}}_{u,v,w,r_B}[B(u, v, w, r, 1^s) = 1] \leq 10 \cdot \left(2^{-10} + (3/n)\right) \leq 1/10. \tag{8}$$

Fix any such r so that Equation (8) holds. This also fixes the string y_r.

Next we show, using a hybrid argument, that $B(-, \mathcal{U}_{2t}, r, 1^s)$ *cannot* distinguish between the uniform distribution and the distribution $(\mathsf{DP}_k(x; z), \mathsf{DP}_{k'}(y_r; z'))$, for random independent z, z', where $k \approx \mathsf{pK}^{p_{\mathsf{DP}}(t)}(x \mid r)$ and $k' \approx \mathsf{pK}^{p_{\mathsf{DP}}(t)}(y_r \mid x, r)$. This will imply that

$$\mathop{\mathbf{Pr}}_{w}[(\mathsf{DP}_k(x; z), \mathsf{DP}_{k'}(y_r; z'), w, r, 1^s) \in L] < 2/3$$

for some z, z', yielding the desired lower bound on $\mathsf{pK}^{2t}(\mathsf{DP}_k(x; z), \mathsf{DP}_{k'}(y_r; z') \mid r)$ via our choice of s and the definition of L. We give the details of the hybrid argument next.

First, toward a contradiction, suppose

$$\mathop{\mathbf{Pr}}_{z,v,w,r_B}[B(\mathsf{DP}_k(x; z), v, w, r, 1^s) = 1] > 1/2.$$

In this case, comparing with Equation (8), we get a randomized distinguisher (with advice) for $\mathsf{DP}_k(x; \mathcal{U}_{nk})$, defined by sampling $v \sim \mathcal{U}_{mk'+k'}$ and $w \sim \mathcal{U}_{2t}$, and outputting $B(-, v, w, r, 1^s)$. By Lemma 22,

$$\mathsf{pK}^{q(t)}(x \mid r) \leq k + \log q(t) \tag{9}$$

for some polynomial q such that $q(t) \geq p_{\mathsf{DP}}(t_B \cdot 3 \cdot n)$ whenever $t \geq n + m$ and $s \leq n^3$, where p_{DP} is the polynomial from Lemma 22 and t_B denotes the time required to compute $B(-, v, w, r, 1^s)$.

Define $k := \mathsf{pK}^{q(t)}(x \mid r) - \log q(t) - 1$ so that Equation (9) *does not hold*. Assume for now that $k > 0$. Hence,

$$\Pr_{z,v,w,r_B}[B(\mathsf{DP}_k(x;z), v, w, r, 1^s) = 1] \leq 1/2. \tag{10}$$

Again, toward a contradiction, suppose that for *all* z, z',

$$\Pr_w[(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z'), w, r, 1^s) \in L] \geq 2/3.$$

By definition of B, this implies that

$$\Pr_{z,z',w,r_B}[B(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z'), w, r, 1^s) = 1] \geq (2/3)(1-1/n) > 5/8.$$

In this case, comparing with Equation (10), we get a randomized distinguisher (with advice) B' for $\mathsf{DP}_{k'}(y_r;\mathcal{U}_{mk'})$, defined by sampling $z \sim \mathcal{U}_{nk}$, $w \sim \mathcal{U}_{2t}$, and outputting $B(\mathsf{DP}_k(x;z), -, w, r, 1^s)$. By Lemma 22,

$$\mathsf{pK}^{q'(t)}(y_r \mid x, r) \leq k' + \log q'(t) \tag{11}$$

for some polynomial q' with $q'(t) \geq p_{\mathsf{DP}}(t_{B'} \cdot 8 \cdot m)$ whenever $t \geq n+m$ and $s \leq n^3$, where $t_{B'}$ denotes the time required to compute B'.

We now choose $k' := \mathsf{pK}^{q'(t)}(y_r \mid x, r) - \log q'(t) - 1$ so that Equation (11) does not hold. Assume for now that $k' > 0$. Hence, there exist z and z' such that

$$\begin{aligned}&\Pr_w\left[(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z'), w, r, 1^s) \in L\right] \\ &= \Pr_w\left[\exists \mathcal{M} \in \{0,1\}^s, \mathcal{M}(w,r) \text{ outputs } (\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z')) \text{ within } |w| \text{ steps}\right] < 2/3,\end{aligned}$$

which implies that

$$\mathsf{pK}^{2t}(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z') \mid r) > s.$$

For this choice of z, z', by definition of s we get that $s = |u|+|v|-10 = |z|+k+|z'|+k'-10$, and so

$$\mathsf{pK}^{2t}(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z') \mid r) > |z| + k + |z'| + k' - 10.$$

Combining this inequality with Equation (6), we get

$$\begin{aligned}\mathsf{pK}^t(x, y_r \mid r) &\geq \mathsf{pK}^{2t}(\mathsf{DP}_k(x;z), \mathsf{DP}_{k'}(y_r;z') \mid r) - |z| - |z'| - d\log t \\ &> k + k' - d\log t - 10.\end{aligned} \tag{12}$$

The definitions of k and k' then imply that

$$\mathsf{pK}^t(x, y_r \mid r) > \mathsf{pK}^{q(t)}(x \mid r) + \mathsf{pK}^{q'(t)}(y_r \mid x, r) - \log q(t) - \log q'(t) - d\log t - 12.$$

For the polynomial $p(t) := (t^{d+1}) \cdot q(t) \cdot q'(t)$ and for every $t \geq p_0(|x|, |y|)$, we get

$$\mathsf{pK}^t(x, y_r \mid r) > \mathsf{pK}^{p(t)}(x \mid r) + \mathsf{pK}^{p(t)}(y_r \mid x, r) - \log p(t),$$

as desired.

Finally, consider the case that $k \leq 0$ or $k' \leq 0$. If $k \leq 0$, then $\mathsf{pK}^{q(t)}(x \mid r) \leq \log q(t) + 1$, implying that $\mathsf{pK}^{p(t)}(x \mid r) < \log p(t)$. But then the lemma simply follows from the fact that $\mathsf{pK}^t(x, y_r \mid r) \geq \mathsf{pK}^{p(t)}(y_r \mid x, r)$. Similarly, if $k' \leq 0$, then $\mathsf{pK}^{p(t)}(y_r \mid x, r) < \log p(t)$, and the lemma follows from the fact that $\mathsf{pK}^t(x, y_r \mid r) \geq \mathsf{pK}^{p(t)}(x \mid r)$. ◀

Proof of Item 2. Argue as in the proof of Item 1, but at Equation (9), apply Lemma 23 instead of Lemma 22 to get

$$\mathsf{rK}^{q(t)}(x \mid r) \leq k + \log q(t)$$

for some polynomial q such that $q(t) \geq p'_{\mathsf{DP}}(t_B \cdot 3 \cdot n)$, where p'_{DP} is the polynomial from Lemma 23 and t_B denotes the time required to compute $B(-, v, w, r, 1^s)$. We then define $k := \mathsf{rK}^{q(t)}(x \mid r) - \log q(t) - 1$ so the above equation does not hold. Similarly, at Equation (11), apply Lemma 23 to get

$$\mathsf{rK}^{q'(t)}(y_r \mid x, r) \leq k' + \log q'(t)$$

for some polynomial q' with $q'(t) \geq p'_{\mathsf{DP}}(t_{B'} \cdot 8 \cdot m)$, where $t_{B'}$ denotes the time required to compute the distinguisher B' defined as above. Then define $k' := \mathsf{rK}^{q'(t)}(y_r \mid x, r) - \log q'(t) - 1$.

Following the proof of Item 1 up to Equation (12), we get that

$$\mathsf{pK}^t(x, y_r \mid r) > k + k' - d \log t - \log n.$$

The definitions of k, k', and rK^t then imply that

$$\mathsf{rK}^t(x, y_r \mid r) > \mathsf{rK}^{q(t)}(x \mid r) + \mathsf{rK}^{q'(t)}(y_r \mid x, r) - \log q(t) - \log q'(t) - d \log t - 12,$$

and so

$$\mathsf{rK}^t(x, y_r \mid r) > \mathsf{rK}^{p(t)}(x \mid r) + \mathsf{rK}^{p(t)}(y_r \mid x, r) - \log p(t)$$

for the polynomial $p(t) := (t^{d+1}) \cdot q(t) \cdot q'(t)$. ◀

3.5 Approximating pK^t under randomized average-case easiness

▶ **Lemma 27.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then there exists a polynomial* τ *such that the following promise problem is in* $\mathsf{promiseBPP}$*:*

$$\Pi_{\mathsf{YES}} := \left\{ \left(x, 1^s, 1^t\right) \mid \mathsf{pK}^t(x) \leq s \right\},$$
$$\Pi_{\mathsf{NO}} := \left\{ \left(x, 1^s, 1^t\right) \mid \mathsf{pK}^{\tau(t)}(x) > s + \log \tau(t) \right\}.$$

Proof. Let $k = s + \log n$. Consider the language

$$L := \{(\mathsf{DP}_k(x; z), w, 1^s, 1^n) \mid \exists \mathcal{M} \in \{0,1\}^s,\ \mathcal{M}(w) \text{ outputs } x \in \{0,1\}^n \text{ within } |w| \text{ steps}\}.$$

Note that $L \in \mathsf{NP}$. Define a distribution family $D = \{D_{\langle nk+k,t,s,n\rangle}\}$, each member of which does the following: sample $u \sim \mathcal{U}_{nk+k}$ and $w \sim \mathcal{U}_t$, and then output $(u, w, 1^s, 1^n)$. By assumption, $(L, D) \in \mathsf{AvgBPP}$. Let B be a randomized heuristic algorithm for (L, D) as described in Lemma 6.

Now, define an algorithm B':

> On input $(x, 1^s, 1^t)$ with $x \in \{0,1\}^n$, sample $z \sim \mathcal{U}_{nk}$ and $w \sim \mathcal{U}_t$, and then output $B(\mathsf{DP}_k(x; z), w, 1^s, 1^n)$.

Below, we show that B' solves $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ correctly with high probability in the worst case.

First, consider the case that $(x, 1^s, 1^t) \in \Pi_{\mathsf{YES}}$. By the definitions of L and pK, for any choice of $z \in \{0,1\}^{nk}$,

$$\Pr_w [(\mathsf{DP}_k(x; z), w, 1^s, 1^n) \in L] \geq 2/3.$$

The definition of B then implies that

$$\Pr_{w,z,r_B} [B(\mathsf{DP}_k(x; z), w, 1^s, 1^n) = 1] > 1/2,$$

and so

$$\Pr_{r_{B'}} \left[B'(x, 1^s, 1^t) = 1\right] > 1/2, \tag{13}$$

where r_B denotes the internal randomness of B, and $r_{B'} = (w, z, r_B)$ that of B'.

Now consider the case that $(x, 1^s, 1^t) \in \Pi_{\mathsf{NO}}$. For a contradiction, suppose that

$$\Pr_{w,z,r_B} [B(\mathsf{DP}_k(x; z), w, 1^s, 1^n) = 1] > 1/3. \tag{14}$$

By a counting argument, for randomly selected u and w,

$$\Pr_{u,w} [(u, w, 1^s, 1^n) \in L] \leq \frac{2^s \cdot 2^{nk} \cdot 2^{|w|}}{2^{nk+k+|w|}} = \frac{1}{n},$$

where the last line follows from the definition of $k = s + \log n$. Then by definition of B,

$$\Pr_{u,w,r_B} [B(u, w, 1^s, 1^n) = 1] \leq 4/n = o(1). \tag{15}$$

Comparing Equations (14) and (15), it is clear that $B(-, \mathcal{U}_t, 1^s, 1^n)$ $(1/4)$-distinguishes $\mathsf{DP}_k(x; \mathcal{U}_{nk})$ from $\mathcal{U}_{nk+k}$. Lemma 22 implies that

$$\begin{aligned} \mathsf{pK}^{p'(t)}(x) &\leq k + \log p'(t) \\ &= s + \log n + \log p'(t), \end{aligned}$$

for some polynomial p' with $p'(t) \geq p_{\mathsf{DP}}(t_B \cdot 4 \cdot n)$, where p_{DP} is the polynomial from Lemma 22 and t_B denotes the time required to compute $B(-, \mathcal{U}_t, 1^s, 1^n)$. For the polynomial $\tau(t) = t \cdot p'(t)$, this means that $(x, 1^s, 1^t)$ is *not* in Π_{NO}, which gives the desired contradiction. By definition of B', we have that

$$\Pr_{r_{B'}} \left[B'(x, 1^s, 1^t) = 1\right] \leq 1/3. \tag{16}$$

By Equations (13) and (16), B' yields a $\mathsf{promiseBPP}$ algorithm for $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ via standard success amplification techniques. ◀

▶ **Lemma 28** ($\mathsf{Approximating\ pK}^t$). *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then there exist a polynomial* τ*, a constant* $C \geq 1$*, and a randomized algorithm* $\mathsf{Approx}_\tau\text{-}\mathsf{pK}$ *that, on input* $(x, 1^t)$ *where* $x \in \{0,1\}^n$ *and* $t \geq Cn$*, runs in time* $\mathsf{poly}(n, t)$ *and with probability at least* $1 - o(1)$ *outputs an integer* $\widetilde{s}$ *such that*

$$\mathsf{pK}^{\tau(t)}(x) - \log \tau(t) \ \leq \ \widetilde{s} \ \leq \ \mathsf{pK}^t(x).$$

Proof. Consider a polynomial-time randomized algorithm A that solves the promise problem from Lemma 27. Assume without loss of generality that on the inputs satisfying the promise its error is at most $1/n^2$, where $n = |x|$. Algorithm $\mathsf{Approx}_\tau\text{-}\mathsf{pK}$ runs A on $(x, 1^s, 1^t)$ for $s = 1, 2, \ldots, n + \log n$, and outputs the first $\widetilde{s}$ such that $A(x, 1^{\widetilde{s}}, 1^t) = 1$.

The correctness of $\mathsf{Approx}_\tau\text{-}\mathsf{pK}$ follows by a union bound. Indeed, if $s < \mathsf{pK}^{\tau(t)}(x) - \log \tau(t)$, i.e., $\mathsf{pK}^{\tau(t)}(x) > s + \log \tau(t)$, using the promise we get that $\Pr_A[A(x, 1^s, 1^t) = 1] \leq 1/n^2$. On the other hand, if $s = \mathsf{pK}^t(x)$, which implies that $\mathsf{pK}^t(x) \leq s$ and the promise is satisfied, we have $\Pr_A[A(x, 1^s, 1^t) = 1)] \geq 1 - 1/n^2$. Since $\mathsf{pK}^t(x) \leq n + \log n$ if $t \geq Cn$, where C is a sufficiently large constant, with high probability over the internal randomness of $\mathsf{Approx}_\tau\text{-}\mathsf{pK}$, it outputs a value $\widetilde{s}$ such that $\mathsf{pK}^{\tau(t)}(x) - \log \tau(t) \leq \widetilde{s} \leq \mathsf{pK}^t(x)$. ◀

3.6 Language compression under randomized average-case easiness

In some results that rely on language compression, it is useful to consider languages and promise problems that consist of strings of the form $(x, 1^\ell)$, where $|x| = \alpha(\ell)$ for some function α. More specifically, we need the following definition.

▶ **Definition 29** (Ensembles of Promise Problems). [19] *Let $\alpha\colon \mathbb{N} \to \mathbb{N}$. We say that a promise problem $\Pi = (\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ is an* ensemble of promise problems with input size α *if*

$$\Pi_{\mathsf{YES}} \cup \Pi_{\mathsf{NO}} \subseteq \left\{(x, 1^\ell) \mid \ell \in \mathbb{N} \text{ and } |x| = \alpha(\ell)\right\}.$$

In this case, we let

$$\Pi_{\mathsf{YES},\ell} := \left\{x \in \{0,1\}^{\alpha(\ell)} \mid (x, 1^\ell) \in \Pi_{\mathsf{YES}}\right\},$$

and similarly

$$\Pi_{\mathsf{NO},\ell} := \left\{x \in \{0,1\}^{\alpha(\ell)} \mid (x, 1^\ell) \in \Pi_{\mathsf{NO}}\right\}.$$

Below we show a language compression result for problems in $\mathsf{promiseAM}$. Before stating and proving this result, note that it is not immediately clear what it means to have language compression for such a *promise* class. A reasonable definition may be that for every $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}}) \in \mathsf{promiseAM}$ and every $x \in \Pi_{\mathsf{YES}} \cap \{0,1\}^n$, we have $\mathsf{pK}^{\mathsf{poly}(n)}(x) \lesssim \log |\Pi_{\mathsf{YES}} \cap \{0,1\}^n|$. However, it is unclear how to show such a strong theorem. On the other hand, we manage to show a weaker version which instead says $\mathsf{pK}^{\mathsf{poly}(n)}(x) \lesssim \log \left|\overline{\Pi_{\mathsf{NO}}} \cap \{0,1\}^n\right|$ for every $x \in \Pi_{\mathsf{YES}} \cap \{0,1\}^n$, and it turns out that such a language compression for $\mathsf{promiseAM}$ suffices in some applications (see Section 4.6).

▶ **Theorem 30** (Language compression for pK^t under average-case easiness of NP). *Let $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}}) \in \mathsf{promiseAM}$ be an ensemble of promise problems with input size $\alpha\colon \mathbb{N} \to \mathbb{N}$. Assume that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. Then there is a polynomial p such that for every $\ell \in \mathbb{N}$ and every $x \in \Pi_{\mathsf{YES},\ell}$,*

$$\mathsf{pK}^{p(\alpha(\ell)+\ell)}(x) \leq \log \left|\{0,1\}^{\alpha(\ell)} - \Pi_{\mathsf{NO},\ell}\right| + \log p(\alpha(\ell) + \ell).$$

[19] In the original definition of ensemble languages from [11], then length of x is not fixed and is required to be less than $\ell^{O(1)}$. For simplicity, here we require that this length is fixed according to ℓ.

Proof. Let $c > 0$ be a constant and V be a (deterministic) polynomial-time algorithm such that

$$(x, 1^\ell) \in \Pi_{\mathsf{YES}} \Rightarrow \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^{(\alpha(\ell)+\ell)^c}} \left[\exists\, y \in \{0,1\}^{(\alpha(\ell)+\ell)^c}, V\big(x, 1^\ell, y, r\big) = 1 \right] \geq 1 - 1/2^{\alpha(\ell)},$$

$$(x, 1^\ell) \in \Pi_{\mathsf{NO}} \Rightarrow \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^{(\alpha(\ell)+\ell)^c}} \left[\forall\, y \in \{0,1\}^{(\alpha(\ell)+\ell)^c}, V\big(x, 1^\ell, y, r\big) = 0 \right] \geq 1 - 1/2^{\alpha(\ell)},$$

Define the language

$$L' := \Big\{ \Big(\mathsf{DP}_k(x; z), r, 1^\ell, 1^{\alpha(\ell)} \Big) \;\Big|\; |x| = \alpha(\ell), |z| = \alpha(\ell)k, |r| = (\alpha(\ell)+\ell)^c \text{ and } \exists\, y \in \{0,1\}^{(\alpha(\ell)+\ell)^c} \text{ such that } V\big(x, 1^\ell, y, r\big) = 1 \Big\}.$$

Note that $L' \in \mathsf{NP}$. Define a distribution family $D = \left\{ D_{\langle \alpha(\ell)k+k, (\alpha(\ell)+\ell)^c, \ell, \alpha(\ell) \rangle} \right\}$, each of which does the following: sample $u \sim \mathcal{U}_{\alpha(\ell)k+k}$, $r \sim \mathcal{U}_{(\alpha(\ell)+\ell)^c}$, and output $\big(u, r, 1^\ell, 1^{\alpha(\ell)}\big)$. By assumption, $(L', D) \in \mathsf{AvgBPP}$. Let B be a randomized heuristic algorithm for (L', D) as described in Lemma 6.

Consider any $\ell \in \mathbb{N}$ and any $x \in \Pi_{\mathsf{YES},\ell}$. Note that for any $z \in \{0,1\}^{\alpha(\ell)k}$,

$$\mathop{\mathbf{Pr}}_{r} \left[\Big(\mathsf{DP}_k(x; z), r, 1^\ell, 1^{\alpha(\ell)} \Big) \in L' \right] \geq 2/3.$$

Then by property of B,

$$\mathop{\mathbf{Pr}}_{z, r, r_B} \left[B\Big(\mathsf{DP}_k(x; z), r, 1^\ell, 1^{\alpha(\ell)} \Big) = 1 \right] \geq 1/2. \tag{17}$$

On the other hand, for u and r selected uniformly at random, by a counting argument,

$$\begin{aligned} \mathop{\mathbf{Pr}}_{u, r} \left[\Big(u, r, 1^\ell, 1^{\alpha(\ell)} \Big) \in L' \right] &\leq \frac{2^{\alpha(\ell)k} \cdot 2^{(\alpha(\ell)+\ell)^c} \cdot \Big(\Big| \{0,1\}^{\alpha(\ell)} - \Pi_{\mathsf{NO},\ell} \Big| + |\Pi_{\mathsf{NO},\ell}| / 2^{\alpha(\ell)} \Big)}{2^{\alpha(\ell)k+k} \cdot 2^{(\alpha(\ell)+\ell)^c}} \\ &\leq 2^{-3}, \end{aligned}$$

where the last equality holds if we set $k := \log \Big| \{0,1\}^{\alpha(\ell)} - \Pi_{\mathsf{NO},\ell} \Big| + 4$. Then again by property of B,

$$\mathop{\mathbf{Pr}}_{u, r, r_B} \left[B\Big(u, r, 1^\ell, 1^{\alpha(\ell)} \Big) = 1 \right] \leq 1/10.$$

Comparing with Equation (17), it is clear that $B\big(-, \mathcal{U}_{(\alpha(\ell)+\ell)^c}, 1^\ell, 1^{\alpha(\ell)}\big)$ is a randomized distinguisher for $\mathsf{DP}_k\big(x; \mathcal{U}_{\alpha(\ell)k}\big)$. Lemma 22 implies that

$$\begin{aligned} \mathsf{pK}^{\mathsf{poly}(\alpha(\ell)+\ell)}(x) &\leq k + O(\log(\alpha(\ell)+\ell)) \\ &\leq \log \Big| \{0,1\}^{\alpha(\ell)} - \Pi_{\mathsf{NO},\ell} \Big| + O(\log(\alpha(\ell)+\ell)), \end{aligned}$$

as desired. ◀

3.7 Probabilistic computational depth

The next lemma is a straightforward adaptation of an argument from [11]. We provide a proof for completeness.

▶ **Lemma 31** (Computational Depth Upper Bound). *For all $x \in \{0,1\}^n$ and $r \in \{0,1\}^*$, and for all non-decreasing polynomials p_0 and p, if n is large enough then*

1. *there exists a time bound t_1 such that $p_0(n) \leq t_1 \leq 2^{n/\log n}$ and*

$$\mathsf{pK}^{t_1}(x \mid r) - \mathsf{pK}^{p(t_1)}(x \mid r) \leq O(n/\log n);$$

2. *there exists a time bound t_2 such that $p_0(n) \leq t_2 \leq 2^{n/\log n}$ and*

$$\mathsf{rK}^{t_2}(x \mid r) - \mathsf{rK}^{p(t_2)}(x \mid r) \leq O(n/\log n).$$

Proof. We will start by proving Item 1. Given $x \in \{0,1\}^n$ and polynomials p_0 and p, define the polynomial $\tau := p \circ p_0$. For an integer $I \geq 1$, consider the following telescoping sum:

$$\begin{aligned}
&\mathsf{pK}^{\tau(n)}(x \mid r) - \mathsf{pK}^{\tau^{I+1}(n)}(x \mid r) \\
&= \left(\mathsf{pK}^{\tau(n)}(x \mid r) - \mathsf{pK}^{\tau^2(n)}(x \mid r)\right) + \left(\mathsf{pK}^{\tau^2(n)}(x \mid r) - \mathsf{pK}^{\tau^3(n)}(x \mid r)\right) \\
&\quad + \cdots + \left(\mathsf{pK}^{\tau^I(n)}(x \mid r) - \mathsf{pK}^{\tau^{I+1}(n)}(x \mid r)\right),
\end{aligned}$$

where $\tau^i(-)$ denotes the composition of τ with itself i times. For any choice of x, p_0, and p as in the statement of the lemma, $\mathsf{pK}^{\tau(n)}(x \mid r) \leq n + d$, for some universal constant $d \geq 0$; hence, the above sum is at most $n + d$. By averaging, there is some index $i_0 \in [I]$ such that

$$\mathsf{pK}^{\tau^{i_0}(n)}(x \mid r) - \mathsf{pK}^{\tau^{i_0+1}(n)}(x \mid r) \leq \frac{n+d}{I}.$$

For this i_0, define $t_1 := \tau^{i_0}(n)$. Note that $t_1 \geq \tau(n) \geq p_0(n)$, since $i_0 \geq 1$ and $p(\ell) \geq \ell$ for every input ℓ. Letting $c \in \mathbb{N}$ be such that $\tau(n) \leq n^c$ for sufficiently large n, define $I := \log_c(n/(\log_2 n)^2)$. Then $t_1 \leq n^{c^I} = 2^{n/\log n}$. Moreover,

$$\begin{aligned}
\mathsf{pK}^{t_1}(x \mid r) - \mathsf{pK}^{p(t_1)}(x \mid r) &\leq \mathsf{pK}^{t_1}(x \mid r) - \mathsf{pK}^{\tau(t_1)}(x \mid r) \\
&\leq O(n/\log n),
\end{aligned}$$

as desired.

The proof of Item 2 is similar. ◀

4 Probabilistic Worst-Case to Average-Case Reductions

4.1 Auxiliary lemmas

Given a language $L \in \mathsf{NP}$ with a corresponding verifier V, and $x \in L$, let y_x be the lexicographically first witness that $x \in L$. It is not hard to compute y_x from x if an NP oracle is available, by performing a standard search-to-decision reduction. For this reason, the time-bounded Kolmogorov complexity of the pair (x, y_x) is essentially that of x, in the presence of an NP oracle. We will show, more generally, that the oracle can be eliminated under an appropriate average-case easiness assumption, even for the following randomized versions of Σ_ℓ^P, denoted $\mathsf{BP}_\delta \circ \Sigma_\ell^\mathsf{P}$.

▶ **Definition 32** ($\mathsf{BP}_\delta \circ \Sigma_\ell^\mathsf{P}$). *A language L is in the class $\mathsf{BP}_\delta \circ \Sigma_\ell^\mathsf{P}$ if there is a $\mathsf{BP} \circ \Sigma_\ell \circ \mathsf{P}$ formula*

$$\phi(x) = \mathsf{BP}r\ \exists y_1\ \forall y_2\ \ldots\ Qy_\ell\ \ R(x, y_1, y_2, \ldots, y_\ell, r)$$

for a polytime predicate R and all string variables $y_1, \ldots, y_\ell, r$ of length $\mathsf{poly}(|x|)$*, such that, for all $x \in \{0,1\}^n$,*

$$x \in L \implies \Pr_r [\exists\, y_1\ \forall\, y_2\ \ldots\ Q\, y_\ell\ \ R(x, y_1, y_2, \ldots, y_\ell, r)] \geq \delta, \text{ and}$$
$$x \notin L \implies \Pr_r \left[\forall\, y_1\ \exists\, y_2\ \ldots\ \bar{Q}\, y_\ell\ \ \neg R(x, y_1, y_2, \ldots, y_\ell, r)\right] \geq \delta.$$

▶ **Lemma 33** (Oracle Elimination)**.** *For any $\ell \geq 1$, suppose* $\mathsf{Dist}\Sigma^{\mathsf{P}}_{\ell+1} \subseteq \mathsf{AvgBPP}$*. For an arbitrary $L \in \mathsf{BP}_\delta \circ \Sigma^{\mathsf{P}}_\ell$, let $x \in L_n$ be sufficiently large, let $r \in \{0,1\}^{n^c}$ be any random string under the* BP *quantifier such that there exists a witness for the left-most $\exists$ quantifier in the definition of L, and let $y_{x,r} \in \{0,1\}^{n^c}$ be the lexicographically first such L-witness for x and randomness r, for some constant $c > 0$.*

1. *There exist polynomials q and q_0 such that, for all sufficiently large $n \in \mathbb{N}$, all $x \in L_n$, and all $t \geq q_0(n)$, we have with probability at least $\delta - 1/10$ over $r \in \{0,1\}^{n^c}$ that an L-witness $y_{x,r}$ exists, and*

$$\mathsf{pK}^{q(t)}(x, y_{x,r} \mid r) \leq \mathsf{pK}^t(x \mid r) + \log q(t).$$

2. *Assume in addition that* $\mathsf{DistP}^{\Sigma^{\mathsf{P}}_2} \subseteq \mathsf{AvgBPP}$ *(which holds in particular for $\ell \geq 2$). Then there exist polynomials q' and q'_0 such that, for all sufficiently large $n \in \mathbb{N}$, all $x \in L_n$, and all $t \geq q'_0(n)$, we have with probability at least $\delta - 1/10$ over $r \in \{0,1\}^{n^c}$ that an L-witness $y_{x,r}$ exists, and*

$$\mathsf{rK}^{q'(t)}(x, y_{x,r} \mid r) \leq \mathsf{rK}^t(x \mid r) + \log q'(t).$$

Moreover, for the case of no BP *quantifier, i.e., for $L \in \Sigma^{\mathsf{P}}_\ell$, we get the same conclusions (with probability 1) without conditioning on r.*

Proof of Item 1. Define

$$L' := \left\{ \left(\mathsf{DP}_k(x,y;z), w, w', 1^k, 1^s, 1^n\right) \,\middle|\, \exists\, \mathcal{M} \in \{0,1\}^s, \mathcal{M}^{\Sigma^{\mathsf{P}}_\ell}(w,w') \text{ prints } (x,y) \in \{0,1\}^{n+n^c} \right.$$
$$\left. \text{within } |w| \text{ steps, where } |w'| = n^c, \text{ and } s = k - 10 \right\}.$$

More formally, in this and subsequent proofs, the above notation should be interpreted as follows: a string $(u, w, w', 1^k, 1^s, 1^n)$ belongs to L' iff there exist some $(x,y) \in \{0,1\}^{n+n^c}$ and $\mathcal{M} \in \{0,1\}^s$ such that $\mathsf{DP}_k(x,y;z) = u$, where $u = (z, \alpha)$ for some $z \in \{0,1\}^{(n+n^c)k}$ and $\alpha \in \{0,1\}^k$, and $\mathcal{M}^{\Sigma^{\mathsf{P}}_\ell}(w,w')$ outputs (x,y) within $|w|$ steps, for $|w'| = n^c$.

Note that $L' \in \mathsf{NP}^{\Sigma^{\mathsf{P}}_\ell}$. Define a distribution family $D = \{D_{\langle (n+n^c)k+k, 2t, n^c, k, s, n\rangle}\}$ as follows: sample $u \sim \mathcal{U}_{(n+n^c)k+k}$, $w \sim \mathcal{U}_{2t}$, $w' \sim \mathcal{U}_{n^c}$, and output $(u, w, w', 1^k, 1^s, 1^n)$, for $s = k - 10$. By assumption, $(L', D) \in \mathsf{AvgBPP}$. Let B be a randomized algorithm for (L', D) as in Lemma 6.

For all (sufficiently large) n, k, and s, and for independent uniformly random u, w, and w',

$$\begin{aligned}
&\Pr_{u,w,w'} \left[(u, w, w', 1^k, 1^s, 1^n) \in L'\right] \\
&= \Pr_{u,w,w'} \Big[\exists\, (x,y) \in \{0,1\}^{n+n^c}, \exists\, z \in \{0,1\}^{(n+n^c)k}, \exists\, \mathcal{M} \in \{0,1\}^s, \\
&\qquad\qquad \mathcal{M}^{\Sigma^{\mathsf{P}}_\ell}(w,w') = (x,y)\ \wedge\ u = \mathsf{DP}_k(x,y;z)\Big] \\
&\leq \frac{2^{s+|w|+|w'|+|z|}}{2^{k+|w|+|w'|+|z|}} = 2^{s-k} = 2^{-10},
\end{aligned}$$

where the inequality is by a union bound and a counting argument, and the last equality by the condition that $s = k - 10$ in the definition of L'. Hence,

$$\begin{aligned}
&\Pr_{u,w,w',r_B}\left[B(u,w,w',1^k,1^s,1^n)=1\right]\\
&\le \Pr_{u,w,w'}\left[(u,w,w',1^k,1^s,1^n)\in L'\right]\\
&\quad + \Pr_{u,w,w',r_B}\left[B(u,w,w',1^k,1^s,1^n)\neq L'(u,w,w',1^k,1^s,1^n)\right]\\
&\le 2^{-10}+(3/n).
\end{aligned}$$

By Markov's inequality, for at least $9/10$ fraction of strings r,

$$\Pr_{u,w,r_B}\left[B(u,w,r,1^k,1^s,1^n)=1\right]\le 10\cdot(2^{-10}+(3/n))\le 1/10. \tag{18}$$

Let $x\in L_n$ be arbitrary. By definition of the BP_δ quantifier, for at least δ fraction of random strings $r\in\{0,1\}^{n^c}$, there exists an L-witness, and hence, the lexicographically first L-witness $y_{x,r}$. By the above, for at least $\delta - 1/10$ of random r, we have that both a witness $y_{x,r}$ exists and Equation (18) holds. Fix any such r.

Observe that for some polynomial q_0 (dependent on L) and some constant d, for any $t\ge q_0(n)$,

$$\mathsf{pK}^{2t,\Sigma_\ell^{\mathsf{P}}}(x,y_{x,r}\mid r)\le \mathsf{pK}^t(x\mid r)+d\log n =: s(x,r). \tag{19}$$

In particular, $q_0(n)$ reflects the time required to deterministically compute $y_{x,r}$, given x and r, by a search-to-decision procedure for L using a Σ_ℓ^{P}-oracle.

Define $s := s(x,r)$ (which determines $k = s+10$), and let $t\ge q_0(n)$. By the definitions of $\mathsf{pK}^{2t,\Sigma_\ell^{\mathsf{P}}}$ and L', for every $z\in\{0,1\}^{(n+n^c)k}$,

$$\Pr_{w}\left[(\mathsf{DP}_k(x,y_{x,r};z),w,r,1^k,1^s,1^n)\in L'\right]\ge 2/3.$$

Then by the definition of B,

$$\begin{aligned}
&\Pr_{z,w,r_B}\left[B(\mathsf{DP}_k(x,y_{x,r};z),w,r,1^k,1^s,1^n)=1\right]\\
&\ge \Pr_{z,w}\left[(\mathsf{DP}_k(x,y_{x,r};z),w,r,1^k,1^s,1^n)\in L'\right]\cdot\Pr_{r_B}\left[B(\omega)=L'(\omega)\mid\omega\in L'\right]\\
&\ge \frac{2}{3}\cdot(1-(1/n))\\
&\ge 2/3-o(1).
\end{aligned} \tag{20}$$

Comparing Equation (18) with Equation (20), it is clear that $B(-,\mathcal{U}_{2t},r,1^k,1^s,1^n)$ is a randomized algorithm (with advice) that $(1/2)$-distinguishes $\mathsf{DP}_k(x,y_{x,r};\mathcal{U}_{(n+n^c)k})$ from uniform. Lemma 22 implies that

$$\begin{aligned}
\mathsf{pK}^{p'(t)}(x,y_{x,r}\mid r,s,n,t) &\le k+\log p'(t) \\
&= s+10+\log p'(t)\\
&= \mathsf{pK}^t(x\mid r)+d\log n+\log p'(t)+10,
\end{aligned} \tag{21}$$

for some polynomial p' with $p'(t)\ge p_{\mathsf{DP}}(t_B\cdot 3\cdot(n+n^c))$, where p_{DP} is the polynomial from Lemma 22 and t_B denotes the time required to compute $B(-,\mathcal{U}_{2t},r,1^k,1^s,1^n)$. As the advice (s,n,t) can be encoded with $\log s+\log n+\log t+O(\log\log n)\le(3.1)\cdot\log t$ bits, it follows that $\mathsf{pK}^{q(t)}(x,y_{x,r}\mid r)\le\mathsf{pK}^t(x\mid r)+\log q(t)$ for the polynomial $q(t):=t^{(d+4)}\cdot p'(t)$.

The "Moreover" statement follows the same argument, dropping any mention of w' (and r). ◀

Proof of Item 2. The proof is similar to that of Item 1, except at Equation (19), we observe that for all $t \geq q_0(n)$,

$$\begin{aligned}\mathsf{pK}^{2t,\Sigma_\ell^\mathsf{p}}(x, y_{x,r} \mid r) &\leq \mathsf{pK}^t(x \mid r) + d \log n \\ &\leq \mathsf{rK}^t(x \mid r) + d \log n,\end{aligned}$$

and then define $s(x,r) := \mathsf{rK}^t(x \mid r) + d \log n$. We then follow the proof of Item 1 up to Equation (21), where we apply Lemma 23 instead of Lemma 22 to obtain

$$\mathsf{rK}^{p''(t)}(x, y_{x,r} \mid r, s, n, t) \leq \mathsf{rK}^t(x \mid r) + d \log n + \log p''(t) + 10$$

for some polynomial p'' with $p''(t) \geq p'_{\mathsf{DP}}(t_B \cdot 3 \cdot (n + n^c))$, where p'_{DP} is the polynomial from Lemma 23 and t_B denotes the time required to compute $B(-, \mathcal{U}_{2t}, r, 1^k, 1^s, 1^n)$. It follows that $\mathsf{rK}^{q'(t)}(x, y_{x,r} \mid r) \leq \mathsf{rK}^t(x \mid r) + \log q'(t)$ for the polynomial $q'(t) := t^{(d+4)} \cdot p''(t)$. ◀

▶ **Lemma 34** (Witness Compression)**.** *For any $\ell \geq 1$, suppose $\mathsf{Dist}\Sigma_{\ell+1}^\mathsf{P} \subseteq \mathsf{AvgBPP}$. For an arbitrary $L \in \mathsf{BP}_\delta \circ \Sigma_\ell^\mathsf{P}$, let $x \in L_n$ be sufficiently large, $r \in \{0,1\}^{n^c}$ be any good random string, and let $y_{x,r} \in \{0,1\}^{n^c}$ be the lexicographically first L-witness for x with respect to randomness r, for some constant $c > 0$.*

1. *Let p_0, p and q_0, q be the polynomials from Items 1 of Lemmas 26 and 33 respectively. Then for every $t \geq \max\{p_0(n), q_0(n + n^c)\}$, with probability at least $\delta - 1/5$ over r, an L-witness $y_{x,r}$ exists, and*

$$\mathsf{pK}^{p(q(t))}(y_{x,r} \mid x, r) \leq \Big(\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{p(q(t))}(x \mid r)\Big) + 2 \log p(q(t)).$$

2. *Assume in addition that $\mathsf{DistP}^{\Sigma_2^\mathsf{P}} \subseteq \mathsf{AvgBPP}$ (which holds in particular when $\ell \geq 2$). Let p'_0, p' and q'_0, q' be the polynomials from Items 2 of Lemmas 26 and 33 respectively. Then for every $t \geq \max\{p'_0(n), q'_0(n+n^c)\}$, with probability at least $\delta - 1/5$ over r, an L-witness $y_{x,r}$ exists, and*

$$\mathsf{rK}^{p'(q'(t))}(y_{x,r} \mid x, r) \leq \Big(\mathsf{rK}^t(x \mid r) - \mathsf{rK}^{p'(q'(t))}(x \mid r)\Big) + 2 \log p'(q'(t)).$$

Moreover, for $L \in \Sigma_\ell^\mathsf{P}$, we get the same conclusions without conditioning on r.

Proof. We start by proving Item 1. By Item 1 of Lemma 33 and Lemma 26, we get by a union bound that, with probability at least $\delta - 2/10$ over r,

$$\mathsf{pK}^{q(t)}(x, y_{x,r} \mid r) \leq \mathsf{pK}^t(x \mid r) + \log q(t),$$

and

$$\mathsf{pK}^{q(t)}(x, y_{x,r} \mid r) > \mathsf{pK}^{p(q(t))}(x \mid r) + \mathsf{pK}^{p(q(t))}(y_{x,r} \mid x, r) - \log p(q(t)).$$

Combining the previous two inequalities,

$$\begin{aligned}\mathsf{pK}^{p(q(t))}(y_{x,r} \mid x, r) &< \mathsf{pK}^{q(t)}(x, y_{x,r} \mid r) - \mathsf{pK}^{p(q(t))}(x \mid r) + \log p(q(t)) \\ &\leq \Big(\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{p(q(t))}(x \mid r)\Big) + 2 \log p(q(t)).\end{aligned}$$

The "Moreover" part follows by applying the "no random r" versions of Lemma 33 and Lemma 26. The proof of Item 2 is the same, except we apply Items 2 of Lemmas 33 and 26 respectively. ◀

We will also need the following analogue of Lemma 33 for the case of unique witnesses.

▶ **Lemma 35** (Oracle Elimination for UP). *Suppose* DistNP $\subseteq$ AvgBPP*, and let* $L \in$ UP*. Let* $x \in L_n$ *be sufficiently large, and let* $y_x \in \{0,1\}^{n^c}$ *be the unique* L*-witness for* x*, for some constant* $c > 0$*. There exists a polynomial* q *such that for all* $t \geq n + n^c$*,*

$$\mathsf{pK}^{q(t)}(x, y_x) \leq \mathsf{pK}^t(x) + \log q(t).$$

Proof. Let $L \in \mathsf{UP}$ with deterministic verifier V running in time n^c on inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^{n^c}$. Let

$$L' := \Big\{ (\mathsf{DP}_k(x, y; z), w, 1^s, 1^n) \mid \exists\, \mathcal{M} \in \{0,1\}^s, \mathcal{M}(w) \text{ outputs } x \in \{0,1\}^n \text{ within } |w| \text{ steps}, \\ \text{where } k = s + 10, \text{ and } V(x, y) = 1 \Big\}.$$

Note that $L' \in \mathsf{NP}$. Define a distribution family $D = \{D_{\langle (n+n^c)k+k, t, s, n\rangle}\}$ as follows: sample $u \sim \mathcal{U}_{(n+n^c)k+k}$, $w \sim \mathcal{U}_t$, and output $(u, w, 1^s, 1^n)$. By assumption, $(L', D) \in \mathsf{AvgBPP}$. Let B be a randomized heuristic algorithm for (L', D) as described in Lemma 6.

Let $x \in L_n$ be sufficiently large, and let $y_x \in \{0,1\}^{n^c}$ be the *unique* L-witness for x. Define $s := \mathsf{pK}^t(x)$, and let $t \geq n + n^c$. By definition of pK^t and L', for any $z \in \{0,1\}^{(n+n^c)k}$,

$$\Pr_w[(\mathsf{DP}_k(x, y_x; z), w, 1^s, 1^n) \in L'] \geq 2/3.$$

Then by definition of B,

$$\begin{aligned} &\Pr_{z,w,r_B}[B(\mathsf{DP}_k(x, y_x; z), w, 1^s, 1^n) = 1] \\ &\geq \Pr_{z,w}[(\mathsf{DP}_k(x, y_x; z), w, 1^s, 1^n) \in L'] \cdot \Pr_{r_B}[B(\omega) = L'(\omega) \mid \omega \in L'] \\ &\geq \frac{2}{3} \cdot (1 - (1/n)) \geq 1/2. \end{aligned} \tag{22}$$

On the other hand, for u and w selected uniformly at random,

$$\begin{aligned} \Pr_{u,w}[(u, w, 1^s, 1^n) \in L'] = \Pr_{u,w}\Big[&\exists\,(x, y) \in \{0,1\}^{n+n^c}, \exists\, z \in \{0,1\}^{(n+n^c)k}, \exists\, \mathcal{M} \in \{0,1\}^s, \\ &\mathcal{M}(w) = x \ \wedge\ V(x, y) = 1 \ \wedge\ u = \mathsf{DP}_k(x, y; z)\Big] \\ \leq \frac{2^{s+|w|+|z|}}{2^{k+|w|+|z|}} &= 2^{-10}, \end{aligned}$$

where the inequality is by a union bound and a counting argument, and the last equality by definition of $k = s + 10 = \mathsf{pK}^t(x) + 10$. Then

$$\begin{aligned} &\Pr_{u,w,r_B}[B(u, w, 1^s, 1^n) = 1] \\ &\leq \Pr_{u,w}[(u, w, 1^s, 1^n) \in L'] + \Pr_{u,w,r_B}[B(u, w, 1^s, 1^n) \neq L'(u, w, 1^s, 1^n)] \\ &\leq 2^{-10} + (3/n) \leq 1/10. \end{aligned}$$

Comparing with Equation (22), it is clear that $B(-, \mathcal{U}_t, 1^s, 1^n)$ is a randomized distinguisher for $\mathsf{DP}_k\big(x, y_x; \mathcal{U}_{(n+n^c)k}\big)$. Lemma 22 implies that

$$\begin{aligned} \mathsf{pK}^{p'(t)}(x, y_x) &\leq k + \log p'(t) \\ &= \mathsf{pK}^t(x) + 10 + \log p'(t) \end{aligned}$$

for some polynomial p' with $p'(t) \geq p_{\mathsf{DP}}(t_b \cdot 3 \cdot (n + n^c))$, where p_{DP} is the polynomial from Lemma 22 and t_B denotes the time required to compute $B(-, \mathcal{U}_t, 1^s, 1^n)$. It follows that $\mathsf{pK}^{q(t)}(x, y_x) \leq \mathsf{pK}^t(x) + \log q(t)$ for the polynomial $q(t) := t \cdot p'(t)$. ◀

4.2 Case of AM

▶ **Theorem 36.** *If* $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, *then* $\mathsf{AM} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.

Proof. Consider an arbitrary $L \in \mathsf{AM}$ given by a $\mathsf{BP} \circ \Sigma_1^{\mathsf{P}}$ formula such that, for every $x \in \{0,1\}^n$,

$$x \in L \implies \mathsf{BP}_{(1-2^{-n})} r \; \exists y \; V(x, y, r),$$
$$x \notin L \implies \mathsf{BP}_{(1-2^{-n})} r \; \forall y \; \neg V(x, y, r),$$

for a polytime verifier $V(x, y, r)$, where $|r|, |y| \leq n^c$ for some constant $c > 0$, and the notation $\mathsf{BP}_{(\delta)} r \; \phi(-, r)$, for a formula ϕ, means $\mathbf{Pr}_r[\phi(-, r)] \geq \delta$.

For a given (sufficiently large) $x \in L_n$, and a good random string r, let $y_{x,r}$ be the lexicographically first L-witness for x, r. By Lemma 34, for some constant $a > 0$, and for all large enough time bounds t, we have, with probability at least $1 - 2^{-n} - 1/5$ over random strings r, that

$$\mathsf{pK}^{t^a}(y_{x,r} \mid x, r) \leq \left(\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{t^a}(x \mid r)\right) + (2a) \log t.$$

By Lemma 31, there exists a time bound $t \leq 2^{O(n/\log n)}$ such that $\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{t^a}(x \mid r) \leq bn/\log n$, for some universal constant $b > 0$. Hence, for some universal constant $d > 0$ (independent of x, r),

$$\mathsf{pK}^{2^{dn/\log n}}(y_{x,r} \mid x, r) \leq dn/\log n.$$

A randomized algorithm for L is now obvious:

> Given $x \in \{0,1\}^n$, sample uniformly random strings $r \in \{0,1\}^{|x|^c}$ and $w \in \{0,1\}^{2^{dn/\log n}}$, and then exhaustively search over all $dn/\log n$-bit machines $\mathcal{M}$, running each $\mathcal{M}(w, x, r)$ for $2^{dn/\log n}$ steps to produce a candidate witness y, accepting if $V(x, y, r)$ accepts.

The runtime $2^{O(n/\log n)}$ is clear. For correctness, for every $x \in L_n$, we know there is at least a $(4/5 - 2^{-n})$ fraction of good random strings r for which a witness $y_{x,r}$ will be found by our algorithm, with probability at least $2/3$ over random w's (by the definition of pK). So our algorithm accepts $x \in L$ with probability at least $(2/3)(4/5) - o(1) > 1/2$. On the other hand, for any $x \notin L$, our algorithm may find a witness y and accept x for at most 2^{-n} fraction of r's. ◀

4.3 Case of UP

▶ **Theorem 37.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, *then* $\mathsf{UP} \subseteq \mathsf{RTIME}\left[2^{O(n/\log n)}\right]$.

Proof. Let $L \in \mathsf{UP}$, $x \in L_n$ for n sufficiently large, and $y_x \in \{0,1\}^{n^c}$ the unique L-witness for x, for some constant $c > 0$. Let V be a deterministic verifier for L running in time n^c. Let p_0, p, and q be the polynomials from Lemmas 26 (Item 1) and 35 respectively.

By Lemma 31, there exists $p_0(n + n^c) \leq t \leq 2^{n/\log n}$ such that

$$\mathsf{pK}^t(x) - \mathsf{pK}^{p(q(t))}(x) \leq O(n/\log n).$$

For such a t, combining Lemma 26 and Lemma 35 as in the proof of Lemma 34, we obtain

$$\mathsf{pK}^{p(q(t))}(y_x \mid x) \leq \left(\mathsf{pK}^t(x) - \mathsf{pK}^{p(q(t))}(x)\right) + 2\log p(q(t)).$$

The above inequalities imply that for some constant $d \in \mathbb{N}$,

$$\mathsf{pK}^{2^{dn/\log n}}(y_x \mid x) \leq dn/\log n.$$

We are now ready to define a probabilistic algorithm A deciding L.

On input $x \in \{0,1\}^n$, A samples $w \sim \{0,1\}^{2^{dn/\log n}}$. It then exhaustively searches over all machines $\mathcal{M} \in \{0,1\}^{dn/\log n}$, running $\mathcal{M}(w,x)$ for $2^{dn/\log n}$ steps to produce some output y. A accepts iff a y is obtained such that $V(x,y) = 1$.

If $x \in L$, then A accepts with probability at least $2/3$ over w; if $x \notin L$, then A never accepts. ◀

4.4 Case of PH

As a warm-up, we consider a special case of PH languages L, where a Σ_ℓ^P verifier for $L \in \Sigma_\ell^\mathsf{P}$ has all its $\exists/\forall$ quantifiers over binary strings of length $O(|x|)$ for a given input x. Denote the class of such Σ_ℓ^P languages by $\mathsf{lin}\Sigma_\ell^\mathsf{P}$.

▶ **Theorem 38.** *For an arbitrary $\ell \geq 1$, suppose that $\mathsf{Dist}\Sigma_{\ell+1}^\mathsf{P} \subseteq \mathsf{AvgBPP}$. Then*

$$\mathsf{lin}\Sigma_\ell^\mathsf{P} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right].$$

Proof. The proof is by induction on ℓ. The base case of $\ell = 1$ follows from Theorem 36. We will argue the case of $\ell \geq 2$. Consider an arbitrary language $L \in \mathsf{lin}\Sigma_\ell^\mathsf{P}$. By definition, there is a polytime predicate R and a constant $a > 0$ such that, for every $x \in \{0,1\}^n$,

$$x \in L \iff \exists\, y_1\, \forall\, y_2\ \ldots\ Q\, y_\ell\ \ R(x, y_1, y_2, \ldots, y_\ell),$$

where $|y_i| \leq an$ for all $1 \leq i \leq \ell$. Define the language

$$L' = \{(x,y) \mid \forall\, y_2\ \ldots\ Q\, y_\ell\ \ R(x, y, y_2, \ldots, y_\ell)\}.$$

Note that $L' \in \mathsf{lin}\Pi_{\ell-1}^\mathsf{P}$, and so, by the Inductive Hypothesis, $L' \in \mathsf{BPTIME}[2^{O(n/\log n)}]$, since $|(x,y)| \leq O(|x|)$ by assumption. By standard success amplification, we may assume that this probabilistic algorithm for L' has error probability at most 2^{-n}.

On the other hand, by Items 1 of Lemmas 34 and 31, we have that, for some universal constant $d > 0$, for every sufficiently large input $x \in L_n$,

$$\mathsf{pK}^{2^{dn/\log n}}(y_x \mid x) \leq dn/\log n,$$

where y_x is the lexicographically first L-witness for $x \in L$.

We are now ready to define a probabilistic algorithm A deciding L.

On input $x \in \{0,1\}^n$, A samples $w \sim \{0,1\}^{2^{dn/\log n}}$. It then exhaustively searches over all machines $\mathcal{M} \in \{0,1\}^{dn/\log n}$, running $\mathcal{M}(w,x)$ for $2^{dn/\log n}$ steps to produce some output y. A accepts iff a y is obtained such that $(x,y) \in L'$, where the latter is checked using a $\mathsf{BPTIME}[2^{O(n/\log n)}]$ algorithm for L' shown to exist earlier.

The overall runtime $2^{O(n/\log n)}$ of the described algorithm A is clear. For correctness, consider an arbitrary $x \in L_n$. With probability at least $2/3$ over w's, A will check if $(x, y_x) \in L'$. The latter check will succeed with probability at least $1 - 2^{-n}$, by assumption. Therefore, A will correctly accept x with probability at least $(2/3)(1 - 2^{-n}) \geq 6/10$. Next consider an arbitrary $x \in \{0,1\}^n$ such that $x \notin L$. For this x, no witnesses exist, and hence every string y tried by A is such that $(x, y) \notin L'$. The probability that at least one such y is incorrectly accepted by A is, by the union bound, at most $2^{dn/\log n} \cdot 2^{-n} \leq 2^{-n/2}$, for all sufficiently large n. ◀

It is observed in [7, Corollary 17] that, for every $\ell \geq 1$, $\Sigma_\ell^{\mathsf{P}} = \mathsf{lin}\Sigma_\ell^{\mathsf{P}}$, under the assumption that $\mathsf{Dist}\Sigma_{\ell+1}^{\mathsf{P}} \subseteq \mathsf{AvgP}$. We will show a similar result for the class $\mathsf{promise}\text{-}\Sigma_\ell^{\mathsf{BP}}$ defined below.

▶ **Definition 39** ($\mathsf{promise}\text{-}\Sigma_\ell^{\mathsf{BP}}$ and $\mathsf{promise}\text{-}\mathsf{lin}\Sigma_\ell^{\mathsf{BP}}$)**.** *A promise problem* $\Pi = (\Pi_Y, \Pi_N) \in \mathsf{promise}\text{-}\Sigma_\ell^{\mathsf{BP}}$ *for* $\ell \geq 0$*, if there is a* $\Sigma_\ell \circ \mathsf{BP} \circ \mathsf{P}$ *formula*

$$\phi(x) = \exists\, y_1\, \forall\, y_2\ \ldots\ Q\, y_\ell\ \mathsf{BP}r\ \ R(x, y_1, y_2, \ldots, y_\ell, r),$$

for a polytime predicate R *and all string variables* $y_1, \ldots, y_\ell, r$ *of length polynomial in* $|x|$*, such that, for all* $x \in \{0,1\}^n$*,*

$$\begin{aligned} x \in \Pi_Y \quad &\Longrightarrow \quad \exists\, y_1\, \forall\, y_2\ \ldots\ Q\, y_\ell\ \mathsf{BP}_{(2/3)}r\ \ R(x, y_1, y_2, \ldots, y_\ell, r), \text{ and} \\ x \in \Pi_N \quad &\Longrightarrow \quad \forall\, y_1\, \exists\, y_2\ \ldots\ \bar{Q}\, y_\ell\ \mathsf{BP}_{(2/3)}r\ \ \neg R(x, y_1, y_2, \ldots, y_\ell, r), \end{aligned}$$

where $\mathsf{BP}_{(\delta)}r\ \ R(-, r)$ *means that* $\mathbf{Pr}_r\,[R(-, r)] \geq \delta$*. The constant* $2/3$ *in the definition above can be changed to* $1 - 2^{-\mathsf{poly}(n)}$ *by standard success amplification techniques (even if we start with any pair* $\delta_Y > 1 - \delta_N + \varepsilon$ *of thresholds, for some* $\varepsilon \geq 1/\mathsf{poly}(n)$*, where* δ_Y *is the probability* $R(-, r)$ *accepts, and* δ_N *the probability* $\neg R(-, r)$ *accepts).*

The promise class $\mathsf{promise}\text{-}\mathsf{lin}\Sigma_\ell^{\mathsf{BP}}$ *is defined the same way as above, with an extra condition that, for some constant* $a > 0$*,* $|y_i| \leq an$ *for all* $1 \leq i \leq \ell$ *(but the randomness* r *may still be of any* $\mathsf{poly}(n)$ *length).*

▶ **Theorem 40** (Witness Compression for PH)**.** *For any* $\ell \geq 0$*, suppose* $\mathsf{Dist}\Sigma_{\ell+2}^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$*. Then*

$$\mathsf{promise}\text{-}\Sigma_\ell^{\mathsf{BP}} = \mathsf{promise}\text{-}\mathsf{lin}\Sigma_\ell^{\mathsf{BP}}.$$

Proof. The proof is by induction on ℓ. The base case of $\ell = 0$ is trivially true.

For any $\ell > 0$, suppose the claim is true for $\ell - 1$. Consider an arbitrary promise problem $\Pi = (\Pi_Y, \Pi_N) \in \mathsf{promise}\text{-}\Sigma_\ell^{\mathsf{BP}}$. That is, for every $x \in \{0,1\}^n$,

$$\begin{aligned} x \in \Pi_Y \quad &\Longrightarrow \quad \exists\, y\, \Pi_{\ell-1} z\ \mathsf{BP}_{(1-2^{-n})}r\ \ R(x, y, z, r), \\ x \in \Pi_N \quad &\Longrightarrow \quad \forall\, y\, \Sigma_{\ell-1} z\ \mathsf{BP}_{(1-2^{-n})}r\ \ \neg R(x, y, z, r), \end{aligned}$$

for some polytime predicate R, where z is a sequence of $\ell - 1$ strings $z_1, \ldots, z_{\ell-1}$ under the corresponding $\ell - 1$ quantifiers of the $\Pi_{\ell-1}$ pattern, and $|y|, |r|, |z_i| \leq q(n)$, for some polynomial q, for all $1 \leq i \leq \ell - 1$.

As in the proof of the inclusion $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ [22], we replace the BP quantifier by the $\exists\forall$ pattern if ℓ is odd, or by the $\forall\exists$ pattern if ℓ is even. We get that there exists a polytime predicate R' such that, for every $x \in \{0,1\}^n$,

$$\begin{aligned} x \in \Pi_Y \quad &\Longrightarrow \quad \exists\, y\, \Pi_{\ell-1} z\, Q_0 z' Q_1 z''\ \ R'(x, y, z, z', z''), \\ x \in \Pi_N \quad &\Longrightarrow \quad \forall\, y\, \Sigma_{\ell-1} z\, Q_1 z' Q_0 z''\ \ \neg R'(x, y, z, z', z''), \end{aligned}$$

where z', z'' have the lengths polynomial in n and $Q_0 Q_1 = \exists\forall$ if ℓ is odd, and $Q_0 Q_1 = \forall\exists$ if ℓ is even.

Note that the above transformation shows that our promise problem $\Pi \in \mathsf{promise}\text{-}\Sigma^{\mathsf{P}}_{\ell+1}$. For any given $x \in \Pi_Y$, let y_x be the lexicographically first string y such that

$$\Pi_{\ell-1} z \; Q_0 z' \; Q_1 z'' \; R'(x, y, z, z', z''). \tag{23}$$

▷ Claim 41. For x and y_x as above,

$$\Pi_{\ell-1} z \; \mathsf{BP}_{(1/\mathsf{poly}(n))} r \;\; R(x, y_x, z, r).$$

Proof. Immediate from the properties of Lautemann's proof of $\mathsf{BPP} \subseteq \Sigma^{\mathsf{P}}_2$ [22]. Recall that in that proof, the subformula $\mathsf{BP} r \;\; R(-, r)$ is replaced by

$$\exists u_1 \ldots u_k \; \forall r \;\; \vee_{i=1}^{k} \;\; R(-, r \oplus u_i),$$

for some $k \leq \mathsf{poly}(|r|)$, where $|u_i| = |r|$, for all $1 \leq i \leq k$, and $\oplus$ is the bit-wise XOR. If $\mathbf{Pr}_r [R(-, r)] < 1/k$, then such a collection of $u_1, \ldots, u_k$ cannot exist. Indeed, each "shifted" predicate $R(-, r \oplus u_i)$, for $1 \leq i \leq k$, accepts less than $1/k$ fraction of $r \in \{0,1\}^{|r|}$, and hence, by the union bound, the OR of any k such shifts accepts less than $k \cdot (1/k) = 1$ fraction of $r \in \{0,1\}^{|r|}$. ◁

On the other hand, for every $x \in \Pi_N$ and every y, we have

$$\Sigma_{\ell-1} z \; \mathsf{BP}_{(1-2^{-n})} r \;\; \neg R(x, y, z, r).$$

Using Claim 41, by standard success amplification, for a random string r' of length $q'(n)$, for some polynomial q', and a polytime predicate R^{boost}, we get for every $x \in \{0,1\}^n$ and y_x as defined above,

$$x \in \Pi_Y \implies \Pi_{\ell-1} z \; \mathsf{BP}_{(1-2^{-n})} r' \;\; R^{boost}(x, y_x, z, r'), \tag{24}$$

$$x \in \Pi_N \implies \forall \, y \; \Sigma_{\ell-1} z \; \mathsf{BP}_{(1-2^{-n})} r' \;\; \neg R^{boost}(x, y, z, r'). \tag{25}$$

▷ Claim 42. For some universal constant $a > 0$ and some polynomial p', for every n-bit $x \in \Pi_Y$ with a witness y_x as defined above, we have

$$\mathsf{rK}^{p'(n)}(y_x \mid x) \leq an.$$

Proof. Using Equation (23), we get by Item 2 of Lemma 34 (since $\mathsf{Dist}\Sigma^{\mathsf{P}}_3 \subseteq \mathsf{AvgBPP}$), that, for some polynomial p and all large enough t,

$$\mathsf{rK}^{p(t)}(y_x \mid x) \leq \mathsf{rK}^{t}(x) - \mathsf{rK}^{p(t)}(x) + 2 \log p(t).$$

Since, for any t, $\mathsf{rK}^t(x) \leq O(n)$, the claim follows. ◁

Consider the formula

$$\phi(x) = \exists \, y' (|y'| \leq an) \; \Pi_{\ell-1} z \; \mathsf{BP} w \;\; R''(x, y', z, w), \tag{26}$$

where the predicate R'' is computed by the following polytime algorithm A:

> On input x, y', z, and a string $w = w'r'$, where w' is of length $p'(n)$, and r' is of length $q'(n)$, first run the universal machine $U(y', x, w')$ for $p'(n)$ steps, getting an output y. Then run $R^{boost}(x, y, z, r')$, accepting iff R^{boost} accepts.

We claim that Equation (26) is a correct $\Sigma_\ell^{\mathsf{BP}}$ formula for the promise problem Π. Indeed, for any $x \in \Pi_Y$, by Claim 42, there is a string y' that is decompressed to y_x with high probability over random strings w' (say, with probability at least $3/4$). By Equation (24), $R^{boost}(x, U(y', x, w'), z, r')$ accepts with probability at least $(3/4)(1-2^{-n}) > 2/3$ over strings $w = w'r'$. On the other hand, for any $x \in \Pi_N$, Equation (25) implies that

$$\forall\, y'\ \Sigma_{\ell-1} z\ \mathsf{BP}_{(1-2^{-n})} w\ \neg R''(x, y', z, w).$$

Next, using Equation (26), define a new promise problem $\Pi' = (\Pi'_Y, \Pi'_N) \in \mathsf{promise}\text{-}\Pi^{\mathsf{BP}}_{\ell-1}$:

$$\Pi'_Y = \{(x, y') \mid |y'| \le a|x|,\ \ \Pi_{\ell-1} z\ \mathsf{BP} w\ R''(x, y', z, w)\},$$
$$\Pi'_N = \{(x, y') \mid |y'| \le a|x|,\ \ \Sigma_{\ell-1} z\ \mathsf{BP} w\ \neg R''(x, y', z, w)\}.$$

By the Inductive Hypothesis, $\Pi' \in \mathsf{promise}\text{-}\mathsf{lin}\Pi^{\mathsf{BP}}_{\ell-1}$, with all quantified binary strings of length $O(|x| + |y'|) \le O(n)$. It follows that $\Pi \in \mathsf{promise}\text{-}\mathsf{lin}\Sigma^{\mathsf{BP}}_\ell$. ◀

▶ **Theorem 43.** *For any $\ell \ge 0$,*

$$\mathsf{Dist}\Sigma^{\mathsf{P}}_{\ell+2} \subseteq \mathsf{AvgBPP} \implies \mathsf{promise}\text{-}\Sigma^{\mathsf{P}}_\ell \subseteq \mathsf{promise}\text{-}\mathsf{BPTIME}[2^{O(n/\log n)}].$$

Proof. The proof is by induction on ℓ. The base case of $\ell = 0$ is trivially true. Consider any $\ell \ge 1$, and assume the claim for $\ell - 1$. Let $\Pi = (\Pi_Y, \Pi_N) \in \mathsf{promise}\text{-}\Sigma^{\mathsf{P}}_\ell$ be arbitrary. By Theorem 40, $\Pi \in \mathsf{promise}\text{-}\mathsf{lin}\Sigma^{\mathsf{BP}}_\ell$ via some formula

$$\phi(x) = \exists\, y\ \Pi_{\ell-1} z\ \mathsf{BP} r\ \ R(x, y, z, r),$$

with a polytime R, linearly-bounded y and $z = (z_1, \ldots, z_{\ell-1})$ (under the $\Pi_{\ell-1}$ quantifiers), and a poly-bounded r.

Arguing as in the proof of Theorem 40 (see the proof of Claim 42), but using Lemma 31 (for the case of pK) to bound the computational depth of an input x by $O(n/\log n)$ with a time bound $t \le 2^{O(n/\log n)}$, we get that, for some universal constant $a > 0$, every $x \in \Pi_Y$ has a witness y_x with

$$\mathsf{pK}^{2^{an/\log n}}(y_x \mid x) \le an/\log n. \tag{27}$$

Applying success amplification to R, we get a new polytime predicate R^{boost} with poly-bounded randomness r' such that we get for every $x \in \{0,1\}^n$ and y_x as defined above,

$$x \in \Pi_Y \implies \Pi_{\ell-1} z\ \mathsf{BP}_{(1-2^{-n})} r'\ \ R^{boost}(x, y_x, z, r'), \tag{28}$$
$$x \in \Pi_N \implies \forall\, y\ \Sigma_{\ell-1} z\ \mathsf{BP}_{(1-2^{-n})} r'\ \ \neg R^{boost}(x, y, z, r'). \tag{29}$$

Define a promise problem $\Pi' = (\Pi'_Y, \Pi'_N) \in \mathsf{promise}\text{-}\Pi^{\mathsf{BP}}_{\ell-1}$:

$$\Pi'_Y = \left\{(x, y) \mid \Pi_{\ell-1} z\ \mathsf{BP} r'\ \ R^{boost}(x, y, z, r')\right\},$$
$$\Pi'_N = \left\{(x, y) \mid \Sigma_{\ell-1} z\ \mathsf{BP} r'\ \ \neg R^{boost}(x, y, z, r')\right\}.$$

By the Inductive Hypothesis, $\Pi' \in \mathsf{promise}\text{-}\mathsf{BPTIME}[2^{O(n/\log n)}]$, where $|x| = n$, via some randomized algorithm A, with error probability at most 2^{-n}.

We now solve Π with the following randomized algorithm B:

Given $x \in \{0,1\}^n$, choose a random string w of length $2^{an/\log n}$, and then enumerate over all machines $\mathcal{M} \in \{0,1\}^{an/\log n}$, and run $\mathcal{M}(w, x)$ for $2^{an/\log n}$ steps, getting a candidate witness y of length $O(n)$. Run $A(x, y)$, accepting iff A accepts.

The time complexity of the described algorithm B is clearly $2^{O(n/\log n)}$. For correctness, for every $x \in \Pi_Y$, with probability at least $3/4$ over w, our algorithm will consider a short description of a TM $\mathcal{M}$ such that $\mathcal{M}(w,x) = y_x$. Since $(x, y_x) \in \Pi'_Y$, we get that $A(x, y_x)$ will accept with probability at least $1 - 2^{-n}$ over its internal randomness. Hence, the algorithm B accepts x with probability at least $(3/4)(1 - 2^{-n}) > 2/3$. On the other hand, for every $x \in \Pi_N$, for every w and all $\mathcal{M}$, the string $y = \mathcal{M}(w,x)$ is such that $(x,y) \in \Pi'_N$, and so $A(x,y)$ accepts with probability at most 2^{-n}. It follows that the probability that $B(x)$ accepts in this case is

$$\mathop{\mathbf{E}}_{w,A}\left[\exists\, v \in \{0,1\}^{an/\log n} \;:\; A(x, U(v,x,w))\right] \leq 2^{an/\log n} \cdot \mathop{\mathbf{E}}_{w}\left[2^{-n}\right] \leq 2^{-n/2},$$

for all sufficiently large n. ◀

▶ **Corollary 44.** *If* $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{PH} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$*. More precisely, for any* $\ell \geq 0$*,*

$$\mathsf{Dist}\Sigma^{\mathsf{P}}_{\ell+2} \subseteq \mathsf{AvgBPP} \implies \Sigma^{\mathsf{P}}_{\ell} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right].$$

Proof. Immediate from Theorem 43. ◀

4.5 Fine-grained case

Recall the quasilinear-time complexity classes $\mathsf{QL} = \mathsf{DTIME}[\widetilde{O}(n)]$, $\mathsf{NQL} = \mathsf{NTIME}[\widetilde{O}(n)]$, and the quasilinear-time analog $\mathsf{QLH} = \cup_{\ell \geq 0}\Sigma^{\mathsf{QL}}_{\ell}$ of the polynomial-time hierarchy PH; see, e.g., [36] for more details. Define $\mathsf{BPQL} = \mathsf{BPTIME}[\widetilde{O}(n)]$, the quasilinear-time version of BPP. We show better probabilistic worst-case to average-case reductions under fine-grained average-case easiness assumptions.

▶ **Theorem 45.** $\Sigma^{\mathsf{QL}}_2 \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL} \implies \mathsf{AMTIME}\left[2^{O(\sqrt{n \log n})}\right] = \mathsf{BPTIME}\left[2^{O(\sqrt{n \log n})}\right]$.

It suffices to prove the inclusion $\mathsf{AMTIME}[2^{O(\sqrt{n \log n})}] \subseteq \mathsf{BPTIME}[2^{O(\sqrt{n \log n})}]$ (as the other inclusion is obvious). We will need the following analog of Lemma 31. The difference is that the "blow-up" of the time bound in pK^t here is *linear* in t, while in Lemma 31, this blow-up is *polynomial* in t.

▶ **Lemma 46** (Fine-Grained Computational Depth)**.** *Let* $x \in \{0,1\}^n$ *and* $r \in \{0,1\}^*$ *be arbitrary strings,* f *any function, and* $\tau_n(t) := t \cdot \mathsf{poly}(n)$*. There exists a constant* $a > 0$ *and a time bound* t *such that* $f(n) \leq t \leq f(n) \cdot 2^{a\sqrt{n \log n}}$ *and*

$$\mathsf{pK}^t(x \mid r) - \mathsf{pK}^{\tau_n(t)}(x \mid r) \leq O\left(\sqrt{n \log n}\right).$$

Proof. Fix $x \in \{0,1\}^n$. Let $\tau_n(t) := t \cdot n^c$. Define $t_0 := f(n)$, and for $i \geq 1$, $t_i := \tau_n(t_{i-1})$. Consider the following telescoping sum:

$$\begin{aligned}\mathsf{pK}^{t_0}(x \mid r) - \mathsf{pK}^{t_I}(x \mid r) = &\left(\mathsf{pK}^{t_0}(x \mid r) - \mathsf{pK}^{t_1}(x \mid r)\right) + \left(\mathsf{pK}^{t_1}(x \mid r) - \mathsf{pK}^{t_2}(x \mid r)\right) \\ &+ \cdots + \left(\mathsf{pK}^{t_{I-1}}(x \mid r) - \mathsf{pK}^{t_I}(x \mid r)\right).\end{aligned}$$

Note that $\mathsf{pK}^{f(n)}(x \mid r) \leq n + d$, for some universal constant $d \geq 0$; hence, the above sum is at most $n + d$. By averaging, there is some index $1 \leq i_* \leq I$ such that

$$\mathsf{pK}^{t_{i_*-1}}(x \mid r) - \mathsf{pK}^{t_{i_*}(n)}(x \mid r) \leq \frac{n+d}{I},$$

which is at most $O\big(\sqrt{n \log n}\big)$ for $I := \sqrt{n/\log n}$. Finally, by induction, for every $i \geq 0$, $t_i \leq f(n) \cdot 2^{c \cdot i \cdot \log n}$. The bound $t_{i_*-1} \leq f(n) \cdot 2^{O(\sqrt{n \log n})}$ follows. ◀

We will also need the following fine-grained version of Lemma 26.

▶ **Lemma 47** (Fine-Grained Symmetry of Information for pK^t). *If* $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$, *then there exists a polynomial* p *such that for all sufficiently large* $n, m \in \mathbb{N}$, *all* $t \in \mathbb{N}$ *such that* $t \geq 2^{(n \cdot m)^\varepsilon}$ *for some* $\varepsilon > 0$, *and all time-constructible* $0 \leq \tau(n,m) \leq t$, *the following holds: for every* $x \in \{0,1\}^n$ *and every family* $\left\{ y_r \in \{0,1\}^m \mid r \in \{0,1\}^{\tau(n,m)} \right\}$, *with probability at least* $9/10$ *over* $r \in \{0,1\}^{\tau(n,m)}$,

$$\mathsf{pK}^t(x, y_r \mid r) > \mathsf{pK}^{\widetilde{O}(t)}(x \mid r) + \mathsf{pK}^{\widetilde{O}(t)}(y_r \mid x, r) - \log p(t).$$

Proof Sketch. The proof is adapted from that of Lemma 26. Define

$$L := \left\{ \left(u, v, w, w', 1^t, 1^s, 1^n, 1^m\right) \middle| \exists\, \mathcal{M} \in \{0,1\}^s, \mathcal{M}(w, w') \text{ prints } uv \text{ within } 2t \text{ steps, where } |w'| = \tau(n,m) \leq t,\ |w| = 2t, \text{ and } s = |u| + |v| - 10 \right\}.$$

A distribution D will be given parameters $\langle nk + k, mk' + k', 4t, s, n, m \rangle =: N$. It will be defined to randomly sample $u \sim \mathcal{U}_{nk+k}$, $v \sim \mathcal{U}_{mk'+k'}$, $w \sim \mathcal{U}_{2t}$, and $w' \sim \mathcal{U}_{\tau(n,m)}$, and then output $(u, v, w, w', 1^t, 1^s, 1^n, 1^m)$.

The key observation is that L can be solved in NQL, so using our fine-grained average-case easiness assumption, we obtain a randomized heuristic algorithm B that solves the distributional problem (L, D) in quasilinear time. In particular, since $t \geq 2^{(n \cdot m)^\varepsilon}$ for some $\varepsilon > 0$, $N \leq \widetilde{O}(t)$. This ensures that B runs in time $\widetilde{O}(t)$ (see Proposition 16 and Definition 5).

The rest of the proof follows that of Lemma 26, except when B is used later on as a distinguisher, where we invoke the reconstruction lemma (Lemma 22), we get time bounds of $\widetilde{O}(t)$ instead of $\mathsf{poly}(t)$. ◀

Using similar ideas, we can get an analog of Lemma 33 for a special case of $\ell = 1$. We will use the following definitions, similar to Definition 32, which respectively allow for larger time-bounds and impose a separate length restriction on witnesses.

▶ **Definition 48** ($\mathsf{BP}_\delta \circ \mathsf{NTIME}$ and $\mathsf{BP}_\delta \circ \mathsf{NTIMEGUESS}$). *A language* L *is in the class* $\mathsf{BP}_\delta \circ \mathsf{NTIME}[\tau]$ *for a time bound* τ *if there is a formula*

$$\phi(x) = \mathsf{BP}r\, \exists\, y\ \ R(x, y, r)$$

for a predicate R *running in time* τ *and variables* y, r *of length at most* τ, *such that, for all* $x \in \{0,1\}^n$,

$$x \in L \implies \Pr_r\left[\exists\, y\ \ R(x,y,r)\right] \geq \delta, \text{ and}$$
$$x \notin L \implies \Pr_r\left[\forall\, y\ \ \neg R(x,y,r)\right] \geq \delta.$$

The class $\mathsf{BP}_\delta \circ \mathsf{NTIMEGUESS}[\tau, \ell]$ *is defined the same way as above, with an extra condition that* $|y| \leq \ell$ *(but the randomness* r *may still be of length at most* τ*).*

▶ **Lemma 49** (Fine-Grained Oracle Elimination). *Suppose* $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$. *Let* $\tau(n) = 2^{O(\sqrt{n \log n})}$ *be some time-constructible function and* $c > 0$ *a constant. For an arbitrary* $L \in \mathsf{BP}_\delta \circ \mathsf{NTIMEGUESS}[\tau(n), n^c]$, *let* $x \in L_n$ *be sufficiently large, let* $r \in \{0,1\}^{\tau(n)}$ *be any random string under the* BP *quantifier such that there exists a witness for* $x \in L$, *and let* $y_{x,r} \in \{0,1\}^{n^c}$ *be the lexicographically first such* L*-witness for* x *and randomness* r.

There exist polynomials q *and* q_0 *such that, for all sufficiently large* $n \in \mathbb{N}$, *all* $x \in L_n$, *and all* $t \geq q_0(\tau(n))$, *we have with probability at least* $\delta - 1/10$ *over* $r \in \{0,1\}^{\tau(n)}$ *that an* L*-witness* $y_{x,r}$ *exists, and*

$$\mathsf{pK}^{\widetilde{O}(t)}(x, y_{x,r} \mid r) \leq \mathsf{pK}^t(x \mid r) + \log q(t).$$

Proof Sketch. The proof is adapted from that of Lemma 33. Define

$$L' := \left\{ \left(\mathsf{DP}_k(x,y;z), w, w', 1^t, 1^k, 1^s, 1^n\right) \middle| \begin{array}{r} \exists\, \mathcal{M} \in \{0,1\}^s, \text{ such that } \mathcal{M}^{\mathsf{SAT}}(w,w') \text{ prints} \\ (x,y) \in \{0,1\}^{n+n^c} \text{ within } 2t \text{ steps, where} \\ |w'| = \tau(n) \leq t,\ |w| = 2t, \text{ and } s = k - 10 \end{array} \right\}.$$

Since SAT is NQL-complete under many-one QL reduction, we have, for $x \in \{0,1\}^n$, $y \in \{0,1\}^{n^c}$, $w \in \{0,1\}^{2t}$, and $w' \in \{0,1\}^{\tau(n)}$ that $L' \in \Sigma_2^{\mathsf{QL}}$.

Again, using our fine-grained average-case easiness assumption, we get a randomized heuristic algorithm B that solves L' in time $\widetilde{O}(t)$. Also note that given x and good randomness $r \in \{0,1\}^{\tau(n)}$, we can compute $y_{x,r}$ via search-to-decision reduction using a SAT oracle, which takes time $\mathsf{poly}(\tau) =: q_0(\tau)$. Therefore, similar to Equation (19) in the original proof, we get

$$\mathsf{pK}^{2t,\mathsf{SAT}}(x, y_{x,r} \mid r) \leq \mathsf{pK}^t(x \mid r) + d \log t \tag{30}$$

for some constant $d > 0$, provided $t \geq q_0(\tau)$. The rest of the proof is the same as that of Lemma 33, except when B is used later in the proof as a distinguisher, where we invoke the reconstruction lemma (Lemma 22), we get a time bound of $\widetilde{O}(t)$ instead of $\mathsf{poly}(t)$. ◀

We will also use the following simple generalization of (non-fine-grained) oracle elimination, Lemma 33, to higher time bounds.

▶ **Lemma 50.** *Suppose* $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$. *For an arbitrary* $L \in \mathsf{NTIME}[\tau(n)]$, *for some* $\tau(n) = 2^{O(\sqrt{n \log n})}$, *let* $x \in L_n$ *be sufficiently large, and let* $y_x \in \{0,1\}^{\tau(n)}$ *be the lexicographically first* L*-witness for* x. *There exist polynomials* q *and* q_0 *such that, for all* $t \geq q_0(\tau(n))$, *we have*

$$\mathsf{pK}^{q(t)}(x, y_x) \leq \mathsf{pK}^t(x) + \log q(t).$$

Proof Sketch. The proof follows that of the "Moreover" statement of Lemma 33, except the search-to-decision computation of y_x, where $|y_x| = \tau(n)$, now takes time at least $q_0(\tau(n))$ for some polynomial q_0, yielding

$$\mathsf{pK}^{2t,\mathsf{NP}}(x, y_{x,r} \mid r) \leq \mathsf{pK}^t(x \mid r) + d \log n$$

for $t \geq q_0(\tau(n))$ in Equation (19). ◀

We are now ready to show Theorem 45.

Proof of Theorem 45. Assume $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$. Let $\tau = 2^{a\sqrt{n \log n}}$ for some constant $a > 0$, and let $L \in \mathsf{AMTIME}[\tau]$ be arbitrary. That is, for every $x \in \{0,1\}^n$,

$$x \in L \implies \mathsf{BP}_{(1-2^{-n})} r\, \exists y\, V(x,y,r),$$
$$x \notin L \implies \mathsf{BP}_{(1-2^{-n})} r\, \forall y\, \neg V(x,y,r),$$

for a polytime verifier $V(x,y,r)$, where $|r|, |y| = \tau$. Let $x \in L_n$ and good randomness $r \in \{0,1\}^\tau$ be given, with lexicographically first witness $y_{x,r} \in \{0,1\}^\tau$.

Note that $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$ implies $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$ by a simple padding argument. Combining Lemma 50 and Lemma 26 as in the proof of Lemma 34, for some constant $b > 0$, with probability at least $1 - 2^{-n} - 1/5$ over $r \in \{0,1\}^\tau$,

$$\mathsf{pK}^{\tau^b}(y_x \mid x, r) \leq \left(\mathsf{pK}^{\tau}(x \mid r) - \mathsf{pK}^{\tau^b}(x \mid r)\right) + O(\log \tau), \tag{31}$$

from which it follows that $\mathsf{pK}^{\tau^b}(y_x \mid x, r) \leq 2n$ by the trivial upper bound on $\mathsf{pK}^{\tau}(x|r)$. In other words, by definition of pK,

$$\Pr_{r' \sim \{0,1\}^{\tau^b}} \left[\exists \mathcal{M} \in \{0,1\}^{2n},\ \mathcal{M}(x,r,r') \text{ outputs } y_x \text{ within time } \tau^b\right] \geq 2/3. \tag{32}$$

Define a verifier $V'(x,z,r,r')$, which does the following:

> Simulate the machine described by z on input (x,r,r') for τ^b steps to produce some output y_z, and then return the output of $V(x,y_z,r)$.

Note that V' runs in time τ^c for some constant $c > 0$. For $x \in L$, there exists $z \in \{0,1\}^{2n}$ such that $V'(x,z,r,r')$ so long as Equation (31) and Equation (32) both hold, which occurs with probability at least $(2/3)(1 - 2^{-n} - 1/5) =: \delta > 1/2$ over r, r'. For $x \notin L$, $V'(x,z,r,r')$ rejects unless there is an L-witness $y_{x,r}$ under the original verifier V, which occurs with probability at most 2^{-n}. This leads to the following characterization of L:

$$x \in L \implies \mathsf{BP}_{(\delta)} r, r'\, \exists z\, V'(x,z,r.r'),$$
$$x \notin L \implies \mathsf{BP}_{(1-2^{-n})} r, r'\, \forall z\, \neg V'(x,z,r,r'),$$

where $|r| = \tau$, $|r'| = \tau^b$, and $|z| = 2n$, which implies that $L \in \mathsf{BP}_\delta \circ \mathsf{NTIMEGUESS}[\tau^c, 2n]$.

Combining Lemma 47 and Lemma 49 as in the proof of Lemma 34, we have that with probability at least $\delta - 1/5 > 1/4$ over $(r,r') \in \{0,1\}^{\tau+\tau^b}$ that for all $t \geq p_0'(\tau^c)$, for some polynomials p' and p_0',

$$\mathsf{pK}^{t \cdot p'(n)}(z_{x,r,r'} \mid x, r, r') \leq \left(\mathsf{pK}^{t}(x \mid r, r') - \mathsf{pK}^{t \cdot p'(n)}(x \mid r, r')\right) + O(\log t) \tag{33}$$

where $z_{x,r,r'}$ is such that $V'(x, z_{x,r,r'}, r, r')$ accepts. Lemma 46 then implies the existence of some $t \in \mathbb{N}$ such that $p_0'(\tau^c) \leq t \leq 2^{O(\sqrt{n \log n})}$ and $\mathsf{pK}^t(x \mid r, r') - \mathsf{pK}^{t \cdot p'(n)}(x \mid r, r') \leq O(\sqrt{n \log n})$. So by Equation (33), for some constant $d > 0$,

$$\mathsf{pK}^{\tau^d}(z_{x,r} \mid x, r, r') \leq d\sqrt{n \log n}.$$

This suggests the following algorithm to decide L:

> Given $x \in \{0,1\}^n$, sample uniformly random strings $r \sim \{0,1\}^\tau$, $r' \in \{0,1\}^{\tau^b}$, and $w \in \{0,1\}^{\tau^d}$. Then exhaustively search over all $d\sqrt{n \log n}$-bit machines $\mathcal{M}$, running each $\mathcal{M}(x,r,r',w)$ for τ^d steps to produce a candidate witness z, accepting if $V'(x,z,r,r')$ accepts.

The runtime $\mathsf{poly}(\tau) = 2^{O(\sqrt{n \log n})}$ is clear. For correctness, for every $x \in L_n$, there is at least a $1/4$ fraction of good random strings r, r' such that Equation (33) holds, in which case a witness $z_{x,r,r'}$ will be found by our algorithm with probability at least $2/3$ over w (by definition of pK). So, our algorithm accepts $x \in L_n$ with probability at least $(1/4)(2/3) = 1/6$. On the other hand, for $x \notin L$, our algorithm will accept with probability at most 2^{-n}, as this upper-bounds the probability of r being good for the original verifier V. Thus, we have $L \in \mathsf{BPTIME}[2^{O}\sqrt{n \log n}]$ via standard success amplification. ◀

A proof of the analogous statement for the class $\mathsf{UTIME}[2^{O(\sqrt{n \log n})}]$, Item 1 of Theorem 2,

$$\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL} \implies \mathsf{UTIME}\left[2^{O(\sqrt{n \log n})}\right] \subseteq \mathsf{RTIME}\left[2^{O(\sqrt{n \log n})}\right],$$

is presented in Appendix B. Though its proof requires some additional ideas, it does not need to use an extremely efficient PRG as the one developed in [6].

Finally, we note that one immediately obtains the main results of [6], under *deterministic* average-case easiness assumptions, as corollaries of the theorems above via a simpler proof. For example, we will prove the following.

▶ **Theorem 51** ([6]). $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgQL} \implies \mathsf{AMTIME}\left[2^{O(\sqrt{n \log n})}\right] = \mathsf{DTIME}\left[2^{O(\sqrt{n \log n})}\right]$.

Proof. If $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgQL}$, then trivially $\Sigma_2^{\mathsf{QL}} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$, and by Theorem 45, we get that $\mathsf{AMTIME}[2^{O(\sqrt{n \log n})}] = \mathsf{BPTIME}[2^{O(\sqrt{n \log n})}]$. On the other hand, $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (implied by the assumption of the theorem) yields $\mathsf{BPP} = \mathsf{P}$ [5], which implies by padding that $\mathsf{BPTIME}[2^{O(\sqrt{n \log n})}] = \mathsf{DTIME}[2^{O(\sqrt{n \log n})}]$. ◀

▶ Remark 52. The main technical work in [6] was to construct a particular efficient PRG to argue that the DPG reconstruction yields a *quasilinear*-time *deterministic* Kolmogorov complexity. Instead, by using the probabilistic time-bounded Kolmogorov complexity measure pK, we forgo the need for a PRG within the DPG reconstruction lemma. This yields a *randomized* algorithm for any given language in $\mathsf{NTIME}[2^{O(\sqrt{n \log n})}]$. Finally, at the very end, we derandomize this randomized algorithm, with polynomial overhead, using a standard hardness-based PRG [37, 19], which is shown to exist by [5] assuming $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.

4.6 Time-computable heuristic schemes

One drawback of the worst-case to average-case reductions in Theorem 1 is that assuming average-case easiness for a class such as NP, we only get non-trivial worst-case algorithms for a smaller class, such as UP. A natural question then is whether we can get non-trivial worst-case algorithms for NP assuming only average-case easiness for NP. While this remains an interesting open problem, Hirahara [11] showed that such a worst-case to average-case reduction exists for NP if one considers a stronger notion of average-case easiness called P-*computable average-case polynomial time* (denoted as $\mathsf{Avg_P P}$), where, given an input x, the running time of the average-case algorithm on x can be efficiently estimated.

The proof of Hirahara's result built on ideas developed by [1], who showed that under a strong derandomization assumption, a language L is average-case easy *if and only if* it can be solved in time $2^{O\left(\mathsf{K}^{\mathsf{poly}(n)}(x) - \mathsf{K}(x) + \log n\right)}$ for every input $x \in \{0,1\}^n$. The proof of this result in [1] makes use of a fundamental theorem in Kolmogorov complexity called *language compression*, which states that for every (computable) language L, $\mathsf{K}(x) \lesssim \log |L \cap \{0,1\}^n|$

for all $x \in L \cap \{0,1\}^n$. The intuition is that for an average-case easy language, the set of "hard" instances that requires large running time must be small, and by the language compression theorem has low Kolmogorov complexity. This indicates that an instance with high (time-unbounded) Kolmogorov complexity has small running time.

Under the assumption $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, Hirahara proved a K^t version of the language compression theorem for languages in NP. By further assuming that one can efficiently estimate the running times of the average-case algorithms for NP, which is required to apply the ideas of [1] in the time-bounded case, he showed that NP can be solved in worst-case time $2^{O\left(\mathsf{K}^t(x)-\mathsf{K}^{\mathsf{poly}(t)}(x)+\log t\right)}$ for every input x and large enough t. As explained above, for every $x \in \{0,1\}^n$ there exists some $t \leq 2^{O(n/\log n)}$ such that $\mathsf{K}^t(x) - \mathsf{K}^{\mathsf{poly}(t)}(x) \leq O(n/\log n)$ so this yields non-trivial worst-case running times for NP. We remark that the reason why a language compression theorem only for NP is sufficient in the above argument is linked to the fact that K^t complexity can be checked in NP.

Here, we consider an analogy of $\mathsf{Avg_PP}$ in the randomized setting, called $\mathsf{Avg_{BPP}BPP}$, and we show that $\mathsf{DistNP} \subseteq \mathsf{Avg_{BPP}BPP}$ implies $\mathsf{NP} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$. While it is not surprising that our proof will require to show a pK^t version of the language compression theorem (under probabilistic average-case easiness of NP), there are also subtleties in terms of which language class that we need to compress. First of all, using ideas from previous sections, we can show that for every $L \in \mathsf{NP}$, we have $\mathsf{pK}^{\mathsf{poly}(n)}(x) \lesssim \log |L \cap \{0,1\}^n|$ for all $x \in L \cap \{0,1\}^n$. However, it turns out that such a language compression theorem for NP is not sufficient, and we actually need language compression for $\mathsf{promiseAM}$ (Section 3.6), again, due to the fact that pK^t is a more complicated notion and can only be estimated in $\mathsf{promiseAM}$.

▶ **Definition 53** ($\mathsf{Avg_{BPP}BPP}$). *For a language L and a family $D = \{D_n\}_{n\in\mathbb{N}}$ of distributions, we say that $(L, D) \in \mathsf{Avg_{BPP}BPP}$ if there exist a randomized algorithm A, a function $t\colon \{0,1\}^* \times \{1\}^* \to \mathbb{N}$ and constants $\varepsilon, b > 0$ such that for every $n \in \mathbb{N}$,*

1. $\mathbf{Pr}_A[A(x,1^n) = L(x)] \geq 2/3$, *for every $x \in \mathsf{Supp}(D_n)$,*
2. *$A(x,1^n)$ halts within time $t(x,1^n)$ (on each of its computational path), for every $x \in \mathsf{Supp}(D_n)$,*
3. $\mathbf{E}_{x\sim D_n}\left[\frac{t(x,1^n)^\varepsilon}{n}\right] \leq b$, *and*
4. *there is a polynomial-time randomized algorithm such that given x, 1^n and θ, decides whether $t(x,1^n) \leq \theta$.*

▶ **Theorem 54.** *If $\mathsf{DistNP} \subseteq \mathsf{Avg_{BPP}BPP}$, then $\mathsf{AM} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.*

For the proof of this result, we will also need the following weak symmetry of information, which follows as a simple corollary from Lemma 47 and Lemma 20.

▶ **Lemma 55** (Weak Symmetry of Information for pK^t). *If $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then there exist polynomials p_0 and p_w such that for every sufficiently large $x \in \{0,1\}^n$, $m \in \mathbb{N}$ and $p_0(n,m) \leq t \leq 2^{n+m}$,*

$$\Pr_{u\sim\{0,1\}^m}\left[\mathsf{pK}^t(x,u) > \mathsf{pK}^{t\cdot p_w(nm)}(x) + m - \log p_w(t)\right] \geq 0.99.$$

We are now ready to prove Theorem 54.

Proof of Theorem 54.

A "universal" distribution. We first define a distribution family $D = \{D_{\langle m,t_0\rangle}\}$, each of which does the following. On input $1^{\langle m,t_0\rangle}$,

1. Sample $s \in [2m]$,
2. Sample $w \in \{0,1\}^{t_0}$,
3. Sample $\mathcal{M} \in \{0,1\}^s$,
4. Run $\mathcal{M}(w)$ for t_0 steps, if it outputs a string y of length m, then output x; otherwise output 1^m.

Note that D is polynomial-time samplable. Moreover, it *dominates* the "universal distribution" for pK^{t_0}, which assigns each y the probability mass $2^{-\mathsf{pK}^{t_0}(y)}$. In particular, it is easy to see that for every $y \in \{0,1\}^m$,

$$D_{\langle m,t_0\rangle}(y) \geq \frac{2^{-\mathsf{pK}^{t_0}(y)}}{O(m)}. \tag{34}$$

Worst-case running times for NP. Let $L \in \mathsf{AM}$ be the language that we want to solve in the worst case. Let $L' \in \mathsf{NP}$ and $c > 0$ be a constant such that for every input $x \in \{0,1\}^n$ to L,

$$\mathop{\mathbf{Pr}}_{r\in\{0,1\}^{n^c}}[L'(x,r) = L(x)] \geq 1 - 1/n.$$

Therefore, to decide $L(x)$ (probabilisticaly), it suffices to decide $L'(x,r)$ for a typical r. For the rest of the proof, we will use x to denote an input to L and $y := (x,r)$ to L'.

By assumption, $(L', D) \in \mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$. Let A be an algorithm that solves L' on average with respect to D. Let $t_A\big(y, 1^{\langle m,t_0\rangle}\big)$ denote the running time of A on input $y \in \mathsf{Supp}(D_{\langle m,t_0\rangle})$. We will also write $t_A(y)$ instead of $t_A\big(y, 1^{\langle m,t_0\rangle}\big)$ when the input size to the distribution is clear in the context.

Let B be a $\mathsf{promiseBPP}$ algorithm that solves the promise problem from Lemma 27 and let τ be the polynomial there. Now, consider the following ensemble of promise problems:

$$\Pi_{\mathsf{YES}} := \Big\{ \Big(y, 1^{\langle m,i,s,t\rangle}\Big) \;\Big|\; |y| = m,\ t_A\Big(y, 1^{\langle m,\tau(t)\rangle}\Big) \in \big[2^i, 2^{i+1}\big] \text{ and } B\big(y, 1^s, 1^t\big) = 1 \text{ with probability} \geq 2/3 \Big\},$$

$$\Pi_{\mathsf{NO}} := \Big\{ \Big(y, 1^{\langle m,i,s,t\rangle}\Big) \;\Big|\; |y| = m,\ t_A\Big(y, 1^{\langle m,\tau(t)\rangle}\Big) \notin \big[2^i, 2^{i+1}\big] \text{ or } B\big(y, 1^s, 1^t\big) = 0 \text{ with probability} \geq 2/3 \Big\}.$$

Note that the above problem is in $\mathsf{PromiseBPP}$, since B is a $\mathsf{promiseBPP}$ algorithm and t_A is probabilistically polynomial-time checkable. Let

$$H_{\langle m,i,s,t\rangle} := \{0,1\}^m - \Pi_{\mathsf{NO},\langle m,i,s,t\rangle}.$$

Note that by property of B (recall Lemma 27), for every $y \in H_{\langle m,i,s,t\rangle}$, we have

$$\mathsf{pK}^{\tau(t)}(y) < s + \log \tau(t). \tag{35}$$

Now consider any $m, i, s, t \in \mathbb{N}$. Since A runs in time polynomial on average with respect to D, there are constants b and ε such that

$$\begin{aligned} b &\geq \sum_{y \in H_{\langle m,i,s,t\rangle}} \frac{t_A(y)^\varepsilon}{\langle m, \tau(t)\rangle} \cdot D_{\langle m,\tau(t)\rangle}(y) \\ &\geq \sum_{y \in H_{\langle m,i,s,t\rangle}} \frac{t_A(y)^\varepsilon}{\langle m, \tau(t)\rangle} \cdot \frac{2^{-\mathsf{pK}^{\tau(t)}(y)}}{O(m)} && \text{(Equation (34))} \\ &\geq |H_{\langle m,i,s,t\rangle}| \cdot \frac{2^{\varepsilon \cdot i}}{\tau(t)^3} \cdot 2^{-s - \log \tau(t)} && \text{(Equation (35))} \\ &= 2^{\varepsilon \cdot i + \log(|H_{\langle m,i,s,t\rangle}|) - s - 4\log \tau(t)}. \end{aligned}$$

By rearranging, we get that for every $x \in H_{\langle m,i,s,t\rangle}$,

$$\varepsilon \cdot i \leq s - \log |H_{\langle m,i,s,t\rangle}| + O(\log t). \tag{36}$$

Also, applying language compression (Theorem 30) to $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$, we have that for every $y \in \Pi_{\mathsf{YES},\langle n,i,s,t\rangle} \subseteq H_{\langle m,i,s,t\rangle}$,

$$\mathsf{pK}^{p_{\mathsf{LC}}(\langle m,i,s,t\rangle)}(y) \leq \log |H_{\langle m,i,s,t\rangle}| + \log p_{\mathsf{LC}}(\langle m, i, s, t\rangle), \tag{37}$$

where p_{LC} is some polynomial. Combining Equations (36) and (37), and using the fact that $t_A\big(y, 1^{\langle m,\tau(t)\rangle}\big) \leq 2^{i+1}$ for every $y \in \Pi_{\mathsf{YES},\langle m,i,s,t\rangle}$, we have for every such y,

$$t_A\Big(y, 1^{\langle m,\tau(t)\rangle}\Big) \leq 2^{\varepsilon^{-1} \cdot \left(s - \mathsf{pK}^{p_1(\langle m,i,s,t\rangle)}(y) + \log p_1(\langle m,i,s,t\rangle)\right)},$$

where p_1 is some polynomial.

Worst-case running times for AM. Now fix any input $x \in \{0,1\}^n$ to L, and let $m := |(x, r)|$ for $r \in \{0,1\}^{n^c}$. Consider any $r \in \{0,1\}^{n^c}$. Let $s := \mathsf{pK}^t(x, r) \leq m^2$ and let i be an integer such that

$$2^i \leq t_A\Big((x, r), 1^{\langle m,\tau(t)\rangle}\Big) \leq 2^{i+1}.$$

By definition, $(x, r) \in \Pi_{\mathsf{YES},\langle m,i,s,t\rangle}$. Note that $i \leq m^2$, since A runs in polynomial time on average and every string y of length m in the support of the distribution has probability mass at least $2^{-O(m)}$. Consequently, from the above, we have for every $t \geq m^2$,

$$t_A\Big((x, r), 1^{\langle m,\tau(t)\rangle}\Big) \leq 2^{O\left(\mathsf{pK}^t(x,r) - \mathsf{pK}^{p_2(t)}(x,r) + \log p_2(t)\right)}, \tag{38}$$

where p_2 is some polynomial. Now observe that by weak symmetry of information (Lemma 55), as long as t is greater than $q(m)$ for some polynomial q, with probability at least 0.99 over r (which we consider *good*), we have

$$\begin{aligned} &\mathsf{pK}^t(x, r) - \mathsf{pK}^{p_2(t)}(x, r) \\ &\leq \Big(\mathsf{pK}^{t/2}(x) + |r| + O(\log n)\Big) - \Big(\mathsf{pK}^{p_w(p_2(t))}(x) + |r| + \log p_w(p_2(t))\Big) \\ &\leq \mathsf{pK}^{t/2}(x) - \mathsf{pK}^{p_3(t/2)}(x) + \log p_3(t/2), \end{aligned} \tag{39}$$

where p_3 is some polynomial. Then by Equation (38), Equation (39), and Lemma 31, there exist a constant $d > 0$ and $t^* \leq 2^{dn/\log n}$ such that for every good r,

$$t_A\Big((x, r), 1^{\langle m,\tau(t^*)\rangle}\Big) \leq 2^{dn/\log n}.$$

This suggests the following randomized algorithm for solving L.

Given $x \in \{0,1\}^n$, randomly pick $r \sim \{0,1\}^{n^c}$ and let $m := |(x,r)|$. For $t = q(m), \ldots, 2^{dn/\log n}$, check if $t_A\big((x,r), 1^{\langle m, \tau(t)\rangle}\big) \leq 2^{dn/\log n}$. If so, run $A\big((x,r), 1^{\langle n, \tau(t^*)\rangle}\big)$ for at most $2^{dn/\log n}$ steps and output whatever it outputs.

It is easy to verify that the above probabilistic algorithm runs in time $2^{O(n/\log n)}$ and decides L. ◀

5 Learning in Randomized Heuristica

The main result of this section is the following.

▶ **Theorem 56.** *If* DistNP $\subseteq$ AvgBPP*, then for any time constructible functions* $s, T, a\colon \mathbb{N} \to \mathbb{N}$*, and* $\varepsilon \in [0,1]$*,* SIZE$[s(n)]$ *is agnostic learnable on* Samp$[T(n)]/a(n)$ *in time* $\mathsf{poly}\big(n, \varepsilon^{-1}, s(n), T(n), a(n)\big)$ *with sample complexity*

$$\left(\frac{(n + s(n) + a(n) + \log T(n))^3}{\varepsilon^8}\right)^{1+o(1)}.$$

5.1 Ingredients

In this section, we will use D to denote a family of distributions (one for each n) and $\mathcal{D}$ to denote a class of family of distributions (e.g., those that are efficiently samplable). For a distribution D and $m \in \mathbb{N}$, let D^m denote the distribution $x_1 \circ x_2 \circ \cdots \circ x_m$ where $x_1, x_2, \ldots, x_m \sim D$.

5.1.1 Learning from RRHS Refutation

▶ **Definition 57** (Correlative RRHS-Refutation [21, Definition 3])**.** *Let $\mathcal{C}$ be a concept class, and let $\mathcal{D}$ be a class of example distributions. A randomized algorithm A is a correlatively random-right-hand-side-refuter (correlative RRHS-refuter) for $\mathcal{C}$ on $\mathcal{D}$ with sample complexity m if A satisfies the following. A takes as input $n \in \mathbb{N}$, $\varepsilon \in (0,1)$ and a set $S = \big(\langle x^{(1)}, b^{(1)}\rangle, \ldots, \langle x^{(m)}, b^{(m)}\rangle\big)$ of samples, where $x^{(i)} \in \{0,1\}^n$ and $b^{(i)} \in \{0,1\}$ for every $i \in [m]$, and*

- ***Soundness:*** *if the samples S are i.i.d. from a distribution D' on $\{0,1\}^n \times \{0,1\}$ such that the marginal on $\{0,1\}^n$ equals D_n for some $D_n \in \mathcal{D}_n$ and there exists $f \in \mathcal{C}_n$ with*

$$\Pr_{\langle x^{(i)}, b^{(i)}\rangle \sim D'}\left[b^{(i)} = f(x^{(i)})\right] \geq \frac{1}{2} + \frac{\varepsilon}{2},$$

then

$$\Pr_{S,A}\left[A(n,\varepsilon,S) = \textit{correlative}\ \right] \geq 2/3.$$

- ***Completeness:*** *if the samples S are selected i.i.d. so that $x^{(1)}, \ldots, x^{(m)} \sim D_n$ for some $D_n \in \mathcal{D}_n$ and $b^{(1)}, \ldots, b^{(m)} \sim \mathcal{U}$, then*

$$\Pr_{S,A}\left[A(n,\varepsilon,S) = \textit{random}\ \right] \geq 2/3.$$

▶ **Theorem 58** (Agnostic Learning from RRHS-Refutation [21]; see also [13, Theorem 5])**.** *Let $\mathcal{C}$ be a concept class, and let $\mathcal{D}$ be a class of example distributions. If there exists a correlative RRHS-refuter for $\mathcal{C}$ on $\mathcal{D}$ with sample complexity $m(n,\varepsilon)$ and running time $T(n,\varepsilon)$, then $\mathcal{C}$ is agnostic learnable with*

$$\text{sample complexity} = O\left(\frac{m(n,\varepsilon/2)^3}{\varepsilon^2}\right) \quad \textit{and} \quad \text{running time} = O\left(T(n,\varepsilon/2) \cdot \frac{m(n,\varepsilon/2)^2}{\varepsilon^2}\right).$$

5.1.2 Probabilistic sampling depth

▶ **Definition 59** (Probabilistic Sampling Depth). *Let $t, t' \in \mathbb{N}$, where $t' > t$. For a family $D = \{D_n\}_{n\geq 1}$ of distributions, we introduce the (t,t')-*probabilistic-sampling-depth functions $\mathsf{psd}_D^{t,t'} := \left\{\mathsf{psd}_{D,n}^{t,t'}\right\}_{n\geq 1}$, *where*

$$\mathsf{psd}_{D,n}^{t,t'}(m) = \mathop{\mathbf{E}}_{X\sim D_n^m}\left[\mathsf{pK}^t(X) - \mathsf{pK}^{t'}(X)\right].$$

For a collection $\mathcal{D}$ of families of distributions, we define $\mathsf{psd}_{\mathcal{D}}^{t,t'} := \left\{\mathsf{psd}_{\mathcal{D},n}^{t,t'}\right\}_{n\geq 1}$, where

$$\mathsf{psd}_{\mathcal{D},n}^{t,t'}(m) = \max_{D\in\mathcal{D}} \mathsf{psd}_{D,n}^{t,t'}(m).$$

▶ **Lemma 60** (Small Probabilistic Sampling Depth for Samplable Distributions). *There exists a polynomial $p_1 \colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ such that for any $T, a \colon \mathbb{N}\to\mathbb{N}$ and $n, m \in \mathbb{N}$, the following holds. For every $t \geq p_1(T(n), m)$, and every $t' > t$,*

$$\mathsf{psd}_{\mathsf{Samp}[T(n)]/a(n),n}^{t,t'}(m) \leq O(\log m + \log T(n) + a(n) + \log t').$$

Proof. The proof is essentially the same as that of [13, Lemma 6], but uses the unconditional coding theorem for *probabilistic* Kolmogorov complexity (Lemma 25).

Fix any $D_n \in \mathsf{Samp}[T(n)]/a(n)$. Let p_1 be the polynomial p in Lemma 25. We have for every $t \geq p_1(T(n), m)$,

$$\begin{aligned}
\mathop{\mathbf{E}}_{x\sim D_n^m}\left[\mathsf{pK}^t(x)\right] &\leq \mathop{\mathbf{E}}_{x\sim D_n^m}\left[\mathsf{pK}^{p_1(T(n),m)}(x)\right] \\
&\leq \mathop{\mathbf{E}}_{x\sim D_n^m}\left[\log(1/D_n^m(x))\right] + O(\log(m) + \log T(n) + a(n)) && \text{(Lemma 25)} \\
&= H(D_n^m) + O(\log(m) + \log T(n) + a(n)) \\
&\leq \mathop{\mathbf{E}}_{x\sim D_n^m}\left[\mathsf{K}(x)\right] + O(\log(m) + \log T(n)) + a(n) \\
&\leq \mathop{\mathbf{E}}_{x\sim D_n^m}\left[\mathsf{pK}^{t'}(x)\right] + O(\log(m) + \log T(n) + a(n) + \log t'), && \text{(Lemma 18)}
\end{aligned}$$

where the second last inequality uses the fact that for any distribution D, its Shannon entropy $H(D) \leq \mathbf{E}_{y\sim D}\left[\mathsf{K}(y)\right]$ (see [27, Theorem 8.1.1] and [13, Lemma 5]). Rearranging the above, we get

$$\mathop{\mathbf{E}}_{x\sim D_n^m}\left[\mathsf{pK}^t(x) - \mathsf{pK}^{t'}(x)\right] \leq O(\log(m) + \log T(n) + a(n) + \log t'),$$

as desired. ◀

5.2 RRHS refuters from probabilistic average-case easiness

We first prove the following technical theorem.

▶ **Theorem 61.** *If $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then for any time constructible functions $s, T, a \colon \mathbb{N}\to\mathbb{N}$ and for any constant $\zeta > 0$, there is a correlative RRHS-refuter for $\mathsf{SIZE}[s(n)]$ under $\mathsf{Samp}[T(n)]/a(n)$ with sample complexity*

$$m := \left(\frac{n + s(n) + a(n) + \log T(n)}{\varepsilon^2}\right)^{1+\zeta}$$

and running time $\mathsf{poly}(n, m, T(n)) = \mathsf{poly}(n, 1/\varepsilon, T(n), a(n), s(n))$.

Proof. The proof closely follows that of [13, Theorem 8].

The (correlative) RRHS-refuter R. Let τ be the polynomial from Lemma 28. Let $\ell_s(n) \leq O(s(n)\log s(n))$ denote the number of bits needed to encode a function $f \in \mathsf{SIZE}[s(n)]$. On input $n \in \mathbb{N}$, $\varepsilon > 0$, and a set $S = \left(\left\langle x^{(1)}, b^{(1)}\right\rangle, \ldots, \left\langle x^{(m)}, b^{(m)}\right\rangle\right)$ of samples, R operates as follows.

1. Compute $t := \max\{p_0\left(n \cdot m^2\right), p_1(T(n), m)\}$, where p_0 is the polynomial from Lemma 26 and p_1 is the polynomial from Lemma 60. Also compute $t' := t'(t)$, where t' is some polynomial specified later (in Claim 62).
2. Compute

$$\beta := \mathsf{Approx}_{\tau}\text{-}\mathsf{pK}\left(X, 1^t\right), \text{ and}$$
$$\beta' := \mathsf{Approx}_{\tau}\text{-}\mathsf{pK}\left(X \circ b, 1^{t'}\right),$$

   where $X := x^{(1)} \circ \cdots \circ x^{(m)}$, $b := b^{(1)} \circ \cdots \circ b^{(m)}$, and $\mathsf{Approx}_{\tau}\text{-}\mathsf{pK}$ is the randomized algorithm from Lemma 28.
3. Output "correlative" if $\beta' - \beta \leq \theta$, where $\theta := m(1-\varepsilon^2/8) + \ell_s(n) + n + \log\tau(t)$, and output "random" otherwise.

It is easy to verify that the running time of R is $\mathsf{poly}(n, m, T(n))$. Next, we argue its correctness.

Soundness. Suppose we are in the "correlative" case. That is, the labelled examples in S are sampled i.i.d from some distribution D' on $\{0,1\}^n \times \{0,1\}$, whose marginal on $\{0,1\}^n$ is given by some $D \in \mathsf{Samp}[T(n)]/a(n)$, and there exists $f \in \mathsf{SIZE}[s(n)]$ such that

$$\mathop{\mathbf{Pr}}_{\left\langle x^{(i)}, b^{(i)}\right\rangle \sim D'}\left[b^{(i)} = f(x^{(i)})\right] \geq \frac{1}{2} + \frac{\varepsilon}{2}.$$

In this case, by a standard concentration bound, the probability over $S \sim (D')^m$ that

$$\left|\{i \in [m] \mid b_i = f(x^{(i)})\}\right| < (1/2 + \varepsilon/4) \cdot m$$

is at most $\exp(-2m(\varepsilon/4)^2) \leq o(1)$, where the last inequality relies on our choice of m. Now observe the following.

▷ Claim 62. There exists a polynomial t' such that for any $b \in \{0,1\}^m$ satisfying

$$\left|\{i \in [m] \mid b_i = f(x^{(i)})\}\right| \geq (1/2 + \varepsilon/4) \cdot m,$$

we have

$$\mathsf{pK}^{t'(t)}(X \circ b) \leq \mathsf{pK}^{\tau(t)}(X) + \ell_s(n) + \left(1 - \varepsilon^2/8\right) \cdot m.$$

Proof of Claim 62. Note that given X, we can compute $f(x^{(1)}), \ldots, f(x^{(m)})$ in time $\mathsf{poly}(m \cdot \ell_s(n))$ using the encoding of f, which is of $\ell_s(n)$ bits. Note that b and $f(x^{(1)}), \ldots, f(x^{(m)})$ disagree on at most $(1/2 - \varepsilon/4) \cdot m$ coordinates. Then to recover b, we can define a string $e \in \{0,1\}^m$ such that $e_i = 1$ iff $f(x^{(i)}) \neq b_i$. Note that e has hamming weight at most $(1/2 - \varepsilon/4) \cdot m$. Using the inequality

$$\sum_{i=0}^{k} \binom{m}{i} \leq 2^{H_2(k/m) \cdot m},$$

where H_2 is the binary entropy function, e can be encoded with $H_2(1/2+\varepsilon/4)\cdot m$ bits, by lexicographic indexing among binary strings of bounded hamming weight. Using the Taylor series of H_2 in the neighborhood of $1/2$, for $\delta := \varepsilon/4$, we have

$$H_2(1/2+\delta) = 1 - \frac{1}{2\ln 2}\sum_{i=1}^{\infty}\frac{(2\delta)^{2i}}{i(2i-1)} \leq 1 - \frac{2}{\ln 2}\delta^2 \leq 1 - 2\delta^2.$$

Moreover, using such an encoding, it is not hard to see that e can be reconstructed in time $\mathsf{poly}(n,m)$. Therefore, given X, we can compute b in time $\mathsf{poly}(n,m)$ using strings of length $\ell_s(n)$ (which encodes f) and $(1-\varepsilon^2/8)\cdot m$ (which encodes e). In other words, if X has probabilistic $\tau(t)$-time-bounded Kolmogorov complexity k, then

$$\mathsf{pK}^{t'(t)}(X \circ b) \leq k + \ell_s(n) + \left(1-\varepsilon^2/8\right)\cdot m,$$

for some $t'(t) := \mathsf{poly}(\tau(t))$. This completes the proof of the claim. ◁

Now, suppose in Step 2 of R, β and β' output by the algorithm $\mathsf{Approx}_{\tau}\text{-}\mathsf{pK}$ are good approximations as given in Lemma 28. Note that this happen with high probability. Then by a union bound, with probability at least $2/3$, over the samples $S \sim (D')^m$ and the internal randomness of R, we have

$$\begin{aligned}
\beta' &\leq \mathsf{pK}^{t'}(X \circ b) && (\beta' \text{ is a good approximation})\\
&\leq \mathsf{pK}^{\tau(t)}(X) + \ell_s(n) + \left(1-\varepsilon^2/8\right)\cdot m && (\text{Claim 62})\\
&\leq \beta + \log\tau(t) + \ell_s(n) + \left(1-\varepsilon^2/8\right)\cdot m, && (\beta \text{ is a good approximation})
\end{aligned}$$

which implies

$$\beta' - \beta \leq \left(1-\varepsilon^2/8\right)\cdot m + \ell_s(n) + n + \log\tau(t) = \theta,$$

and R will output "correlative".

Completeness. Suppose we are in the "random" case. That is, b is sampled from $\mathcal{U}_m$. Assuming $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, by combining symmetry of information (Lemma 26) and probabilistic incompressibility (Lemma 20), for any $X \in \{0,1\}^{nm}$ and some polynomial p_w, we have

$$\Pr_{b\sim\mathcal{U}_m}\left[\mathsf{pK}^{\tau(t')}(X\circ b) \geq \mathsf{pK}^{p_w(\tau(t'))}(X) + (m-10) - \log p_w(\tau(t'))\right] \geq 1 - 1/8. \tag{40}$$

Let us define

$$p_\tau(t) := p_w(\tau(t'(t))).$$

By Markov, we have

$$\begin{aligned}
&\Pr_{X\sim D^m}\left[\mathsf{pK}^t(X) - \mathsf{pK}^{p_\tau(t)}(X) > 8\cdot \mathsf{psd}^{t,p_\tau(t)}_{D,n}(m)\right]\\
&\leq \frac{\mathbf{E}_{X\sim D^m}\left[\mathsf{pK}^t(X) - \mathsf{pK}^{p_\tau(t)}(X)\right]}{8\cdot \mathsf{psd}^{t,p_\tau(t)}_{D,n}(m)} = \frac{1}{8}.
\end{aligned} \tag{41}$$

Again, let us assume that in Step 2 of R, β and β' output by the algorithm $\mathsf{Approx}_{\tau}\text{-}\mathsf{pK}$ are good approximations. Then with with probability at least $1-(1/8+1/8+o(1)) \geq 2/3$ over $b \sim \mathcal{U}_m$, $X \sim D^m$ and the internal randomness of R, we have

$$\begin{aligned}
\beta' &\geq \mathsf{pK}^{\tau(t')}(X \circ b) - \log \tau(t') && (\beta' \text{ is a good approximation}) \\
&\geq \mathsf{pK}^{p_w\left(\tau(t')\right)}(X) + m - \log\left(p_w\left(\tau(t')\right) \cdot \tau(t')\right) && (\text{Equation (40)}) \\
&= \mathsf{pK}^{t}(X) - \left(\mathsf{pK}^{t}(X) - \mathsf{pK}^{p_\tau(t)}(X)\right) + m - 10 - \log(p_\tau(t) \cdot \tau(t')) \\
&\geq \beta - 8 \cdot \mathsf{psd}_{D,n}^{t,p_\tau(t)}(m) + m - 10 - \log(p_\tau(t) \cdot \tau(t'))\,, && (\text{Equation (41)})
\end{aligned}$$

which implies

$$\beta' - \beta \geq m - 8 \cdot \mathsf{psd}_{D,n}^{t,p_\tau(t)}(m) - \log(p_\tau(t) \cdot \tau(t')) - 10.$$

Now we want the above to be greater than θ. We have

$$\begin{aligned}
&(\beta' - \beta) - \theta \\
&\geq \left(m - 8 \cdot \mathsf{psd}_{D,n}^{t,p_\tau(t)}(m) - \log(p_\tau(t) \cdot \tau(t')) - 10\right) \\
&\quad - \left(m + \ell_s(n) + n + \log \tau(t) - m\varepsilon^2/8\right) \\
&= m\varepsilon^2/8 - \left(n + \ell_s(n) + 8 \cdot \mathsf{psd}_{D,n}^{t,p_\tau(t)}(m) + \log(p_\tau(t) \cdot \tau(t') \cdot \tau(t)) + 10\right) \\
&\geq m\varepsilon^2/8 - \left[n + \ell_s(n) + c \cdot (\log m + \log T(n) + a(n)) + \log p_2(n, m, T(n)) + 10\right], \\
&\qquad\qquad (\text{Lemma 60})
\end{aligned}$$

where $c > 0$ is some constant and p_2 is some polynomial (that depends on τ, p_0, p_1 and p_w). From here, it can be easily seen that the above is greater than 0 by our choice of m, provided that n is sufficiently large. Hence R will output "random". This completes the proof of the theorem. ◀

5.3 Concluding the proof

We are now ready to prove Theorem 56.

▶ **Theorem 63** (Reminder of Theorem 56). *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, *then for any time constructible functions* $s, T, a\colon \mathbb{N} \to \mathbb{N}$, *and* $\varepsilon \in [0,1]$, $\mathsf{SIZE}[s(n)]$ *is agnostic learnable on* $\mathsf{Samp}[T(n)]/a(n)$ *in time* $\mathsf{poly}\left(n, \varepsilon^{-1}, s(n), T(n), a(n)\right)$ *with sample complexity*

$$\left(\frac{(n + s(n) + a(n) + \log T(n))^3}{\varepsilon^8}\right)^{1+\zeta},$$

where $\zeta > 0$ *is any arbitrary small constant.*

Proof. The theorem follows by combining Theorem 61 and Theorem 58. ◀

References

1 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 298–303. IEEE Computer Society, 2009. doi:10.1109/CCC.2009.12.

2 Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992. doi:10.1016/0022-0000(92)90019-F.

3 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006. doi:10.1561/0400000004.

4 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.

5 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.

6 Lijie Chen, Shuichi Hirahara, and Neekon Vafa. Average-case hardness of NP and PH from worst-case fine-grained assumptions. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 45:1–45:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.45.

7 Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022. URL: https://eccc.weizmann.ac.il/report/2022/007/.

8 Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. doi:10.1017/CBO9780511546891.

9 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 25–32, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73010.

10 Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 50–60. IEEE, 2020. doi:10.1109/FOCS46700.2020.00014.

11 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 292–302. ACM, 2021. doi:10.1145/3406325.3451065.

12 Shuichi Hirahara. Symmetry of information in heuristica. Manuscript, 2021.

13 Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized heuristica. In *Symposium on Foundations of Computer Science* (FOCS), 2021.

14 Shuichi Hirahara and Rahul Santhanam. Errorless versus error-prone average-case complexity. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 84:1–84:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.84.

15 Shuichi Hirahara and Rahul Santhanam. Excluding PH pessiland. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 85:1–85:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.85.

16 Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.

17 Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 104–114, 2011. doi:10.1109/CCC.2011.34.

18 Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89604.

19 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.

20 Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Mach. Learn.*, 17(2-3):115–141, 1994. doi:10.1007/BF00993468.

21 Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 55:1–55:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.55.

22 Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983. doi:10.1016/0020-0190(83)90044-3.

23 Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

24 Leonid A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

25 Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.

26 Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2019.

27 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019. doi:10.1007/978-3-030-11298-1.

28 Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.

29 Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. *Electron. Colloquium Comput. Complex.*, page 59, 2021. URL: https://eccc.weizmann.ac.il/report/2021/059.

30 Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on exp$\neq$ BPP. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, pages 11–40, 2021. doi:10.1007/978-3-030-84242-0_2.

31 Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. *Inf. Comput.*, 121(1):14–22, 1995. doi:10.1006/inco.1995.1120.

32 Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 94:1–94:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.94.

33 Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 303–316. ACM, 2021. doi:10.1145/3406325.3451085.

34 Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded kolmogorov complexity. In *49th International Colloquium on Automata, Languages and Programming*, 2022.

35 Peter Bro Miltersen, N. V. Vinodchandran, and Osamu Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In Takao Asano, Hiroshi Imai, D. T. Lee, Shin-Ichi Nakano, and Takeshi Tokuyama, editors, *Computing and Combinatorics, 5th Annual International Conference, COCOON '99, Tokyo, Japan, July 26-28, 1999, Proceedings*, volume 1627 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 1999. doi:10.1007/3-540-48686-0_21.

36 Ashish V. Naik, Kenneth W. Regan, and D. Sivakumar. On quasilinear-time complexity theory. *Theor. Comput. Sci.*, 148(2):325–349, 1995. doi:10.1016/0304-3975(95)00031-Q.

37 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.

38 Igor C. Oliveira. Randomness and intractability in kolmogorov complexity. 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 32:1–32:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.32.

39 Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. 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 665–677. ACM, 2017. doi:10.1145/3055399.3055500.

40 Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, pages 35:1–35:58, 2021. doi:10.4230/LIPIcs.CCC.2021.35.

41 Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014. URL: http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms.

42 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005. doi:10.1145/1059513.1059516.

43 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

44 Salil P. Vadhan. On learning vs. refutation. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, volume 65 of *Proceedings of Machine Learning Research*, pages 1835–1848. PMLR, 2017. URL: http://proceedings.mlr.press/v65/vadhan17a.html.

45 Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 183–197, 2005. doi:10.1109/CCC.2005.16.

A Additional Properties of pK^t Complexity

A.1 Upper bound on the complexity of estimating pK^t

The following says that the problem of estimating the pK^t complexity of a given string for a given t is in promise AM.

▶ **Definition 64** (Gap-MINpKT). *For a function $\tau\colon \mathbb{N} \to \mathbb{N}$, let Gap_τ-MINpKT be the following promise problem $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$:*

$$\Pi_{\mathsf{YES}} := \left\{ (x, 1^s, 1^t) \mid \mathsf{pK}^t(x) \leq s \right\},$$
$$\Pi_{\mathsf{NO}} := \left\{ (x, 1^s, 1^t) \mid \mathsf{pK}^{\tau(t)}(x) > s + \log \tau(t) \right\}.$$

▶ **Proposition 65.** *There is a constant $c \geq 1$ for which the following holds. For any function τ such that $\tau(t) \geq c \cdot t$, Gap_τ-MINpKT $\in$ promiseAM.*

Proof. We first observe that the following problem $(\Pi'_{\mathsf{YES}}, \Pi'_{\mathsf{NO}})$ is in promiseAM.

$$\Pi'_{\mathsf{YES}} = \left\{ (x, 1^s, 1^t) \mid \mathsf{pK}^t_{2/3}(x) \leq s \right\},$$
$$\Pi'_{\mathsf{NO}} = \left\{ (x, 1^s, 1^t) \mid \mathsf{pK}^t_{1/3}(x) > s \right\}.$$

To solve the above problem, on input x, the verifier first sends a random $w \in \{0,1\}^t$ to the prover, who then sends back a program $\mathcal{M}_w$. Finally, the verifier accepts if and only if $|\mathcal{M}_w| \leq s$ and $\mathcal{M}_w(w)$ outputs x within t steps. It is easy to see that if $x \in \Pi'_{\mathsf{YES}}$, then the above protocol accepts with probability at least $2/3$. If $x \in \Pi'_{\mathsf{NO}}$, then the fraction of $w \in \{0,1\}^t$ such that x can be generated by some size-s program in time t is less than $1/3$ (otherwise $\mathsf{pK}^t_{1/3}(x)$ would be at most s), so the protocol accepts with probability less than $1/3$. Now the proposition follows from Lemma 21, which implies that the set Π_{NO} of NO instances in Gap_τ-MINpKT satisfies $\Pi_{\mathsf{NO}} \subseteq \Pi'_{\mathsf{NO}}$, provided that c is a large enough constant. ◀

A.2 Relations between time-bounded Kolmogorov complexity notions

In this section, we present the tight relations between K^t, rK^t, and pK^t (see Section 2.5) that hold under derandomization assumptions.

▶ **Proposition 66.** *The following results hold.*

- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{rK}^t(x) + \log p(t)$, *for every* t *and* x.
- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{pK}^t(x) + \log p(t)$, *for every* t *and* x.
- *If* $\mathsf{BPE} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{rK}^{p(t)}(x) \leq \mathsf{pK}^t(x) + \log p(t)$, *for every* t *and* x.

Proof. Let x be any string $\{0,1\}^n$ and t be any integer such that $t \geq n$.

For the first item, note that the assumption $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$ implies that there is a PRG $G\colon \{0,1\}^{O(\log s)} \to \{0,1\}^s$ that $(1/s)$-fools size-s circuits and has running time $\mathsf{poly}(s)$ [19]. Suppose $\mathsf{rK}^t(x) \leq k$. Let $\mathcal{M} \in \{0,1\}^k$ be a randomized program with running time t that outputs x with probability at least $2/3$. Consider the following function C on inputs of length t:

$$C(w) = 1 \iff \mathcal{M}(w) = x.$$

It is clear that C can be implemented as a $\mathsf{poly}(t)$-size circuit, and by definition the acceptance probability of C is at least $2/3$. Then there exists some seed $z \in \{0,1\}^{O(\log t)}$ such that $C(G(z)) = 1$, which implies $\mathcal{M}(G(z)) = x$. This means given $\mathcal{M}$ and z, we can deterministically compute x in time $\mathsf{poly}(t)$. In other words, $\mathsf{K}^{\mathsf{poly}(t)} \leq k + O(\log t)$.

Next, we show the second item. The assumption $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$ implies that there is a PRG $G'\colon \{0,1\}^{O(\log s)} \to \{0,1\}^s$ that $(1/s)$-fools size-s nondeterministic circuits and has running time $\mathsf{poly}(s)$ [42]. Now suppose $\mathsf{pK}^t(x) \leq k$. Consider the following function C' on t bits.

$$C'(w) = 1 \iff \exists\, \mathcal{M} \in \{0,1\}^k \text{ such that } \mathcal{M}(w) \text{ outputs } x \text{ within } t \text{ steps.}$$

It is easy to see that C' can be implemented as a size-$\mathsf{poly}(t)$ nondeterministic circuit. Then there exists some seed $z \in \{0,1\}^{O(\log t)}$ such that $C'(G'(z)) = 1$. This means for such z, there exists some program $\mathcal{M}' \in \{0,1\}^k$ (which can depend on z) such that $\mathcal{M}'(G'(z))$ runs t steps and outputs x. Then given such z and $\mathcal{M}'$, we can computes x in time $\mathsf{poly}(t)$, by first computing $w := G'(z)$ and executing $\mathcal{M}(w)$. This yields $\mathsf{pK}^{\mathsf{poly}(t)}(x) \leq k + O(\log t)$.

The proof of the third item is essentially the same as that of the second item. The only difference here is that the assumption $\mathsf{BPE} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$ implies a *pseudodeterministic* PRG that fools size-s nondeterministic circuits with seed length $O(\log s)$, which follows from

the construction of the PRG in [42]. Therefore, at the end of the proof for the second item above, we can successfully compute $w := G'(z)$ with high probability (over the internal randomness of the algorithm computing G'), and hence we get $\mathsf{rK}^{\mathsf{poly}(t)}(x) \leq k + O(\log t)$. ◀

B Probabilistic Fine-Grained Reduction for UTIME$\left[2^{O(\sqrt{n \log n})}\right]$

▶ **Theorem 67.** $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL} \implies \mathsf{UTIME}\left[2^{O(\sqrt{n \log n})}\right] \subseteq \mathsf{RTIME}\left[2^{O(\sqrt{n \log n})}\right]$.

We will need the following fine-grained version of Lemma 27.

▶ **Lemma 68.** *If* $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$*, then there exists a polynomial* τ *such that the following promise problem is in* $\mathsf{promiseBPP}$*:*

$$\Pi_{\mathsf{YES}} := \left\{ \left(x, 1^s, 1^t\right) \mid \mathsf{pK}^t(x) \leq s \right\},$$
$$\Pi_{\mathsf{NO}} := \left\{ \left(x, 1^s, 1^t\right) \mid \mathsf{pK}^{\widetilde{O}(t) \cdot \tau(|x|)}(x) > s + \log \tau(t) \right\}.$$

Proof. Define

$$L := \left\{ \left(\mathsf{DP}_k(x; z), w, 1^s, 1^n\right) \middle| \exists \mathcal{M} \in \{0,1\}^s,\ \mathcal{M}(w) \text{ outputs } x \in \{0,1\}^n \text{ within } |w| \text{ steps} \right\},$$

where $c > 0$ is some large constant so that $L \in \mathsf{NQL}$. Define a distribution family

$$D := \left\{ D_{\langle nk+k, t, s, n \rangle} \right\},$$

each member of which does the following: sample $u \sim \mathcal{U}_{nk+k}$ and $w \sim \mathcal{U}_t$ and then output $(u, w, 1^s, 1^n)$. (Again, for the simplicity of presentation, we omit the padding, which is of length at most $\widetilde{O}(t) \cdot \mathsf{poly}(nk)$.) By assumption, $(L, D) \in \mathsf{AvgBPQL}$. Let B be a randomized heuristic algorithm for (L, D) as described in Lemma 6. Now, define an algorithm B':

> On input $(x, 1^s, 1^t)$ with $x \in \{0,1\}^n$, set $k = s + 10$, sample $z \sim \mathcal{U}_{nk}$ and $w \sim \mathcal{U}_t$, and then output $B(\mathsf{DP}_k(x; z), w, 1^s, 1^n)$.

Note that B' runs in time

$$t_D := \widetilde{O}(t) \cdot p(n)$$

for some polynomial p. Below, we show that B' solves the mentioned promise problem correctly with high probability in the worst case.

First, consider the case that $(x, 1^s, 1^t) \in \Pi_{\mathsf{YES}}$. By the definitions of L and pK, for any choice of $z \in \{0,1\}^{nk}$,

$$\Pr_w \left[\left(\mathsf{DP}_k(x; z), w, 1^s, 1^n\right) \in L \right] \geq 2/3.$$

The definition of B then implies that

$$\Pr_{w, z, r_B} \left[B(\mathsf{DP}_k(x; z), w, 1^s, 1^n) = 1 \right] > 1/2,$$

and so

$$\Pr_{r_{B'}} \left[B'(x, 1^s, 1^t) = 1 \right] > 1/2, \tag{42}$$

where r_B denotes the internal randomness of B, and $r_{B'} = (w, z, r_B)$ that of B'.

Now consider the case that $(x, 1^s, 1^t) \in \Pi_{\mathsf{NO}}$. For a contradiction, suppose that

$$\Pr_{w,z,r_B}[B(\mathsf{DP}_k(x;z), w, 1^s, 1^n) = 1] > 1/3. \tag{43}$$

Recall $k = s + 10$. By a counting argument, for randomly selected u and w,

$$\Pr_{u,w}[(u, w, 1^s, 1^n) \in L] \leq \frac{2^s \cdot 2^{nk} \cdot 2^t}{2^{nk+k+t}} = \frac{1}{10}.$$

Then by definition of B,

$$\Pr_{u,w,r_B}[B(u, w, 1^s, 1^n) = 1] \leq 1/9. \tag{44}$$

Comparing Equations (43) and (44), we see that $B(-, \mathcal{U}_t, 1^s, 1^n)$ (2/9)-distinguishes the distribution $\mathsf{DP}_k(x;\mathcal{U}_{nk})$ from uniform. Lemma 22 implies that

$$\begin{aligned}\mathsf{pK}^{\widetilde{O}(t)\cdot\tau_0(n)}(x) &\leq k + \log \tau_0(t)\\ &= s + 10 + \log \tau_0(t),\end{aligned}$$

for some polynomial τ_0 that depends on the polynomials p and p_{DP} from Lemma 22. This means that $(x, 1^s, 1^t)$ is *not* in Π_{NO}, by setting τ properly. This gives the desired contradiction. By definition of B', we have that

$$\Pr_{r_{B'}}\left[B'(x, 1^s, 1^t) = 1\right] \leq 1/3. \tag{45}$$

By Equations (13) and (16), B' yields a promiseBPP algorithm for $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ via standard error reduction techniques. ◀

The following is a simple corollary of Lemma 47 and Lemma 20.

▶ **Lemma 69** (Fine-Grained Weak Symmetry of Information for pK^t)**.** *If* $\mathsf{NQL} \times \mathsf{QLSamp} \subseteq \mathsf{AvgBPQL}$*, then there exist polynomials* p_0 *and* p *such that for every sufficiently large* $x \in \{0,1\}^n$*,* $m \in \mathbb{N}$ *and* $p_0(n, m) \leq t \leq 2^{n+m}$*,*

$$\Pr_{u\sim\{0,1\}^m}\left[\mathsf{pK}^t(x \circ u) > \mathsf{pK}^{t\cdot p(nm)}(x) + m - \log p(t)\right] \geq 0.99.$$

We will use the following black-box hitting set generator whose outputs can be small relative to the "hard string" used in its construction.

▶ **Lemma 70** ([10, Theorem 4.3])**.** *For any* $T, m \in \mathbb{N}$ *with* $m \leq 2T$*, there exists a function* $H_{T,m}\colon \{0,1\}^T \times \{0,1\}^d \to \{0,1\}^m$ *and a deterministic procedure* $\mathsf{Recon}^{(-)}\colon \{0,1\}^a \to \{0,1\}^T$*, where* $d = O(\log T + \log^3 m)$ *and* $a = 2m + O(\log T + \log^3 m)$ *such that, for any* $x \in \{0,1\}^T$ *and any function* $D\colon \{0,1\}^m \to \{0,1\}$ *that* 0.1*-avoids* $H_{T,m}(x, -)$*, there exists advice* $\alpha \in \{0,1\}^a$ *such that* $\mathsf{Recon}^D(\alpha) = x$*. Moreover,* $H_{T,m}$ *can be computed in time* $\mathsf{poly}(T)$ *and* Recon^D *can be computed in time* $\mathsf{poly}(T)$ *with oracle access to* D*.*

We are now ready to show Theorem 67.

Proof of Theorem 67. Let $L \in \mathsf{UTIME}[T(n)]$ with verifier V, where $T(n) = 2^{O(\sqrt{n \log n})}$. Fix an input $x \in \{0,1\}^n$. Let $y_x \in \{0,1\}^{T(n)}$ be the unique L-witness for x. Let B be a promiseBPP algorithm that solves the promise problem $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ from Lemma 68.

Let t be *any* number such that $q_0(T(n)) \leq t \leq 2^{n/2}$, where q_0 is some polynomial specified later. Let $H := H_{T(n),m}$ be the hitting set generator from Lemma 70, where $m := m_x(t) \leq O(n)$ is defined later, and let $d := O\big(\log T(n) + \log^3 m\big)$ be the seed length of H.

Using H and B, we will argue that the time-bounded pK complexity of y_x is small given x. First note the following.

▷ Claim 71. There exists a polynomial q such that for every $q_0(T(n)) \leq t \leq 2^{n/2}$ and every $z_H \in \{0,1\}^d$,

$$\mathsf{pK}^{t\cdot q(nm)}\left(x \circ H(y_x; z_H)\right) \leq \mathsf{pK}^t(x) + d + \log q(t).$$

Proof of Claim 71. Let L be the following language

$$L' := \left\{ \left(\mathsf{DP}_k(x \circ H(y; z'); z), w01^{T(n)^c}, 1^s, 1^n, 1^m\right) \;\middle|\; \exists\, \mathcal{M} \in \{0,1\}^s, \mathcal{M}(w) \text{ outputs } x \in \{0,1\}^n \right.$$
$$\left. \text{within } |w| \text{ steps, and } V(x,y) = 1 \right\},$$

where $c > 0$ is some constant so that $L' \in \mathsf{NQL}$. Define a distribution family

$$D := \left\{D_{\langle (n+m)k+k, t+T(n)^c+1, s, n, m\rangle}\right\},$$

each member of which samples $u \sim \mathcal{U}_{(n+m)k+k}$, $w \sim \mathcal{U}_t$ and outputs

$$\left(u, w01^{T(n)^c}, 1^s, 1^n, 1^m\right).$$

By assumption, $(L', D) \in \mathsf{AvgBPQL}$. Let B' be a randomized heuristic algorithm for (L', D) as described in Lemma 6.

Define $s := \mathsf{pK}^t(x)$. By definition of pK^t and L', for any fixed $z \in \{0,1\}^{(n+m)k}$,

$$\mathop{\mathbf{Pr}}_{w}\left[\left(\mathsf{DP}_k(x \circ H(y_x; z_H); z), w01^{T(n)^c}, 1^s, 1^n, 1^m\right) \in L'\right] \geq 2/3.$$

Then by definition of B', it is easy to see that

$$\mathop{\mathbf{Pr}}_{z,w,r'_B}\left[B'\left(\mathsf{DP}_k(x \circ H(y_x; z_H); z), w01^{T(n)^c}, 1^s, 1^n, 1^m\right) = 1\right] \geq 1/2. \tag{46}$$

On the other hand, by a counting argument it is easy to see that for u and w selected uniformly at random, we have

$$\mathop{\mathbf{Pr}}_{u,w}\left[\left(u, w01^{T(n)^c}, 1^s, 1^n, 1^m\right) \in L'\right] \leq \frac{2^s \cdot 2^t \cdot 2^d \cdot 2^{(n+m)k}}{2^{|u|} \cdot 2^{|w|}}$$
$$\leq 1/n.$$

where the last inequality holds by setting $k := s + d + \log n$. By the definition of B', we have

$$\mathop{\mathbf{Pr}}_{u,w,r'_B}\left[B'\left(u, w01^{T(n)^c}, 1^s, 1^n, 1^m\right) = 1\right] \leq o(1). \tag{47}$$

Given Equations (46) and (47), it is clear that $D(-) := B'\big(-, \mathcal{U}_t 01^{T(n)^c}, 1^s, 1^n, 1^m\big)$ is a randomized distinguisher for $\mathsf{DP}_k\big(x \circ H(y_x; z_H); \mathcal{U}_{(n+m)k}\big)$. Note that provided that q_0 is a large enough polynomial, D needs only $O(\log t)$ bits as advice, and then it runs in time

$$t_D := \widetilde{O}(t + T(n)^c) \cdot \mathsf{poly}(nm) \leq \widetilde{O}(t) \cdot \mathsf{poly}(nm).$$

Lemma 22 implies that

$$\mathsf{pK}^{\widetilde{O}(t)\cdot q(nm)}\left(x \circ H(y_x; z_H)\right) \leq k + O(\log t) \leq \mathsf{pK}^t(x) + d + \log q(t),$$

for some large polynomial q. This completes the proof of Claim 71. ◁

By Lemma 69, there exist polynomial p_0 and p such that for every $p_0(n,m) \leq t_1 \leq 2^{n+m}$

$$\mathop{\mathbf{Pr}}_{u\sim\{0,1\}^m}\left[\mathsf{pK}^{t_1}(x \circ u) > \mathsf{pK}^{t_1\cdot p(nm)}(x) + m - \log p(t_1)\right] \geq 0.99. \tag{48}$$

We set $t_1 := \widetilde{O}(t \cdot q(nm)) \cdot \tau(n+m)$, where τ is the function from Lemma 68. Note that we can make the polynomial q_0 to be large enough so that t_1 satisfies the condition for which Equation (48) holds. Now we choose m so that

$$\mathsf{pK}^{t_1\cdot p(nm)}(x) + m - \log p(t_1) \geq s + \log \tau(t \cdot q(nm)). \tag{49}$$

That is, we set

$$m := \left(\mathsf{pK}^t(x) - \mathsf{pK}^{t\cdot n^c}(x)\right) + c \cdot \log t,$$

where $c > 0$ is some large constant, and we use the fact that $t \leq 2^{n/2}$ to simplify t_1. Define the following (probabilistic) function:

$$D(-) := \neg B\left(x \circ -, 1^s, 1^{t\cdot q(nm)}\right).$$

Observe the following properties of D.

- For every $z \in \{0,1\}^d$, $\mathbf{Pr}_D\left[D(H(y_x; z)) = 0\right] \geq 2/3$. This follows from Claim 71 and the correctness of B on solving the promise problem $(\Pi_{\mathsf{YES}}, \Pi_{\mathsf{NO}})$ from Lemma 68.
- For at least 0.99 fraction of $\beta \in \{0,1\}^m$, $\mathbf{Pr}_D\left[D(\beta) = 1\right] \geq 2/3$. This follows from Equations (48) and (49) (and the correctness of B).
- D runs in time $\mathsf{poly}(t)$.

At this point, we can see that D is a randomized function that *distinguishes* $H(y_x;\mathcal{U}_d)$ from $\mathcal{U}_m$, and by randomly fixing its internal randomness we get a deterministic distinguisher with good probability. However, in order to utilize the reconstruction procedure in Lemma 70 to recover y_x, we need a deterministic function that *avoids* H. By amplifying the success probability of D on the "promised" inputs, we can show that randomly fixing the internal randomness of D gives a function that avoids H with high probability. More specifically, using standard error reduction techniques, we can make both the probability in the first two items above at least $1 - 1/(4 \cdot 2^m)$, which only causes a multiplicative overhead of $\mathsf{poly}(m)$ in the running time. By abusing notation, we use the same D to denote this new function with small error. Now consider the set S of inputs for which D has this small error. That is,

$$S := \left\{H(y_x; z) \mid z \in \{0,1\}^d\right\} \cup \left\{\beta \in \{0,1\}^m \mid \mathop{\mathbf{Pr}}_D[D(\beta) = 1] \geq 1 - 1/(4 \cdot 2^m)\right\}.$$

Note that $|S| \leq 2^m$. For $\beta \in S$, let us say that the *correct* answer of β is 0 (resp. 1) if β is in the first (resp. second) subset in the definition of S. By a union bound, we have

$$\mathop{\mathbf{Pr}}_{r_D\sim\{0,1\}^{\mathsf{poly}(t)}}\left[\exists\, \beta \in S \text{ such that } D(\beta; r_D) \text{ is not correct}\right] \leq |S| \cdot \frac{1}{4 \cdot 2^m} \leq \frac{1}{4}, \tag{50}$$

where r_D above denotes the internal randomness of D. This means with probability at least $2/3$ over a uniformly random r_D, $D(-;r_D)$ is a deterministic function that is correct on *all* the inputs in S. In particular, such a function 0.1-avoids H. Let us say that r_D is *good* if this is true. Let $\mathsf{Recon}^{(-)}\colon \{0,1\}^a \to \{0,1\}^{T(n)}$ be the reconstruction procedure in Lemma 70, where $a := 2m + O\big(\log T + \log^3 m\big)$. We have

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{r_D\sim\{0,1\}^{\mathsf{poly}(t)}}\left[\exists\,\alpha\in\{0,1\}^a \text{ such that } \mathsf{Recon}^{D(-;r_D)}(\alpha)=x\right]\\
&\geq \mathop{\mathbf{Pr}}_{r_D}\left[\exists\,\alpha\in\{0,1\}^a \text{ such that } \mathsf{Recon}^{D(-,r_D)}(\alpha)=x \,\middle|\, r_D \text{ is good}\right]\cdot \mathop{\mathbf{Pr}}_{r_D}[r_D \text{ is good}]\\
&\geq \frac{2}{3}. \qquad \text{(Lemma 70 and Equation (50))}
\end{aligned}$$

Note that given x, D can be constructed using the integers s, t, n, m and the code for B, which can be encoded using $O(\log t)$ bits. Then the above implies

$$\mathsf{pK}^{\mathsf{poly}(t\cdot T)}\left(y_x \mid x\right) \leq a + O(\log t) \leq 2m + O\big(\log t + \log^3 m\big)\,. \tag{51}$$

Note that the above holds for every $q_0(T(n)) \leq t \leq 2^{n/2}$. Also, recall that $m = \mathsf{pK}^t(x) - \mathsf{pK}^{t^c}(x) + c\cdot\log t$ for some constant $c>0$. By Lemma 46, there exists some t_* such that

$$q_0(T(n)) \leq t_* \leq q_0(T(n))\cdot 2^{c\cdot\sqrt{n\log n}} \leq 2^{O(\sqrt{n\log n})},$$

and

$$\mathsf{pK}^{t_*}(x) - \mathsf{pK}^{t_*\cdot n^c}(x) \leq O\Big(\sqrt{n\log n}\Big)\,.$$

Then for such t_*, we have $m \leq O\big(\sqrt{n\log n}\big)$. Therefore, plugging this t_* into Equation (51) for t, we have that there exists some constant $d>0$ such that

$$\mathsf{pK}^{2^{d\cdot\sqrt{n\log n}}}\left(y_x \mid x\right) \leq d\cdot\sqrt{n\log n}.$$

This suggests the following probabilistic algorithm A for solving L:

> On input $x\in\{0,1\}^n$, A samples $w\sim\{0,1\}^{d\cdot\sqrt{n\log n}}$. It then exhaustively searches over all $\mathcal{M}\in\{0,1\}^{d\cdot\sqrt{n\log n}}$, running $\mathcal{M}(w,x)$ for $2^{d\cdot\sqrt{n\log n}}$ steps to produce some output y. A accepts iff a y is obtained such that $V(x,y)=1$.

This completes the proof. ◀

Nisan–Wigderson Generators in Proof Complexity: New Lower Bounds

Erfan Khaniki
Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
Institute of Mathematics of the Czech Academy of Sciences, Prague, Czech Republic

Abstract

A map $g : \{0,1\}^n \to \{0,1\}^m$ $(m > n)$ is a hard proof complexity generator for a proof system P iff for every string $b \in \{0,1\}^m \setminus \mathsf{Rng}(g)$, formula $\tau_b(g)$ naturally expressing $b \notin \mathsf{Rng}(g)$ requires superpolynomial size P-proofs. One of the well-studied maps in the theory of proof complexity generators is Nisan–Wigderson generator. Razborov [37] conjectured that if A is a suitable matrix and f is a $\mathsf{NP} \cap \mathsf{CoNP}$ function hard-on-average for $\mathsf{P/poly}$, then $\mathsf{NW}_{f,A}$ is a hard proof complexity generator for Extended Frege. In this paper, we prove a form of Razborov's conjecture for AC^0-Frege. We show that for any symmetric $\mathsf{NP} \cap \mathsf{CoNP}$ function f that is exponentially hard for depth two AC^0 circuits, $\mathsf{NW}_{f,A}$ is a hard proof complexity generator for AC^0-Frege in a natural setting. As direct applications of this theorem, we show that:

1. For any f with the specified properties, $\tau_b(\mathsf{NW}_{f,A})$ (for a natural formalization) based on a random b and a random matrix A with probability $1 - o(1)$ is a tautology and requires superpolynomial (or even exponential) AC^0-Frege proofs.
2. Certain formalizations of the principle $f_n \notin (\mathsf{NP} \cap \mathsf{CoNP})/\mathsf{poly}$ requires superpolynomial AC^0-Frege proofs.

These applications relate to two questions that were asked by Krajíček [21].

2012 ACM Subject Classification Theory of computation → Proof complexity; Theory of computation → Complexity theory and logic

Keywords and phrases Proof complexity, Bounded arithmetic, Bounded depth Frege, Nisan-Wigderson generators, Meta-complexity, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.17

Funding This work was supported by the GACR grant 19-27871X, the institute grant RVO: 67985840, and the Specific university research project SVV-2020-260589. Part of this work was done while the author was participating in the program Satisfiability: Theory, Practice, and Beyond at the Simons Institute for the Theory of Computing.

Acknowledgements We are grateful to Jan Bydžovský, Susanna de Rezende, Emil Jeřábek, Jan Krajíček, Jan Pich and Pavel Pudlák for their different forms of help in different stages of this work. We are also indebted to anonymous referees for their helpful suggestions, which led to a better presentation of the paper.

1 Introduction

Proving superpolynomial lower bounds for every proof system is one of the ultimate goals in proof complexity. For this matter, we need to prove that for every proof system P, there exists an infinite family of tautologies $\{\phi_n\}_{n\in\mathbb{N}}$ such that P does not have polynomial-size proofs for $\{\phi_n\}_{n\in\mathbb{N}}$. It is known that some weak proof systems require superpolynomial (or even exponential) size proofs for some families of tautologies (see [21] for more information). No superpolynomial lower bounds are known for strong proof systems such as Frege or Extended Frege. We do not even know superpolynomial lower bounds for $\mathsf{AC}^0(\oplus)$-Frege. It

seems that one of the main issues in proving lower bounds is the lack of good candidate hard formulas. There are three prominent candidates of formulas that are believed to be hard for any proof system.

The first candidate of these formulas is random CNFs. Some experts believe that these formulas should be hard for any proof system (see [21]). Another family of conjectured hard formulas is finite consistency statements. These formulas have tight connections to important conjectures in proof complexity and experts believed that they are hard for any proof system (For a detailed discussion, see [23, 35]). The third candidate is proof complexity generators.

1.1 Proof complexity generators

Let $g : \{0,1\}^n \to \{0,1\}^m$ $(m > n)$ be a function which is computable in a *reasonable low complexity class* such as $\mathsf{FP/poly}$. As $m > n$, $\{0,1\}^m \setminus \mathsf{Rng}(g)$ is nonempty. Let $b \in \{0,1\}^m \setminus \mathsf{Rng}(g)$, then as g is computable in $\mathsf{FP/poly}$, we can naturally express the true statement $b \notin \mathsf{Rng}(g)$ as a propositional formula which is denoted by $\tau_b(g)$. If for a proof system P, $\tau_b(g)$ requires superpolynomial size P-proofs for every $b \in \{0,1\}^m \setminus \mathsf{Rng}(g)$, then g is a hard proof complexity generator for P. The concept of proof complexity generators were defined independently by Alekhnovich *et. al.* [2] and Krajíček [11].

As pseudorandom generators are an important topic in computational complexity, Alekhnovich *et al.* [2] asked the following natural question: *which mappings $g : \{0,1\}^n \to \{0,1\}^m$ should be considered hard from the point of view of proof complexity?* To understand this concept, different mappings were investigated from different aspects in [2]. In particular, they investigated conditions that make a Nisan–Wigderson generator hard for proof systems such as Resolution and Polynomial Calculus.

Krajíček [11] investigated the hardness of different variants of the Pigeonhole principle in proof systems and their provability in related theories of bounded arithmetic. One of these variants is the dual weak Pigeonhole principle (dWPHP^n_{2n}) which says that for every function $g : [n] \to [2n]$, g cannot be onto. An interesting theory of bounded arithmetic is $\mathsf{BT} := \mathsf{S}^1_2 + \mathsf{dWPHP(PV)}$ which has several nice properties (see [7, 8]). Here S^1_2 is the base bounded arithmetic theory in the Buss's Bounded arithmetic hierarchy which is related to the polynomial-time reasoning (see [5]) and $\mathsf{dWPHP(PV)}$ consists of $\mathsf{dWPHP}^n_{2n}(f)$ for every polynomial-time computable function f. A natural question is whether S^1_2 and BT are actually the same theory. Krajíček introduced the concept of proof complexity generators as functions which violate $\mathsf{dWPHP(PV)}$ and formulated a conjecture about them in the setting of model theory of arithmetic that implies $\mathsf{S}^1_2 \neq \mathsf{BT}$ (see [22] for a proof of separation of PV and $\mathsf{PV} + \mathsf{dWPHP(PV)}$ under a different assumption). Moreover, this conjecture implies that proof complexity generators are hard for Extended Frege.

Later, Krajíček [12, 13, 14, 15, 16, 18, 19] investigated proof complexity generators from different aspects, developed the theory of proof complexity generators in great length and proposed some conjectures. In particular, Krajíček [18] defined the generator $\mathsf{nw}_{n,c}$ based on the gadget generators of [16] and conjectured that $\mathsf{nw}_{n,c}$ is a hard proof complexity generator for any proof system.

Razborov [37] made a significant contribution to the lower bound problem for proof complexity generators. He proved that Nisan–Wigderson generators based on suitable matrices and suitable functions are hard not only for Resolution but also for k-DNF Resolution, which improved the previous lower bounds in terms of the stretch of the generator and the strength of the proof system in [2, 14]. Moreover, he formulated the following intriguing conjecture:

▶ **Conjecture 1** (Razborov [37]). *Any Nisan–Wigderson generator based on suitable matrices and any function in* NP ∩ CoNP *that is hard on average for* P/poly*, is hard for Extended Frege.*

Conjecture 1 initiated new investigations in the theory of proof complexity generators from different aspects. We refer the reader for comprehensive dissections of the conjectures about the proof complexity generators to read Chapter 30 of [18] and Section 19.4 of [21].

Regarding Razborov's conjecture, Pich [28] proved that this conjecture is true for proof systems that enjoy different forms of the feasible interpolation property.

The strongest argument that supports Conjecture 1 was done by Krajíček in [19]. He proved that assuming the existence of a function $f \in \mathsf{NP} \cap \mathsf{CoNP}$ which is hard on average for P/poly; it is consistent with the universal theory PV that for any Nisan–Wigderson generator based on f (or for a function closely related to f) and suitable matrices is an onto function. It is worth noting that the arithmetical sentence that Krajíček used to formalize the sentence *Nisan–Wigderson generator is onto* is a natural one. However, it is not clear whether this consistency result based on this formalization implies Razborov's conjecture or not. Note that PV is a fairly strong theory as it proves a reasonable fragment of computational complexity theorems (see [30] for more information). It is worth mentioning that those investigations of the Nisan–Wigderson generators in proof complexity led to advancements in other areas as well, such as [29, 32] which proved unprovability of circuit lower bounds in bounded arithmetic and [31] which proved the existence of learning algorithms from circuit lower bounds.

Razborov's conjecture is inherently different from other conjectures in proof complexity that imply that strong proof systems are not p-bounded. The reason is that this conjecture describes a situation where *the hardness of computation implies the hardness of proof* for strong proof systems. For weak proof systems, such a relation exists, which is called *feasible interpolation property*. Krajíček defined this property in [10] and proved that several proof systems such as Resolution have the feasible interpolation property, which implied lower bounds for new formulas. Proving lower bounds using feasible interpolation proved to be very fruitful and led to several lower bounds for different proof systems such as Cutting Planes [34]. Unfortunately, this property does not hold for strong proof systems such as Extended Frege [24], and even AC^0-Frege [4] assuming cryptographic hardness assumptions (for more information, see chapter 17 of [21]). To overcome the barrier against the feasible interpolation property, different attempts were made to prove *hardness of computation implies hardness of proof* theorems for strong proof systems. Krajíček [17] proved a form of feasible interpolation for AC^0-Frege that is different from the original definition of the feasible interpolation property. Moreover, he developed the method of *Forcing with random variables* in [18] intending to prove *hardness of computation to hardness of proofs* theorems for strong proof systems (bounded arithmetics) and proved types of this theorem for AC^0-Frege and $\mathsf{AC}^0(\oplus)$-Frege (for a finitary proof of the theorem for $\mathsf{AC}^0(\oplus)$-Frege see [20]). Pudlák [36] characterized the canonical disjoint NP-pairs of AC^0-Frege and proved a generalized feasible interpolation theorem for them.

1.2 Our results

This paper aims to find sufficient conditions that make a Nisan–Wigderson generator hard for proof systems such as AC^0-Frege. Our main contribution is the proof of Razborov's conjecture for AC^0-Frege in a natural setting which was not known before. The following theorem states a natural restriction of Razborov's conjecture.

▶ **Theorem 2** (Main theorem, informal version). *Let $f \in \mathsf{NP} \cap \mathsf{CoNP}$ be a symmetric function that requires $2^{n^{\Omega(1)}}$ depth two AC^0 circuits. Then for any $\Sigma^1_1 \cap \Pi^1_1$ pair (ϕ_0, ϕ_1) that defines f, any suitable matrix A, and any $b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$, $\tau_b(\mathsf{NW}_{f,A})$ requires superpolynomial or exponential size AC^0-Frege proofs based on whether the stretch is exponential or polynomial when the Paris-Wilkie translation of (ϕ_0, ϕ_1) is used to form the formula $\tau_b(\mathsf{NW}_{f,A})$.*

Theorem 2 unconditionally implies that $\mathsf{NW}_{f,A}$ for suitable functions f (such as Parity or Majority) and suitable matrices A are hard proof complexity generators for AC^0-Frege even when the stretch is exponential. No lower bounds for Nisan–Wigderson generators were known for this system. It is worth noting that before this work, the only known hard proof complexity generators for AC^0-Frege, were the PHP-generator of [16] and the more general generator $\mathsf{nw}_{n,c}$ of [18]. Moreover, Theorem 2 implies the following results:

1. For any f that satisfies the conditions of Theorem 2 such as Parity, the formula $\tau_b(\mathsf{NW}_{f,A})$ (for a natural formalization) based on a random b and a random matrix A is a tautology with probability $1 - o(1)$ and requires superpolynomial (exponential) AC^0-Frege proofs.
2. Certain formalizations of the principle $f_n \notin (\mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k))/\mathsf{poly}$ requires superpolynomial AC^0-Frege proofs.

These results relate to two questions asked by Krajíček [21] (problems 19.4.5 and 19.6.1). The first problem asks whether random linear generators (random systems of linear equations over $\mathbb{F}_2$) are hard for AC^0-Frege or not. The second problem asks whether linear generators are iterable for AC^0-Frege or not, which relates to the question of the hardness of proving the principle $f_n \notin \mathsf{SIZE}(n^k)$ in AC^0-Frege. It seems that because of the formalization that we used in the main theorem, our lower bounds do not imply the hardness of proving the principle $f_n \notin \mathsf{SIZE}(n^k)$ for AC^0-Frege.

2 Preliminaries

2.1 Nisan–Wigderson generators

For the rest of the paper for any two real numbers $r_1 \leq r_2$, define $[r_1, r_2) := \{i \in \mathbb{N} : \lfloor r_1 \rfloor \leq i < \lceil r_2 \rceil\}$ and $[r_1, r_2] := \{i \in \mathbb{N} : \lfloor r_1 \rfloor \leq i \leq \lceil r_2 \rceil\}$.

Let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function. For a natural number n, f_n denotes the function f restricted to $\{0,1\}^n$. Let A be an $m \times n$ $0-1$ matrix such that each row of A has exactly l ones. Such a matrix is called an l-sparse matrix. For such a $m \times n$ l-sparse matrix A, $J_i(A) := \{j \in [0, n) : A_{i,j} = 1\}$.

For every pair (f, A) where $f : \{0,1\}^l \to \{0,1\}$ is a Boolean function and A is a $m \times n$ l-sparse matrix, Nisan and Wigderson [26] defined the generator $\mathsf{NW}_{f,A} : \{0,1\}^n \to \{0,1\}^m$ as follows:

- For every input $a \in \{0,1\}^n$, the i'th bit of the output of $\mathsf{NW}_{f,A}(a)$ is $f(a|J_i(A))$.

It was proved in the seminal paper [26] that if f is a hard function (depending on the application) and A satisfies specific combinatorial properties, then $\mathsf{NW}_{f,A}$ is a *good* pseudorandom generator (depending on the parameters).

Let $f \in \mathsf{NP} \cap \mathsf{CoNP}$. A pair of propositional formulas $(\sigma_0(\mathbf{p}, \mathbf{q}), \sigma_1(\mathbf{p}, \mathbf{r}))$ is a representation of f_n for a natural number n iff:

1. $|\mathbf{p}| = n$ and moreover $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ variables are disjoint.
2. (σ_0, σ_1) defines the function f_n which means:
   a. $\neg\sigma_0 \vee \neg\sigma_1$ is a tautology.
   b. For every $a \in \{0,1\}^n$, $f(a) = i$ iff $\sigma_i(a, \mathbf{t})$ is satisfiable where $i \in \{0,1\}$.

Note that as $f \in \mathsf{NP} \cap \mathsf{CoNP}$, for every n, f_n has a representation.

Suppose $f \in \mathsf{NP} \cap \mathsf{CoNP}$, (σ_0, σ_1) is a representation for f_l, and A is a $m \times n$ l-sparse matrix. Then for any $b \in \{0,1\}^m$, $\tau_b(\mathsf{NW}_{f,A})$ based on (σ_0, σ_1) is the following propositional formula:

$$\bigvee_{b_i=1} \neg\sigma_1(\mathbf{p}|J_i(A), \mathbf{q}_i) \vee \bigvee_{b_i=0} \neg\sigma_0(\mathbf{p}|J_i(A), \mathbf{q}_i)$$

where $\mathbf{q}_i$'s are disjoint variables. Note that if $b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$, then $\tau_b(\mathsf{NW}_{f,A})$ is tautology.

As it was discussed in previous works [2, 15, 37, 19], $\mathsf{NW}_{f,A}$ can be a hard proof complexity generator for a proof system P for the following four reasons:

- The complexity of f.
- The properties that A satisfies.
- The representation of f that is used in the formula $\tau_b(\mathsf{NW}_{f,A})$.
- The string $b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$.

As we will see, our main result also imposes different conditions on $\tau_b(\mathsf{NW}_{f,A})$ to make sure that it requires long proofs.

The following parts explain the properties that we need for the matrices and representations to prove our theorems.

2.1.1 Representations

The hardness of $\tau_b(\mathsf{NW}_{f,A})$ can depend on the pair (σ_0, σ_1) that is used in it. This matter has been investigated in [2], and they examined different representations. Recently, Sokolov [38] answered one of the open problems that was stated about a representation of $\tau_b(\mathsf{NW}_{f,A})$ in [2]. Here we investigate representations based on definability over finite structures in logic, which is a well-studied concept in descriptive complexity and finite model theory.

$\mathbf{\Sigma^1_1 \cap \Pi^1_1}$ representation

Let $\mathcal{L}$ be a finite relational language and X be a unary relational symbol which is not in $\mathcal{L}$. A Σ^1_1 formula $\psi(X)$ in the language $\mathcal{L} \cup \{X\}$ with equality defines a function $f \in \mathsf{NP}$ iff:

1. $\psi := \exists \bar{Y} \phi(X, \bar{Y})$ where $\phi(X, \bar{Y})$ is a first-order formula in the language $\mathcal{L} \cup \{X\}$ with equality.
2. X is not in $\bar{Y}$.
3. For every n, every $a \in \{0,1\}^n$, $f_n(a) = 1$ iff $([0, n), a) \models \psi(X)$ when X is interpreted by a.

Fagin's theorem [6] directly implies that for every symmetric $f \in \mathsf{NP}$, a Σ^1_1 formula $\psi_f(X)$ exists in a language $\mathcal{L} \cup \{X\}$ that defines f. Therefore, the set of functions that are Σ^1_1 definable is exactly symmetric NP and hence this set is quite rich. As an example, we explain how the negation of Parity function can be defined as a Σ^1_1 formula. Let $\mathcal{L} = \{Y\}$ where Y is a binary relation symbol. Then

$$\bar{\oplus}(X, Y) := \forall i \, (X(i) \to \exists j (j \neq i \wedge X(j) \wedge Y(i, j) \wedge Y(j, i) \wedge \forall k (k = i \vee \neg Y(i, k) \vee j = k)) \, .$$

Then $\psi_{\bar{\oplus}}(X) := \exists Y \bar{\oplus}(X, Y)$ defines the negation of Parity function (parity of $a \in \{0,1\}^n$ is 0 iff the number of 1's in a is even).

The class of Σ^1_1 formulas is a natural and important class in finite model theory and descriptive complexity. Moreover, this class has appeared in different places in proof complexity, too (for example, see [17]).

To prove Theorem 2, the following lemma is needed. This lemma states that the truth of first-order formulas in a relational language does not change under permutations.

If A is a set and Q is a relation on it, i.e. $Q \subseteq A^k$ for some k, and $h : A \to A$ is a function, then $h(Q) := \{(h(a_0), ..., h(a_{k-1})) : (a_0, ..., a_{k-1}) \in Q\}$.

▶ **Lemma 3.** *Let $\mathcal{L} = \{Y_0, ..., Y_k\}$ be a finite relational language and $\mathcal{A}_0 = (A, \{Q_0^0, ..., Q_k^0\})$ be an $\mathcal{L}$-structure. Let h be a bijective function from A onto A. Consider the $\mathcal{L}$-structure $\mathcal{A}_1 := (A, \{Q_0^1, ..., Q_k^1\})$ where $Q_i^1 = h(Q_i^0)$, for every $i \in [0, k]$. Then for every first-order formula $\phi(x_0, ..., x_{p-1})$ in $\mathcal{L}$ with equality, every $(a_0, ..., a_{p-1}) \in A^p$:*

$$\mathcal{A}_0 \models \phi(a_0, ..., a_{p-1}) \Leftrightarrow \mathcal{A}_1 \models \phi(h(a_0), ..., h(a_{p-1})).$$

Proof. This lemma can be proved by induction on the complexity of ϕ. ◀

Let $\exists \bar{Y} \phi(X, \bar{Y})$ be a Σ_1^1 formula. Then for any n, the Paris–Wilkie translation [27] (see also Section 8.2 of [21]) of $\phi^{<n}(X, \bar{Y})$ ($\phi^{<n}$ is ϕ when every first-order quantifier is bounded by n) is denoted by $\langle \phi \rangle_n (\mathbf{p}, \mathbf{q})$ which is a constant depth formula (without loss of generality we can assume that it is a CNF using extension variables). The number n indicates the size of the universe in which $\phi(X, Y)$ has been considered. For example the Paris-Wilkie translation of $\bar{\oplus}(X, Y)$ in the universe of size n is

$$\langle \bar{\oplus}(X, Y) \rangle_n := \bigwedge_{i=0}^{n-1} \left(\neg p_i \vee \bigvee_{j=0, j \neq i}^{n-1} \left(p_j \wedge q_{i,j} \wedge q_{j,i} \wedge \bigwedge_{k=0, k \neq i, k \neq j}^{n-1} \neg q_{i,k} \right) \right).$$

Let $f \in \mathsf{NP} \cap \mathsf{CoNP}$ be a symmetric function. Then a pair of Σ_1^1 formulas $(\exists \bar{Y} \phi_0(X, \bar{Y}), \exists \bar{Z} \phi_1(X, \bar{Z}))$ defines f iff:

1. $\exists \bar{Y} \phi_1(X, \bar{Y})$ defines f.
2. $\exists \bar{Z} \phi_0(X, \bar{Z})$ defines $\neg f$.

Such a pair is called a $\Sigma_1^1 \cap \Pi_1^1$ definition of f. Moreover, for any n, $(\langle \phi_0 \rangle_n, \langle \phi_1 \rangle_n)$ is a representation of f_n. For the sake of easiness, by $\langle \psi \rangle_n$ we mean $\langle \phi \rangle_n$ where $\psi(X) := \exists \bar{Y} \phi(X, \bar{Y})$ is a Σ_1^1 formula.

2.2 Proof systems

We assume the reader knows the basic facts about proof complexity, proof systems, and bounded arithmetics (for a detailed discussion, see [21, 9]). Here we state some useful facts about AC^0-Frege, which will be used in the results.

2.2.1 AC^0-Frege

AC^0-Frege is the name for a family of proof systems that work with constant-depth de Morgan formulas. For each $d \geq 1$, F_d denotes AC^0-Frege proof system of depth d, which is the Frege proof system that works with formulas of depth at most d.

To prove Theorem 2, we need some known relations between AC^0-Frege and V_1^0, which is a two-sorted bounded arithmetic (see [5, 9]). These relations are related to the model theory of V_1^0.

Let $\mathcal{M}$ be an arbitrary nonstandard model of true arithmetic and $n \in \mathcal{M} \setminus \mathbb{N}$. Then

$$\mathcal{M}_n := \{a \in \mathcal{M} : \text{There exists a } b \in \mathcal{M} \setminus \mathbb{N} \text{ such that } a < 2^{n^{1/b}}\}.$$

The following theorems explain the relationship between AC^0-Frege and V_1^0 from the point of view of proof complexity. These theorems state that lower bounds for AC^0-Frege correspond to unprovability in V_1^0 in a certain sense.

For a set A, $\mathcal{P}(A)$ denotes the power set of A.

▶ **Theorem 4** (Section 9.4 of [9]). *Let $(\mathcal{M}, \chi) \models \mathsf{V}_1^0$ and $\sigma \in \chi$ be a constant depth propositional formula (depth of σ is standard). If $\neg\sigma$ is satisfiable by an assignment in χ, then for every standard d, there is no F_d-proof of σ in $(\mathcal{M}, \chi)$.*

Note that Theorem 4 also holds in the case where σ is the Paris-Wilkie translation of a bounded arithmetical formula such as $\phi(x, \bar{R})$ ($\sigma = \langle \phi(n, \bar{R}) \rangle_n$ for some $n \in \mathcal{M}$), i.e. if there is an $\bar{\alpha} \in \chi$ such that $(\mathcal{M}, \chi) \models \neg\phi(n, \bar{\alpha})$, then $\neg\sigma$ is satisfiable by an assignment from χ and therefore σ does not have any F_d-proof in $\mathcal{M}$.

▶ **Theorem 5** (Section 9.4 of [9]). *Let $\mathcal{M}$ be a countable nonstandard model of true arithmetic and $\phi(x, R)$ be a bounded arithmetical formula such that for every d, the family $\{\langle \phi(n, R) \rangle_n\}_{n \in \mathbb{N}}$ requires exponential F_d-proofs. Then for every $m \in \mathcal{M} \setminus \mathbb{N}$, there exists a $\chi \subseteq \mathcal{P}(\mathcal{M}_m)$ such that:*

1. *Every bounded subset of $\mathcal{M}_m$ which is definable in $\mathcal{M}$ is in χ.*
2. *$(\mathcal{M}_m, \chi) \models \mathsf{V}_1^0$.*
3. *There is an $\alpha \in \chi$ such that $(\mathcal{M}_m, \chi) \models \neg\phi(m, \alpha)$.*

3 Razborov's conjecture for AC^0-Frege

In this section, we state and prove the main result of the paper.

A Boolean function f is symmetric iff for every n, f_n is invariant under any permutation of its inputs. Let $\mathsf{S}_{\mathsf{AC}_2^0}$ denote the depth two AC^0 circuit complexity of functions, then:

▶ **Theorem 6.** *Let $f \in \mathsf{NP} \cap \mathsf{CoNP}$ be a symmetric function such that $\mathsf{S}_{\mathsf{AC}_2^0}(f) = 2^{n^{\Omega(1)}}$ and (ϕ_0, ϕ_1) be a $\Sigma_1^1 \cap \Pi_1^1$ definition of f. Then for every d:*

1. *For every positive $c \in \mathbb{N}$, every $0 < \epsilon < 1$, there exits an $\epsilon' > 0$ such that for every large enough n, every $n^c \times n$ $\lfloor n^\epsilon \rfloor$-sparse matrix A, any $b \notin \mathsf{Rng}$, $\tau_b(\mathsf{NW}_{f,A})$ based on $(\langle \phi_0 \rangle_{\lfloor n^\epsilon \rfloor}, \langle \phi_1 \rangle_{\lfloor n^\epsilon \rfloor})$ does not have F_d-proofs of size less than $2^{n^{\epsilon'}}$.*
2. *For every positive $r \in \mathbb{N}$, every large enough s, every $t \in [s/r, s]$, every $c \in \mathbb{N}$, every large enough n, every $2^n \times n^s$ n^t-sparse matrix A, any $b \notin \mathsf{Rng}$, $\tau_b(\mathsf{NW}_{f,A})$ based on $(\langle \phi_0 \rangle_{n^t}, \langle \phi_1 \rangle_{n^t})$ does not have F_d-proofs of size less than 2^{cn}.*

Note that in Theorem 6, the size of the formula $\tau_b(\mathsf{NW}_{f,A})$ is $n^{O(1)}$ in the first part and it is $2^{O(n)}$ in the second part. This theorem is proved by a model-theoretic argument based on the relations explained in Preliminaries in combination with the hardness of the Pigeonhole principle in AC^0-Frege. Model theoretic arguments have been used previously in proof complexity and they were very fruitful (for example see [1, 11, 17] and [9] for a detailed explanation). See [39, 12] for discussions about the importance and benefits of the model-theoretic arguments (and in general, the logical point of view) in proof complexity.

Note that an immediate consequence of Theorem 6 is that $\mathsf{NW}_{f,A}$ based on a hard enough function f with suitable parameters is a hard proof complexity generator for AC^0-Frege. As the Parity function or the Majority function satisfies the required assumptions of Theorem 6, we get that NW-generators based on these functions are hard proof complexity generators for AC^0-Frege.

Proof of Theorem 6

We state the proof as a series of lemmas for more clarity. We prove the second part of this theorem. The first part can be proved in the same way. For the rest of the paper, $[n] := [0, n)$.

Intuitively, the proof goes as follows. Let n be large enough. As f is symmetric and has high depth two circuit complexity, then there is a u such that f_{n^t} on strings with u many 1's is different from f_{n^t} on strings with $u+1$ many 1's and moreover, u is not too big and not too small. This implies that there is an input a for the NW generator such that the number of 1's of a restricted to each row is close to u. Now, assuming PHP fails, for every row i we can find a witness for $\phi_{b_i}(a|J_i(A))$ by constructing permutations between sets that actually have different sizes and this forces any string b into the range of the generator and this is possible as the truth of representations are closed under permutations. The next lemmas formalize this intuitive proof and their combination proves the theorem.

▶ **Lemma 7.** *Let $f : \{0,1\}^* \to \{0,1\}$ be a symmetric Boolean function such that $\mathsf{S}_{\mathsf{AC}^0_2}(f_n) = \Omega(2^{n^\epsilon})$ for an $\epsilon > 0$. Then there is a natural m such that for every $n \geq m$ there is natural number $u \in [n^{\epsilon/2}, n - n^{\epsilon/2}]$ such that*

$$f_n(1^u 0^{n-u}) \neq f_n(1^{u+1} 0^{n-u-1}).$$

Proof. Let $g : \{0,1\}^n \to \{0,1\}$ be a symmetric function. If there exists a $r \leq n/2$ such that for every $r \leq k \leq n - r$, $g(1^k 0^{n-k}) = 0$, then

$$\mathsf{S}_{\mathsf{DNF}}(g) \leq 2n \cdot \sum_{i=0}^{r} \binom{n}{i} \leq 2n(\frac{en}{r})^r$$

where $\mathsf{S}_{\mathsf{DNF}}$ denotes the DNF complexity of functions. Writing this inequality (1) for f_n, we get $c2^{n^\epsilon} \leq 2n(\frac{en}{r})^r$ for a $c > 0$. So if we put $r = n^\delta$ and rewriting this inequality we have

$$c2^{n^\epsilon} \leq 2n(en^{1-\delta})^{n^\delta} \leq 2e^{n^\delta} n^{1+n^\delta} \leq 2n^{1+2n^\delta} = 2^{(2n^\delta+1)\log n+1}.$$

So assuming $\delta = \epsilon/2$, we have $(2n^\delta + 1)\log n + 1 = o(n^\epsilon)$. Therefore for every large enough n, there exists a $v \in [n^{\epsilon/2}, n - n^{\epsilon/2}]$ such that $f_n(1^v 0^{n-v}) = 1$. Following the same argument for $\neg f_n$, we can deduce that for every large enough n, there exists a $v' \in [n^{\epsilon/2}, n - n^{\epsilon/2}]$ such that $\neg f_n(1^{v'} 0^{n-v'}) = 1$. So we have found $v, v' \in [n^{\epsilon/2}, n - n^{\epsilon/2}]$ such that $f_n(1^v 0^{n-v}) \neq f_n(1^{v'} 0^{n-v'})$, hence there exists a $u \in [n^{\epsilon/2}, n - n^{\epsilon/2}]$ such that

$$f_n(1^u 0^{n-u}) \neq f_n(1^{u+1} 0^{n-u-1}).$$ ◀

Now let $\mathcal{M}$ be a countable nonstandard model of true arithmetic. Let n, s, t, A, b be arbitrary elements of $\mathcal{M}$ such that:

1. $n, t \in \mathcal{M} \setminus \mathbb{N}$.
2. $A \in \mathcal{M} \setminus \mathbb{N}$ encodes a $2^n \times n^s$ n^t-sparse matrix where $t \in [s/r, s]$, $n^s < 2^n$, and $n^s 2^n \leq 2^{n^{t/u}}$ for a nonstandard u.
3. $b \in \mathcal{M} \setminus \mathbb{N}$ is a binary string of length 2^n such that $b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$.

Let χ be the set of all bounded subsets of $\mathcal{M}_{n^t}$ encoded in $\mathcal{M}$. So in particular $A, b \in \chi$.

As $\mathsf{S}_{\mathsf{AC}^0_2}(f_m) = 2^{m^{\Omega(1)}}$, there is a standard rational $\epsilon > 0$ such that $\mathsf{S}_{\mathsf{AC}^0_2}(f_m) = \Omega(2^{m^\epsilon})$. Let $\delta := \epsilon/2$, then there exists $u \in [n^{\delta t}, n^t - n^{\delta t}]$ that is guaranteed to exist by Lemma 7 for f_{n^t}. Let $v := \min\{u, n^t - u\}$, then

▶ **Lemma 8.** *There exists a binary string $\alpha \in \chi$ of length n^s such that for every $i \in [2^n]$,*

$$\#_1(\alpha|J_i(A)) \in [v(1-\frac{1}{\sqrt[3]{v}}), v(1+\frac{1}{\sqrt[3]{v}})]$$

where $\#_s(w)$ is the number of occurrences of symbol s in the string w.

Proof. Let $X_0, ..., X_{n^s-1}$ be independent random variables taking values in $\{0,1\}$ such that for every i, $Pr[X_i = 1] = \frac{v}{n^t}$. For every $i \in [2^n]$, let $Y_i = \sum_{j \in J_i(A)} X_j$ and hence $\mathbb{E}[Y_i] = v$. By the Chernoff bound we have the following inequalities for every $i \in [2^n]$:

1. $Pr[Y_i \leq v(1-\frac{1}{\sqrt[3]{v}})] \leq e^{\frac{-\sqrt[3]{v}}{2}}$.
2. $Pr[Y_i \geq v(1+\frac{1}{\sqrt[3]{v}})] \leq e^{\frac{-\sqrt[3]{v}}{3}}$.

Let X' be the concatenation of $X_0, ..., X_{n^s-1}$, hence it is a random string of length of n^s. Now combining the above inequalities with the union bound we get:

$$\begin{aligned}
\mathbf{P} = Pr\left[\bigvee_{i=0}^{2^n-1} \#_1\left(X'|J_i(A)\right) \notin [v(1-\frac{1}{\sqrt[3]{v}}), v(1+\frac{1}{\sqrt[3]{v}})]\right] \leq \\
\sum_{i=0}^{2^n-1} Pr\left[\#_1\left(X'|J_i(A)\right) \notin [v(1-\frac{1}{\sqrt[3]{v}}), v(1+\frac{1}{\sqrt[3]{v}})]\right] \leq \\
\sum_{i=0}^{2^n-1} \left(Pr[Y_i \leq v(1-\frac{1}{\sqrt[3]{v}})] + Pr[Y_i \geq v(1+\frac{1}{\sqrt[3]{v}})]\right) \leq \\
2^n \cdot 2e^{\frac{-\sqrt[3]{v}}{3}}.
\end{aligned}$$

We know that $v \geq n^{\delta t}$, t is a nonstandard number, and δ is a standard rational, so

$$n+1 < \frac{n^{\delta t/3}}{3} \leq \frac{\sqrt[3]{v}}{3}$$

which implies $2^n \cdot 2e^{\frac{-\sqrt[3]{v}}{3}} < 1$, and hence $\mathbf{P} < 1$. This implies that there exists a string $\alpha \in \chi$ that satisfies the desired property. ◀

▶ **Lemma 9.** *The following functions exist in χ:*

1. *$\gamma : [2^n] \to [n^t + 1]$ such that for every $i \in [2^n]$, $\gamma(i) = \#_1(\alpha|J_i(A))$.*
2. *$\omega : [2^n] \times [n^t] \to [n^t]$ such that for every $i \in [2^n]$, $\omega(i, .)$ defines a permutation over $[n^t]$ and moreover $\beta_j = \left(\alpha|J_i(A)\right)_{\omega(i,j)}$ where $\beta = 1^{\gamma(i)}0^{n^t-\gamma(i)}$.*

Proof.

1. The function γ exists in $\mathcal{M}$ as it is definable by an arithmetical formula with parameter α. To prove that γ is in χ, we observe that encoding of γ as a binary string requires at most $c2^n \cdot \log n^t$ (for some $c \in \mathbb{N}$) which is less than $2^{n^{\sqrt{t}}}$, hence $\gamma \in \chi$.
2. Like the previous part, ω exists in $\mathcal{M}$ as it is definable by an arithmetical formula with parameters α and γ, and its bit representation requires at most $c2^n \cdot n^t \log n^t$ (for some $c \in \mathbb{N}$) which is again less than $2^{n^{\sqrt{t}}}$, and therefore $\omega \in \chi$. ◀

To continue the proof, we need the celebrated result about the hardness of the Pigeonhole principle for AC^0-Frege.

▶ **Theorem 10** ([1, 25, 33])**.** *For any natural number d, there exists an $\epsilon_d > 0$ such that for large values of n, any F_d-proof of PHP_n^{n+1} has size at least $2^{\Omega(n^{\epsilon_d})}$.*

Now let $l = \lfloor \sqrt[4]{v} \rfloor$, then we have the following lemma.

▶ **Lemma 11.** *There exists $\chi' \subseteq \mathcal{P}(\mathcal{M}_{n^t})$ such that:*
1. $\chi \subseteq \chi'$.
2. *There exists a function $\sigma \in \chi'$ such that σ is a bijection from $[l]$ onto $[l-1]$.*
3. $(\mathcal{M}_{n^t}, \chi') \models \mathsf{V}_1^0$.

Proof. By Theorem 10 we know that PHP_{m-1}^m requires exponential size F_d-proofs for every d. Therefore by Theorem 5, there exists a $\chi' \subseteq \mathcal{P}(\mathcal{M}_l)$ such that every bounded subset of $\mathcal{M}_l$ is in χ', $(\mathcal{M}_l, \chi') \models \mathsf{V}_1^0$ and there exists a $\sigma \in \chi'$ such that it is bijection from $[l]$ onto $[l-1]$. Note that if $a \in \mathcal{M}_{n^t}$, then there exists a $b \in \mathcal{M} \setminus \mathbb{N}$ such that $a < 2^{n^{t/b}}$. Let $b' = \lfloor \frac{\delta b}{4} \rfloor$, then $a < 2^{l^{1/b'}}$ as we know $l \geq n^{\delta t/4}$. This implies that $\mathcal{M}_l = \mathcal{M}_{n^t}$, and moreover $\chi \subseteq \chi'$ which completes the proof. ◀

The following lemma shows that we can simultaneously falsify some weak Pigeonhole principle instances.

▶ **Lemma 12.** *There exists a function $F : [2^n] \times \{-1, 0, 1\} \times [n^t] \to [n^t]$ in χ' such that for every $i \in [2^n]$*
1. *$F(i, a, .)$ restricted to $[v + a]$, is a bijection from $[v + a]$ onto $[\gamma(i)]$.*
2. *$F(i, a, .)$ restricted to $[v + a, n^t)$, is a bijection from $[v + a, n^t)$ onto $[\gamma(i), n^t)$.*

Proof. Let $\sigma \in \chi'$ be the function that Lemma 11 provides. Let
1. $w_{i,a} = |\gamma(i) - v - a|$.
2. $M_{i,a} = \max\{v + a, \gamma(i)\}$.
3. $m_{i,a} = \min\{v + a, \gamma(i)\}$.

Then we define the function $G_0(i, a, b)$ as follows:

$$G_0(i, a, b) := \begin{cases} \sigma(b - lk) + (l-1)k & b \in [lk, l(k+1)) \wedge k \in [w_{i,a}] \\ b - w_{i,a} & b \in [lw_{i,a}, M_{i,a}) \end{cases}$$

where $i \in [2^n]$, $a \in \{-1, 0, 1\}$, and $b \in [M_{i,a}]$.

Note that $v - 1 \leq M_{i,a}$, hence

$$w_{i,a} \leq \frac{v-1}{\sqrt[4]{v}} \leq \frac{v-1}{l} \leq \frac{M_{i,a}}{l}$$

as $w_{i,a} \leq \sqrt[3]{v^2} + 1$ by Lemma 8. So $G_0(i, a, .)$ is a bijection from $[lw_{i,a}]$ onto $[(l-1)w_{i,a}]$ and moreover is a bijection from $[lw_{i,a}, M_{i,a})$ onto $[(l-1)w_{i,a}, m_{i,a})$ as

$$M_{i,a} - lw_{i,a} = m_{i,a} - (l-1)w_{i,a}.$$

Therefore the conclusion is that $G(i, a, .)$ is a bijection from $[M_{i,a}]$ onto $[m_{i,a}]$ for every $i \in [2^n]$ and $a \in \{-1, 0, 1\}$. Now, we define the function $G_1(i, a, b)$ as the inverse of G_0 which means that

$$G_1(i, a, G_0(i, a, b)) = b$$

where $i \in [2^n]$, $a \in \{-1, 0, 1\}$, and $b \in [M_{i,a}]$. So $G_1(i, a, .)$ is a bijection from $[m_{i,a}]$ onto $[M_{i,a}]$.

Using G_0 and G_1, we can fulfill (1) from the lemma. Now we want to construct two other functions H_0 and H_1 to fulfill (2).

The task is to define $H_0(i, a, .)$ as a function that defines a bijection from $[\max\{n^t - v - a, n^t - \gamma(i)\}]$ onto $[\min\{n^t - v - a, n^t - \gamma(i)\}]$ where $i \in [2^n]$ and $a \in \{-1, 0, 1\}$ and moreover H_1 would be the inverse of H_0. Let

1. $M'_{i,a} = \max\{n^t - v - a, n^t - \gamma(i)\}$.
2. $m'_{i,a} = \min\{n^t - v - a, n^t - \gamma(i)\}$.

Then we define $H_0(i,a,b)$ as follows:

$$H_0(i,a,b) := \begin{cases} \sigma(b - lk) + (l-1)k & b \in [lk, l(k+1)) \wedge k \in [w_{i,a}] \\ b - w_{i,a} & b \in [lw_{i,a}, M'_{i,a}) \end{cases}$$

where $i \in [2^n]$, $a \in \{-1, 0, 1\}$, and $b \in [M'_{i,a}]$.

Note that $n^t - v - 1 \leq M'_{i,a}$ and moreover $v \leq n^t/2$, therefore $n^t/2 - 1 \leq M'_{i,a}$. This implies that

$$w_{i,a} \leq \sqrt[3]{v^2}+1 \leq \sqrt[3]{(n^t/2)^2}+1 \leq \sqrt[4]{(n^t/2)^3}-1 \leq \frac{n^t/2-1}{\sqrt[4]{n^t/2}} \leq \frac{n^t/2-1}{\sqrt[4]{v}} \leq \frac{n^t/2-1}{l} \leq \frac{M'_{i,a}}{l}.$$

Therefore $H_0(i,a,.)$ is a bijection from $[lw_{i,a}]$ onto $[(l-1)w_{i,a}]$ and moreover is a bijection $[lw_{i,a}, M'_{i,a})$ onto $[(l-1)w_{i,a}, m'_{i,a})$. Hence $H_0(i,a,.)$ is a bijection from $[M'_{i,a}]$ onto $[m'_{i,a}]$ for every $i \in [2^n]$ and $a \in \{-1,0,1\}$. Now we define the function $H_1(i,a,b)$ as the inverse again as follows:

$$H_1(i,a,H_0(i,a,b)) = b$$

where $i \in [2^n]$, $a \in \{-1,0,1\}$, and $b \in [M'_{i,a}]$. Hence $H_1(i,a,.)$ is a bijection from $[m'_{i,a}]$ onto $[M'_{i,a}]$.

Now $F(i,a,b)$ is:

$$F(i,a,b) := \begin{cases} G_0(i,a,b) & b \in [v+a] \wedge v + a = M_{i,a} \\ G_1(i,a,b) & b \in [v+a] \wedge v + a = m_{i,a} \\ H_0(i,a,b-v-a) & b \in [v+a, n^t) \wedge n^t - v - a = M'_{i,a} \\ H_1(i,a,b-v-a) & b \in [v+a, n^t) \wedge n^t - v - a = m'_{i,a} \end{cases}$$

As G_0, G_1, H_0, H_1 are definable by a bounded arithmetical formula based on γ and σ, therefore F is also definable by a bounded arithmetical formula based on γ and σ and this implies that $F \in \chi'$ as $(\mathcal{M}_{n^t}, \chi') \models \mathsf{V}^0_1$. ◀

Without the loss of generality we can assume $f_{n^t}(1^u 0^{n^t-u}) = 0$. Consider the following relations in χ:

1. $\theta_0 = 1^u 0^{n^t-u}$.
2. $\theta'_0 = 0^{n^t-u} 1^u$.
3. $\theta_1 = 1^{u+1} 0^{n^t-u-1}$.
4. $\theta'_1 = 0^{n^t-u-1} 1^{u+1}$.
5. $\bar{\lambda_0}$ such that $\phi_0(\theta_0, \bar{\lambda_0})$ holds in $(\mathcal{M}_{n^t}, \chi)$.
6. $\bar{\lambda'_0}$ such that $\phi_0(\theta'_0, \bar{\lambda'_0})$ holds in $(\mathcal{M}_{n^t}, \chi)$.
7. $\bar{\lambda_1}$ such that $\phi_1(\theta_1, \bar{\lambda_1})$ holds in $(\mathcal{M}_{n^t}, \chi)$.
8. $\bar{\lambda'_1}$ such that $\phi_1(\theta'_1, \bar{\lambda'_1})$ holds in $(\mathcal{M}_{n^t}, \chi)$.

Now we are ready to describe the assignments $\mathcal{X}$, $\{\bar{\mathcal{Y}}_i\}_{i\in[2^n]}$, and $\{\bar{\mathcal{Z}}_i\}_{i\in[2^n]}$ such that

$$(\mathcal{M}_{n^t}, \chi') \models \forall i < 2^n \left(b_i = 0 \to \phi_0(\mathcal{X}|J_i(A), \bar{\mathcal{Y}}_i)\right) \wedge \left(b_i = 1 \to \phi_1(\mathcal{X}|J_i(A), \bar{\mathcal{Z}}_i)\right)$$

which implies that $\tau_b(\mathsf{NW}_{f,A})$ based on $(\langle\phi_0\rangle_{n^t}, \langle\phi_1\rangle_{n^t})$ fails under an assignment in $(\mathcal{M}_{n^t}, \chi')$. We define these assignments as follows:

1. If $u = v$:
   a. $\mathcal{X} = \alpha$.
   b. $\bar{\mathcal{Y}}_i = \omega(i, F(i, 0, \bar{\lambda}_0))$.
   c. $\bar{\mathcal{Z}}_i = \omega(i, F(i, 1, \bar{\lambda}_1))$.
2. If $u = n^t - v$:
   a. $\mathcal{X}$ is the complement of α, i.e., $\mathcal{X}_j = 1 - \alpha_j$ $j \in [n^s]$.
   b. $\bar{\mathcal{Y}}_i = \omega(i, F(i, 0, \bar{\lambda}'_0))$.
   c. $\bar{\mathcal{Z}}_i = \omega(i, F(i, -1, \bar{\lambda}'_1))$.

Without loss of generality assume $v = u$. Then for an arbitrary $i \in [2^n]$, we know that

$$\mathcal{X}|J_i(A) = \omega(i, F(i, 0, \theta_0)),$$

hence $\sigma_0(\mathcal{X}|J_i(A), \bar{\mathcal{Y}}_i)$ holds by Lemma 3 as $\omega(i, F(i, 0, .))$ is a bijection from $[n^t]$ onto itself and the fact that $\sigma_0(\theta_0, \bar{\lambda}_0)$ holds. The same argument works for $\sigma_1(\mathcal{X}|J_i(A), \bar{\mathcal{Z}}_i)$. Moreover if $v = n^t - u$, the same argument works by using $\theta'_0, \theta'_1, \bar{\lambda}'_0, \bar{\lambda}'_1$.

To complete the proof, we argue as follows. Suppose the statement of the theorem is not true. This means that there exist standard d and r such that the following arithmetical sentence is true in $\mathbb{N}$:

- $\mathbf{H} := \forall s_1 \exists s \geq s_1, \exists t \in [s/r, s], \exists c > 0, \forall m, \exists n > m \exists$ $2^n \times n^s$ n^t-sparse matrix A, $\exists b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$, $\exists$ F_d-proof π for $\tau_b(\mathsf{NW}_{f,A})$ such that $|\pi| \leq |\tau_b(\mathsf{NW}_{f,A})|^c$.

Let $\mathcal{M}$ be a countable nonstandard model of true arithmetic. This means that $\mathcal{M} \models \mathbf{H}$. To simplify the presentation let

$$\mathbf{H} := \forall s_1 \exists s, t, c \forall m \exists n, A, b, \pi \Phi(s_1, s, t, c, m, n, A, b, \pi).$$

Let $s_1 \in \mathcal{M} \setminus \mathbb{N}$, then there exist $s, t \in \mathcal{M} \setminus \mathbb{N}$ and $c \in \mathcal{M}$ such that

$$\mathcal{M} \models \forall m \exists n, A, b, \pi \Phi(s_1, s, t, c, m, n, A, b, \pi).$$

We choose an $m \in \mathcal{M} \setminus \mathbb{N}$ such that for all $m_1 \geq m$, $m_1^{ct} 2^{cm_1} \leq 2^{m_1^{\sqrt{t}/2}}$, hence there exist an $n > m$, a $2^n \times n^s$ n^t-sparse matrix $A \in \mathcal{M} \setminus \mathbb{N}$, a $b \in \mathcal{M} \setminus \mathbb{N}$ such that $b \notin \mathsf{Rng}(\mathsf{NW}_{f,A})$, and an F_d-proof $\pi \in \mathcal{M}$ for $\tau_b(\mathsf{NW}_{f,A})$ such that $|\pi| \leq |\tau_b(\mathsf{NW}_{f,A})|^c$. Now we consider $\mathcal{M}_{n^t}$ and by the argument in this section, there exists a $\chi' \subseteq \mathcal{P}(\mathcal{M}_{n^t})$ such that it has every bounded $\mathcal{M}$-definable subset of $\mathcal{M}_{n^t}$ and moreover

1. $(\mathcal{M}_{n^t}, \chi') \models \mathsf{V}_1^0$.
2. There exists an $\alpha \in \chi'$ which falsifies $\tau_b(\mathsf{NW}_{f,A})$.

Then by Theorem 4 there is no F_d-proof of $\tau_b(\mathsf{NW}_{f,A})$ in $(\mathcal{M}_{n^t}, \chi')$. Note that there is a standard number e such that

$$|\pi| \leq |\tau_b(\mathsf{NW}_{f,A})|^c \leq (n^{ct} 2^{cn})^e \leq 2^{en^{\sqrt{t}/2}}$$

which implies that $\pi \in \chi$, but this leads to a contradiction and completes the proof.

4 What are the implications of the hardness of NW-generators for a proof system?

Some experts believe that random DNFs with suitable parameters give hard formulas to prove in any proof system. The hardness of random DNFs has been proved for several proof systems. One way of proving the hardness of these formulas is by proving the hardness of certain NW-generators. Let A be a $m \times n$ l-sparse matrix such that $m \geq 2n$ and l is a

constant or it is at most $O(\log n)$. Let the base function be the Parity function $\oplus$. Then if we choose a random $b \in \{0,1\}^m$ uniformly, with probability $1-o(1)$, $b \notin \mathsf{Rng}(\mathsf{NW}_{\oplus,A})$. Now, if we choose a random A and a random b uniformly, then with probability $1-o(1)$ $\tau_b(\mathsf{NW}_{\oplus,A})$ is a tautology (here we use DNF representation of the Parity function in the definition of the τ formula). The interesting point about these formulas is that if $\tau_b(\mathsf{NW}_{\oplus,A})$ is hard with probability $1-o(1)$ for a proof system P, then random l-DNFs are hard with probability $1-o(1)$ for P. This strategy was used to prove the hardness of random DNFs for some proof systems (for example, see [14, 3]). For more information, see Section 13.4 of [21]. In this regard, Krajíček [21] asked whether random systems of linear equations over $\mathbb{F}_2$ are hard for AC^0-Frege or not (problem 19.4.5). We note that Theorem 6 partially answers this question as follows.

Let (ϕ_0, ϕ_1) be a $\Sigma^1_1 \cap \Pi^1_1$ definition of a function $f \in \mathsf{NP} \cap \mathsf{CoNP}$ (for example we can take f as the Parity function). Then a random formula $F \sim \mathcal{F}(\phi_0, \phi_1, m, n, l)$ is generated as follows:

1. we choose m subsets $J_0, ..., J_{m-1}$ independently uniformly randomly such that $J_i \subseteq [n]$ and $|J_i| = l$ for every $i \in [m]$. These subsets specify a random $m \times n$ l-sparse matrix A.
2. We choose a random $b \in \{0,1\}^m$ uniformly randomly.
3. Then $F := \tau_b(\mathsf{NW}_{f,A})$ based on $(\langle\phi_0\rangle_l, \langle\phi_1\rangle_l)$.

The following corollary partially answers Krajíček's question.

▶ **Corollary 13.** *Let $f \in \mathsf{NP} \cap \mathsf{CoNP}$ be a symmetric function such that $\mathsf{S}_{\mathsf{AC}^0_2}(f_n) = 2^{n^{\Omega(1)}}$. Let (ϕ_0, ϕ_1) be a $\Sigma^1_1 \cap \Pi^1_1$ definition of f. Then for every d, for every $c > 1$ and every $0 < \epsilon < 1$, if n is large enough, then $F \sim \mathcal{F}(\phi_0, \phi_1, n^c, n, \lfloor n^\epsilon \rfloor)$ is a tautology with probability $1-o(1)$ and it requires exponential F_d-proofs.*

Another implication of the hardness of NW-generators for a proof system P is that it implies that it is hard for P to prove circuit lower bounds effectively. Razborov [37] pointed out that if the base function is in $\mathsf{P/poly}$ and the $2^n \times n^{O(1)}$ matrix A is *efficiently constructible* (an example of such matrices was constructed in [26]), and moreover $\mathsf{NW}_{f,A}$ is a hard proof complexity generator for a proof system P, the P cannot prove circuit lower bounds effectively. Moreover, this implies that $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ does not have efficient proofs in P. Razborov proved such a result for k-DNF Resolution in [37]. In this regard, our results imply a partial answer for the question of the hardness of circuit lower bounds for proof systems. A related question about AC^0-Frege was asked by Krajíček [21] (problem 19.6.1). Let $f \in \mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k)$ and A be a $2^n \times n^s$ n^t-sparse matrix which is *effectively constructible*. Then for any fixed $w \in \{0,1\}^{n^c}$, $\mathsf{NW}_{f,A}(w)$ defines a function $C_w \in (\mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k))/\mathsf{poly}$ as follows:

- For every $i \in \{0,1\}^n$, $C_w(i) = f\left(w|J_{n(i)}(A)\right)$ where $n(i)$ is the number with the binary representation i.

This means that if $\tau_b(\mathsf{NW}_{f,A})$ is a tautology (for a fixed representation of f), then the function with the truth-table b does not have a C_w circuit for any $w \in \{0,1\}^{n^s}$. As Theorem 6 (part 2) implies that NW-generators based on suitable $\mathsf{NP} \cap \mathsf{CoNP}$ functions, suitable matrices, and suitable representations are hard proof complexity generators AC^0-Frege, we get the fact that proving certain $(\mathsf{NP} \cap \mathsf{CoNP})/\mathsf{poly}$ lower bounds (b does not have C_w circuits) for Boolean functions are hard for AC^0-Frege. Note that in contrast with with the principle $f_n \notin \mathsf{SIZE}(n^k)$ which can be written as a propositional formula, it is not clear how the principle $f_n \notin (\mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k))/\mathsf{poly}$ can be written as a propositional formula. So one way of considering this principle in proof complexity is to consider $\tau_{f_n}(\mathsf{NW}_{f,A})$ for any

$g \in \mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k)$, any representation of g and any effectively constructible A. In this regard, Theorem 6 states lower bounds for a lot of possible natural formalizations (but not all) of the principle $f_n \notin (\mathsf{NTime}(n^k) \cap \mathsf{CoNTime}(n^k))/\mathsf{poly}$.

References

1 M. Ajtai. The complexity of the pigeonhole principle. *Combinatorica*, 14(4):417–433, 1994.

2 M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM Journal on Computing*, 34(1):67–88, 2004.

3 E. Ben-Sasson and R. Impagliazzo. Random CNF's are hard for the polynomial calculus. *Computational Complexity*, 19(4):501–519, 2010.

4 M. L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, and T. Pitassi. Non-automatizability of bounded-depth Frege proofs. *Computational Complexity*, 13(1-2):47–68, 2004.

5 S. R. Buss. Bounded arithmetic. Studies in Proof Theory. Lecture Notes, 3. Napoli: Bibliopolis. VII, 221 p. (1986)., 1986.

6 R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. Complexity of Comput., Proc. Symp. appl. Math., New York City 1973, 43-73 (1974)., 1974.

7 E. Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1-3):1–37, 2004.

8 E. Jeřábek. Approximate counting in bounded arithmetic. *The Journal of Symbolic Logic*, 72(3):959–993, 2007.

9 J. Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60. Cambridge: Cambridge Univ. Press, 1995.

10 J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.

11 J. Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-2):123–140, 2001.

12 J. Krajíček. Tautologies from pseudo-random generators. *The Bulletin of Symbolic Logic*, 7(2):197–212, 2001.

13 J. Krajíček. Diagonalization in proof complexity. *Fundamenta Mathematicae*, 182(2):181–192, 2004.

14 J. Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *The Journal of Symbolic Logic*, 69(1):265–286, 2004.

15 J. Krajíček. Structured pigeonhole principle, search problems and hard tautologies. *The Journal of Symbolic Logic*, 70(2):616–630, 2005.

16 J. Krajíček. A proof complexity generator. In *Logic, methodology and philosophy of science. Proceedings of the 13th international congress, Beijing, China, August 2007*, pages 185–190. London: College Publications, 2009.

17 J. Krajíček. A form of feasible interpolation for constant depth Frege systems. *The Journal of Symbolic Logic*, 75(2):774–784, 2010.

18 J. Krajíček. *Forcing with random variables and proof complexity*, volume 382. Cambridge: Cambridge University Press, 2011.

19 J. Krajíček. On the proof complexity of the Nisan-Wigderson generator based on a hard NP $\cap$ coNP function. *Journal of Mathematical Logic*, 11(1):11–27, 2011.

20 J. Krajíček. A reduction of proof complexity to computational complexity for $\mathrm{AC}^0[p]$ Frege systems. *Proceedings of the American Mathematical Society*, 143(11):4951–4965, 2015.

21 J. Krajíček. *Proof complexity*, volume 170. Cambridge: Cambridge University Press, 2019.

22 J. Krajíček. Small Circuits and Dual Weak PHP in the Universal Theory of P-Time Algorithms. *ACM Transactions on Computational Logic*, 22(2), 2021.

23 J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

24 J. Krajíček and P. Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.

25 J. Krajíček, P. Pudlák, and A. Woods. An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures & Algorithms*, 7(1):15–39, 1995.

26 N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

27 J. Paris and A. Wilkie. Counting problems in bounded arithmetic. Methods in mathematical logic, Proc. 6th Latin Amer. Symp., Caracas/Venez. 1983, Lect. Notes Math. 1130, 317-340 (1985)., 1985.

28 J. Pich. Nisan-Wigderson generators in proof systems with forms of interpolation. *Mathematical Logic Quarterly (MLQ)*, 57(4):379–383, 2011.

29 J. Pich. Circuit lower bounds in bounded arithmetics. *Annals of Pure and Applied Logic*, 166(1):29–45, 2015.

30 J. Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. *Logical Methods in Computer Science*, 11(2):38, 2015.

31 J. Pich. Learning algorithms from circuit lower bounds. *arXiv*, 2020. `arXiv:2012.14095`.

32 J. Pich and R. Santhanam. Strong Co-Nondeterministic Lower Bounds for NP Cannot Be Proved Feasibly. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 223–233, 2021.

33 T. Pitassi, P. Beame, and R. Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

34 P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997.

35 P. Pudlák. Incompleteness in the finite domain. *The Bulletin of Symbolic Logic*, 23(4):405–441, 2017.

36 P. Pudlák. The canonical pairs of bounded depth Frege systems. *Annals of Pure and Applied Logic*, 172(2):42, 2021. Id/No 102892.

37 A. A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics. Second Series*, 181(2):415–472, 2015.

38 D. Sokolov. Pseudorandom Generators, Resolution and Heavy Width. In S. Lovett, editor, *37th Computational Complexity Conference (CCC 2022)*, volume 234 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.CCC.2022.15`.

39 A. R. Woods. Approximating the structures accepted by a constant depth circuit or satisfying a sentence – a nonstandard approach. In *Logic and random structures. DIMACS workshop, November 5–7, 1995*, pages 109–130. DIMACS/AMS, 1997.

High-Dimensional Expanders from Chevalley Groups

Ryan O'Donnell
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Kevin Pratt
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Let Φ be an irreducible root system (other than G_2) of rank at least 2, let $\mathbb{F}$ be a finite field with $p = \operatorname{char} \mathbb{F} > 3$, and let $\mathrm{G}(\Phi, \mathbb{F})$ be the corresponding Chevalley group. We describe a strongly explicit high-dimensional expander (HDX) family of dimension $\operatorname{rank}(\Phi)$, where $\mathrm{G}(\Phi, \mathbb{F})$ acts simply transitively on the top-dimensional faces; these are λ-spectral HDXs with $\lambda \to 0$ as $p \to \infty$. This generalizes a construction of Kaufman and Oppenheim (STOC 2018), which corresponds to the case $\Phi = A_d$. Our work gives three new families of spectral HDXs of any dimension ≥ 2, and four exceptional constructions of dimension 4, 6, 7, and 8.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases High-dimensional expanders, simplicial complexes, group theory

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.18

Related Version *Full Version*: https://arxiv.org/abs/2203.03705

1 Introduction

In 1989, Babai, Kantor, and Lubotzky made a conjecture that significantly guided research on expander graphs:

▶ **Conjecture 1** ([7]). *There are constants $k \in \mathbb{N}$ and $\lambda < 1$ such that for every nonabelian finite simple group G, there is a symmetric set $S \subset G$ of $2k$ generators such that the Cayley graph $\mathrm{Cay}(G, S)$ is a λ-spectral expander graph.*

(Here we say that a graph is a λ-spectral expander if all the eigenvalues of its random walk matrix, excluding the largest, are at most λ.)

Notable achievements toward the conjecture include: Kassabov's proof [36] for the alternating groups; work of Lubotzky and Nikolov [37] proving the conjecture for non-Suzuki groups of Lie type (the Chevalley groups and their twisted versions); and, the Breuillard–Green–Tao [11] proof for the Suzuki groups. In light of the Classification of Finite Simple Groups [6], these completed the proof of Conjecture 1. An immediate consequence is that for every nonabelian simple group G, there is a $2k$-regular λ-spectral expander $\mathfrak{K}$ such that G acts transitively on the vertices of $\mathfrak{K}$.

Having expander graphs with such nontrivial symmetry properties (or even stronger ones) has played an important role in applications to computer science. For example, motivated by the search for locally testable codes (see [42]), Kaufman and Wigderson [44] made substantial progress on finding so-called "highly symmetric" LDPC codes of constant rate and relative distance ("good") using expanding Cayley graphs of nonabelian groups; at the same time, they showed that highly symmetric LDPC codes arising from abelian – or even solvable – groups cannot work. Later, notable work of Kaufman and Lubotzky [38]

(see also [9]) positively resolved the problem, giving explicit, highly symmetric, good LDPC codes; the main tool was the use of explicit *edge-transitive* (not just vertex-transitive), highly expanding (indeed, Ramanujan) Cayley graphs of $\mathrm{PSL}_2(\mathbb{F}_q)$ (for $q = 4093$). In turn, the existence of these highly-symmetric expanders arose from the construction of Ramanujan *high-dimensional expanders* (HDXs) [8, 14, 47, 49, 48, 52] from Bruhat–Tits buildings, relying on the Lafforgue's work [45] on the Langlands correspondence.

High-dimensional expanders – defined, say, as simplicial complexes where the 1-skeleton of every link is a λ-spectral expander – have been crucial in many new works in theoretical computer science, either through inspiration, their spectral analysis, or their direct construction. Example applications include results in analysis of Boolean functions [17], computational geometry [25], inapproximability [3, 19], list-decoding [2, 20], Markov chain mixing [4], property testing [21, 16, 39, 41], and quantum codes [24, 43]; particularly notable examples including the resolution of the Mihail–Vazirani Conjecture on the bases-exchange walk for matroids [5] and the construction of locally testable codes of constant rate, distance, and locality [18, 51].

1.1 Our goal

In this paper, we investigate a problem similar to Conjecture 1 for high-dimensional expanders. Namely, for nonabelian finite simple groups G, we seek:

1. bounded-degree λ-spectral HDXs whose top-dimensional faces are acted on transitively by G,
2. with λ arbitrarily close to 0, as opposed to merely bounded away from 1.

(Recall that existence of highly symmetric good LDPC codes was resolved by obtaining *one*-dimensional HDXs – i.e., expander graphs – with both properties.) The aforementioned HDXs built from Bruhat–Tits buildings [47, 49, 48, 52] have property (2) above, and the work of Kaufman and Lubotzky [38] also verified property (1) for $G = \mathrm{PSL}_3(\mathbb{F})$ (for $\mathrm{char}\,\mathbb{F}$ sufficiently large). Later, Kaufman and Oppenheim [40] gave a new (and elementary) construction of HDX families of any dimension d satisfying both (1) and (2) with $G = \mathrm{PSL}_{d+1}(\mathbb{F})$. These two constructions are the only previous examples of bounded-degree λ-spectral HDXs of which we are aware. To quote the final remark from [49]: "*Of course one hopes eventually to define and construct Ramanujan complexes as quotients of the Bruhat–Tits buildings of other simple groups as well.*"

Results

We give strongly explicit constructions of d-dimensional HDX families satisfying properties (1) and (2) above, for any rank-d *Chevalley group* G (except for "G_2") over any field $\mathbb{F}$ of characteristic exceeding 3.[1] Informally, Chevalley groups (also known as the untwisted groups of Lie type) are the finite-field analogues of continuous Lie groups. These groups are specified by two pieces of data: a *root system* Φ, consisting of a set of vectors in $\mathbb{R}^d$ with certain symmetry properties, and a finite field $\mathbb{F}$. Our work gives a general recipe that produces HDX families from Chevalley groups with Φ and the characteristic of $\mathbb{F}$ being fixed, and with $|\mathbb{F}|$ growing. Our approach generalizes that of [40], which corresponds to the case $\Phi = A_d$. As with their work, our construction incidentally gives new families of strongly explicit Δ-degree-bounded λ-spectral expander graphs, with $\lambda \to 0$ as $\Delta \to \infty$.

[1] In fact, we can show that our construction works for characteristic 3 when one excludes the case of G_2. But for simplicity of presentation we will just assume the characteristic exceeds 3.

1.2 Our approach

As in [40], we associate to G a *coset complex*, a kind of d-dimensional simplicial complex determined by G and a choice of subgroups $H_1, \ldots, H_{d+1}$ of G. A few challenges arise in generalizing the construction of [40] to Chevalley groups G of type other than A_d. One immediate question is: what is a "good" choice of $H_1, \ldots, H_{d+1}$? We give one such choice, which has an elegant description in terms of the root system Φ associated with G: the H_i's are certain unipotent subgroups of G (these are essentially groups of upper unitriangular matrices), obtained from a set of fundamental roots of Φ. While all of our constructions can be realized with matrices (see the examples in the next section), it is more convenient in our analysis to work with a set of generators and relations of G known as the *Steinberg presentation*. In particular, the *Chevalley commutator formula* gives us workable descriptions of the subgroups H_i, and the links of our complexes.

As in [40], we apply the *trickling down* theorem of [50] (originating in work of Garland [27]) to show that these coset complexes have expanding links. This theorem says that under a mild connectivity condition, it suffices to show that the links of the $(d-2)$-dimensional faces are good expander graphs. The connectivity condition will follow from some calculations using the properties of Chevalley groups and root systems. In their case of $\Phi = A_d$, Kaufman–Oppenheim establish expansion of links by appealing to a general result of Ershov–Jaikin-Zapirain [22] on expansion in certain groups of nilpotency class two. Unfortunately, to handle root systems Φ that are not "simply-laced", one would need an analogous result for groups of nilpotency class three (and higher, when $\Phi = G_2$). Related results were given in [23] (see its Sec. 10.3), but these are not strong enough for our setting. An alternative, and much simpler, proof of expansion of the Kaufman–Oppenheim complexes was given by Harsha and Saptharishi [31]; their proof was quite specific to the $\Phi = A_d$ case, but we were much inspired its elementary nature.

We prove expansion by observing that, when $\Phi \neq G_2$, the squares of the links of the $(d-2)$-dimensional faces are Cayley graphs of abelian groups. This allows us to express their eigenvalues as character sums, which we bound with an elementary argument that ultimately boils down to the Schwartz–Zippel lemma. (In the G_2 case, the squared links are not abelian Cayley graphs, but we discuss some approach that might be used ito show their expansion.)

1.3 Example constructions

In this section we explicitly give the easiest new HDX family implied by our work. We start by recalling the basic construction of [40], arising from the group $G = \mathrm{SL}_3(\mathbb{F})$.[2] Let $\mathbb{F} = \mathbb{F}_p[x]/(f)$ where p is prime and $f \in \mathbb{F}_p[x]$ is irreducible of degree m. Now define the following three subgroups of G:

[2] In [40] they work over the ring $\mathbb{F}_p[x]/(x^m)$ rather than the field $\mathbb{F}_{p^m}$, but this does not materially change their result, and we prefer to work with the field. Also, regarding the distinction between $\mathrm{SL}_3(\mathbb{F})$ and $\mathrm{PSL}_3(\mathbb{F})$, see Footnote 3.

$$H_1 = \left\{ \begin{bmatrix} 1 & \ell_1 & Q \\ & 1 & \ell_2 \\ & & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1, \deg(Q) \le 2 \right\},$$

$$H_2 = \left\{ \begin{bmatrix} 1 & \ell_1 & \\ & 1 & \\ \ell_2 & Q & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1, \deg(Q) \le 2 \right\},$$

$$H_3 = \left\{ \begin{bmatrix} 1 & & \\ Q & 1 & \ell_1 \\ \ell_2 & & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1, \deg(Q) \le 2 \right\}.$$

Let $\mathfrak{K}(p, m)$ be the 2-dimensional simplicial complex whose vertices are the cosets of these subgroups inside $\mathrm{SL}_3(\mathbb{F})$, and where a "triangle" (2-dimensional face) is added between a triple of cosets g_1H_1, g_2H_2, g_3H_3 whenever $g_1H_1 \cap g_2H_2 \cap g_3H_3 \neq \emptyset$. Edges are included between any two cosets contained in a common triangle. This is an example of a *coset complex*, a well-studied construction from the theory of algebraic groups.

In [40] it was shown that for any fixed $\lambda > 0$, and for sufficiently large p, the complex $\mathfrak{K}(p, m)$ is a bounded-degree λ-spectral HDX. (Here "bounded-degree" means that each vertex of $\mathfrak{K}(p, m)$ is contained in a number of triangles depending only on p.) Moreover, $\mathrm{SL}_3(\mathbb{F})$ acts simply transitively on the set of triangles in $\mathfrak{K}(p, m)$. In a similar manner, Kaufman and Oppenheim show how a d-dimensional HDX family can be associated to $\mathrm{SL}_{d+1}(\mathbb{F})$.

Following this, the most basic new construction provided by our work is as follows. Again, we form a coset complex, but this time we will consider cosets of subgroups of the 4×4 *symplectic group*, $\mathrm{Sp}_4(\mathbb{F})$,[3] defined by

$$\mathrm{Sp}_4(\mathbb{F}) = \left\{ A \in \mathbb{F}^{4\times 4} : A \begin{bmatrix} 0 & I_{2\times 2} \\ -I_{2\times 2} & 0 \end{bmatrix} A^{\intercal} = \begin{bmatrix} 0 & I_{2\times 2} \\ -I_{2\times 2} & 0 \end{bmatrix} \right\}.$$

The vertices of our coset complex $\mathfrak{K}_{\mathrm{Sp}_4}(p, m)$ will be the cosets of the following subgroups of $\mathrm{Sp}_4(\mathbb{F})$:

$$H_1 = \left\{ \begin{bmatrix} 1 & \ell_1 & C & \ell_1\ell_2 + Q \\ & 1 & Q & \ell_2 \\ & & 1 & \\ & & -\ell_1 & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1, \deg(Q) \le 2, \deg(C) \le 3 \right\},$$

$$H_2 = \left\{ \begin{bmatrix} 1 & & \ell_1 & \\ & & 1 & \\ & \ell_2 & Q & 1 \\ \ell_1\ell_2 + Q & C & -\ell_1 & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1, \deg(Q) \le 2, \deg(C) \le 3 \right\},$$

$$H_3 = \left\{ \begin{bmatrix} 1 & & & \\ & 1 & & \ell_1 \\ \ell_2 & & 1 & \\ & & & 1 \end{bmatrix} : \deg(\ell_1), \deg(\ell_2) \le 1 \right\}.$$

The triangles in $\mathfrak{K}_{\mathrm{Sp}_4}(p, m)$ are again added between triples of cosets whenever they have a nontrivial intersection. Our work shows that for any $\lambda > 0$, provided $p \ge 2\frac{(1+\lambda)^2}{\lambda^2}$, the 2-dimensional complexes $(\mathfrak{K}_{\mathrm{Sp}_4}(p, m))_m$ form a (strongly explicit) λ-spectral HDX family of

[3] Technically, this group is not simple; it only becomes the simple group $\mathrm{PSp}_4(\mathbb{F})$ upon identifying the matrices A and $-A$. This is an example of the (very minor) distinction between "universal" and "adjoint" Chevalley groups that is explained in Definition 25.

size $\Theta(p^{10m-4})$ in which each vertex participates in at most p^{22} triangles. Moreover, the group $\mathrm{Sp}_4(\mathbb{F}_{p^m})$ acts on $\mathfrak{K}_{\mathrm{Sp}_4}(p,m)$, with the action being transitive on triangles. Finally, we remark that the underlying skeleton of $\mathfrak{K}_{\mathrm{Sp}_4}(p,m)$ is a (strongly explicit) λ-spectral expander graph of degree at most p^{11} and with $\Theta(p^{10m-4})$ vertices. Since this graph is tripartite, its smallest eigenvalue is at least $-1/2$, and it is therefore also a *two*-sided $1/2$-spectral expander.

1.4 Outline

In Section 2 we give an overview of high-dimensional spectral expansion and coset complexes. We then briefly discuss Chevalley groups and root systems, making explicit all facts about Chevalley groups that we will need.

In Section 3 we give the choice of subgroups used in our coset complex construction. We show in Corollary 55 that these have the connectivity properties needed to apply the trickling down Theorem 3. In Section 3.3 we show that the links of the $(d-2)$-dimensional faces in these complexes are good expander graphs. By Claim 10, this conveniently reduces to studying the expansion of vertex links in three different 2-dimensional complexes, two of which are the examples in Section 1.3.

We conclude with further questions. It is interesting to ask if our analogue of Conjecture 1 has an affirmative answer when G is a more "combinatorial" group; for example, the symmetric/alternating group. We also leave open the case of the Chevalley group based on root system G_2; we conjecture it has the desired expansion properties, and suggest an approach to proving this.

2 Preliminaries

Let $\mathbb{N} = \{0,1,2,\ldots\}$, $\mathbb{Z}_+ = \{1,2,3,\ldots\}$. We identify elements in a finite field $\mathbb{F}$ of size p^m (where p is prime) with polynomials in $\mathbb{F}_p[x]/(f)$ for some irreducible polynomial $f \in \mathbb{F}_p[x]$ of degree m. When we write $\deg(t)$ for $t \in \mathbb{F}$ we mean the degree of the corresponding polynomial in the quotient ring. If g and h are elements of a group, we use the notation $[g,h] = g^{-1}h^{-1}gh$ for their commutator.

2.1 High-dimensional spectral expansion

In this section we recall the notion of spectral HDX families. Let $\mathfrak{K}(0)$ be a finite set. A *simplicial complex* $\mathfrak{K}$ with vertex set $\mathfrak{K}(0)$ is a collection of subsets of $\mathfrak{K}(0)$ satisfying the following conditions:

1. $\{v\} \in \mathfrak{K}$ for all $v \in \mathfrak{K}(0)$;
2. If $\sigma \in \mathfrak{K}$, then $\tau \in \mathfrak{K}$ for all $\tau \subseteq \sigma$.

Said differently, $\mathfrak{K}$ is a downward-closed hypergraph on the set $\mathfrak{K}(0)$. For $i = -1, 0, 1, 2, \ldots$, we denote by $\mathfrak{K}(i)$ the set of subsets of size $i+1$ in $\mathfrak{K}$. An element of $\mathfrak{K}(i)$ is called an *i-dimensional face*. $\mathfrak{K}$ is said to be *pure* if all maximal faces are d-dimensional for some d; in this case we say that d is the *dimension* of $\mathfrak{K}$, denoted $\dim \mathfrak{K}$. (In this work, all simplicial complexes will be pure.) Note that a 1-dimensional simplicial complex can be identified with an ordinary graph. We say that $\mathfrak{K}$ is of *Δ-bounded degree* if every vertex participates in at most Δ maximal faces; and, we say that $\mathfrak{K}$ is *k-partite* if there is a partition of $\mathfrak{K}(0)$ into k parts such that each face has intersection size at most 1 with each part. (Pure $(d+1)$-partite complexes are sometimes called *balanced*, or *numbered*.)

The *link* of a face $\sigma \in \mathfrak{K}$ is the simplicial complex $\mathrm{Link}_\sigma(\mathfrak{K}) = \{\tau \setminus \sigma : \tau \in \mathfrak{K}, \sigma \subseteq \tau\}$. In particular, the link of the (-1)-dimensional face $\emptyset$ is $\mathfrak{K}$. For a pure d-dimensional complex $\mathfrak{K}$, we define the 1-*skeleton* of $\mathfrak{K}$ to be the *multigraph* on vertex set $\mathfrak{K}(0)$ in which $j,k \in \mathfrak{K}(0)$

are connected by a number of edges equal to the number of d-dimensional faces containing $\{j,k\}$. We will say that $\mathfrak{K}$ is *connected* if its 1-skeleton is a connected (multi)graph. Finally, for a face σ, we introduce the notation K_σ for the 1-skeleton of $\mathrm{Link}_\sigma(\mathfrak{K})$, and we will write $\lambda_2(K_\sigma)$ for the second largest eigenvalue of the standard random walk matrix of K_σ. (This refers to the walk on the vertices of K_σ in which a random out-edge is taken at each step.)

By now, the most common definition of expansion for HDXs is probably the following:

▶ **Definition 2** ([50, 40])**.** *A d-dimensional pure simplicial complex $\mathfrak{K}$ is a* λ-spectral HDX *(also known as* λ-link *or* λ-local-spectral *HDX) if $\lambda_2(K_\sigma) \le \lambda$ for all faces σ of dimension at most $d-2$.*

(Note that the $d=1$ case yields the usual notion of a λ-expander graph, one in which the second eigenvalue of the random walk matrix is at most λ.) The trickling down theorem [50] essentially shows that a d-dimensional complex is an HDX provided the links of its $(d-2)$-dimensional faces are λ-expander graphs for $\lambda < \frac{1}{d}$:

▶ **Theorem 3** ([50])**.** *Let $\mathfrak{K}$ be a d-dimensional pure simplicial complex in which* $\mathrm{Link}_\sigma(\mathfrak{K})$ *is connected for all $\sigma \in \mathfrak{K}(i)$, $i \le d-2$. Further suppose that $\lambda_2(K_\sigma) \le \gamma \le \frac{1}{d}$ for all $\sigma \in \mathfrak{K}(d-2)$. Then $\mathfrak{K}$ is a $\left(\frac{\gamma}{1-(d-1)\gamma}\right)$-spectral HDX.*

(We remark that in the case $d=2$, if $\mathfrak{K}$ is a Cayley graph then the conclusion of this theorem can be improved by a factor of $2/\sqrt{3}$; see [55].)

The objects we seek are (highly symmetric versions of) the following:

▶ **Definition 4.** *A* d-dimensional, Δ-bounded degree, λ-spectral HDX family *is a sequence $(\mathfrak{K}_n)_{n\in\mathbb{N}}$ of pure d-dimensional, Δ-bounded degree complexes, with $\mathfrak{K}_n$ having some $n' = \Theta(n)$ vertices, such that $\mathfrak{K}_n$ is a λ-spectral HDX for sufficiently large n. We also say the family is* explicit *if there is a* $\mathrm{poly}(n)$*-time algorithm for computing the description of $\mathfrak{K}_n$, and* strongly explicit *if there is a* $\mathrm{polylog}(n)$*-time algorithm. (See the proof of Theorem 42, item 1 for more details.)*

2.2 Coset complexes

The following notion has been studied since at least the 1950 PhD thesis of Lannér [46]:

▶ **Definition 5.** *Let G be a finite group and let $\mathcal{H} = (H_1, \ldots, H_{d+1})$ be a sequence of subgroups. The associated* coset complex *$\mathcal{CC}(G;\mathcal{H})$ is the pure d-dimensional, $(d+1)$-partite simplicial complex with vertices being the cosets $\bigsqcup_i G/H_i$, and with maximal faces $\{gH_1, \ldots, gH_{d+1} : g \in G\}$. Equivalently, a set of cosets forms a face if all cosets have an element in common.*

Some well-studied instances of coset complexes are *Coxeter complexes* and *Tits buildings* [10].

▶ **Definition 6.** *The ith part of the $(d+1)$-partite coset complex $\mathcal{CC}(G;\mathcal{H})$ is the coset G/H_i, and the* type *of a face σ refers to the subset of parts $[d+1]$ to which its vertices belong.*

The group G naturally acts on $\mathcal{CC}(G;\mathcal{H})$ by left-multiplication, and it is easy to see the following:

▷ Claim 7. The action of G on the $(d+1)$-partite complex $\mathcal{CC}(G;\mathcal{H})$ is *type-preserving* (it does not change the type of any face), and transitive on the maximal faces. Moreover, the action is simply transitive if $H_1 \cap H_2 \cap \cdots \cap H_{d+1} = \{1\}$.

(In fact, Lannér [46] showed that whenever there is a G-action on some $(d+1)$-partite complex that is type-preserving and transitive on maximal faces, then the complex must be of the form $\mathcal{CC}(G;\mathcal{H})$ for some subgroups $H_1,\dots,H_{d+1}$.)

We can also easily understand the connectivity and link structure of coset complexes, as the following facts show.

▶ **Definition 8.** *Given $\mathcal{H}$ and $T \subseteq [d+1]$ we write $H_T = \bigcap_{i\in T} H_i$, with the convention that $H_\emptyset = \langle H_1,\dots,H_{d+1}\rangle$, the subgroup of G generated by $\mathcal{H}$.*

The following facts are easy to prove:

▷ Claim 9 ([1]). $\mathcal{CC}(G;\mathcal{H})$ is connected if and only if $H_\emptyset = G$.

▷ Claim 10 ([28],p. 13, [31]). Let σ be a face in $\mathcal{CC}(G;\mathcal{H})$ of type $T \neq \emptyset$. Then the link of F is isomorphic to the coset complex $\mathcal{CC}(H_T;(H_{T\cup\{i\}} : i \notin T))$.

Note that Claim 10 says that, up to isomorphism, the link of a face only depends on its type. This will help us apply Theorem 3, as we will only have to consider a small number of cases. Finally we quote another easy-to-prove claim from Kaufman and Oppenheim, which we can use to pass between the (very slightly different) different universal and adjoint Chevalley groups:

▷ Claim 11 ([40], essentially Prop. 2.12). Let $\overline{\mathfrak{K}} = \mathcal{CC}(\overline{G};\overline{\mathcal{H}})$ be a coset complex with $\overline{\mathcal{H}} = (\overline{H}_1,\dots,\overline{H}_{d+1})$, suppose $Z \triangleleft \overline{G}$ is a normal subgroup (e.g., if Z is the center of $\overline{G}$), and suppose that $Z \cap \overline{H}_i = \{1\}$ for all $i \in [d+1]$. Then for $G = \overline{G}/Z$ and $\mathcal{H} = (H_1,\dots,H_{d+1})$, where $H_i = \overline{H}_i Z/Z$, the coset complex $\mathfrak{K} = \mathcal{CC}(G;\mathcal{H})$ is "covered" by $\overline{\mathfrak{K}}$, and the following property holds: every link in $\mathfrak{K}$ of type $T \neq \emptyset$ is isomorphic to every link of type T in $\overline{\mathfrak{K}}$.

▶ Remark 12. A consequence of Claim 11 is that if $\overline{\mathfrak{K}}$ is a Δ-bounded degree, λ-spectral HDX, then so too is $\mathfrak{K}$; moreover, provided $H_1Z \cap H_2Z \cap \cdots \cap H_{d+1}Z = Z$, the group G acts simply transitively on the maximal faces of $\mathfrak{K}$. We note that our complexes satisfy this condition in Observation 53.

2.3 Root systems

Killing and Cartan [12] classified simple Lie algebras over $\mathbb{C}$ via root systems:

▶ **Definition 13.** *A* (reduced) root system *of rank d is a finite set Φ of nonzero vectors spanning a d-dimensional real vector space such that for each $\alpha \in \Phi$:*

- *Φ is closed under w_α, where w_α is the reflection through the hyperplane orthogonal to α;*
- *$w_\alpha(\beta) - \beta$ is an integer multiple of α for all $\beta \in \Phi$;*
- *for $\lambda \in \mathbb{R}$ we have $\lambda\alpha \in \Phi$ (if and) only if $\lambda \in \{\pm 1\}$.*

The root system Φ is irreducible *if it cannot be written as $\Phi_1 \sqcup \Phi_2$ with Φ_1, Φ_2 nonempty and lying in orthogonal subspaces. Root system Φ' is said to be* isomorphic *to Φ if there is bijection between them that preserves inner products up to a fixed positive scalar multiple.*

Figure 1 shows the three non-isomorphic rank-2 root systems (all of which are irreducible). The irreducible root systems have been completely classified:

▶ Note 14. Up to isomorphism, the irreducible root systems are classified as the families A_d $(d \geq 1)$, B_d $(d \geq 2)$, C_d $(d \geq 3)$, D_d $(d \geq 4)$, and the exceptional systems G_2, F_4, E_6, E_7, E_8. In all cases, the subscript gives the dimension of the root system. For explicit descriptions of these root systems, see e.g. [13, Sec. 3.6].

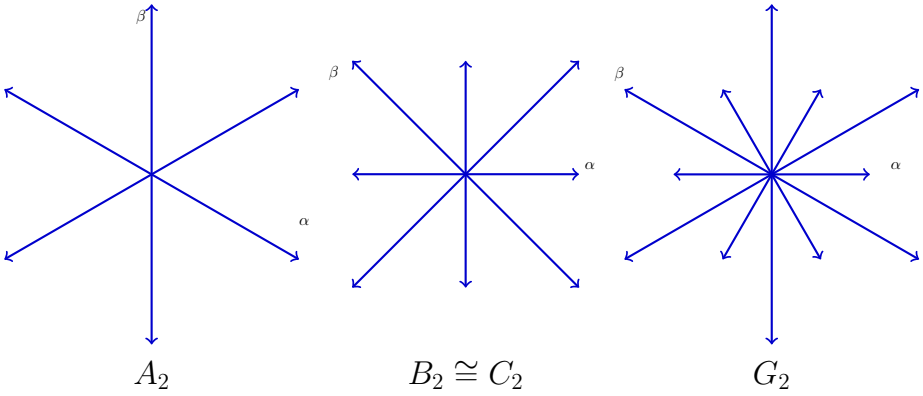


Figure 1 The rank 2 (irreducible) root systems, with a simple set $\{\alpha, \beta\}$ shown.

▶ Remark 15. The restriction of a root system to a subspace is also a root system. Thus if Φ is a root system containing roots α, β, and $\alpha + \beta$, then the restriction of Φ to the subspace spanned by α and β must be (isomorphic to) A_2, B_2, or G_2. In fact, since G_2 is the only irreducible root system containing vectors at an angle of 30° (see, e.g., [13, Sec. 3.6]), an irreducible root system containing G_2 as a subsystem must in fact be isomorphic to G_2.

Let us now record a handy fact involving the inner product $\alpha \cdot \beta$ of two roots:

▷ Claim 16 ([34], p. 45, Lem. 9.4).] Let α, β be roots. If $\alpha \cdot \beta < 0$ then $\alpha + \beta \in \Phi \cup \{0\}$, and if $\alpha \cdot \beta > 0$ then $\alpha - \beta \in \Phi \cup \{0\}$.

This fact can be used to prove another simple result (which is surely well known, though we could not find a reference):

▷ Claim 17. Let Φ be an irreducible root system of rank at least 2, and let $\alpha \in \Phi$. Then α is the sum of two other roots.

Proof. We claim there must exist a root $\beta \neq \pm\alpha$ with $\alpha \cdot \beta \neq 0$. Otherwise, every root is either orthogonal to α or parallel to α, meaning Φ is either irreducible or of rank 1. We may assume $\alpha \cdot \beta > 0$, by replacing β by the root $-\beta$, if necessary. Thus Claim 16 tells us that $\alpha - \beta \in \Phi$. But now $\alpha - \beta$ and β are roots summing to α. ◁

We now discuss "simple" subsets of roots:

▶ **Definition 18.** *Let Φ be a root system spanning $\mathbb{R}^d$. A set of roots $\Pi = \{\alpha_1, \ldots, \alpha_d\} \subseteq \Phi$ is called* simple *(or a* base*) if it is a basis for $\mathbb{R}^d$, and every root $\gamma \in \Phi$ may be expressed as*

$$\gamma = n_1\alpha_1 + \cdots + n_d\alpha_d$$

either with $n_1, \ldots, n_d \in \mathbb{N}$ or with $-n_1, \ldots, -n_d \in \mathbb{N}$. (Since Π is a basis, there is a unique such expression.) In the former case, γ is called a positive root*; in the latter case, a* negative root*. One also defines the* height *of γ (with respect to Π, or more generally a set of linearly independent roots whose span contains γ), denoted $\mathrm{ht}_\Pi(\gamma)$, to be $\sum_{i=1}^{d} |n_i|$.*

(In Figure 1, each root system has labeled a simple set $\{\alpha, \beta\}$.)

In a certain sense, up to symmetries there is a unique choice of simple roots for a given root system:

▷ Claim 19 ([13], Prop. 2.1.2, Cor 2.2.5). Every root system has a set of simple roots. Further, for any two simple sets, there is a unique reflection w_α mapping one to the other.

▶ **Definition 20.** *For any subset $\Psi \subseteq \Phi$, we write $\Psi^+ = \Phi \cap \{\sum_{\alpha\in\Psi} n_\alpha \alpha : n_\alpha \in \mathbb{N}\}$, and $\Psi^- = -\Psi^+$.*

▷ Claim 21. Let $\Psi \subset \Phi$ be a set of linearly independent roots. Then there is set of simple roots Π of Φ where $\Psi \subseteq \Pi^+$.

Proof. We can always find a hyperplane H not containing any root, and where all of Ψ is contained on one side of H. Then by [29, Thm. 8.16], there is a set of simple roots Π such that the roots in Φ on this side of H are positive with respect to Π. ◁

The following fact is very similar to a standard one about root systems, but it is usually only stated when $\{\alpha, \dots, \alpha_m\}$ form a simple set (see, e.g., [13, Lem. 3.6.2]):

▷ Claim 22. Let $A = \{\alpha_1, \dots, \alpha_\ell\} \subseteq \Phi$ be *any* set of roots, and suppose that $\gamma = \sum_{i=1}^{\ell} n_i\alpha_i \in \Phi$ for $n_1, \dots, n_\ell \in \mathbb{N}$. Then we may express $\gamma = \sum_{j=1}^{m} \alpha_{i_j}$ for certain $i_1, \dots, i_m \in [\ell]$ in such a way that all the prefix-sums $\sum_{j=1}^{k} \alpha_{i_j}$ $(1 \le k \le \ell)$ are in Φ.

Proof. By induction, it suffices to show that if γ is not already in A, then there exists $i_0 \in [\ell]$ with $n_{i_0} > 0$ such that $\gamma - \alpha_{i_0} \in \Phi$. To do this, note that $0 < \gamma \cdot \gamma = \sum_{i=1}^{\ell} n_i(\gamma \cdot \alpha_i)$, and since the n_i's are nonnegative we must have $\gamma \cdot \alpha_{i_0} > 0$ for (at least) one i_0. By Claim 16 we conclude that $\gamma - \alpha_{i_0} \in \Phi \cup \{0\}$, and the case $\gamma - \alpha_{i_0} = 0$ (i.e., $\gamma = \alpha_{i_0}$) is impossible because γ is assumed not already in A. ◁

Finally, we need the following known fact [32]:

▷ Claim 23. Let Φ be an irreducible root system with simple roots $\Pi = \{\alpha_1, \dots, \alpha_d\}$. Then $\sum_{i=1}^{d} \alpha_i \in \Phi$.

2.4 Chevalley groups

We may now define the Chevalley groups, via the *Steinberg presentation* (see, e.g., [13, Thm. 12.1.1]).

▶ **Definition 24.** *Corresponding to any irreducible root system Φ of rank at least 2, and any finite field $\mathbb{F}$, there is an associated* universal (or simply connected) Chevalley group*, denoted $\overline{\mathrm{G}}(\Phi, \mathbb{F})$. Abstractly, it is generated by symbols $x_\alpha(t)$ for $\alpha \in \Phi$ and $t \in \mathbb{F}$, subject to the relations*

$$
\begin{aligned}
x_\alpha(t)x_\alpha(u) &= x_\alpha(t+u) \\
[x_\alpha(t), x_\beta(u)] &= \prod_{i,j>0} x_{i\alpha+j\beta}(C_{ij}^{\alpha,\beta} t^i u^j) \qquad \textit{(for } \alpha+\beta \neq 0\textit{)} \\
h_\alpha(t)h_\alpha(u) &= h_\alpha(tu) \quad \textit{(for } tu \neq 0\textit{)}, \\
\textit{where} \qquad h_\alpha(t) &= n_\alpha(t)n_\alpha(-1) \\
\textit{and} \qquad n_\alpha(t) &= x_\alpha(t)x_{-\alpha}(-t^{-1})x_\alpha(t).
\end{aligned}
$$

The second relation above is the Chevalley commutator formula*, and it is elaborated upon in Theorem 28 below.*

Table 1 The Chevalley groups corresponding to classical root systems.

Type of Φ	$\mathrm{G}(\Phi,\cdot)$	$\overline{\mathrm{G}}(\Phi,\cdot)$
A_d	PSL_{d+1}	SL_{d+1}
B_d	SO_{2d+1}	Spin_{2d+1}
C_d	PSp_{2d}	Sp_{2d}
$D_{2\ell}$	$\mathrm{PSO}_{4\ell}$	$\mathrm{Spin}_{4\ell}$
$D_{2\ell+1}$	$\mathrm{PSO}_{4\ell+2}$	$\mathrm{Spin}_{4\ell+2}$

▶ **Definition 25.** *Let* $\mathrm{Z}(\Phi,\mathbb{F})$ *denote the center of* $\overline{\mathrm{G}}(\Phi,\mathbb{F})$*. The* adjoint Chevalley group*, which we denote by* $\mathrm{G}(\Phi,\mathbb{F})$*, is the quotient* $\overline{\mathrm{G}}(\Phi,\mathbb{F})/\mathrm{Z}(\Phi,\mathbb{F})$*. In all cases,* $\mathrm{Z}(\Phi,\mathbb{F})$ *is a constant-sized subgroup (of size* $d+1$ *for* $\Phi = A_d$*, and of size at most* 4 *otherwise).*[4] *It is generated by certain products* $\prod_{\alpha\in\Pi} h_\alpha(t_\alpha)$ *(i.e., diagonal matrices in the matrix realizations), where* Π *is a simple set of roots and the* t_α*'s are roots of unity in* $\mathbb{F}$*.*

▶ Remark 26. The Classification of Finite Simple Groups [6] states that as $\mathbb{F}$ ranges over all finite fields, the adjoint Chevalley groups (excluding $\mathrm{G}(A_1,\mathbb{F}_2)$, $\mathrm{G}(A_1,\mathbb{F}_3)$, $\mathrm{G}(B_2,\mathbb{F}_2)$, $\mathrm{G}(G_2,\mathbb{F}_2)$, but including the "twisted" versions, which we do not discuss in this work) constitute the finite simple groups, together with the cyclic, alternating, and sporadic simple groups.

Although, strictly speaking, it is the adjoint Chevalley groups that are the simple ones, it is more convenient to work with the very slightly larger universal Chevalley groups. If one wants to precisely fulfill the goal concerning simple (adjoint) Chevalley groups described in Section 1.1, one may use do so by appealing to Remark 12. But henceforth we work exclusively with the universal Chevalley groups, and we will drop the adjective "universal".

Although we have defined the Chevalley groups abstractly, we have [54, Sec. 3.3] the isomorphisms with classical groups shown in Table 1, for the "classical" root systems of types A, B, C, and D.

Identifications of the root elements $x_\alpha(t)$ of the Chevalley groups of classical type as elements of the corresponding matrix groups can be found in [13, Sec. 11.3], and matrix realizations for the exceptional Chevalley groups can be found in [33].

As we discuss in Section 2.5, we have

$$n := |\overline{\mathrm{G}}(\Phi,\mathbb{F})| = |\mathbb{F}|^{\Theta(1)} = \exp(\Theta(m))$$

for fixed p and Φ; indeed, an exact formula for $|\overline{\mathrm{G}}(\Phi,\mathbb{F})|$ is known, and one can compute within $\overline{\mathrm{G}}(\Phi,\mathbb{F})$ (and $\mathrm{G}(\Phi,\mathbb{F})$) in $\mathrm{poly}(m) = \mathrm{polylog}(n)$ time (see Section 2.5).

▶ Remark 27. From the first relation of Definition 24, it follows that the subgroup $\langle x_\alpha(r) : r \in \mathbb{F}\rangle$ of $\overline{\mathrm{G}}(\Phi,\mathbb{F})$ is isomorphic to the additive group of $\mathbb{F}$. This subgroup is called the *root subgroup* associated to α.

The second relation in Definition 24 will be used to give explicit descriptions of the links in our constructions, so we elaborate on it here.

[4] Specifically, it is isomorphic to $\mathbb{Z}_{d+1}$ when $\Phi = A_d$, to $\mathbb{Z}_2$ when $\Phi \in \{B_d, C_d, E_7\}$, to $\mathbb{Z}_4$ or $\mathbb{Z}_2\times\mathbb{Z}_2$ when $\Phi = D_d$ (for odd, even d respectively), to $\mathbb{Z}_3$ when $\Phi = E_7$, and is trivial otherwise. [54, Sec. 3.3].

▶ **Theorem 28.** *The* Chevalley commutator formula *asserts that within* $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ *and* $\mathrm{G}(\Phi, \mathbb{F})$*, if* $\alpha, \beta \in \Phi$ *with* $\alpha + \beta \neq 0$*, and* $t, u \in \mathbb{F}$*, then*

$$[x_\alpha(t), x_\beta(u)] = \prod_{\substack{i,j \in \mathbb{Z}_+ \\ i\alpha + j\beta \in \Phi}} x_{i\alpha+j\beta}(C_{ij}^{\alpha,\beta} t^i u^j)$$

for certain structure constants $C_{ij}^{\alpha,\beta} \in \{\pm 1, \pm 2, \pm 3\}$ *that can be found in, e.g., [13, Sec. 5.2]. Here the product above is taken in order of increasing* $i + j$*.*[5] *In addition, the structure constants only depend on the set* $\{(i, j) : i\alpha + j\beta \in \Phi\}$*.*

▶ Remark 29. In particular, the commutator formula implies that if $\alpha + \beta \notin \Phi \cup \{0\}$, then $[x_\alpha(r), x_\beta(r)] = 1$.

▶ Remark 30. The constants $C_{ij}^{\alpha,\beta}$ are determined uniquely by Φ up to signs. Different signs can arise from different choices of a *Chevalley basis*. The resulting groups are isomorphic, however. See [13, Prop. 4.2.2] and the preceding discussion.

Although not strictly necessary for our work, we give explicit structure constants in the following description of the commutator formula for root systems of rank 2:

▶ **Proposition 31** ([35], Sec. 33.3–33.5)**.** *Let* Φ *be one of* A_2, B_2*, or* G_2 *and let* $t, u \in \mathbb{F}$*. Then:*[6]

- *If* $\Phi = A_2$ *with positive roots* $\alpha, \beta, \alpha + \beta$*, then*

$$[x_\alpha(t), x_\beta(u)] = x_{\alpha+\beta}(tu).$$

- *If* $\Phi = B_2$ *with positive roots* $\alpha, \beta, \alpha + \beta, 2\alpha + \beta$*, then*

$$[x_\beta(t), x_\alpha(u)] = x_{\alpha+\beta}(tu)x_{2\alpha+\beta}(t^2u)$$
$$[x_{\alpha+\beta}(t), x_\alpha(u)] = x_{2\alpha+\beta}(2tu).$$

- *If* $\Phi = G_2$ *with positive roots* $\alpha, \beta, \alpha + \beta, 2\alpha + \beta, 3\alpha + \beta, 3\alpha + 2\beta$*, then*

$$[x_\beta(t), x_\alpha(u)] = x_{\alpha+\beta}(tu)x_{2\alpha+\beta}(tu^2)x_{3\alpha+\beta}(tu^3)x_{3\alpha+2\beta}(-t^2u^3)$$
$$[x_{\alpha+\beta}(t), x_\alpha(u)] = x_{2\alpha+\beta}(2tu)x_{3\alpha+\beta}(3tu^2)x_{3\alpha+2\beta}(-3t^2u)$$
$$[x_{2\alpha+\beta}(t), x_\alpha(u)] = x_{3\alpha+\beta}(3tu)$$
$$[x_{3\alpha+\beta}(t), x_\beta(u)] = x_{3\alpha+2\beta}(tu)$$
$$[x_{2\alpha+\beta}(t), x_{\alpha+\beta}(u)] = x_{3\alpha+2\beta}(-3tu).$$

Finally, we will require two more key facts:

▶ **Proposition 32** ([54], Lem. 17)**.** *In the Chevalley group* $\overline{\mathrm{G}}(\Phi, \mathbb{F})$*, suppose* $S \subset \Phi$ *is a set of roots with the following two properties: (i)* $\alpha, \beta \in S$ *and* $\alpha + \beta \in \Phi$ *implies* $\alpha + \beta \in S$*; (ii)* $\alpha \in S$ *implies* $-\alpha \notin S$*. Then each element of the subgroup* $\langle x_\alpha(t) : \alpha \in S, t \in \mathbb{F} \rangle$ *can be expressed uniquely as* $\prod_{\alpha \in S} x_\alpha(t_\alpha)$ *for some* $t_\alpha \in \mathbb{F}$*, where the product is taken in some fixed order (and this is true for any fixed ordering of* S *for the product).*

▶ **Proposition 33.** *Let* Π *be a set of simple roots, and define the two subgroups* $U^\pm = \langle x_\alpha(t) : \alpha \in \Pi^\pm \rangle$*. Then* $U^+ \cap U^- = \{1\}$*.*

Proof. By [54, Lem. 18, Cor. 3], $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ can be realized as a group of matrices over $\mathbb{F}$ where the subgroup U^+ is upper-unitriangular and U^- is lower-unitriangular. The proposition follows. ◀

[5] Ties may be broken arbitrarily, as it turns out that elements with equal $i + j$ commute.

[6] For G_2 we fix the signs implemented in the GAP [26] package Unipot [30], which we used for calculations in Remark 60.

2.5 Computation within the Chevalley groups

Given field $\mathbb{F} = \mathbb{F}_q = \mathbb{F}_{p^m}$ and root system Φ of rank d, let us treat Φ and p as fixed, and $m \to \infty$ as an asymptotically growing parameter. Here we recap the known facts that the Chevalley group $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ has order $n = \exp(\Theta(m))$ and that one can compute within $\overline{G}$ in deterministic $\mathrm{poly}(m) = \mathrm{polylog}(n)$ time. (The same is true for the adjoint Chevalley group $\mathrm{G}(\Phi, \mathbb{F})$.)

First, field arithmetic is efficient, thanks to Shoup:

▶ **Theorem 34** ([53]). *For a fixed prime p, there is an deterministic $\mathrm{poly}(m)$-time algorithm for finding an irreducible $f \in \mathbb{F}_p[x]$ of degree d, and thereby "constructing" the field $\mathbb{F} = \mathbb{F}_q = \mathbb{F}_{p^m}$. The elements of $\mathbb{F}$ are encoded by bit-strings of length $\Theta(m)$, and field operations may be computed in deterministic $\mathrm{poly}(m)$ time – this includes computing all kth roots of unity in $\mathrm{poly}(k, m)$ time.*

Next, we note that there is an easy-to-compute formula for the order of a given Chevalley group:

▶ **Theorem 35.** *For Φ of rank d, the order of the group $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ is of the form $q^{\Theta(d^2)} = p^{\Theta(d^2 m)}$, where the constant hidden in the $\Theta(\cdot)$ depends only on Φ. Moreover there is a precise formula for $|\overline{\mathrm{G}}(\Phi, \mathbb{F})|$ that can easily be computed in $\mathrm{poly}(d, \log p, m)$ time; see, e.g. [54, Thm. 25]. (All of this is also true of $\mathrm{G}(\Phi, \mathbb{F})$.)*

Finally, we appeal to the work of Cohen, Murray, and Taylor [15] to show that one can efficiently construct and compute within Chevalley groups:

▶ **Theorem 36** ([15], see especially Sec. 8.1). *For Φ of rank d and $\mathbb{F} = \mathbb{F}_{p^m}$, there is a canonical representation ("Bruhat normal form") for each element of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$, encoded by a bit-string of length $\mathrm{poly}(d, \log q, m)$. One can pass between this form, a natural matrix representation, and an expression in the Steinberg presentation – and also compute group products and inverses – via deterministic $\mathrm{poly}(d, \log q, m)$-time algorithms. (Since kth roots of unity can also be computed efficiently (Theorem 34), the $O(d)$-size center Z of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ can also be constructed efficiently, and hence this whole theorem is also true for $\mathrm{G}(\Phi, \mathbb{F})$.)*

3 The Construction

For the rest of the paper we fix a field $\mathbb{F}$ of size p^m where $p > 3$, an irreducible root system Φ of rank at least 2, and a set of simple roots $\Pi = \{\alpha_1, \ldots, \alpha_d\} \subset \Phi$. With this in mind, $x_\alpha(t)$ refers to the corresponding root element of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$.

▶ **Definition 37.** *For $S \subseteq \Phi$ and $d \in \mathbb{N}$, let $X_{S,d} = \langle x_\alpha(t) : \alpha \in S, t \in \mathbb{F}, \deg(t) \leq d \rangle$. For shorthands we write $X_S = X_{S,1}$ and also $X_{\alpha,d} = X_{\{\alpha\},d}$.*

▶ **Definition 38.** *Recalling $\Pi = \{\alpha_1, \ldots, \alpha_d\}$, we define $\mathcal{S}$ to be the following particular set of roots:*

$$\mathcal{S} = \Pi \cup \{-(\alpha_1 + \cdots + \alpha_d)\}. \tag{1}$$

(The last of these is a root by by Claim 23.)

▶ Remark 39. Since Π is a basis, it follows that every subset of $\mathcal{S}$ of cardinality d is linearly independent.

▶ **Definition 40.** *For each $\alpha \in \mathcal{S}$, we introduce the following subgroup of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$:*

$$H_\alpha = X_{\mathcal{S}\setminus\{\alpha\}}.$$

Finally, we can introduce our coset complex:

▶ **Definition 41.** $\mathfrak{K} = \mathfrak{K}_m := \mathcal{CC}(\overline{\mathrm{G}}(\Phi, \mathbb{F}); (H_\alpha)_{\alpha\in\mathcal{S}})$.

▶ **Theorem 42.** *For $d = \mathrm{rank}(\Phi)$, it holds that $\mathfrak{K}$ is a d-dimensional pure simplicial complex, where:*

1. *$|\mathfrak{K}(0)| = p^{\Theta(m)}$, where the constant hidden by $\Theta(\cdot)$ depends only on Φ; moreover, the family that arises as $m \to \infty$ is strongly explicit.*
2. *Every vertex participates in at most $\Delta = \Delta(\Phi, p) = p^{\Theta(1)}$ maximal faces, where the $\Theta(\cdot)$ constant (independent of m) depends only on Φ (indeed, it is $\Theta(d^2)$).*
3. *If $p > 3$,[7] then $\mathrm{Link}_\sigma(\mathfrak{K})$ is connected for all $\sigma \in \mathfrak{K}(i)$, $i \le d-2$.*
4. *If $\Phi \neq G_2$ and $p > 2$, then for all $\sigma \in \mathfrak{K}(d-2)$ it holds that K_σ is a p^2-regular bipartite graph with $\lambda_2(K_\sigma) \le \sqrt{2/p}$.*
5. *$\overline{\mathrm{G}}(\Phi, \mathbb{F})$ acts simply transitively on the maximal faces of $\mathfrak{K}$ (and this is also true if one constructs $\mathfrak{K}$ from $\mathrm{G}(\Phi, \mathbb{F})$ rather than $\overline{\mathrm{G}}(\Phi, \mathbb{F})$).*

By Theorem 3, we conclude our final goal:

▶ **Corollary 43.** *Fixing $\Phi \neq G_2$ of rank $d \ge 2$, $p > 3$ prime, and taking $m \to \infty$, the sequence $(\mathfrak{K}_m)$ forms a strongly explicit d-dimensional, Δ-bounded degree ($\Delta = p^{\Theta(d^2)}$), λ-spectral HDX family, where*

$$\lambda \le \frac{1}{\sqrt{p/2} - d + 1}.$$

(Hence for large p, we have $\lambda \sim 1/\Delta^{\Theta(1/d^2)}$.) Moreover, the universal Chevalley group $\overline{\mathrm{G}}(\Phi, \mathbb{F})$ acts simply transitively on $\mathfrak{K}_m$'s maximal faces.

We add that the results all remain true if uses the (simple) adjoint Chevalley groups $\mathrm{G}(\Phi, \mathbb{F})$ in place of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$.

3.1 Global connectivity of the coset complex

The main goal of this section is to show that the subgroups H_α for $\alpha \in \mathcal{S}$ generate $\overline{\mathrm{G}}(\Phi, \mathbb{F})$. By Claim 9, this is necessary to ensure that the 1-skeleton of $\mathfrak{K}$ is connected.

▶ **Theorem 44.** *Let $S \subseteq \Phi$ be a subset of $\mathrm{rank}(\Phi) + 1$ roots where $S^+ = \Phi$. Then $X_S = \overline{\mathrm{G}}(\Phi, \mathbb{F})$.*

The particular set of roots $\mathcal{S}$ we selected in Equation (1) has the desired property, as the following shows:

▶ **Proposition 45.** *For $\mathcal{S}$ as in* Equation (1) *we have $\mathcal{S}^+ = \Phi$.*

Proof. We have $\mathcal{S}^+ \supseteq \Pi^+$, and so $\mathcal{S}^+$ certainly contains all positive roots in Φ (recall Definition 18). It remains to show that $\mathcal{S}^+$ contains each negative root $\gamma \in \Phi$. Writing $\gamma = -n_1\alpha_1 - \cdots - n_d\alpha_d$, it follows that we can reexpress it as

$$\gamma = r(-(\alpha_1 + \cdots + \alpha_d)) + r_1\alpha_1 + \cdots + r_d\alpha_d$$

for a sufficiently large positive integer r, and positive integers $r_1, \ldots, r_d$. Thus indeed $\gamma \in \mathcal{S}^+$. ◀

[7] Recall Footnote 1.

▶ **Example 46.** A_d is the set of vectors $\{e_i - e_j, i \neq j\} \subset \mathbb{R}^d$. A set of simple roots is given by $\Pi = \{e_i - e_{i+1} : i \in [d]\}$; in this case $-\sum_{\alpha\in\Pi} \alpha = e_d - e_1$. It is straightforward to check that $S = \{e_i - e_{i+1} : i \in [d]\} \cup \{e_d - e_1\} \subset A_d$ satisfies the hypothesis of Theorem 44. This is the set of roots implicitly used in [40].

▶ Remark 47. There are other choices of S besides our $\mathcal{S}$ from Equation (1) that satisfy the condition of Theorem 44. These can be used to obtain slightly different constructions. For example, referring to Figure 1 one see that in B_2 one can take $S = \{\alpha, \beta, -\beta - 2\alpha\}$, or in G_2 one can take $S = \{\alpha, \alpha + \beta, -2\alpha - \beta\}$.

We will require the following (presumably known) fact:

▶ **Lemma 48.** *For $i, j, d_1, d_2 \in \mathbb{N}$ with $\mathrm{char}(\mathbb{F}) > \max(i, j)$, write $d = id_1 + jd_2$. Then*

$$\mathbb{F}[x]^{\leq d} = \mathrm{span}\{f^i g^j : f \in \mathbb{F}[x]^{\leq d_1},\ g \in \mathbb{F}[x]^{\leq d_2}\}.$$

where $\mathbb{F}[x]^{\leq k}$ represents the polynomials of degree at most k.

Proof. It suffices to establish that x^e is in the span, for any $e \leq d$. Express $e = a_1 + \cdots + a_i + b_1 + \cdots + b_j$, with each a being a natural number at most d_1 and each b being a natural number at most d_2. Now note that the monomial

$$x_{a_1} x_{a_2} \cdots x_{a_i} x_{b_1} x_{b_2} \cdots x_{b_j} \tag{2}$$

becomes equal to x^e if each indeterminate x_c is substituted with x^c. Next, we use the identity

$$x_{a_1} x_2 \cdots x_{a_i} = \frac{1}{i!} \sum_{s\in\{0,1\}^i} (-1)^{|s|+i} \left(\sum_{\ell=1}^{i} s_\ell x_{a_\ell}\right)^i,$$

with the constant $\frac{1}{i!}$ being sensible in the field $\mathbb{F}$ since $\mathrm{char}(\mathbb{F}) > i$. (This is the "higher order polarization identity", or Ryser's formula applied to the matrix where every row is $\begin{bmatrix} x_{a_1} & x_{a_2} & \cdots & x_{a_i} \end{bmatrix}$.) Multiplying this against the analogous identity with the b's (and using $\mathrm{char}(\mathbb{F}) > j$), we get that (2) can be expressed as a linear combination of multivariate polynomials $F^i G^j$, where each F is a linear combination of x_c's with $c \leq d_1$ and each G is a linear combination of x_c's with $c \leq d_2$. Now substituting $x_c = x^c$ yields the desired univariate expression for x^e. ◀

A key goal now is to establish the below Lemma 49. We remark that several times it will use Lemma 48; in each application we will have "i" and "j" at most 3, less than $\mathrm{char}(\mathbb{F}) = p > 3$ as required.

▶ **Lemma 49.** *Fix roots $\beta \neq -\alpha$ and any $d_1, d_2 \in \mathbb{N}$. Then*

$$\langle X_{\alpha,d_1}, X_{\beta,d_2}\rangle = \langle X_{i\alpha+j\beta,id_1+jd_2} : i, j \in \mathbb{N}, i\alpha + j\beta \in \Phi\rangle.$$

Proof. The inclusion $\subseteq$ is immediate by taking $(i, j) \in \{(1, 0), (0, 1)\}$, so it suffices to prove the reverse inclusion $\supseteq$. The case $\beta = \alpha$ is trivial, so we may assume that α, β span some 2-dimensional subspace H. Let $R \coloneqq \{i\alpha + j\beta \in \Phi : i, j \in \mathbb{N}\}$, a subset of the 2-dimensional root system $\Phi' = \Phi \cap H$. If $R = \{\alpha, \beta\}$ only then the lemma is immediate. Otherwise, R must also contain $\alpha + \beta$ (using Claim 22) and hence Φ' is isomorphic to A_2, B_2, or G_2 as explained in Remark 15. This allows us to classify the possibilities for R; with the assistance of Figure 1, we see there are four cases, namely $R = \{\alpha, \beta\} \cup R'$ for R' equal to...

1. $\{\alpha+\beta\}$, 2. $\{\alpha+\beta, 2\alpha+\beta\}$, 3. $\{\alpha+\beta, 2\alpha+\beta, \alpha+2\beta\}$, 4. $\{\alpha+\beta, 2\alpha+\beta, 3\alpha+\beta, 3\alpha+2\beta\}$.

In each case, we need to show for every $\gamma = i\alpha + j\beta \in R'$ that $x_\gamma(w) \in \langle X_{\alpha,d_1}, X_{\beta,d_2}\rangle$ for all $w \in \mathbb{F}$ of degree at most $d = id_1 + jd_2$. By virtue of Lemma 48 (and using $\mathrm{char}(\mathbb{F}) > 3 \geq i, j$), it suffices to show this for w's that are linear combinations of field elements of the form $t^i u^j$, where t has degree d_1 and u has degree d_2. Further, since $x_\gamma(r+s) = x_\gamma(r)x_\gamma(s)$, it suffices to handle w of the form ct^iu^j for arbitrary $c \in \mathbb{F}_p$. Finally, it suffices to handle just one specific $c \neq 0$, because if $x_\gamma(ct^iu^j)$ is in $\langle X_{\alpha,d_1}, X_{\beta,d_2}\rangle$ then so too is its kth power $x_\gamma(ct^iu^j)^k = x_\gamma(kct^iu^j)$, and kc varies over all $\mathbb{F}_p$ as k varies in $\mathbb{N}$. We will always use a c which is the product of structure constants $C_{i',j'}^{\alpha',\beta'}$, and such are never 0 in $\mathbb{F}_p$ because $1 \leq |C_{i',j'}^{\alpha',\beta'}| \leq 3 < p$.

Summarizing, for fixed $t, u \in \mathbb{F}$ of degree at most d_1, d_2 (respectively), it suffices to show the following in Cases 1–4: For each $\gamma = i\alpha + j\beta \in R'$ we have $x_\gamma(ct^iu^j) \in \langle x_\alpha(t), x_\beta(u)\rangle$ for some product of structure constants c.

Case 1: $R' = \{\alpha + \beta\}$. This case arises when $\Phi' = A_2$ and $\angle(\alpha,\beta) = 120°$, or when $\Phi' = B_2$ and α, β are short roots with $\angle(\alpha,\beta) = 90°$, or when $\Phi' = G_2$ and α, β are short roots with $\angle(\alpha,\beta) = 60°$. We handle $\gamma = \alpha + \beta$ via the commutator formula $[x_\alpha(t), x_\beta(u)] = x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu)$.

Case 2: $R' = \{\alpha + \beta, 2\alpha + \beta\}$, which arises for $\Phi' = B_2$. We first treat the root $\gamma = 2\alpha + \beta$. By the commutator formula we have

$$[[x_\alpha(t), x_\beta(u)], x_\alpha(t)] = [x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu) x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u), x_\alpha(t)].$$

In this latter commutator we can delete $x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u)$ because it commutes with the other two elements. (This is since no root is a nontrivial $\mathbb{N}$-linear combination involving $2\alpha + \beta$.) Thus

$$[[x_\alpha(t), x_\beta(u)], x_\alpha(t)] = [x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu), x_\alpha(t)] = x_{2\alpha+\beta}(C_{1,1}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} t^2u). \tag{3}$$

Thus $\gamma = 2\alpha + \beta$ is handled. As for $\gamma = \alpha + \beta$, the commutator formula gives

$$\begin{aligned}[x_\alpha(t), x_\beta(u)] \cdot x_{2\alpha+\beta}(-C_{2,1}^{\alpha,\beta} t^2u) &= x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu) x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u) \cdot x_{2\alpha+\beta}(-C_{2,1}^{\alpha,\beta} t^2u) \\ &= x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu),\end{aligned}$$

and so $\gamma = \alpha + \beta$ is also handled (since we already know $x_{2\alpha+\beta}(-C_{2,1}^{\alpha,\beta} t^2u)$ is in $\langle x_\alpha(t), x_\beta(u)\rangle$ via Equation (3)).

Case 3: $R' = \{\alpha + \beta, 2\alpha + \beta, \alpha + 2\beta\}$. This case only arises for $\Phi' = G_2$. We start by treating $\gamma = 2\alpha + \beta$. We have

$$[[x_\alpha(t), x_\beta(u)], x_\alpha(t)] = [x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu) y, x_\alpha(t)] \quad \text{for } y = x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u) x_{\alpha+2\beta}(C_{3,1}^{\alpha,\beta} tu^2),$$

and similar to Case 2 we can delete y from this commutator as it commutes with the other two elements (by virtue of the height of $2\alpha + \beta$ and $\alpha + 2\beta$). Hence

$$[[x_\alpha(t), x_\beta(u)], x_\alpha(t)] = [x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu), x_\alpha(t)] = x_{2\alpha+\beta}(C_{1,1}^{\alpha,\beta} C_{1,1}^{\alpha+\beta,\alpha} t^2u)$$

and we've handled $\gamma = 2\alpha + \beta$. The case of $\gamma = \alpha = 2\beta$ is similar. Finally the treatment of $\gamma = \alpha + \beta$ is similar to Case 2; it follows from

$$[x_\alpha(t), x_\beta(u)] x_{\alpha+2\beta}(-C_{3,1}^{\alpha,\beta} tu^2) x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u) = x_{\alpha+\beta}(C_{1,1}^{\alpha,\beta} tu).$$

Case 4: $R' = \{\alpha+\beta, 2\alpha+\beta, 3\alpha+\beta, 3\alpha+2\beta\}$. This case only arises for $\Phi' = G_2$. To reduce clutter in this case, we will sometimes abbreviate $x_{i\alpha+j\beta}(ct^iu^j)$ to $x_{i\alpha+j\beta}$. We start with

$$[x_\alpha(t), x_\beta(u)] = x_{\alpha+\beta} \cdot x_{2\alpha+\beta} \cdot x_{3\alpha+\beta} \cdot x_{3\alpha+2\beta}, \tag{4}$$

which implies

$$[[x_\alpha(t), x_\beta(u)], x_\beta(u)] = [x_{\alpha+\beta} \cdot x_{2\alpha+\beta} \cdot x_{3\alpha+\beta}, x_\beta],$$

where we deleted the $x_{3\alpha+2\beta}$ element since it commutes with everything else. Now since x_β commutes with $x_{\alpha+\beta}$ and $x_{2\alpha+\beta}$, we get

$$[x_{\alpha+\beta} \cdot x_{2\alpha+\beta} \cdot x_{3\alpha+\beta}, x_\beta] = [x_{3\alpha+\beta}, x_\beta] = x_{3\alpha+2\beta} = x_{3\alpha+2\beta}(C_{1,1}^{3\alpha+\beta,\beta} C_{3,1}^{\alpha,\beta} t^3u^2),$$

where in the last step we explicitly wrote in the argument to $x_{3\alpha+2\beta}$ that arises. Thus we have handled $\gamma = 3\alpha+2\beta$. Taking care of $\gamma = 3\alpha+\beta$ is somewhat more tedious. Considerations similar to the above lead us to

$$[[x_\alpha(t), x_\beta(u)], x_\alpha(t)] = [x_{\alpha+\beta} \cdot x_{2\alpha+\beta}, x_\alpha],$$

which in turn equals

$$x_{2\alpha+\beta}(-C_{2,1}^{\alpha,\beta} t^2u) \cdot [x_{\alpha+\beta}, x_\alpha] \cdot x_{2\alpha+\beta}(C_{2,1}^{\alpha,\beta} t^2u) \cdot [x_{2\alpha+\beta}, x_\alpha], \tag{5}$$

where we explicitly wrote in the arguments to $x_{2\alpha+\beta}$ that arise. We now observe that when the commutator rule is twice applied in the above, the resulting elements are $x_{2\alpha+\beta} \cdot x_{3\alpha+2\beta} \cdot x_{3\alpha+\beta}$ (first commutator) and $x_{3\alpha+\beta}$ (second commutator), and these all commute with the $x_{2\alpha+\beta}(\pm C_{2,1}^{\alpha,\beta} t^2u)$ in Equation (5). Thus said $x_{2\alpha+\beta}(\pm C_{2,1}^{\alpha,\beta} t^2u)$ cancel out, and we end up deducing that $[[x_\alpha(t), x_\beta(u)], x_\alpha(t)]$ equals

$$x_{2\alpha+\beta}(C_{1,1}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} t^2u) x_{3\alpha+2\beta}(C_{2,1}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} t^3u^2) x_{3\alpha+\beta}((C_{1,2}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} + C_{1,1}^{2\alpha+\beta,\alpha} C_{2,1}^{\alpha,\beta}) t^3u). \tag{6}$$

Finally, we take one more commutator with $x_\alpha(t)$. The latter two elements in the above commute with $x_\alpha(t)$ and thus may be deleted; we are left with

$$\begin{aligned}[[[x_\alpha(t), x_\beta(u)], x_\alpha(t)], x_\alpha(t)] &= [x_{2\alpha+\beta}(C_{1,1}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} t^2u), x_{\alpha(t)}] \\ &= x_{3\alpha+\beta}(C_{1,1}^{2\alpha+\beta,\alpha} C_{1,1}^{\alpha+\beta,\alpha} C_{1,1}^{\alpha,\beta} t^3u).\end{aligned}$$

Thus we have handled $\gamma = 3\alpha+\beta$. Since $\gamma = 3\alpha+2\beta$ has also been treated, we get $\gamma = 2\alpha+\beta$ from Equation (6), and then $\gamma = \alpha+\beta$ from Equation (4). ◀

We may now complete our goal for this section:

Proof of Theorem 44. We first show that $X_\alpha = X_{\alpha,1} \subseteq X_S$ for all $\alpha \in \Phi$. Since we are assuming $S^+ = \Phi$, we can write $\alpha = \sum_{\beta \in S} n_\beta \beta$ with $n_\beta \in \mathbb{N}$. Then by Claim 22 we can write $\alpha = p_{i_1} + p_{i_2} + \cdots + p_{i_\ell}$ with $p_{i_j} \in S$ and where all prefix sums are roots. Clearly we may assume that $p_j \neq -(p_1 + \cdots + p_{j-1})$ does not occur for any j, as otherwise the first j terms could be excised from the expression for α. Then by Lemma 49 it follows that $X_{p_{i_1}+p_{i_2}} \subseteq \langle X_{p_{i_1}}, X_{p_{i_2}} \rangle$, $X_{p_{i_1}+p_{i_2}+p_{i_3}} \subseteq \langle X_{p_{i_1}}, X_{p_{i_2}}, X_{p_{i_3}} \rangle$, and so on, eventually yielding $X_\alpha \subseteq \langle X_\beta : \beta \in \Phi \rangle$.

Now suppose by induction on $i \geq 0$ that $X_{\alpha,2^i} \subseteq X_S$ for all $\alpha \in \Phi$. By Claim 17, for any root $\gamma \in \Phi$ we can write $\gamma = \alpha + \beta$ for some $\alpha, \beta \in \Phi$, and it follows from Lemma 49 that $X_{\gamma,2^i+2^i} \subseteq \langle X_{\alpha,2^i}, X_{\beta,2^i} \rangle$. Thus indeed $X_{\gamma,2^{i+1}} \subseteq X_S$, completing the induction. ◀

3.2 Structure of the links

In this section we describe the structure of the subgroups X_T where $T \subset \mathcal{S}$. This will be used to show that the links of all faces of $\mathfrak{K}$ are connected.

We will first need a "graded" version of Proposition 32.

▶ **Proposition 50.** *Fix any ordering $\prec$ of the roots Φ, and let $\Psi \subseteq \Phi$ be linearly independent. Then the elements of X_Ψ are in* 1-1 *correspondence with expressions of the form $\prod_{\gamma \in \Psi^+} x_\gamma(t_\gamma)$ with $x_\gamma(t_\gamma) \in X_{\gamma,\mathrm{ht}_\Psi(\gamma)}$ (and the product taken in order $\prec$).*

Proof. We first prove that every expression of the given form is indeed in X_Ψ. Precisely, we show by induction on h that X_Ψ contains all subgroups $X_{\gamma,h}$ with $h = \mathrm{ht}(\gamma)$. The base case of $h = 1$ is immediate. For general h, take any $\gamma \in \Psi^+$ with height h and write $\gamma = \alpha + \beta$ with $\alpha, \beta \in \Psi^+$ of height smaller than h. (This is possible by Claim 22.) Now it follows from Lemma 49 that $X_{\gamma,\mathrm{ht}(\gamma)} = X_{\gamma,\mathrm{ht}(\alpha)+\mathrm{ht}(\beta)} \subseteq \langle X_{\alpha,\mathrm{ht}(\alpha)}, X_{\beta,\mathrm{ht}(\beta)} \rangle$, and this is in X_Ψ by induction.

We next show that every element in X_Ψ has a unique expression of the given form. In fact, it suffices to show existence, since uniqueness follows from Proposition 32 (note that Ψ^+ satisfies its hypotheses). Let us say that an expression of the form

$$x_{\gamma_1}(u_1)x_{\gamma_2}(u_2)\cdots x_{\gamma_m}(u_m) \tag{7}$$

with $\gamma_i \in \Psi^+$ is *well-bounded* if each u_i has degree at most $\mathrm{ht}(\gamma_i)$. The desired existence result is that every $z \in X_\Psi$ has a well-bounded expression as above, where $\gamma_1, \dots, \gamma_m$ list the elements of Ψ^+ in the order $\prec$. (We remark that it doesn't matter whether we are allowing consecutive duplicate γ_i's in this list, since $x_\gamma(u)x_\gamma(u') = x_\gamma(u+u')$ and this preserves well-boundedness.)

To show this existence, it actually suffices to repeat the existence proof in Proposition 32. At a high level, this works because that proof ultimately only uses the commutator formula, and applications of the commutator formula preserve well-boundedness. That is, starting from an arbitrary $z \in X_\Psi$, by definition we may express z as in Equation (7) with each $\gamma_i \in \Psi$ and each u_i of degree at most 1. This is well-bounded. Then an application of the commutator formula switches some consecutive $x_\gamma(u)x_{\gamma'}(u')$ to $x_{\gamma'}(u')x_\gamma(u)[x_\gamma(u), x_{\gamma'}(u')]$, and this commutator is the product of elements of the form $x_{i\gamma+j\gamma'}(C_{i,j}^{\gamma,\gamma'} u^i (u')^j)$. But this product is indeed well-bounded, presuming the former expression was well-bounded.

For completeness, we sketch why the existence result in Proposition 32 only relies on the commutator formula. We prefer to first follow the existence result in [13, Thm. 5.3.3], which assumes that the order $\prec$ is consistent with heights (meaning $\mathrm{ht}_\Psi(\alpha) \le \mathrm{ht}_\Psi(\beta)$ implies $\alpha \prec \beta$). Under this assumption, we may repeatedly reorder consecutive products $x_\gamma(u)x_{\gamma'}(u')$ whenever $\gamma' \prec \gamma$, as described above. Notice that the new products of elements of the form $x_{i\gamma+j\gamma'}(C_{i,j}^{\gamma,\gamma'} u^i (u')^j)$ that arise are have $\mathrm{ht}(i\gamma + j\gamma') > \mathrm{ht}(\gamma), \mathrm{ht}(\gamma')$. Because of this, and the height-respecting property of $\prec$, this process must eventually terminate with a (well-bounded) expression like Equation (7) where the roots γ_i are in the order $\prec$ (and any missing root $\gamma \in \Phi^+$ can be inserted via $x_\gamma(0)$).

It remains to treat the case that the root order $\prec$ does *not* necessarily respect heights. For this we appeal to [54, Lem. 18], the associated component of the proof of Proposition 32. It says that it suffices to check – when γ *is* a height-respecting order, and $\Psi^+ = \{\gamma_1 \prec \gamma_2 \prec \cdots \prec \gamma_m\}$ – that each subgroup of the form

$$B_i \coloneqq X_{\gamma_i,\mathrm{ht}(\gamma_i)} \cdot X_{\gamma_{i+1},\mathrm{ht}(\gamma_{i+1})} \cdots X_{\gamma_r,\mathrm{ht}(\gamma_m)}$$

is normal in X_Ψ. To see this, take a generic well-bounded expression

$$y = x_{\gamma_i}(t_i)x_{\gamma_{i+1}}(t_{i+1})\cdots x_{\gamma_m}(t_m)$$

in B_i and consider conjugating it by an arbitrary well-bounded expression w as in Equation (7). We have $w^{-1}yw = y[y, w]$, and expanding the commutator yields a well-bounded expression consisting only of $x_\gamma(v)$'s where $\mathrm{ht}(\gamma) \geq \mathrm{ht}(\gamma_i)$. Now as in the previous argument, this may be further rearranged into a well-bounded expression in B_i, showing that B_i is closed under conjugation and hence normal. ◀

We have the following immediate consequence:

▶ **Corollary 51.** *Let Ψ be a set of linearly independent roots. Then $|X_\Psi| = \prod_{\alpha\in\Psi^+} p^{\mathrm{ht}_\Psi(\alpha)+1}$.*

Importantly, $|X_\Psi|$ can be bounded independently of m (where recall $|\mathbb{F}| = p^m$). This will imply that a vertex in $\mathfrak{K}$ belongs to just $p^{O(1)}$ faces where the $O(1)$ does not depend on m.

The proceeding normal form result also helps us show the following:

▶ **Proposition 52.** *Let Ψ and Ψ' be sets of linearly independent roots. Then $X_\Psi \cap X_{\Psi'} = X_{\Psi\cap\Psi'}$.*

Proof. By Claim 21 we may choose a set Π of simple roots with $\Psi \subseteq \Pi^+$. We apply Proposition 50 to any $g \in X_\Psi$ and $h \in X_{\Psi'}$, writing them as $g = \prod_{\alpha\in\Psi^+} x_\alpha(t_\alpha)$ and $h = \prod_{\alpha\in\Psi'^+} x_\alpha(u_\alpha) = U \cdot L$, where we have ordered h as a product U of root elements in Π^+ times a product L of root elements in Π^-. Now supposing $g = h$, we get $U^{-1}g = L$. But by Proposition 33, the only way this equality can hold is if $L = 1$. Hence we have $\prod_{\alpha\in\Psi^+} x_\alpha(t_\alpha) = \prod_{\alpha\in\Psi'^+} x_\alpha(u_\alpha)$, where on both sides α is ranging in Π^+; hence by uniqueness of these expressions (assuming the products are taken in the same order), equality holds just when $t_\alpha = u_\alpha$ for all α. So the elements of $X_\Psi \cap X_{\Psi'}$ are exactly the elements of the form $\prod_{\alpha\in\Psi^+\cap\Psi'^+} x_\alpha(f_\alpha)$ where $\deg(f_\alpha) \leq \min(\mathrm{ht}_\Psi(\alpha), \mathrm{ht}_{\Psi'}(\alpha))$. But note that $\Psi^+ \cap \Psi'^+ = (\Psi \cap \Psi')^+$ and $\mathrm{ht}_\Psi(\alpha) = \mathrm{ht}_{\Psi'}(\alpha) = \mathrm{ht}_{\Psi\cap\Psi'}(\alpha)$ for $\alpha \in (\Psi \cap \Psi')^+$ due to linear independence. So any such an element belongs to $X_{\Psi\cap\Psi'}$ (using Proposition 50 again), which proves the proposition. ◀

▶ **Observation 53.** *In fact, $Z \cdot X_\Psi \cap Z \cdot X_{\Psi'} = Z \cdot X_{\Psi\cap\Psi'}$, where Z denotes the center of $\overline{\mathrm{G}}(\Phi, \mathbb{F})$. The proof proceeds in the same fashion: Under the matrix identification of Proposition 33, Z consists of diagonal matrices. Thus if $D_1U^{-1}g = D_2L$ with D_1 and D_2 diagonal, L lower-unitriangular and $U^{-1}g$ upper-unitriangular, we must have $D_1 = D_2$ and $L = U^{-1}g = 1$. This implies $\prod_{\alpha\in\Psi^+} x_\alpha(t_\alpha) = \prod_{\alpha\in\Psi'^+} x_\alpha(u_\alpha)$, and the rest of the proof follows as before.*

Combining Proposition 52 with Claim 10 lets us understand the structure of the links in $\mathfrak{K}$:

▶ **Theorem 54.** *Let $\sigma \in \mathfrak{K}$ be a face of type $T \subsetneq \mathcal{S}$. Then the link of F is isomorphic to the coset complex $\mathcal{CC}(X_{\mathcal{S}\setminus T}; (X_{\mathcal{S}\setminus T\setminus\{\alpha\}} : \alpha \in \mathcal{S} \setminus T))$.*

Proof. For $T = \emptyset$, this is the combination of Theorem 44 and Proposition 45. Otherwise, by virtue of Claim 10 it suffices to show that for any $U \subseteq \mathcal{S}$,

$$H_U = \bigcap_{\alpha\in U} H_\alpha = \bigcap_{\alpha\in U} X_{\mathcal{S}\setminus\{\alpha\}} = X_{\mathcal{S}\setminus U}.$$

But this follows from Proposition 52 after recalling (Remark 39) that $\mathcal{S} \setminus \{\alpha\}$ is linearly independent for any α. ◀

Finally, whenever $|T| \leq d-1$ the sets $\mathcal{S} \setminus T \setminus \{\alpha\}$ are nonempty, and so we may therefore conclude using Claim 9:

▶ **Corollary 55.** *For all $\sigma \in \mathfrak{K}(i)$ with $i \leq d-2$, K_σ is connected.*

▶ Remark 56. The fact that $\mathfrak{K}$ and all of its links of dimension at most $d-2$ are connected is equivalent to saying that $\mathfrak{K}$ is *strongly gallery connected* [40, Rem. 2.1].

3.3 Expansion of links

▶ **Definition 57.** *For $\alpha, \beta \in \Phi$ with $\alpha \neq -\beta$, let $\mathcal{CC}(\alpha;\beta) = \mathcal{CC}(X_{\{\alpha,\beta\}}; (X_\alpha, X_\beta))$.*

It follows from Theorem 54 that the link of every $(d-2)$-dimensional face in our complex $\mathfrak{K}$ is isomorphic to $\mathcal{CC}(\alpha;\beta)$ for distinct $\alpha, \beta \in \mathcal{S}$. The main goal of this section is to show that the bipartite skeleton graphs of these $\mathcal{CC}(\alpha;\beta)$ are good expanders. (For this we will not even need to recall our specific choice of $\mathcal{S}$.) Combined with Theorem 3 and the connectivity result Corollary 55, it follows that all links of $\mathfrak{K}$ are good expanders.

We begin with a simple observation:

▶ **Proposition 58.** *For $\alpha \neq -\beta$, $\mathcal{CC}(\alpha;\beta)$ is a p^2-regular bipartite (multi)graph.*

Proof. From Claim 10, the link of a vertex in $X_{\alpha,\beta}/X_\beta$ is isomorphic to $\mathcal{CC}(X_\beta; X_\alpha \cap X_\beta) = \mathcal{CC}(X_\beta; 1)$, where we used Proposition 52. But this is equivalent to saying the neighborhood of a vertex in the skeleton is a set of size $|X_\beta| = p^2$ (recalling Corollary 51). The same consideration holds for vertices in $X_{\alpha,\beta}/X_\alpha$. ◀

The key idea we will use in understanding the expansion of the links $\mathcal{CC}(\alpha;\beta)$ will be to look at the graph-theoretic *square*, $\mathcal{CC}(\alpha;\beta)^2$, of (the skeleton of) $\mathcal{CC}(\alpha;\beta)$. Since $\mathcal{CC}(\alpha;\beta)$ is connected and bipartite, we know that its random walk matrix has isolated "trivial" eigenvalues of ± 1, and all other eigenvalues are between $\pm\lambda_2(\mathcal{CC}(\alpha;\beta))$. Thus if we exclude from $\mathcal{CC}(\alpha;\beta)^2$ the "trivial" eigenvalue 1, its maximum eigenvalue will be $\lambda_2(\mathcal{CC}(\alpha;\beta))^2$, the square of what we wish to bound. In fact, since $\mathcal{CC}(\alpha;\beta)$ is bipartite, $\mathcal{CC}(\alpha;\beta)^2$ will have two disconnected components corresponding to the two parts of $\mathcal{CC}(\alpha;\beta)$. It is a simple and well-known linear algebra fact that these two components have the same eigenvalues (possibly up to some eigenvalues of 0). Hence it suffices for us to bound the eigenvalues of $\mathcal{CC}(\alpha;\beta)^2$ on only *one* of the two sides, $X_{\alpha,\beta}/X_\alpha$ or $X_{\alpha,\beta}/X_\beta$.

As we will now show, whenever $\Phi \neq G_2$, at least one of these two sides is an abelian Cayley graph. (Interestingly, we do not know that both sides are.) Thus we can understand the eigenvalues by elementary methods. We discuss a potential approach to handling the G_2 case in Section 4.1.

▶ **Theorem 59.** *Let $\alpha, \beta \in \Phi \neq G_2$, with $\alpha \neq -\beta$. Then the nontrivial eigenvalues of $\mathcal{CC}(\alpha;\beta)^2$ are at most $2/p$; hence $\lambda_2(K_\sigma) \leq \sqrt{2/p}$ for every $\sigma \in \mathfrak{K}(d-2)$.*

Proof. When it is relevant, we will follow the convention of calling the shorter of the two roots α and the longer β. Then, with foresight toward Case 3 below, we choose to study the $X_{\alpha,\beta}/X_\alpha$ side of $\mathcal{CC}(\alpha;\beta)^2$.

By virtue of Proposition 50, we can describe coset representatives for $X_{\alpha,\beta}/X_\alpha$ fairly simply; fixing an ordering for the roots in which α is last, we can take as coset representatives precisely those elements of the form

$$g = \prod \{x_{i\alpha+j\beta}(t_{ij}) : (i,j) \in \mathbb{N} \times \mathbb{N} \setminus \{(1,0)\},\ i\alpha + j\beta \in \Phi,\ \deg(t_{ij}) \leq i + j\}. \tag{8}$$

Moreover, the p^2 neighbors (counted with multiplicity) of vertex gX_α in the squared (multi)graph $\mathcal{CC}(\alpha;\beta)^2$ are the following cosets:

$$(g \cdot x_\alpha(f_0) \cdot x_\beta(f_1))X_\alpha, \quad \text{for } f_0, f_1 \in \mathbb{F} \text{ of degree at most 1.}$$

Via the commutator formula one sees that the associated coset representatives are

$$g \cdot x_\alpha(f_0) \cdot x_\beta(f_1) \cdot x_\alpha(-f_0) = g \cdot x_\beta(f_1) \cdot \prod_{\substack{i,j \in \mathbb{Z}_+ \\ i\alpha + j\beta \in \Phi}} x_{i\alpha+j\beta}(C_{ij}^{\beta,\alpha}(-f_0)^i f_1^j). \tag{9}$$

By Remark 15, either $\alpha + \beta \notin \Phi$, or the root subsystem of Φ spanned by α and β is one of A_2, B_2, or G_2. We will skip the case when α and β span G_2, as it only arises when $\Phi = G_2$. Now as in the proof of Lemma 49, we will do case analysis on the possible sets $R = \{i, j : i\alpha + j\beta \in \Phi\}$.

Case 1: $R = \{\alpha, \beta\}$. If $\alpha + \beta \notin \Phi$, then X_α and X_β commute by Theorem 28, and it is easy to check that $\mathcal{CC}(\alpha;\beta)$ is in fact the complete p^2-regular bipartite graph; hence the nontrivial eigenvalues of $\mathcal{CC}(\alpha;\beta)^2$ are all 0.

Case 2: $\{\alpha, \beta, \alpha+\beta\}$. It was shown in [40], and alternatively in [31, Corollary 5.6], that $\lambda_2(\mathcal{CC}(\alpha;\beta)) = \sqrt{1/p}$; equivalently, the nontrivial eigenvalues of $\mathcal{CC}(\alpha;\beta)$ are at most $1/p$. Here we give a different proof of this fact, the strategy of which will be generalized in Case 3.

From Equations (8) and (9) we have that a typical coset representative $g = x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11})$ is connected in $\mathcal{CC}(\alpha;\beta)^2$ to the following coset representatives, for $f_0, f_1 \in \mathbb{F}$ of degree at most 1:

$$x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11}) \cdot x_\beta(f_1) \cdot x_{\alpha+\beta}(-C_{11}^{\beta,\alpha} f_0 f_1) = x_\beta(t_{01} + f_1) \cdot x_{\alpha+\beta}(t_{11} - C_{11}^{\beta,\alpha} f_0 f_1).$$

Reparameterizing with $f_2 = -C_{11}^{\beta,\alpha} f_0$ (and recalling $C_{11}^{\beta,\alpha} \neq 0$), it is evident that $\mathcal{CC}(\alpha;\beta)^2$ is an abelian Cayley group, wherein each vertex is a pair (ℓ, q) with ℓ linear and q quadratic, hence $(\ell, q) \cong \mathbb{F}_p^5$, and with edges involve adding a pair $(f_1, f_1 f_2)$ for f_1, f_2 linear. With x denoting the field indeterminate, we can write $f_1 = a + bx$ and $f_2 = c + dx$; then the $X_{\alpha,\beta}/X_\alpha$ side of our graph $\mathcal{CC}(\alpha;\beta)^2$ may be identified as an abelian Cayley graph on $\mathbb{F}_p^5$ with symmetric generating set

$$\{(a, b, ac, ad + bc, bd) : a, b, c, d \in \mathbb{F}_p^4\}.$$

Then it is well known that the eigenvalues of this graph are given by the exponential sums

$$\begin{aligned} &\mathop{\mathbf{E}}_{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d} \sim \mathbb{F}_p} \big[\mathrm{Exp}_p(r_1\boldsymbol{a} + r_2\boldsymbol{b} + r_3\boldsymbol{ac} + r_4(\boldsymbol{ad} + \boldsymbol{bc}) + r_5\boldsymbol{bd})\big] \\ &= \mathop{\mathbf{E}}_{\boldsymbol{c},\boldsymbol{d}} \left[\mathop{\mathbf{E}}_{\boldsymbol{a}} \big[\mathrm{Exp}_p(\boldsymbol{a} \cdot h(\boldsymbol{c},\boldsymbol{d}))\big] \mathop{\mathbf{E}}_{\boldsymbol{b}} \big[\mathrm{Exp}_p(\boldsymbol{b} \cdot h'(\boldsymbol{c},\boldsymbol{d}))\big] \right] \end{aligned} \tag{10}$$

for $r_1, \ldots, r_5 \in \mathbb{F}_p$, where $\mathrm{Exp}_p(z) = e^{2\pi i z/p}$, and

$$h(c,d) = r_1 + r_3 c + r_4 d, \qquad h'(c,d) = r_2 + r_4 c + r_5 d.$$

Notice whenever the outcome $\boldsymbol{c}, \boldsymbol{d}$ has $h(\boldsymbol{c},\boldsymbol{d}) \neq 0$, the quantity $\mathbf{E}_{\boldsymbol{a}} \big[\mathrm{Exp}_p(\boldsymbol{a} \cdot h(\boldsymbol{c},\boldsymbol{d}))\big]$ inside Equation (10) becomes 0. On the other hand, if $h(\boldsymbol{c},\boldsymbol{d}) = 0$ then this quantity is 1. Similar considerations hold for h', and we conclude that the eigenvalue in Equation (10) is precisely

$$\mathop{\mathbf{Pr}}_{\boldsymbol{c},\boldsymbol{d}}[h(\boldsymbol{c},\boldsymbol{d}) = h'(\boldsymbol{c},\boldsymbol{d}) = 0].$$

Of course if $r_1 = \cdots = r_5 = 0$ then h, h' are formally 0 and the above is the trivial eigenvalue of 1. But otherwise, at least one of h, h' is nonzero – say, h – and, being an affine linear polynomial over $\mathbb{F}_p$, it has $\mathbf{Pr}_{\boldsymbol{c},\boldsymbol{d}}[h(\boldsymbol{c},\boldsymbol{d})] \leq 1/d$. This shows that indeed the nontrivial eigenvalues of $\mathcal{CC}(\alpha;\beta)^2$ are at most $1/p$.

Case 3: $R = \{\alpha, \beta, \alpha+\beta, 2\alpha+\beta\}$. As mentioned earlier, here we have named the shorter root α and the longer root β. From Equations (8) and (9) we have that a typical coset representative $g = x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11}) \cdot x_{2\alpha+\beta}(t_{21})$ is connected in $\mathcal{CC}(\alpha;\beta)^2$ to the following coset representatives, for $f_0, f_1 \in \mathbb{F}$ of degree at most 1:

$$\begin{aligned} x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11}) \cdot x_{2\alpha+\beta}(t_{21}) \cdot x_\beta(f_1) \cdot x_{\alpha+\beta}(-C_{11}^{\beta,\alpha} f_0 f_1) \cdot x_{2\alpha+\beta}(C_{12}^{\beta,\alpha} f_0^2 f_1) \\ = x_\beta(t_{01}+f_1) \cdot x_{\alpha+\beta}(t_{11} - C_{11}^{\beta,\alpha} f_0 f_1)) \cdot x_{2\alpha+\beta}(t_{21} + C_{12}^{\beta,\alpha} f_0^2 f_1). \end{aligned}$$

(We remark that had we looked at the $X_{\alpha,\beta}/X_\beta$ side of $\mathcal{CC}(\alpha;\beta)^2$, we would not have gotten all of the commutativity in the above calculation.) Reparameterizing again with $f_2 = -C_{11}^{\beta,\alpha} f_0$, this is

$$x_\beta(t_{01}+f_1) \cdot x_{\alpha+\beta}(t_{11} + f_1 f_2) \cdot x_{2\alpha+\beta}(t_{21} + C f_1 f_2^2)$$

for some constant $C \neq 0$ in $\mathbb{F}_p$.

Similar to Case 2, we see that this is an abelian Cayley graph on $\mathbb{F}_p^9$ with symmetric generating set

$$\{(a, b, ac, ad+bc, bd, Cac^2, C(bc^2+2acd), C(2bcd+ad^2), Cbd^2) : a, b, c, d \in \mathbb{F}_p\}.$$

As before, the eigenvalues of this graph are given by

$$\begin{aligned} \mathop{\mathbf{E}}_{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d} \in \mathbb{F}_p} \big[\mathrm{Exp}_p(r_1\boldsymbol{a} + r_2\boldsymbol{b} + r_3\boldsymbol{ac} + r_4(\boldsymbol{ad}+\boldsymbol{bc}) + r_5\boldsymbol{bd} + r_6 C\boldsymbol{ac}^2 + \\ r_7 C(\boldsymbol{bc}^2 + 2\boldsymbol{acd}) + r_8 C(2\boldsymbol{bcd} + \boldsymbol{ad}^2) + r_9 C\boldsymbol{bd}^2) \big] \end{aligned}$$

$$= \mathop{\mathbf{E}}_{\boldsymbol{c},\boldsymbol{d}} \Big[\mathop{\mathbf{E}}_{\boldsymbol{a}} \big[\mathrm{Exp}_p(\boldsymbol{a} \cdot h(\boldsymbol{c},\boldsymbol{d})) \big] \mathop{\mathbf{E}}_{\boldsymbol{b}} \big[\mathrm{Exp}_p(\boldsymbol{b} \cdot h'(\boldsymbol{c},\boldsymbol{d})) \big] \Big], \tag{11}$$

for all $r_1, \ldots, r_9 \in \mathbb{F}_p$, where

$$\begin{aligned} h(c,d) &= r_1 + r_3 c + r_4 d + C r_6 c^2 + 2C r_7 cd + C r_8 d^2, \\ h'(c,d) &= r_2 + r_4 c + r_5 d + C r_7 c^2 + 2C r_8 cd + C r_9 d^2. \end{aligned}$$

The argument is now the same as in Case 2, except we reason that if h is nonzero, then $\mathbf{Pr}_{\boldsymbol{c},\boldsymbol{d}}[h(\boldsymbol{c},\boldsymbol{d})] \leq 2/p$ by Schwartz–Zippel, since now h is quadratic. ◀

▶ Remark 60. When $\Phi = G_2$ two other graphs can arise as $\mathcal{CC}(\alpha;\beta)$. The squares of these graphs are not Cayley graphs of abelian groups, and so the previous approach fails. For completeness we now give explicit description of the squared graphs $\mathcal{CC}(\alpha;\beta)^2$ restricted to the vertices on the side $X_{\alpha,\beta}/X_\alpha$.

From Figure 1 we see that if $\alpha, \beta \in G_2$ and $\alpha+\beta \in G_2$ then $\angle(\alpha,\beta) \in \{60°, 120°, 150°\}$. If $\angle(\alpha,\beta) = 60°$ or if $\angle(\alpha,\beta) = 120°$ and α and β are long roots, the analysis is the same as in Case 2 of the previous proof. There are two remaining cases: (I) α and β are simple roots and $\angle(\alpha,\beta) = 150°$ as in Figure 1; (II) $\angle(\alpha,\beta) = 120°$ and α and β are short roots.

Case I: $\angle(\alpha,\beta) = 150°$

From Equation (8), a typical coset representative in $X_{\alpha,\beta}/X_\alpha$ is

$$g = x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11}) \cdot x_{2\alpha+\beta}(t_{21}) \cdot x_{3\alpha+\beta}(t_{31}) \cdot x_{3\alpha+2\beta}(t_{32})$$

with $\deg(t_{ij}) \le i+j$. By Equation (9), the neighbors of this coset representative in $\mathcal{CC}(\alpha;\beta)^2$ are parameterized by

$$g \cdot x_\beta(f_1 - t_{01}) \cdot [x_\beta(f_1 - t_{01}), x_\alpha(-f_0)]$$

for all f_0, f_1 of degree at most 1. Using Proposition 31, one can show that this is the multigraph with vertices $(t_{01}, t_{11}, t_{21}, t_{31}, t_{32})$ whose neighbors are parameterized by

$$(f_1 + t_{01}, -f_0 f_1 + t_{11}, f_0^2 f_1 + t_{21}, -f_0^3 f_1 + t_{31}, -f_1(t_{31} + 3t_{21} f_0 + f_0^3 f_1) + t_{32}).$$

Case II: $\angle(\alpha,\beta) = 120°$

A typical coset representative in $X_{\alpha,\beta}/X_\alpha$ is

$$g = x_\beta(t_{01}) \cdot x_{\alpha+\beta}(t_{11}) \cdot x_{2\alpha+\beta}(t_{21}) \cdot x_{\alpha+2\beta}(t_{12}),$$

where $\deg(t_{ij}) \le i+j$. The neighbors of this coset representative are parameterized by

$$g \cdot x_\beta(f_1 - t_{01}) \cdot [x_\beta(f_1 - t_{01}), x_\alpha(-f_0)]$$

for all f_0, f_1 of degree at most 1. Using Proposition 31, one can show that this is the multigraph with vertices $(t_{01}, t_{11}, t_{21}, t_{12})$ whose neighbors are parameterized by

$$(f_1 + t_{01}, -2f_0 f_1 + t_{11}, 3f_0^2 f_1 + t_{21}, 3f_1(t_{11} + f_0 f_1) + t_{12}).$$

4 Concluding

Finally we can prove Theorem 42.

Proof of Theorem 42.

1. By Theorem 35, we have $|\overline{\mathrm{G}}(\Phi,\mathbb{F})| = p^{\Theta(m)}$. By Corollary 51, the subgroups H_α have size at most $p^{O(1)}$. (Here the $\Theta(\cdot)$ and $O(\cdot)$ depend only on Φ.) Hence there are $p^{\Theta(m)}$ total cosets, and the claim that $|\mathfrak{K}(0)| = p^{\Theta(m)}$ follows. Note that as m increases by 1, the size of the complex grows by a constant factor $p^{O(1)}$; thus we have the linear growth in size needed for a strongly explicit family, and the *exact* number of vertices n can be computed efficiently in $\mathrm{poly}(m) = \mathrm{polylog}(n)$ time (by Theorem 35). The resulting family is strongly explicit thanks to Theorem 36: one can and construct all the group elements in $\overline{\mathrm{G}}(\Phi,\mathbb{F})$ efficiently, one can identify the vertices (cosets) explicitly and naively by listing all their elements (recall each H_α has constant size), and one can compute the complex's adjacency structure (e.g., list all maximal faces to which a given vertex belongs) thanks to the efficient ($\mathrm{poly}(m) = \mathrm{polylog}(n)$ time) group arithmetic from Theorem 36.
2. Again, by Corollary 51 the subgroups H_α have size at most $p^{O(1)}$. The number of maximal faces containing a vertex is therefore at most $p^{O(1)\cdot(d+1)} = p^{O(1)}$.
3. This is Corollary 55.
4. This is Theorem 59.
5. From Proposition 52, $\bigcap_{\alpha\in\mathcal{S}} H_\alpha = \{1\}$. The claim then follows from Claim 7. In addition, by Observation 53 we have that $\bigcap_{\alpha\in\mathcal{S}} ZH_\alpha = Z$, and so in the quotient group $\mathrm{G}(\Phi,\mathbb{F})$ the subgroups H_iZ/Z intersect trivially. Hence we also get a simply-transitive action for the adjoint Chevalley groups. ◀

4.1 Further questions

As mentioned in Remark 60, we do not know if the 2-dimensional complexes obtained from our construction for $\overline{\mathrm{G}}(G_2, \mathbb{F})$ yield HDX families, either in the case $\mathcal{S} = \{\alpha, \beta, -\alpha - \beta\}$ (as we selected in Definition 38) or in the alternative case $\mathcal{S} = \{\alpha, \alpha + \beta, -2\alpha - \beta\}$ mentioned in Remark 47. We note that this latter case with $\mathcal{S} = \{\alpha, \alpha + \beta, -2\alpha - \beta\}$ is particularly appealing, as all vertex links are isomorphic (i.e., the 1-skeleton is a *graph of constant link*). Additionally, to the best of our knowledge none of the links in this case arise in previous HDX constructions. This is in contrast to our other constructions, which always contain some links isomorphic to those studied in [40].

One approach to prove the expansion of these links is to count the number of closed walks of some fixed length k in one side of their square, which (by the trace method) equals the sum of the kth powers of the eigenvalues of their adjacency matrices. By Remark 60, the number of length-k paths starting and ending at a fixed vertex on the side $X_{\alpha,\beta}/X_\alpha$ equals the number of solutions to the following systems of equations, corresponding to the first and second cases in Remark 60, respectively:

$$0 = \sum_{i=1}^{k} g_i = \sum_{i=1}^{k} f_i g_i = \sum_{i=1}^{k} f_i^2 g_i = \sum_{i=1}^{k} f_i^3 g_i = \sum_{i=1}^{k} -g_i(f_i^3 g_i + \sum_{j=1}^{i-1}(f_j^3 g_j + 3 f_j^2 g_j f_i)),$$

$$0 = \sum_{i=1}^{k} g_i = \sum_{i=1}^{k} f_i g_i = \sum_{i=1}^{k} f_i^2 g_i = \sum_{i=1}^{k} g_i(f_i g_i - 2\sum_{j=1}^{i-1} f_j g_j).$$

Here f_i and g_i are linear polynomials in $\mathbb{F}_p[x]$. The graphs corresponding to the first and second systems have p^n vertices, where $n = 20$ and $n = 13$, respectively. Therefore if for some particular k one could bound the number of solutions to either of these by, say, $p^{4k-n} + p^{3.99k}$, expansion of the corresponding complexes would follow. To show this it would suffice to show that the varieties defined over $\mathbb{C}$ by these systems are irreducible and of dimension at most $4k - 20$ in the first case, or $4k - 13$ in the second case, for some k. This seems potentially tractable for a computer algebra system.

Finally, we have shown that the untwisted groups of Lie type act (simply) transitively on the maximal faces of certain HDXs. Are there more "combinatorial" families of groups – perhaps the symmetric group or the generalized symmetric groups – which admit transitive actions on HDXs?

References

1 Herbert Abels and Stephan Holz. Higher generation by subgroups. *Journal of Algebra*, 160(2):310–341, 1993. doi:10.1006/jabr.1993.1190.

2 Vedat Alev, Fernando Jeronimo, Dylan Quintana, Shashank Srivastava, and Madhur Tulsiani. List decoding of direct sum codes. In *Proceedings of the 14th annual Symposium on Discrete Algorithms (SODA)*, pages 1412–1425, 2020.

3 Vedat Levi Alev, Fernando Granha Jeronimo, and Madhur Tulsiani. Approximating constraint satisfaction problems on high-dimensional expanders. In *Proceedings of the 60th annual Symposium on Foundations of Computer Science (FOCS)*, pages 180–201, 2019. doi:10.1109/FOCS.2019.00021.

4 Vedat Levi Alev and Lap Chi Lau. Improved analysis of higher order random walks and applications. In *Proceedings of the 52nd annual Symposium on Theory of Computing (STOC)*, pages 1198–1211, 2020.

5 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st annual Symposium on Theory of Computing (STOC)*, pages 1–12, 2019.

6 Michael Aschbacher. The status of the classification of the finite simple groups. *Notices of the American Mathematical Society*, 51(7):736–740, 2004.

7 László Babai, William Kantor, and Alexander Lubotsky. Small-diameter Cayley graphs for finite simple groups. *European Journal of Combinatorics*, 10(6):507–522, 1989. doi:10.1016/S0195-6698(89)80067-8.

8 Cristina Ballantine. Ramanujan type buildings. *Canadian Journal of Mathematics*, 52(6):1121–1148, 2000. doi:10.4153/CJM-2000-047-4.

9 Oren Becker. Symmetric unique neighbor expanders and good LDPC codes. *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, 211:211–216, 2016. doi:10.1016/j.dam.2016.04.022.

10 Anders Björner. Some combinatorial and algebraic properties of Coxeter complexes and Tits buildings. *Advances in Mathematics*, 52(3):173–212, 1984. doi:10.1016/0001-8708(84)90021-5.

11 Emmanuel Breuillard, Ben Green, and Terence Tao. Suzuki groups as expanders. *Groups, Geometry, and Dynamics*, 5(2):281–299, 2011. doi:10.4171/GGD/128.

12 Élie Cartan. *Sur la structure des groupes de transformations finis et continus*. PhD thesis, Université de Paris, 1894.

13 Roger Carter. *Simple groups of Lie type*. John Wiley & Sons, 1989.

14 Donald I. Cartwright, Patrick Solé, and Andrzej Żuk. Ramanujan geometries of type $\tilde{A}_n$. *Discrete Mathematics*, 269(1-3):35–43, 2003. doi:10.1016/S0012-365X(02)00748-3.

15 Arjeh Cohen, Scott Murray, and Don Taylor. Computing in groups of Lie type. *Mathematics of Computation*, 73(247):1477–1498, 2004. doi:10.1090/S0025-5718-03-01582-5.

16 Yotam Dikstein and Irit Dinur. Agreement testing theorems on layered set systems. In *Proceedings of the 60th annual Symposium on Foundations of Computer Science (FOCS)*, pages 1495–1524, 2019. doi:10.1109/FOCS.2019.00088.

17 Yotam Dikstein, Irit Dinur, Yuval Filmus, and Prahladh Harsha. Boolean function analysis on high-dimensional expanders. In *Proceedings of the 22nd annual International Conference on Randomization and Computation (RANDOM)*, volume 116, pages Art. No. 38, 20, 2018.

18 Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality, 2021. Announced at https://www.youtube.com/watch?v=pjc6GCRFnpg.

19 Irit Dinur, Yuval Filmus, Prahladh Harsha, and Madhur Tulsiani. Explicit SoS lower bounds from high-dimensional expanders. In *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185, pages 38:1–38:16, 2021.

20 Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta-Shma. List-decoding with double samplers. *SIAM Journal on Computing*, 50(2):301–349, 2021. doi:10.1137/19M1276650.

21 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *Proceedings of the 58th annual Symposium on Foundations of Computer Science (FOCS)*, pages 974–985, 2017.

22 Mikhail Ershov and Andrei Jaikin-Zapirain. Property (T) for noncommutative universal lattices. *Inventiones Mathematicae*, 179(2):303–347, 2010.

23 Mikhail Ershov, Andrei Jaikin-Zapirain, and Martin Kassabov. Property (T) for groups graded by root systems. *Memoirs of the American Mathematical Society*, 249(1186):v+135, 2017. doi:10.1090/memo/1186.

24 Shai Evra, Tali Kaufman, and Gilles Zémor. Decodable quantum LDPC codes beyond the square root distance barrier using high dimensional expanders. In *Proceedings of the 61st annual Symposium on Foundations of Computer Science (FOCS)*, pages 218–227, 2020.

25 Jacob Fox, Mikhail Gromov, Vincent Lafforgue, Assaf Naor, and János Pach. Overlap properties of geometric expanders. *Journal für die Reine und Angewandte Mathematik. (Crelle's Journal)*, 671:49–83, 2012. doi:10.1515/crelle.2011.157.

26 The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.11.1*, 2021. URL: https://www.gap-system.org.

27 Howard Garland. p-adic curvature and the cohomology of discrete subgroups of p-adic groups. *Annals of Mathematics. Second Series*, 97:375–423, 1973. doi:10.2307/1970829.

28 Peter Garst. *Cohen–Macaulay complexes and group actions*. PhD thesis, Unviersity of Wisconsin–Madison, 1979.

29 Brian Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*. Springer, 2003.

30 Sergei Haller and Max Horn. Unipot – a system for computing with elements of unipotent subgroups of Chevalley groups, 2018. URL: https://www.gap-system.org/Packages/unipot.html.

31 Prahladh Harsha and Ramprasad Saptharishi. A note on the elementary HDX construction of Kaufman–Oppenheim. Technical Report 1912.11225, arXiv, 2019.

32 David Hill. Prove that the sum of all simple roots is a root. Mathematics Stack Exchange, 2016. (version: 2016-04-26). URL: https://math.stackexchange.com/q/1760142.

33 Robert Howlett, Leanne Rylands, and Donald Taylor. Matrix generators for exceptional groups of Lie type. *Journal of Symbolic Computation*, 31(4):429–445, 2001. doi:10.1006/jsco.2000.0431.

34 James Humphreys. *Introduction to Lie algebras and representation theory*. Springer–Verlag, 1972.

35 James Humphreys. *Linear algebraic groups*. Springer–Verlag, 1995.

36 Martin Kassabov. Symmetric groups and expander graphs. *Inventiones Mathematicae*, 170(2):327–354, 2007. doi:10.1007/s00222-007-0065-y.

37 Martin Kassabov, Alexander Lubotzky, and Nikolay Nikolov. Finite simple groups as expanders. *Proceedings of the National Academy of Sciences of the United States of America*, 103(16):6116–6119, 2006. doi:10.1073/pnas.0510337103.

38 Tali Kaufman and Alexander Lubotzky. Edge transitive Ramanujan graphs and highly symmetric LDPC good codes. In *Proceedings of the 44th annual Symposium on Theory of Computing (STOC)*, pages 359–366, 2012.

39 Tali Kaufman and David Mass. Local-to-global agreement expansion via the variance method. In *Proceedings of the 11th annual Innovations in Theoretical Computer Science Conference (ITCS)*, pages 74:1–74:14, 2020.

40 Tali Kaufman and Izhar Oppenheim. Construction of new local spectral high dimensional expanders. In *Proceedings of the 50th Annual Symposium on Theory of Computing (STOC)*, pages 773–786, 2018.

41 Tali Kaufman and Izhar Oppenheim. High order random walks: beyond spectral gap. *Combinatorica*, 40(2):245–281, 2020. doi:10.1007/s00493-019-3847-0.

42 Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *Proceedings of the 40th annual Symposium on Theory of Computation (STOC)*, pages 403–412. ACM, New York, 2008. doi:10.1145/1374376.1374434.

43 Tali Kaufman and Ran Tessler. New cosystolic expanders from tensors imply explicit quantum LDPC codes with $\omega(\sqrt{n}\log^k n)$ distance. In *Proceedings of the 53rd annual Symposium on Theory of Computing (STOC)*, pages 1317–1329, 2021.

44 Tali Kaufman and Avi Wigderson. Symmetric LDPC codes and local testing. *Combinatorica*, 36(1):91–120, 2016. doi:10.1007/s00493-014-2715-1.

45 Laurent Lafforgue. Chtoucas de Drinfeld et correspondance de Langlands. *Inventiones Mathematicae*, 147(1):1–241, 2002. doi:10.1007/s002220100174.

46 Folke Lannér. On complexes with transitive groups of automorphisms. *Communications du Séminaire Mathématique de l'Université de Lund*, 11:71, 1950.

47 Wen-Ching Winnie Li. Ramanujan hypergraphs. *Geometric and Functional Analysis*, 14(2):380–399, 2004. doi:10.1007/s00039-004-0461-z.

48 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Explicit constructions of Ramanujan complexes of type $\tilde{A}_d$. *European Journal of Combinatorics*, 26(6):965–993, 2005. doi:10.1016/j.ejc.2004.06.007.

49 Alexander Lubotzky, Beth Samuels, and Uzi Vishne. Ramanujan complexes of type $\tilde{A}_d$. *Israel Journal of Mathematics*, 149:267–299, 2005. Probability in mathematics. doi:10.1007/BF02772543.

50 Izhar Oppenheim. Local spectral expansion approach to high dimensional expanders part I: Descent of spectral gaps. *Discrete & Computational Geometry*, 59(2):293–330, 2018.

51 Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. Technical Report 2111.03654, arXiv, 2021.

52 Alireza Sarveniazi. *Ramanujan Hypergraph Based on Bruhat–Tits Building*. PhD thesis, University of Göttingen, 2004.

53 Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990. doi:10.2307/2008704.

54 Robert Steinberg. *Lectures on Chevalley groups*, volume 66 of *University Lecture Series*. American Mathematical Society, 2016. Notes prepared by John Faulkner and Robert Wilson, revised and corrected edition of the 1968 original. doi:10.1090/ulect/066.

55 Andrzej Żuk. Property (T) and Kazhdan constants for discrete groups. *Geometric and Functional Analysis*, 13(3):643–670, 2003. doi:10.1007/s00039-003-0425-8.

The Composition Complexity of Majority

Victor Lecomte
Stanford University, CA, USA

Prasanna Ramakrishnan
Stanford University, CA, USA

Li-Yang Tan
Stanford University, CA, USA

Abstract

We study the complexity of computing majority as a composition of *local* functions:

$$\textsc{Maj}_n = h(g_1, \ldots, g_m),$$

where each $g_j : \{0,1\}^n \to \{0,1\}$ is an arbitrary function that queries only $k \ll n$ variables and $h : \{0,1\}^m \to \{0,1\}$ is an arbitrary combining function. We prove an optimal lower bound of

$$m \geq \Omega\left(\frac{n}{k} \log k\right)$$

on the number of functions needed, which is a factor $\Omega(\log k)$ larger than the ideal $m = n/k$. We call this factor the *composition overhead*; previously, no superconstant lower bounds on it were known for majority.

Our lower bound recovers, as a corollary and via an entirely different proof, the best known lower bound for bounded-width branching programs for majority (Alon and Maass '86, Babai et al. '90). It is also the first step in a plan that we propose for breaking a longstanding barrier in lower bounds for small-depth boolean circuits.

Novel aspects of our proof include sharp bounds on the information lost as computation flows through the inner functions g_j, and the bootstrapping of lower bounds for a multi-output function (Hamming weight) into lower bounds for a single-output one (majority).

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases computational complexity, circuit lower bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.19

Related Version *Full Version*: https://arxiv.org/abs/2205.02374

Funding Victor, Pras, and Li-Yang are supported by NSF CAREER Award CCF-1942123. Pras is also supported by Moses Charikar's Simons Investigator Award.

Acknowledgements Li-Yang thanks Xi Chen, Rocco Servedio, and Erik Waingarten for numerous discussions about this problem.

1 Introduction

A basic theme in computer science is the representation of certain functions as the combination of simpler ones. Indeed, the field of distributed computing and the widespread principle of divide-and-conquer rely on this property of functions.

In this paper we focus on *locality* as our notion of simplicity: a k-local function over n variables is one that depends only on $k \ll n$ input coordinates. This leads us to the following complexity measure of boolean functions, first studied by Hrubeš and Rao [15], which quantifies how easily they can be represented as the combination of local functions:

▶ **Definition 1** (composition complexity). *The* k-composition complexity *of a function* f*, denoted* $\mathsf{CC}_k(f)$*, is the minimum* m *such that* f *can be expressed as* $h(g_1, \ldots, g_m)$*, where each of the inner functions* g_j *queries only* k *variables.*

Clearly, any function f that actually depends on all n variables must have $\mathsf{CC}_k(f) \geq n/k$, since every variable must be queried at least once. The parity function shows that this bound can be tight: $\mathsf{CC}_k(\textsc{Parity}_n) \leq O(n/k)$, since we can let each g_j compute the parity of a set of k variables, and let h compute the parity of these parities. However, some functions inherently require more than n/k inner functions: they incur a *composition overhead*. This motivates defining the *k-composition overhead* of f to be the ratio $\frac{\mathsf{CC}_k(f)}{n/k}$ between its k-composition complexity and the ideal n/k.

1.1 This work

In this paper we study the composition complexity of the majority function. Prior to our work, this was perhaps the most basic function whose composition complexity was not well understood. For an upper bound, it is not hard to see that $\mathsf{CC}_k(\textsc{Maj}_n) \leq O(\frac{n}{k}\log k)$. We can split the variables into n/k disjoint sets of size k and devote $O(\log k)$ of the inner functions g_i to computing the Hamming weight of each set. Then h can determine the overall Hamming weight, and output 1 if and only if it is at least $n/2$. This uses a total of $m \leq O(\frac{n}{k}\log k)$ inner functions.

As for lower bounds, while it is an easy exercise to improve the trivial lower bound of $\geq n/k$ to $> n/k$ in the case of majority, even a modest lower bound of $\geq 1.1\frac{n}{k}$ seems challenging to establish. Our main result is an asymptotically tight lower bound showing that the construction described above is optimal, and that majority has a composition overhead of $\Theta(\log k)$.

▶ **Theorem 2** (Main theorem). *For all $\epsilon > 0$ and $k \leq n^{1-\epsilon}$, $\mathsf{CC}_k(\textsc{Maj}_n) \geq \Omega(\frac{n}{k}\log k)$.*

In addition to being a natural complexity measure that is of independent interest, our study of composition complexity is further motivated by its relationships to two important models of computation: bounded-width branching programs and small-depth circuits. These are two of the most intensively studied models in circuit complexity, and majority plays a starring role in the efforts at lower bounds for both models. As we now elaborate, Theorem 2 recovers, as a corollary and via an entirely different proof, the current best lower bounds on the length of bounded-width branching programs for majority [2, 4]. It is also the first step in a plan that we propose for lower bounds against depth-3 circuits computing majority, a long-standing open problem that dates back to the 1990s [13]; such lower bounds would represent the first improvements over the state of the art for depth-3 circuits in over three decades [12].

1.2 Motivation and implications

1.2.1 Bounded-width branching programs

There is an easy reduction from branching programs to our model: if function f is computed by a bounded-width branching program of length L, then $\mathsf{CC}_k(f) \leq O(L/k)$. The reduction works by cutting the branching programs into $\sim L/k$ segments of length at most k, then for each segment and for each state s at the start of the segment, use $O(1)$ inner functions to compute which state one would end up at the end of the segment if one started from s. Since the segments have length at most k, each of the inner functions depends on at most k variables.

This means that nontrivial lower bounds on the composition complexity $\mathsf{CC}_k(f)$ for any k directly imply nontrivial lower bounds on the length of bounded-width branching programs: $L \geq \Omega(k \cdot \mathsf{CC}_k(f))$. In particular, setting $k = \sqrt{n}$, by Theorem 2 we obtain

that bounded-width branching programs computing the majority function must have length $L \geq \Omega(\sqrt{n} \cdot \frac{n}{\sqrt{n}} \log \sqrt{n}) = \Omega(n \log n)$, recovering the classic lower bound of Alon and Maass [2] and Babai, Pudlák, Rödl, and Szemerédi [4] as a corollary:

▶ **Theorem 3** ([2, 4]). *Any bounded-width branching program computing* $\textsc{Maj}_n$ *must have length* $\Omega(n \log n)$.

This lower bound remains the current state of the art. Interestingly, our techniques are completely different from the techniques used in [2] and [4]: they first prove a Ramsey-theoretic lemma that identifies two sets of variables that are queried in disjoint segments of the branching program, then conclude by a communication complexity argument between them. We elaborate on our techniques in Section 1.3.

1.2.2 Lower bounds for small-depth circuits computing majority

We view Theorem 2 in part as an essential and necessary first step towards proving stronger lower bounds against small-depth circuits. In this section, we will outline why such lower bounds are particularly interesting, and the relationship between our composition complexity lower bounds and circuit lower bounds.

Lower bounds for small-depth circuits

A fruitful line of work from the '80s [7, 1, 32, 12] managed to prove strong lower bounds against small-depth circuits, but using a surprisingly simple function: parity. In particular, the result that culminated from these works was that any depth-d circuit computing $\textsc{Parity}_n$ must have size $2^{\Omega\left(n^{\frac{1}{d-1}}\right)}$, which is superpolynomial for any $d = o(\log n / \log \log n)$. These bounds are optimal for parity – an extension of the divide-and-conquer scheme mentioned earlier matches this bound. As we now elaborate, these remain essentially the strongest (explicit) lower bounds against small-depth circuits we have for any function, even in the case of $d = 3$, despite the fact that counting arguments give lower bounds of $\Omega(2^n/n)$, and we expect that circuits for hard functions like SAT must also have size $2^{\Omega(n)}$.

Depth-3 circuits and majority

The problem of improving the state of the art for depth-3 circuits ($2^{\Omega(\sqrt{n})}$ for $\textsc{Parity}_n$) in particular has received significant attention as one of the simplest restricted models that are poorly understood. Stronger depth-3 bounds are likely to imply stronger small-depth bounds in general, and *much* stronger bounds of the form $2^{\omega(n/\log \log n)}$ would also give functions that cannot be computed by linear-size log-depth circuits due to a classical result of Valiant [30]. For a detailed exposition of the motivations for depth-3 and the attempts to understand the model, see [16, Chapter 11].

Of the functions that could prove stronger depth-3 lower bounds, majority, being such a basic and simple-to-understand function, is a particularly enticing candidate. The divide-and conquer construction analogous to that for parity gives depth-3 circuits of size $2^{O(\sqrt{n \log n})}$, and the same $2^{\Omega(\sqrt{n})}$ lower bound for parity also applies to majority.[1] In light of this gap, Håstad, Jukna, and Pudlák [13] posed the following challenge.

[1] For general depth d, the upper bound is $2^{O\left(n^{\frac{1}{d-1}} \cdot (\log n)^{1-\frac{1}{d-1}}\right)}$ [17] and the lower bound is $2^{\Omega\left(n^{\frac{1}{d-1}}\right)}$ once again.

▶ **Open Problem 1** ([13]). *Find an explicit function that requires depth-3 circuits of size $2^{\omega(\sqrt{n})}$. In particular, does $\textsc{Maj}_n$ require depth-3 circuits of that size?*

Despite this natural candidate function, all of the improvements on the lower bounds for depth-3 circuits have still been of the form $2^{c\sqrt{n}}$ for successively larger c. [13] found an innovative method of proving lower bounds for circuits from the "top down", and were able to get constants $c = 0.618$ and 0.849 for parity and majority respectively. Paturi, Pudlák, and Zane [24] improved the constant to the optimal $c = 1$ for parity, and later the same authors along with Saks [23] obtained the constant $c = 1.282$ for the membership function of an error-correcting code. Getting a super-constant improvement over this state of the art has been a major frontier of circuit complexity for decades.

For majority, there have also been attempts at improved *upper bounds*. [31] proposed a probabilistic construction of a depth-3 circuit computing $\textsc{Maj}_n$ with size $2^{O(\sqrt{n})}$, but the construction turned out to be mistaken.[2]

The need for new techniques

One view of why previous techniques have fallen short on resolving Open Problem 1 is that at their core, they use sensitivity as the key complexity measure for which small depth circuits are weak (see [5, 28, 20]). With respect to sensitivity, of course $\textsc{Parity}_n$ is the most complex function because it has sensitivity n at every input. The fact that parity is the hardest should suggest why we have struggled to get lower bounds stronger than those for parity. More concretely, these techniques do not establish bounds stronger than $2^{\Omega(s(f)^{\frac{1}{d-1}})}$ where $s(f)$ is the sensitivity of $f : \{0,1\}^n \to \{0,1\}$. But $s(f) \leq n$, so this leaves us stuck the current state of the art ($2^{\Omega(\sqrt{n})}$ for depth-3).

To push beyond our current small-depth circuit lower bounds, there is a need for new techniques. In particular, we need to make use of complexity measures beyond sensitivity, where parity is no longer the hardest function. Moreover, to solve Open Problem 1, such a complexity measure needs to be one where majority is demonstrably harder than parity. The notion of composition complexity and the techniques of this paper meet both of these demands (as shown by Theorem 2). We are hopeful that these techniques (described in Section 1.3) can be extended to prove stronger lower bounds in more general settings.

Composition complexity and depth-3 circuits

If $\mathsf{CC}_k(f) \leq m$, then we can write $f = h(g_1, \ldots, g_m)$ where each of the functions g_j only needs to query k variables. But then we can write h as a DNF (or CNF) of size 2^m, and we can write the inner functions g_j as CNFs (or DNFs) of size 2^k. In this form we have a depth-3 circuit for f of size

$$2^m + m \cdot 2^k \leq 2^{O(m+k)} \tag{1}$$

with bottom fan-in k. The best-known depth-3 circuits for computing $\textsc{Maj}_n$ are obtained in precisely this manner, by using the bound $\mathsf{CC}_k(\textsc{Maj}_n) \leq O(\frac{n}{k} \log k)$ and setting $k \coloneqq \sqrt{n \log n}$.

It follows that in order to prove that $\textsc{Maj}_n$ requires depth-3 circuits of size $2^{\Omega(\sqrt{n \log n})}$, one must first prove that $\max(\mathsf{CC}_k(f), k) \geq \Omega(\sqrt{n \log n})$. This is implied directly by Theorem 2, which we view as a key first step towards proving stronger depth-3 lower bounds. In particular,

[2] Briefly, in the notation of [31], it requires that $kn/c < n$, and so $k < c$, but c then takes on all the values of $n, n/2, n/3, \ldots, 1$ and $k \approx \sqrt{n}$. The third author thanks Srikanth Srinivasan [29] who informed him about this gap in the proof.

it shows that if one wanted to construct a depth-3 circuit for $\textsc{Maj}_n$ of size $2^{o(\sqrt{n \log n})}$, it would have to look very different from the current divide-and-conquer strategy. In Section 6.1 we outline some ways in which our results could be extended beyond the model we consider to depth-3.

1.3 Our techniques

In this section, we briefly describe the techniques we use to prove Theorem 2, and highlight some aspects that we feel are particularly interesting.

Information theory

While there have been some attempts to incorporate information theory into the toolbox of boolean function lower bounds [22, 8], these techniques remain uncommon. Our proof crucially uses *mutual information* to measure information flow from the input variables to the inner functions.

Our key insight can be summed up as the counterintuitive maxim "the less it is queried the more it is revealed". More concretely, suppose we can compute the Hamming weight function as $h(g_1, \ldots, g_m)$, then we show that if few of the inner functions $g_1, \ldots, g_m$ query a particular input variable x_i, then the output of the inner functions must reveal a lot of information about x_i's value – that is, one can guess x_i better than random chance based on $g_1(x), \ldots, g_m(x)$.

▶ **Lemma 4** (key insight, informally). *Suppose that x_i is queried by at most q of the inner functions $g_1, \ldots, g_m$. Then $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \geq 2^{-O(q)}$.*

One of the novelties of our proof is that the information flow can be distilled so cleanly as the above lemma, and that tight lower bounds follow quite easily from it (see Section 3). It is natural to wonder whether this approach can be generalized to stronger models of computation.

From multi-output functions to binary-output functions

Another intriguing aspect of our proof is that in proving the lower bound for majority, it turns out to be easiest to first prove a lower bound for the Hamming weight function $\mathrm{HW}_n : \{0,1\}^n \to \{0, \ldots, n\}$, which maps $x \mapsto |x|$ (i.e., the number input bits that are 1). Notably, this is not a binary-output function, but rather a "multi-output" function (its output takes $\lceil \log(n+1) \rceil$ bits), and the proof of our information-theoretic Lemma 4 fundamentally uses this larger output space.

The way in which we extend the lower bound from HW_n to $\textsc{Maj}_n$ is also worth noting. In Section 4, we give a general framework for, in a sense, forcing binary-output functions to become multi-output. We use "control variables" to manipulate the construction $h(g_1, \ldots, g_m)$ into telling us more about the input, and "buffer variables" to avoid incurring a blowup in how many inner functions are necessary. This allows us to show that an efficient construction for $\textsc{Maj}_n$ would imply a similarly efficient construction computing the Hamming weight on a large fraction of inputs, and we can then conclude with a slightly more general version of Lemma 4.

Perhaps there is more to be found in this direction. Could the frontier of circuit lower bounds be pushed further by first proving lower bounds for multi-output functions, and then bootstrapping these to get lower bounds against the usual single-output functions? Indeed,

our proof technique suggests that proving more lower bounds for multi-output functions could be a valuable endeavor, even when those lower bounds do not seem to immediately lead to lower bounds for binary-output functions we traditionally study.

Natural proofs

The proof of our key lemma in Section 5 is tailored specifically to the Hamming weight function.[3] While this could be seen as a limitation, it can also be seen as a strength. Indeed, Razborov and Rudich [27] showed that lower bound arguments cannot apply to too broad a range of functions, assuming that pseudorandom functions exist. Given that our lower bound argument only works for the Hamming weight and closely related functions like the majority function, it does not seem to fall within the natural proofs framework.

1.4 Related work

Hrubeš-Rao and Nechiporuk's method

Hrubeš and Rao [15] gave a function f for which $\mathsf{CC}_k(f) \geq n^{\Omega(1-k/n)}$.[4] Their proof draws inspiration from Nechiporuk's method [21], which gives lower bounds for functions f for which one can split the variables into disjoint sets $S_1, \ldots, S_\ell$ such that f has many distinct subfunctions on each S_r ($r \in [\ell]$).

Their lower bound is quantitatively stronger than ours, but just like other bounds obtained from Nechiporuk's method, it only applies to a limited set of functions specially created for that purpose, and says nothing about many basic functions like majority. Nechiporuk's method is unable to prove lower bounds for majority because its subfunctions are all threshold functions, of which there are only a handful.[5] Moreover, the functions that [15] uses have no bearing on stronger lower bounds for depth-3 circuits, since their functions actually have depth-2 circuits of size $n^{O(\log n)}$.

It is interesting to contrast our techniques with Nechiporuk's method. At a very high level, Nechiporuk's method considers what one can infer about the function after fixing several variables (particularly, how many subfunctions remain), whereas our method considers what one can infer about the variables given the output of several of the inner functions. This difference in perspective is key in our ability to get tight lower bounds in a case where Nechiporuk's method only gives trivial bounds.

Lower bounds for canonical boolean circuits

Goldreich and Wigderson [11] recently introduced a new restricted of model of "canonical" boolean circuits, with the hope of proving $2^{\omega(\sqrt{n})}$ lower bounds for this model. Their model is inspired by the structure of optimal depth-3 circuits for $\textsc{Parity}_n$, which dissects the computation into disjoint parities of smaller arity (over $\sqrt{n}$ variables). [11] proposes a generalization of this construction, where one aims to represent a multi-linear function as a depth-2 circuit where the gates are *arbitrary multi-linear functions of small arity*. They

[3] At a high level, it uses the facts that the possible Hamming weight values $0, \ldots, n$ are a completely ordered set and that flipping a bit from 0 to 1 increases the Hamming weight by one, in order to find one weight w^* for which the corresponding inputs are biased on a given coordinate.

[4] Their paper denotes k-composition complexity as C^2_k instead of CC_k.

[5] Indeed, there is an exactly analogous situation with branching programs. While Nechiporuk's method can establish strong branching program lower bounds (see [26, Theorem 2], attributed to Beame and Cook), [2, 4] had to introduce new techniques to prove lower bounds for majority.

propose proving strong lower bounds in this model as a "sanity check" for proving better general depth-3 circuit bounds (Open Problem 1) – to do the latter, one must necessarily do the former as well.

Our approach can be viewed in a very similar light. In fact, our model is strictly stronger: we consider gates that compute arbitrary *boolean functions* of small arity, not just functions that are multi-linear over GF(2). In the same way, our model serves as a sanity check for Open Problem 1 – any proof that depth-3 circuits for $\textsc{Maj}_n$ require size $2^{\Omega(\sqrt{n \log n})}$ must prove Theorem 2 as well.

Strong lower bounds have indeed been proven in the model introduced by [11]. Goldreich and Tal [9] proved a lower bound of $2^{\tilde{\Omega}(n^{2/3})}$. [10] also proved lower bounds of $2^{\tilde{\Omega}(n^{3/8})}$ for canonical depth-4 circuits, an improvement over the $2^{\Omega(n^{1/3})}$ bound that is known for the general depth-4 circuits computing $\textsc{Parity}_n$. In our model where the gates can compute arbitrary boolean functions of small arity, such strong bounds are not possible for the $\textsc{Maj}_n$ function, and we pin down exactly the right bounds in this case.

Majority as a majority of majorities

There has been interest [18, 6, 14, 3, 25] in the optimal ways of computing majority as a composition of functions $h(g_1, \ldots, g_m)$ in the restricted model, where the h and $g_1, \ldots, g_m$ are majority functions of smaller fan-in ($\textsc{Maj}_{\leq k}$ for some k). [18] showed that $k \geq n^{0.7}$ was necessary, [6] improved this to $k \geq n^{0.8}$, and [14] further improved this to $k \geq n/2 - o(n)$. In terms of upper bounds, [3] gave $k \leq n+2$ for odd $n \geq 7$, and [25] improved this to $k \leq \frac{2}{3}n+4$ (for all n).

Similar to the state of canonical boolean circuits discussed above, this setting is more restrictive, so it makes sense that their lower bounds are stronger than Theorem 2. It would be interesting to see if our techniques could apply to these more restricted models as well.

2 Preliminaries

2.1 Locality

In this subsection, we define some notation that we will use repeatedly, and we give a formal definition of k-locality.

▶ **Definition 5** ($x^{(i \mapsto b)}$, $x^{\oplus i}$). *For any $x \in \{0,1\}^n$, $i \in [n]$ and $b \in \{0,1\}$, let*

- $x^{(i \mapsto b)} := (x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$;
- $x^{\oplus i} := (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n)$.

▶ **Definition 6** (k-local). *A function $g : \{0,1\}^n \to \{0,1\}$ is k-local if there is a set of variables $I \subseteq [n]$ with $|I| = k$ such that g depends only on the variables in I, i.e. for each $x \in \{0,1\}^n$ and each $i \in [n] \setminus I$, we have $g(x) = g(x^{\oplus i})$.*

▶ **Definition 7** (composition complexity, formal version of Definition 1). *The* k-composition complexity *of a function $f : \{0,1\}^n \to D$, denoted $\mathsf{CC}_k(f)$, is the minimum integer m such that there exist functions $g_1, \ldots, g_m : \{0,1\}^n \to \{0,1\}$ and $h : \{0,1\}^m \to D$ with the following properties:*

- **(i)** *for all $x \in \{0,1\}^n$, $f(x) = h(g_1(x), \ldots, g_m(x))$;*
- **(ii)** *for each $j \in [m]$, g_j is k-local.*

▶ Remark 8. Note that the inner functions g_j are restricted to having binary outputs. This is necessary for making the definition nontrivial: if their output domains were arbitrary, then the composition complexity would always be $O(n/k)$, since we could simply let each inner function output the values of all of the variables they query.

2.2 Number of queries per variable

In this subsection, we show that without loss of generality, we can assume that all variables are queried roughly the same number of times. More precisely, say that $f = h(g_1, \dots, g_m)$ where each g_j is k-local. Then the total number of queries is at most mk, so the average variable is queried at most $\frac{mk}{n}$ times. We will show that for the purpose of proving lower bounds on composition complexity, we can assume that *every* variable is queried at most $\frac{mk}{n}$ times.

▶ **Definition 9** (self-containing). *A family of functions $\{f_n\}_{n\in\mathbb{N}}$ is* self-containing *if for any n and any $I \subseteq [n]$, there is a subfunction of f_n on I that computes $f_{|I|}$.*

▶ **Fact 10.** *Both majority $\{\text{MAJ}_n\}_{n\in\mathbb{N}}$ and Hamming weight $\{\text{HW}_n\}_{n\in\mathbb{N}}$ are self-containing.*

▶ **Lemma 11.** *Let $\{f_n\}_{n\in\mathbb{N}}$ be a self-containing family of functions. Suppose that $\mathsf{CC}_k(f_{2n}) \leq m$. Then we can write $f_n = h(g_1, \dots, g_m)$ where each g_j is k-local and each variable is queried at most $\frac{mk}{n}$ times.*

Proof. Suppose $f_{2n} = h(g_1, \dots, g_m)$, where each g_j is k-local. Let $q \coloneqq \frac{mk}{2n}$. Then the average variable is queried $\leq q$ times, so by Markov's inequality at most half of the variables are queried more than $2q$ times. Let I be any set of n variables, each of which is queried at most $2q = \frac{mk}{n}$ times. Since $\{f_n\}_{n\in\mathbb{N}}$ is self-containing, there is a subfunction of f_{2n} on I that computes f_n. The lemma follows by restricting each g_j to I. ◀

▶ **Corollary 12.** *In order to prove that $\mathsf{CC}_k(f_n) \geq \Omega(\frac{n}{k}\log k)$, it is enough to prove that if $f_n = h(g_1, \dots, g_m)$ and each variable is queried at most $\frac{mk}{n}$ times by the inner functions g_j, then $m \geq \Omega(\frac{n}{k}\log k)$.*

▶ Remark 13. The quantity $\frac{mk}{n}$ corresponds exactly to our definition of composition overhead (recall the discussion following Definition 1). This makes sense, since the composition overhead is the ratio of how many inner functions we need compared to the ideal situation where each variable is queried exactly once. The more inner functions, the more queries per variable (assuming they all query roughly k variables).

2.3 Information theory

In this subsection, we introduce some information-theoretic notions and properties that are used in our proofs. To learn more about information theory from a theoretical computer science perspective, we recommend checking out the Simons Institute workshop titled "Information Theory Boot Camp".[6]

The most important notion in information theory is the *entropy* of a random variable $\boldsymbol{X}$, denoted $\mathbf{H}[\boldsymbol{X}]$, which represents "how much randomness" the variable contains, or how many bits I need to communicate to you on average for you to learn $\boldsymbol{X}$.

▶ **Definition 14** (entropy). *Given a random variable $\boldsymbol{X}$ with support D, the* entropy of $\boldsymbol{X}$ *is the quantity*

$$\mathbf{H}[\boldsymbol{X}] \coloneqq \sum_{x\in D} \mathbf{Pr}[\boldsymbol{X} = x] \log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = x]}\right) = \mathop{\mathbf{E}}_{\boldsymbol{X}'\sim\boldsymbol{X}}\left[\log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = \boldsymbol{X}']}\right)\right]$$

where $\boldsymbol{X}'$ is an independent copy of $\boldsymbol{X}$.

[6] https://simons.berkeley.edu/workshops/inftheory2015-boot-camp

A related notion is the *conditional entropy* $\mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}]$ of two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, which represents the "how much randomness remains" in $\boldsymbol{X}$ once you know $\boldsymbol{Y}$, or how many bits I need to communicate to you on average for you to learn $\boldsymbol{X}$, assuming that you already know $\boldsymbol{Y}$.

▶ **Definition 15** (conditional entropy). *Given two random variables $(\boldsymbol{X}, \boldsymbol{Y})$ over domain D, the* entropy of $\boldsymbol{X}$ conditioned on $\boldsymbol{Y}$ *is the quantity*

$$\begin{aligned}\mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}] &\coloneqq \sum_{(x,y)\in D} \mathbf{Pr}[\boldsymbol{X} = x \wedge \boldsymbol{Y} = y] \log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = x \mid \boldsymbol{Y} = y]}\right) \\ &= \mathop{\mathbf{E}}_{(\boldsymbol{X}',\boldsymbol{Y}')\sim(\boldsymbol{X},\boldsymbol{Y})}\left[\log\left(\frac{1}{\mathbf{Pr}[\boldsymbol{X} = \boldsymbol{X}' \mid \boldsymbol{Y} = \boldsymbol{Y}']}\right)\right]\end{aligned}$$

where $(\boldsymbol{X}', \boldsymbol{Y}')$ is an independent copy of $(\boldsymbol{X}, \boldsymbol{Y})$.

The entropy and conditional entropy have the following properties, which we use in our proofs.

▶ **Fact 16** (bounds). *If $\boldsymbol{X}$ is a random variable over a finite domain D and $\boldsymbol{Y}$ is a random variable, then*

$$0 \le \mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}] \le \mathbf{H}[\boldsymbol{X}] \le \log|D|,$$

where the last inequality is tight iff $\boldsymbol{X}$ is uniform on D.

▶ **Fact 17** (subadditivity). *Let $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ be two random variables. Then $\mathbf{H}[\boldsymbol{X}_1, \boldsymbol{X}_2] \le \mathbf{H}[\boldsymbol{X}_1] + \mathbf{H}[\boldsymbol{X}_2]$, with equality iff $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are independent. Similarly, $\mathbf{H}[\boldsymbol{X}_1, \boldsymbol{X}_2 \mid \boldsymbol{Y}] \le \mathbf{H}[\boldsymbol{X}_1 \mid \boldsymbol{Y}] + \mathbf{H}[\boldsymbol{X}_2 \mid \boldsymbol{Y}]$.*

▶ **Fact 18** (dependence). *Let $\boldsymbol{X}, \boldsymbol{Y}$ be random variables such that $\boldsymbol{X}_2$ is completely determined by $\boldsymbol{X}_1$ (i.e. $\boldsymbol{X}_2 = f(\boldsymbol{X}_1)$ where f is a function). Then we have the following:*

- $\mathbf{H}[\boldsymbol{X}_2 \mid \boldsymbol{X}_1] = 0$;
- $\mathbf{H}[\boldsymbol{Y} \mid \boldsymbol{X}_1] \le \mathbf{H}[\boldsymbol{Y} \mid \boldsymbol{X}_2]$.

Finally, a crucial notion in our proof is the *mutual information* $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}]$ of two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, which represents "how much information $\boldsymbol{X}$ reveals about $\boldsymbol{Y}$", or symmetrically, "how much information $\boldsymbol{Y}$ reveals about $\boldsymbol{X}$".

▶ **Definition 19** (mutual information). *Given two random variables $\boldsymbol{X}, \boldsymbol{Y}$, the* mutual information *of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the quantity $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}] \coloneqq \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid \boldsymbol{Y}]$.*

▶ **Fact 20** (symmetry of mutual information). $\mathbf{I}[\boldsymbol{X} : \boldsymbol{Y}] = \mathbf{I}[\boldsymbol{Y} : \boldsymbol{X}]$.

3 The less it is queried, the more it is revealed

Hamming weight: a multi-output function

Even though the main function of interest in this paper is the majority function, we will first prove a lower bound for the related *Hamming weight* function: a "multi-output" function (as opposed to binary-output) that reveals the entire Hamming weight of the input string.

▶ **Definition 21.** *Let $\mathrm{HW}_n : \{0,1\}^n \to \{0,1,\dots,n\} : x \mapsto |x| = x_1 + \cdots + x_n$ be the Hamming weight function.*

The most natural way to express HW_n as $h(g_1, \ldots, g_m)$ is to split the n variables into groups of k variables, and for each group to create $\lceil \log(k+1) \rceil$ inner functions g_j, each computing one bit of the sum of the k variables in this group. The function h can then compute the Hamming weight of the whole input string by first recovering the sum for each group, then adding them up. Clearly, each of the inner functions is k-local, and there are $O(\frac{n}{k} \log k)$ of them, so $\mathsf{CC}_k(\mathrm{HW}_n) \leq O(\frac{n}{k} \log k)$. We will show that this is optimal: $\mathsf{CC}_k(\mathrm{HW}_n) = \Theta(\frac{n}{k} \log k)$.

Simple counting does not give much

It is easy to see $\mathsf{CC}_k(\mathrm{HW}_n) > n/k$: suppose that there were a way to represent HW_n as $h(g_1, \ldots, g_m)$ with $m = n/k$. Then each of the inner functions g_j would query k variables and each variable would be queried only once. Take g_1, and consider the k variables it queries. Say that we fix all other variables to 0. Then there are still $k+1$ possible values for the Hamming weight. But this fixes the outputs of $g_2, \ldots, g_m$, so there can only be two possible values for $h(g_1, \ldots, g_m)$ (one where g_1 outputs 0 and the other where g_1 outputs 1), so we have a contradiction.

In general, it is easy to see that each set of r variables must collectively be queried by at least $\log(r+1)$ different inner functions. But this observation is not enough to show that the average variable will need to be queried a super-constant number of times.

A very counterintuitive lemma

Basic counting arguments like the above do not seem to say anything that keeps a variable from being queried by only a constant number of inner functions. But it turns out there *is* something we can say about variables that are queried by few inner functions: the outputs of the inner functions must reveal a lot about their value, in terms of mutual information. Put another way, this means that if you are given the outputs of the inner functions, you can often guess the value of such variables better than random chance.

▶ **Lemma 22** (key lemma). *Suppose that* $\mathrm{HW}_n = h(g_1, \ldots, g_m)$ *and that variable* i *is queried at most* q *of the inner functions* $g_1, \ldots, g_m$*. Let* $\boldsymbol{X} \sim \{0,1\}^n$ *be a uniformly random input. Then* $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \geq 2^{-O(q)}$*.*

Consider how counterintuitive this statement is: it says that the *less* the i^{th} variable is queried by the inner functions $g_1, \ldots, g_m$, the *more* it will be revealed by their collective outputs.

A lower bound for Hamming weight, assuming Lemma 22

We will not prove Lemma 22 until Section 5, but let us see how it implies the lower bound we want. First, we will show the intuitive fact that "if the inner functions reveal a lot about many of the variables, then there must be many inner functions".

▶ **Corollary 23.** *Suppose that* $\mathrm{HW}_n = h(g_1, \ldots, g_m)$ *and each variable is queried by at most* q *of the inner functions* $g_1, \ldots, g_m$*. Then* $m \geq n \cdot 2^{-O(q)}$*.*

Proof. We bound the quantity $\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})]$ in two ways. First, we give an upper bound on m based on the fact that there are only m inner functions, each of which outputs a single bit. Then we lower bound this same quantity using Lemma 22. Combining these bounds yields the corollary.

On the one hand,

$$\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] &= \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] - \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X}) \mid \boldsymbol{X}] \\
&= \mathbf{H}[g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\qquad \text{(because } \boldsymbol{X} \text{ determines } g_j(\boldsymbol{X}) \text{ completely)} \\
&\leq \mathbf{H}[g_1(\boldsymbol{X})] + \ldots + \mathbf{H}[g_m(\boldsymbol{X})] \quad \text{(because } \mathbf{H}[\cdot] \text{ is subadditive)} \\
&\leq m. \qquad \text{(because each } g_j \text{ has a binary output)}
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] &= \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&= \left(\sum_{i \in [n]} \mathbf{H}[\boldsymbol{X}_i] \right) - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\qquad \text{(because the } \boldsymbol{X}_i \text{ are independent)} \\
&\geq \sum_{i \in [n]} (\mathbf{H}[\boldsymbol{X}_i] - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})]) \\
&\qquad \text{(because } \mathbf{H}[\cdot \mid \boldsymbol{Y}] \text{ is subadditive for any } \boldsymbol{Y}) \\
&= \sum_{i \in [n]} \mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \\
&\geq \sum_{i \in [n]} 2^{-O(q)} \qquad \text{(by Lemma 22)} \\
&= n \cdot 2^{-O(q)}.
\end{aligned}$$

The corollary follows from combining these two bounds. ◀

We can now use Corollary 23 to deduce the desired lower bound.

▶ **Theorem 24** (composition complexity of HW_n). $\mathsf{CC}_k(\mathrm{HW}_n) \geq \Omega(\frac{n}{k} \log k)$.

Proof. Suppose that $\mathrm{HW}_n = f(g_1, \ldots, g_m)$, where each of the inner functions $g_1, \ldots, g_m$ is k-local. By Corollary 12, we only have to show that $m \geq \Omega(\frac{n}{k} \log k)$ under the assumption that each variable is queried at most $q := \frac{mk}{n}$ times. Then by Corollary 23,

$$\frac{nq}{k} = m \geq n \cdot 2^{-O(q)}.$$

Rearranging, we get

$$q \cdot 2^{O(q)} \geq k \Rightarrow q \geq \Omega(\log k),$$

which means $m = \frac{n}{k} q \geq \Omega(\frac{n}{k} \log k)$. ◀

4 From Hamming weight to majority

In the previous section, we gave a lower bound on the composition complexity for the Hamming weight function HW_n. This is a multi-output function (it has more than two possible outputs), and the proof arguably relied a lot on this fact. In this section, we show how to extend the lower bound to $\textsc{Maj}_n$, a binary-output function, by manipulating it to become multi-output.

▶ **Definition 25** (majority function). *Let*

$$\textsc{Maj}_n : \{0,1\}^n \to \{0,1\} : x \mapsto \begin{cases} 1 & \text{if } |x| \geq n/2 \\ 0 & \text{otherwise} \end{cases}$$

be the majority function.

4.1 Reduction toolkit

Before proving the lower bound, let us introduce the tools we will use to reduce majority to Hamming weight.

Control variables

$\textsc{Maj}_n(x)$ tells us whether x has Hamming weight $\geq n/2$ or $< n/2$, but nothing more. For example, what if we wanted it to also tell us something special when the Hamming weight is *exactly* $n/2$? Suppose n is even, then we could figure this out by looking at the values of $\textsc{Maj}_{n+1}(x_1, \ldots, x_n, 0)$ and $\textsc{Maj}_{n+1}(x_1, \ldots, x_n, 1)$: indeed,

- if $|x| < n/2$ then both return 0;
- if $|x| > n/2$ then both return 1;
- if $|x| = n/2$ then $\textsc{Maj}_{n+1}(x_1, \ldots, x_n, 0) = 0$ but $\textsc{Maj}_{n+1}(x_1, \ldots, x_n, 1) = 1$.

Note how the last variable of $\textsc{Maj}_{n+1}$ does not stay free, but rather is assigned a fixed value that modifies the behavior of $\textsc{Maj}_{n+1}$ as a function of $x_1, \ldots, x_n$. For this reason, we will call it a *control variable*, and we will call the n other variables *free variables*.

Now, assume that $\textsc{Maj}_{n+1}$ can be computed as $h(g_1, \ldots, g_m)$. Then we can compute the function

$$f : \{0,1\}^n \to \{0,1,2\} : x \mapsto \begin{cases} 0 & \text{if } |x| < n/2 \\ 1 & \text{if } |x| = n/2 \\ 2 & \text{if } |x| > n/2 \end{cases}$$

as $h'(g_1^0, \ldots, g_l^0, g_1^1, \ldots, g_l^1)$ where

$$g_j^0(x) \coloneqq g_j(x_1, \ldots, x_n, 0) \qquad g_j^1(x) \coloneqq g_j(x_1, \ldots, x_n, 1).$$

This suggests a potential approach for proving a lower bound on the composition complexity $\textsc{Maj}_n$: add enough control variables to it so that the values we get are enough to determine the Hamming weight of x, then use the lower bound for HW_n as a black box. Indeed, if we know all of the values

$$\begin{cases} \textsc{Maj}_{2n-1}(x_1, \ldots, x_n, 0, 0, \ldots, 0) \\ \textsc{Maj}_{2n-1}(x_1, \ldots, x_n, 1, 0, \ldots, 0) \\ \qquad\qquad\qquad \vdots \\ \textsc{Maj}_{2n-1}(x_1, \ldots, x_n, 1, 1, \ldots, 1) \end{cases}$$

then we can recover $|x|$. However, this would cause a huge blowup in the number of inner functions needed to compute it: assuming $\textsc{Maj}_{2n-1}$ can be computed as $h(g_1, \ldots, g_m)$, this would only guarantee that HW_n can be computed as a function of the nm components

$$g_j(x_1, \ldots, x_n, 0, \ldots, 0), \ldots, g_j(x_1, \ldots, x_n, 1, \ldots, 1)$$

for $j \in [m]$, thus the $\Omega(\frac{n}{k} \log k)$ lower bound we have for HW_n would fail to give any nontrivial lower bound on m. So we need to do this in a smarter way.

Buffer variables

We will avoid this blowup by using some *buffer variables* to "isolate" the control variables from the free variables. Assume that $\textsc{Maj}_n = h(g_1, \ldots, g_m)$. Suppose we split the variables into three sets $I_{\text{free}}, I_{\text{control}}, I_{\text{buffer}} \subseteq [n]$ such that none of the inner functions g_j queries variables from both I_{free} and I_{control} (informally, I_{buffer} acts as a buffer between I_{free} and I_{control}; see Figure 1).

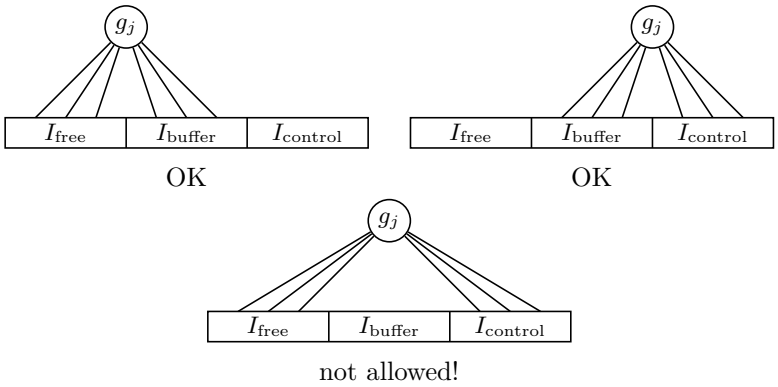


Figure 1 The inner functions g_j are allowed to query variables in both I_{free} and I_{buffer}, or variables in both I_{buffer} and I_{control}, but not variables in both I_{free} and I_{control}.

Then if we fix the value of the variables in I_{buffer}, we can avoid a blow up in the number of times each variable is queried. Let us informally see why. There are two types of inner functions g_j:

- if g_j queries a free variable, then it does not query any control variables, so no blowup will happen;
- if g_j does *not* query any free variables, then it only depends on the control variables and the buffer variables, both of which are known, so in that case we can recompute the output of g_j ourselves.

A bit more formally, let $x^{(I_{\text{control}} \mapsto 0)}$ denote x with all variables in I_{control} set to 0. Then we can recover $\textsc{Maj}_n(x)$ if we know only $g_1(x^{(I_{\text{control}} \mapsto 0)}), \ldots, g_m(x^{(I_{\text{control}} \mapsto 0)})$ as well as the value of the variables in I_{control} and I_{buffer}. Since $g_j(x^{(I_{\text{control}} \mapsto 0)})$ does not depend on the value of the variables in I_{control}, no blowup happens when we start to vary their values.

Partial functions

Even with the help of buffer variables, though, it turns out we will not be able to have enough control variables to compute HW_n from $\textsc{Maj}_n$.[7] We will only be able to compute the Hamming weight correctly on a fraction of the possible inputs. Therefore we need to adapt our techniques from Section 3 to a "partial functions" setting, where $f(x) = |x|$ is only guaranteed when x is in a subset $D \subseteq \{0,1\}^n$ of the possible inputs.

First, the following lemma is a generalization of our key lemma Lemma 22, which gave a lower bound on the mutual information between $\boldsymbol{X}_i$ and the inner function outputs $g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})$ when variable i is queried few times. What differs from Lemma 22 is that

[7] Or more precisely, to compute $\text{HW}_{n'}$ from $\textsc{Maj}_n$ for some $n' = \Omega(n)$.

the result gives a nontrivial bound only when the probability $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D]$ is small. This probability measures how likely you are to leave D if you start out in D then flip the i^{th} bit, so intuitively, the bigger D is, the smaller it will be.

▶ **Lemma 26** (key lemma, "partial functions" version)**.** *Let $D \subseteq \{0,1\}^n$ be a non-empty subset of the hypercube (the "domain" on which f computes the Hamming weight), and let $f : \{0,1\}^n \to \{0,\ldots,n\}$ be a function such that $\forall x \in D$, $f(x) = |x|$. Suppose that $f = h(g_1,\ldots,g_m)$ and that variable i is queried by at most q of the inner functions $g_1,\ldots,g_m$. Let $\boldsymbol{X} \sim D$ be a random input drawn uniformly over D. Then $\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}$.*

We postpone the proof of this lemma until Section 5. For now, let us see why it implies Lemma 22.

Proof of Lemma 22 assuming Lemma 26. Set $f \coloneqq \mathrm{HW}_n$. Then $f(x) = |x|$ for all $x \in \{0,1\}^n$, so we can set $D \coloneqq \{0,1\}^n$. This means that $\boldsymbol{X}$ is a uniformly random input, and $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] = 0$. Therefore, after rearranging, we obtain

$$\begin{aligned}
2^{-O(q)} &\leq 1 - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \\
&= \mathbf{H}[\boldsymbol{X}_i] - \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] && \text{(because } \boldsymbol{X} \text{ is uniformly random)} \\
&= \mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})]. && \text{(by definition of mutual information)}
\end{aligned}$$

◀

We can also prove an analog of Corollary 23, which gave a lower bound on the number of inner functions m assuming that many variables are queried few times by the inner functions.

▶ **Corollary 27.** *There is an integer constant $C > 0$ such that the following holds. Let $D \subseteq \{0,1\}^n$ be a non-empty subset of the hypercube, and let $f : \{0,1\}^n \to \{0,\ldots,n\}$ be a function such that $\forall x \in D$, $f(x) = |x|$. Suppose that $f = h(g_1,\ldots,g_m)$, that each variable is queried by at most $q > 0$ of the inner functions $g_1,\ldots,g_m$, and that $\mathbf{Pr}_{\boldsymbol{X}\sim D}[\boldsymbol{X}^{\oplus i} \notin D] \leq 2^{-Cq}$. Then $m + (n - \log|D|) \geq n \cdot 2^{-O(q)}$.*

Proof. Let C' be a possible integer value for the constant in the $O(\cdot)$ of Lemma 26. We will prove the corollary for $C \coloneqq C' + 1$. The proof works in the same way as the proof of Corollary 23: we will upper and lower bound the quantity $\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})]$. On the one hand,

$$\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] &= \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] - \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X}) \mid \boldsymbol{X}] \\
&= \mathbf{H}[g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \\
&\qquad\qquad \text{(because } \boldsymbol{X} \text{ determines } g_j(\boldsymbol{X}) \text{ completely)} \\
&\leq \mathbf{H}[g_1(\boldsymbol{X})] + \ldots + \mathbf{H}[g_m(\boldsymbol{X})] \quad \text{(because } \mathbf{H}[\cdot] \text{ is subadditive)} \\
&\leq m. \qquad\qquad \text{(because each } g_j \text{ has a binary output)}
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\mathbf{I}[\boldsymbol{X} : g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})] &= \mathbf{H}[\boldsymbol{X}] - \mathbf{H}[\boldsymbol{X} \mid g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})] \\
&\geq \mathbf{H}[\boldsymbol{X}] - \sum_{i \in [n]} \mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \dots, g_m(\boldsymbol{X})] \\
&\qquad \text{(because } \mathbf{H}[\cdot \mid \boldsymbol{Y}] \text{ is subadditive for any } \boldsymbol{Y}) \\
&\geq \mathbf{H}[\boldsymbol{X}] - \sum_{i \in [n]} \left(1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-C'q}\right) \qquad \text{(by Lemma 26)} \\
&\geq \mathbf{H}[\boldsymbol{X}] - n + \sum_{i \in [n]} \left(2^{-C'q} - 2^{-(C'+1)q}\right) \\
&\qquad \text{(by assumption that } \mathbf{Pr}_{\boldsymbol{X} \sim D}[\boldsymbol{X}^{\oplus i} \notin D] \leq 2^{-Cq}) \\
&= \log |D| - n + \sum_{i \in [n]} 2^{-O(q)}. \\
&\qquad \text{(because } \boldsymbol{X} \text{ is uniform on } D, \text{ and because } q > 0)
\end{aligned}$$

The corollary follows from combining these two bounds. ◀

4.2 A lower bound for majority

With all the tools in hand, let us prove a lower bound on the composition for $\textsc{Maj}_n$ by reducing it to a partial version of Hamming weight.

▶ **Theorem 28** (composition complexity of $\textsc{Maj}_n$). $\mathsf{CC}_k(\textsc{Maj}_n) \geq \Omega(\frac{n}{k} \cdot \min(\log k, \log \frac{n}{k}))$.

Proof. Suppose that $\textsc{Maj}_n = h(g_1, \dots, g_m)$ where each of the inner functions g_j is k-local. By Corollary 12, it is enough to show that $m \geq \Omega(\frac{n}{k} \cdot \min(\log k, \log \frac{n}{k}))$ under the assumption that each variable is queried at most $q := \frac{mk}{n}$ times. To do this, we will show that $q \geq \Omega(\min(\log k, \log \frac{n}{k}))$.

The plan of the proof is as follows. We will start by defining the sets of variables I_{free}, I_{buffer} and I_{control}. We will make sure that $|I_{\text{free}}| = \Omega(n)$, and we will play with the variables in I_{control} to show that the inner functions $g_1, \dots, g_m$ must compute the Hamming weight over a large subset D of $\{0,1\}^{I_{\text{free}}}$. From there, we will apply Corollary 27, and only asymptotic calculations will remain, similar to the proof of Theorem 24.

Defining I_{free}, I_{buffer} and I_{control}. Let C be the constant in Corollary 27. Assume that $2^{2Cq+1}qk \leq n/2$ (otherwise, $q = \Omega(\log \frac{n}{k})$ and we are done). Let I_{control} be any 2^{2Cq+1}-element subset of $[n]$, let $J \subseteq [m]$ be the set of inner functions that query some variable in I_{control}, and let $I' \subseteq [n]$ be the set of all variables queried by at least one of the inner functions in J. That is, let I' be the set of variables that are queried by some inner function g_j that also queries a variable in I_{control}.

Since each variable in I_{control} is queried by at most q inner functions, and each inner function queries at most k variables, we have $|I'| \leq |I| \cdot q \cdot k = 2^{2Cq+1}qk \leq n/2$. In case $|I'| < 2|I_{\text{control}}|+1$, extend I' to include some more variables of $[n]$ until it reaches size $2|I_{\text{control}}|+1$. After this, we still have $|I'| \leq \max(n/2, 2|I_{\text{control}}| + 1) = \max(n/2, 2 \cdot 2^{2Cq+1} + 1) \leq n/2$.

Let $I_{\text{free}} := [n] \setminus I'$, and let $I_{\text{buffer}} := I' \setminus I_{\text{control}}$ (see Figure 2). Note that $I_{\text{free}} \cup I_{\text{buffer}} \cup I_{\text{control}} = [n]$. Since I_{free} contains no element of I', it is clear that no inner function queries both variables in I_{free} and I_{control}. Also, defining $n_{\text{free}} := |I_{\text{free}}|$, we have

$$n_{\text{free}} = n - |I'| \geq n - n/2 = n/2. \tag{2}$$

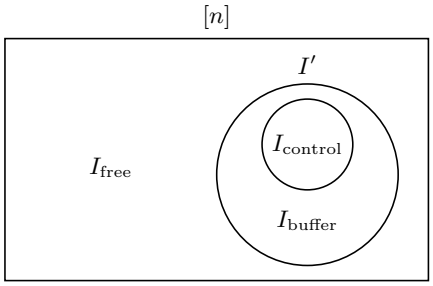


Figure 2 A schematic view of the sets of variables I_{free}, I_{buffer}, and I_{control}.

Computing a partial version of $\text{HW}_{n_{\text{free}}}$ from MAJ_n. We will split the input into three parts $x = x_{\text{free}} \circ x_{\text{buffer}} \circ x_{\text{control}}$, where $x_{\text{free}} \in \{0,1\}^{I_{\text{free}}}$, $x_{\text{buffer}} \in \{0,1\}^{I_{\text{buffer}}}$ and $x_{\text{control}} \in \{0,1\}^{I_{\text{control}}}$. First, let us fix a value for x_{buffer}. We will fix x_{buffer} to be some vector with exactly $|x_{\text{buffer}}| = \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$ ones. To make sure this is possible, we need to show $0 \leq \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \leq |I_{\text{buffer}}|$. Firstly, we have that $n - n_{\text{free}} = |I_{\text{buffer}}| + |I_{\text{control}}| \geq |I_{\text{control}}| = 2^{2Cq+1}$, so $\lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil \geq \left\lfloor \frac{n-n_{\text{free}}}{2} \right\rfloor \geq \left\lfloor \frac{2^{2Cq+1}}{2} \right\rfloor = 2^{2Cq}$, which gives the lower bound. Secondly, we have $|I_{\text{buffer}}| \geq |I'| - |I_{\text{control}}| \geq |I_{\text{control}}| + 1$ (since we made sure that $|I'| \geq 2|I_{\text{control}}| + 1$) and thus

$$|I_{\text{buffer}}| \geq \frac{|I_{\text{buffer}}| + |I_{\text{control}}| + 1}{2} = \frac{n - |I_{\text{free}}| + 1}{2} = \frac{n - n_{\text{free}} + 1}{2} \geq \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil,$$

which gives the upper bound.

Since each inner function $g_j(x)$ is either completely determined by x_{free} and x_{buffer} or completely determined by x_{buffer} and x_{control}, in order to compute $\text{MAJ}_n(x)$, it is enough to know $g_j(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$ for all $j \in [m]$ (where $0^{I_{\text{control}}} \in \{0,1\}^{I_{\text{control}}}$ is the all zeros string) as well as the value of x_{buffer} and x_{control}. So even if $g_j(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$ for $j \in [m]$ is all we know about x_{free}, we can compute the value of $\text{MAJ}_n(x_{\text{free}} \circ x_{\text{buffer}} \circ x_{\text{control}})$ for any x_{control} we desire. Thus, setting $|x_{\text{control}}|$ between 0 and $|I_{\text{control}}| = 2^{2Cq+1}$, we can figure out:

- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 0 \geq \lceil n/2 \rceil$;
- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 1 \geq \lceil n/2 \rceil$;
- ...
- whether $|x_{\text{free}}| + |x_{\text{buffer}}| + 2^{2Cq+1} \geq \lceil n/2 \rceil$.

Given that we set $|x_{\text{buffer}}| = \lceil n/2 \rceil - \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$, this means we can distinguish between the following cases:

- $|x_{\text{free}}| \geq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}$;
- $|x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1$;
- ...
- $|x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$;
- $|x_{\text{free}}| < \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}$.

Therefore, we can form a function $f(x_{\text{free}}) = h'(g_1(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}), \dots, g_m(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}))$ such that

$$f(x_{\text{free}}) = \begin{cases} \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} & \text{if } |x_{\text{free}}| \geq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} \\ \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1 & \text{if } |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} - 1 \\ \dots & \\ \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} & \text{if } |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \\ \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 & \text{if } |x_{\text{free}}| < \lceil n_{\text{free}}/2 \rceil - 2^{2Cq}. \end{cases}$$

Applying Corollary 27. This function f computes the Hamming weight correctly on the set

$$D = \left\{ x_{\text{free}} \in \{0,1\}^{I_{\text{free}}} \;\middle|\; \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 \leq |x_{\text{free}}| \leq \lceil n_{\text{free}}/2 \rceil + 2^{2Cq} \right\}.$$

We now prepare to apply Corollary 27 to f. Clearly, each input variable of f is queried by at most q of the inner functions $g_1(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}}), \dots, g_m(x_{\text{free}} \circ x_{\text{buffer}} \circ 0^{I_{\text{control}}})$. Let us see check that D satisfies the condition of Corollary 27. Observe that $|\boldsymbol{X}^{\oplus i}| = |\boldsymbol{X}| \pm 1$, so when $\lceil n_{\text{free}}/2 \rceil - 2^{2Cq} \leq |\boldsymbol{X}| < \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}$, we have $\boldsymbol{X}^{\oplus i} \in D$. Therefore,

$$\begin{aligned} \Pr_{\boldsymbol{X} \sim D}[\boldsymbol{X}^{\oplus i} \notin D] &\leq \Pr_{\boldsymbol{X} \sim D}[|\boldsymbol{X}| = \lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1 \vee |\boldsymbol{X}| = \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}] \\ &\leq \frac{2}{\#\{\lceil n_{\text{free}}/2 \rceil - 2^{2Cq} - 1, \dots, \lceil n_{\text{free}}/2 \rceil + 2^{2Cq}\}} \\ &\quad \text{(because the further } |\boldsymbol{X}| \text{ is from } n_{\text{free}}/2\text{, the fewer possible values there are for } \boldsymbol{X}\text{)} \\ &= \frac{2}{2^{2Cq+1} + 2} \\ &\leq 2^{-2Cq}. \end{aligned}$$

Thus we can apply Corollary 27 and obtain

$$m + (n_{\text{free}} - \log |D|) \geq n_{\text{free}} \cdot 2^{-O(q)}. \tag{3}$$

Now, we also have

$$\begin{aligned} \log |D| &\geq \log \left| \{ x_{\text{free}} \in \{0,1\}^{I_{\text{free}}} \mid |x_{\text{free}}| = \lceil n_{\text{free}}/2 \rceil \} \right| \\ &\geq \log(\Omega(2^{n_{\text{free}}}/\sqrt{n_{\text{free}}})) \\ &= n_{\text{free}} - O(\log n_{\text{free}}), \end{aligned}$$

so $n_{\text{free}} - \log |D| = O(\log n_{\text{free}}) = O(\log n)$. Plugging this into (3), we get

$$m + O(\log n) \geq n_{\text{free}} \cdot 2^{-O(q)}. \tag{4}$$

Asymptotics and conclusion. Since each inner function $g_1, \dots, g_m$ queries k variables and each variable is queried at most q times, we have $mk \leq nq$. Therefore,

$$\begin{aligned} \frac{nq}{k} + O(\log n) &\geq m + O(\log n) \\ &\geq n_{\text{free}} \cdot 2^{-O(q)} && \text{(by (4))} \\ &\geq \frac{n}{2} \cdot 2^{-O(q)}. && \text{(by (2))} \end{aligned}$$

Rearranging, we get $2^{O(q)}\left(\frac{q}{k} + \frac{O(\log n)}{n}\right) \geq 1/2$, so either

$$\frac{2^{O(q)}q}{k} \geq 1/4 \Rightarrow q = \Omega(\log k)$$

or

$$\frac{2^{O(q)}O(\log n)}{n} \geq 1/4 \Rightarrow q = \Omega(\log n) \geq \Omega(\log k).$$ ◀

5 Proof of Lemma 26

In this section we present the proof of our key lemma Lemma 26, which is a generalization of Lemma 22 to functions that only compute the Hamming weight correctly on a subset of the inputs.

▶ **Lemma 26** (key lemma, "partial functions" version). *Let $D \subseteq \{0,1\}^n$ be a non-empty subset of the hypercube (the "domain" on which f computes the Hamming weight), and let $f : \{0,1\}^n \to \{0,\ldots,n\}$ be a function such that $\forall x \in D$, $f(x) = |x|$. Suppose that $f = h(g_1,\ldots,g_m)$ and that variable i is queried by at most q of the inner functions $g_1,\ldots,g_m$. Let $\boldsymbol{X} \sim D$ be a random input drawn uniformly over D. Then $\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}$.*

We will start by proving a special case to build intuition, then prove the full version using a similar approach.

5.1 Warmup: $q = 1$ for total functions

In preparation for proving Lemma 26 in its full generality, let us prove the special case $q = 1$ of Lemma 22 (the "total functions" version), which is simpler and illustrates the core idea quite well.

▶ **Proposition 29** ("baby version" of Lemma 22). *Suppose that $\mathrm{HW}_n = h(g_1,\ldots,g_m)$ and that variable i is queried by only one of the inner functions $g_1,\ldots,g_m$. Let $\boldsymbol{X} \sim \{0,1\}^n$ be a uniformly random input. Then $\mathbf{I}[\boldsymbol{X}_i : g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})] = 1$. That is, the outputs of the inner functions $g_1(\boldsymbol{X}),\ldots,g_m(\boldsymbol{X})$ determine $\boldsymbol{X}_i$ completely.*

Proof. Say without loss of generality that the only inner function which queries variable i is g_1. Fix any input x. We will show how to recover x_i from the values $g_1(x),\ldots,g_m(x)$.

First, set aside $g_1(x)$, and consider what values $|x|$ could take if we knew only $g_2(x),\ldots,g_m(x)$. By our assumption $\mathrm{HW}_n = h(g_1,\ldots,g_m)$, $|x| = h(g_1(x),\ldots,g_m(x))$, so $|x|$ must belong to the set

$$W = \{h(0,g_2(x),\ldots,g_m(x)), h(1,g_2(x),\ldots,g_m(x))\}.$$

Now, consider the input $x^{\oplus i}$ (x with its i^{th} coordinate flipped). Since only g_1 depends on the i^{th} coordinate, $x^{\oplus i}$ must share the same output values for $g_2,\ldots g_m$. This means that *both* Hamming weights $|x|$ and $|x^{\oplus i}|$ must belong to the set W. But we know that $|x^{\oplus i}|$ is equal to either $|x|+1$ (when $x_i = 0$) or $|x|-1$ (when $x_i = 1$), and $|W| \leq 2$, so that means that W must be of the form

$$W = \{|x|, |x^{\oplus i}|\} = \{w, w+1\}$$

for some w.

So to find the value of x_i, it suffices to check whether $|x| = w$ (in which case $x_i = 0$) or $|x| = w+1$ (in which case $x_i = 1$). Since both W and $|x| = h(g_1(x), \ldots, g_m(x))$ can easily be computed from $g_1(x), \ldots, g_m(x)$, this gives us a way to recover x_i from $g_1(x), \ldots, g_m(x)$. ◀

5.2 General case

The proof of Lemma 26 is a generalization of the trick above: we fix the outputs of the inner functions that *do not* query variable i, then consider the (relatively small) set of possible values for $|\boldsymbol{X}|$ given those outputs, and finally use our knowledge of $|\boldsymbol{X}|$ to guess $\boldsymbol{X}_i$ better than random chance.

Proof of Lemma 26. First, let us show that the inequality holds when $q = 0$. If $q = 0$, then f does not depend on variable i at all, so for any $x \in \{0,1\}^n$, $f(x) = f(x^{\oplus i})$. This means we cannot simultaneously have $f(x) = |x|$ and $f(x^{\oplus i}) = |x^{\oplus i}|$. Thus whenever $x \in D$, $x^{\oplus i}$ cannot also be in D. As a result, $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] = 1$, so the right hand side becomes

$$1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)} = 1 + 1 - 1 = 1,$$

and the inequality is trivially verified. Having proved the inequality for $q = 0$, we may assume that $q \geq 1$ for the remainder of the proof.

Let $J_i \subseteq [m]$ be the set of inner functions that depend on variable i (so $|J_i| \leq q$), and let $\overline{J_i} \coloneqq [m] \setminus J_i$. Using the fact that $g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})$ determine $|\boldsymbol{X}|$ completely, we can weaken the conditioning to get

$$\mathbf{H}[\boldsymbol{X}_i \mid g_1(\boldsymbol{X}), \ldots, g_m(\boldsymbol{X})] \leq \mathbf{H}[\boldsymbol{X}_i \mid \{g_j(\boldsymbol{X})\}_{j \in \overline{J_i}}, |\boldsymbol{X}|],$$

and by definition of conditional entropy,

$$\begin{aligned}
&\mathbf{H}[\boldsymbol{X}_i \mid \{g_j(\boldsymbol{X})\}_{j \in \overline{J_i}}, |\boldsymbol{X}|] \\
&\quad = \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[-\log \mathbf{Pr}[\boldsymbol{X}_i = \boldsymbol{Y}_i \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]] \\
&\quad = \mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])],
\end{aligned}$$

where $H_2(p) \coloneqq -p \log p - (1-p)\log(1-p)$ is the binary entropy function. Thus, to prove the lemma, it suffices to show that

$$\mathop{\mathbf{E}}_{\boldsymbol{Y} \sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \tag{5}$$

Partitioning D according to the outputs of the inner functions that do not query the variable i. To prove this inequality, let us first split into cases according to the outputs of the inner functions that do not query variable i. Fix the output values $v \in \{0,1\}^{\overline{J_i}}$ of the inner functions that do not query variable i, and suppose that those output values are achievable by some input in D (that is, there exists some input $x \in D$ such that $\forall j \in \overline{J_i}, g_j(x) = v_j$). Let $S_v \coloneqq \{x \in \{0,1\}^n \mid \forall j \in \overline{J_i}, g_j(x) = v_j\}$ be the set of inputs that produce these inner function outputs. Note that the sets $D \cap S_v$ form a partition of D. As we will see later, a key property of S_v is that its image under f has cardinality at most 2^q.

Let us call v *good* if $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] \leq 2 \cdot \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D]$ (that is, v is good if $\boldsymbol{X}^{\oplus i}$ is not much more likely to lie outside of D when $\boldsymbol{X}$ is in S_v than on average). By Markov's inequality, the sets S_v with good v account for most of the probability mass: that is, $\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \geq 1/2$.

Our objective will be to show that for every good v,

$$\mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v] \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \quad (6)$$

Informally, the task is: for any good $v \in \{0,1\}^{\overline{J_i}}$ (representing the outputs of the inner functions that do not query variable i), based only on v and the Hamming weight $|\boldsymbol{X}|$, guess the value of $\boldsymbol{X}_i$ slightly better than random chance.

Let us see why (6) implies (5):

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid (\forall j \in \overline{J_i}, g_j(\boldsymbol{X}) = g_j(\boldsymbol{Y})) \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|])] \\
&\quad = \sum_v \mathbf{Pr}[\boldsymbol{X} \in S_v] \cdot \mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v] \\
&\qquad\qquad \text{(by the law of total expectation)} \\
&\quad \leq \sum_v \mathbf{Pr}[\boldsymbol{X} \in S_v] \cdot \begin{cases} 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)} & \text{if } v \text{ is good} \\ 1 & \text{otherwise} \end{cases} \\
&\qquad\qquad \text{(by (6) and because } H_2(\cdot) \leq 1) \\
&\quad = 1 + \left(\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v]\right)\left(\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}\right) \\
&\quad \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - \left(\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v]\right) 2^{-O(q)} \\
&\qquad\qquad \text{(because } \textstyle\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \leq 1) \\
&\quad \leq 1 + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] - 2^{-O(q)}. \qquad \text{(because } \textstyle\sum_{v \text{ good}} \mathbf{Pr}[\boldsymbol{X} \in S_v] \geq 1/2 \text{ and } q > 0)
\end{aligned}$$

Proof overview for (6). Fix some good $v \in \{0,1\}^{\overline{J_i}}$, and let $W_v \coloneqq \{|x| \mid x \in D \cap S_v\}$ be the set of possible Hamming weights for inputs in $D \cap S_v$. When $x \in D$, we know that $|x| = f(x) = h(g_1(x), \ldots, g_m(x))$, and when in addition $x \in S_v$, then for all $j \in \overline{J_i}$, the output value $g_j(x)$ is already fixed to v_j. Therefore, when $x \in D \cap S_v$, the Hamming weight $|x|$ can only depend on the remaining q output values $g_j(x)$ for $j \in J_i$, so there are at most 2^q possible values for $|x|$. In other words, $|W_v| \leq 2^q$.

Informally, using the fact that W_v is small, we will show that there exists a Hamming weight $w^* \in W_v$ that occurs with significant probability within S_v, and such that conditioned on $|\boldsymbol{X}| = w^*$, the probability that $\boldsymbol{X}_i = 1$ is not too close to $1/2$. This means that when $|\boldsymbol{X}| = w^*$ we can guess $\boldsymbol{X}_i$ slightly better than random chance, and even if we just guess randomly when $|\boldsymbol{X}| \neq w^*$, we will have achieved better-than-random-chance accuracy overall. Formally, we will show that $\exists w^* \in W_v$ such that

$$\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid x \in S_v] \geq 2^{-O(q)} \quad (7)$$

and

$$\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*] \leq \frac{1}{2} - 2^{-O(q)}. \quad (8)$$

Let us see why (7) and (8) together imply (6):

$$
\begin{aligned}
&\mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v] \\
&\quad = \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \\
&\qquad\qquad \cdot \mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = |\boldsymbol{Y}|]) \mid \boldsymbol{Y} \in S_v \wedge |\boldsymbol{Y}| = w] \\
&\qquad\qquad\qquad\qquad \text{(by the law of total expectation)} \\
&\quad = \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \\
&\qquad\qquad \cdot \mathop{\mathbf{E}}_{\boldsymbol{Y}\sim D}[H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w]) \mid \boldsymbol{Y} \in S_v \wedge |\boldsymbol{Y}| = w] \\
&\qquad\qquad\qquad\qquad \text{(replace } |\boldsymbol{Y}| \text{ by } w \text{ using the conditioning)} \\
&\quad = \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \cdot H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w]) \\
&\qquad\qquad\qquad\qquad \text{(the expectation was constant)} \\
&\quad \le \sum_{w \in W_v} \mathbf{Pr}[|\boldsymbol{X}| = w \mid \boldsymbol{X} \in S_v] \cdot \begin{cases} H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*]) & \text{if } w = w^* \\ 1 & \text{otherwise} \end{cases} \\
&\qquad\qquad\qquad\qquad \text{(because } H_2(\cdot) \le 1) \\
&\quad = 1 - \mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] \cdot (1 - H_2(\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*])) \\
&\qquad\qquad\qquad\qquad \text{(rearrange and use the fact that probabilities sum to 1)} \\
&\quad \le 1 - 2^{-O(q)}\big(1 - H_2\big(1/2 - 2^{-O(q)}\big)\big) \qquad \text{(by (7) and (8))} \\
&\quad = 1 - 2^{-O(q)}\big(1 - \big(1 - 2^{-O(q)}\big)\big) \qquad \text{(because } H_2(1/2 - p) = 1 - \Omega(p^2)\text{)} \\
&\quad = 1 - 2^{-O(q)}.
\end{aligned}
$$

Existence of the weight w^*. Let us now find this magical Hamming weight w^*.

First of all, assume that $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D] \le 2^{-10q}$ (otherwise, we can replace the right-hand side of (6) by 1 and the inequality becomes trivial). Since v is good, this means that $\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] \le 2 \cdot 2^{-10q} = 2^{-10q+1}$.

When $x \in S_v$, $x^{\oplus i}$ must also be in S_v, since flipping variable i does not affect the output of any inner function g_j for $j \in \overline{J_i}$. Informally, this means that for most inputs $x \in D \cap S_v$ (those for which $x^{\oplus i} \in D$), we can pair it up with another input $x^{\oplus i}$ also in $D \cap S_v$. We can identify each such pair by looking at the value of $x^{(i \mapsto 0)}$ (x with its i^{th} coordinate set to 0). Our argument will rely crucially on this pairing of inputs.

For any integer $-1 \le w \le n$, let $p_w \coloneqq \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge |\boldsymbol{X}^{(i \mapsto 0)}| = w \mid \boldsymbol{X} \in S_v]$ (note that $p_{-1} = 0$ somewhat vacuously). It is clear that $p_w = 0$ whenever $w, w+1 \notin W_v$. This means that the sequence $\{p_w\}_{w \in \{-1,\dots,n\}}$ has at most $2|W_v| \le 2^{q+1}$ nonzero elements. Moreover $\sum_{w=-1}^{n} p_w = \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \mid \boldsymbol{X} \in S_v] \ge 1 - 2^{-10q+1}$, so we must have $\max_{w=-1}^{n} p_w \ge (1 - 2^{-10q+1})/2^{q+1} \ge 2^{-q-2}$. Now, again using the fact that the sequence $\{p_w\}_{w \in \{-1,\dots,n\}}$ has at most 2^{q+1} nonzero elements, to go from $p_{-1} = 0$ to this maximum value of $\ge 2^{-q-2}$, the sequence must contain some "upwards jump" of at least $2^{-q-2}/2^{q+1} = 2^{-2q-3}$, so there must exist a weight $w^* \in \{0, \dots, n\}$ such that $p_{w^*} \ge p_{w^*-1} + 2^{-2q-3}$.

Let us show that $w^* \in W_v$ and that it satisfies (7) and (8). First, note that the set $\{x \in D \cap S_v \mid |x| = w^*\}$ includes at least the following two disjoint sets:

- **(i)** the inputs $x \in D \cap S_v$ such that $x^{\oplus i} \in D$, $x_i = 0$, and $|x^{(i \mapsto 0)}| = w^*$;
- **(ii)** the inputs $x \in D \cap S_v$ such that $x^{\oplus i} \in D$, $x_i = 1$ and $|x^{(i \mapsto 0)}| = w^* - 1$.

It is easy to see that there are exactly $\frac{p_{w^*}}{2}|D \cap S_v|$ inputs of type (i) and $\frac{p_{w^*-1}}{2}|D \cap S_v|$ inputs of type (ii). This means that

$$\begin{aligned}
\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] &\geq \frac{p_{w^*} + p_{w^*-1}}{2} && (9)\\
&\geq \frac{p_{w^*}}{2} && \text{(because } p_{w^*-1} \text{ is a probability)}\\
&\geq \frac{2^{-2q-3}}{2} && \text{(because } p_{w^*} \geq p_{w^*-1} + 2^{-2q-3} \geq 2^{-2q-3}\text{)}\\
&= 2^{-O(q)},
\end{aligned}$$

satisfying (7). Also, given that $\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] \geq 2^{-O(q)} > 0$, there must be some $x \in D \cap S_v$ such that $|x| = w^*$, so $w^* \in W_v$. Finally,

$$\begin{aligned}
&\mathbf{Pr}[\boldsymbol{X}_i = 1 \mid \boldsymbol{X} \in S_v \wedge |\boldsymbol{X}| = w^*]\\
&\quad= \frac{\mathbf{Pr}[\boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]} && \text{(by definition of conditional probability)}\\
&\quad= \frac{\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v] + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}\\
&\qquad\qquad && \text{(split according to } \boldsymbol{X}^{\oplus i} \stackrel{?}{\in} D\text{)}\\
&\quad\leq \frac{\mathbf{Pr}[\boldsymbol{X}^{\oplus i} \notin D \mid \boldsymbol{X} \in S_v] + \mathbf{Pr}[\boldsymbol{X}^{\oplus i} \in D \wedge \boldsymbol{X}_i = 1 \wedge |\boldsymbol{X}^{(i\mapsto 0)}| = w^* - 1 \mid \boldsymbol{X} \in S_v]}{\mathbf{Pr}[|\boldsymbol{X}| = w^* \mid \boldsymbol{X} \in S_v]}\\
&\qquad\qquad && \text{(logical consequences)}\\
&\quad\leq \frac{2^{-10q+1} + \frac{p_{w^*-1}}{2}}{\frac{p_{w^*}+p_{w^*-1}}{2}} && \text{(because there are } \tfrac{p_{w^*-1}}{2}|D \cap S_v| \text{ inputs of type (ii), and by (9))}\\
&\quad= \frac{2^{-10q+2} + p_{w^*-1}}{p_{w^*} + p_{w^*-1}}\\
&\quad\leq \frac{2^{-10q+2} + (p_{w^*} - 2^{-2q-3})}{p_{w^*} + (p_{w^*} - 2^{-2q-3})} && \text{(because } p_{w^*} \geq 2^{-2q-3} > 2^{10q+2} \text{ and } p_{w^*-1} \leq p_{w^*} - 2^{-2q-3}\text{)}\\
&\quad= \frac{1}{2} - \frac{2^{-2q-3}/2 - 2^{-10q+2}}{2p_{w^*} - 2^{-2q-3}}\\
&\quad\leq \frac{1}{2} - \frac{2^{-2q-5}}{2p_{w^*} - 2^{-2q-3}} && \text{(because } q \geq 1\text{)}\\
&\quad< \frac{1}{2} - \frac{2^{-2q-5}}{2} && \text{(because } p_{w^*} \leq 1\text{)}\\
&\quad= \frac{1}{2} - 2^{-O(q)}, && \text{(because } q \geq 1\text{)}
\end{aligned}$$

thus (8) is satisfied, and this concludes the proof. ◀

6 Conclusion

In this section we outline several interesting directions for future work.

6.1 A plan towards better depth-3 lower bounds

As we pointed out in the introduction, if it were possible to compute $\textsc{Maj}_n$ as a fan-in-$O(\sqrt{n})$ function of fan-in-$O(\sqrt{n})$ functions, there would be $2^{O(\sqrt{n})}$-size depth-3 circuits for $\textsc{Maj}_n$. By showing that $\mathsf{CC}_{\Theta(\sqrt{n})}(\textsc{Maj}_n) = \omega(\sqrt{n})$, we ruled out this particular way of obtaining $2^{O(\sqrt{n})}$-size depth-3 circuits for $\textsc{Maj}_n$. Theorem 2 is therefore a necessary first step for showing that $\textsc{Maj}_n$ requires $2^{\omega(\sqrt{n})}$-size depth-3 circuits.

If we keep considering constructions of the form $\textsc{Maj}_n = h(g_1, \ldots, g_m)$, there are several ways we can make both the outer function h and the inner functions $g_1, \ldots, g_m$ more powerful while still giving depth-3 circuit upper bounds. For example, one could try to prove that

$\textsc{Maj}_n$ cannot be computed as a fan-in-$O(\sqrt{n})$ function of depth-$O(\sqrt{n})$ decision trees, or even as a depth-$O(\sqrt{n})$ decision tree of depth-$O(\sqrt{n})$ decision trees. Indeed, if such a construction were possible, the outer function could be transformed into a size-$2^{O(\sqrt{n})}$ DNF (resp. CNF), while the inner functions and their negations could be transformed into size-$2^{O(\sqrt{n})}$ CNFs (resp. DNFs), which would give a $2^{O(\sqrt{n})}$-size Σ^3 (resp. Π^3) circuit for $\textsc{Maj}_n$.

This motivates the following plan towards proving better depth-3 lower bounds for the majority function: prove lower bounds against computing $\textsc{Maj}_n$ as $h(g_1, \ldots, g_m)$, where the inner and outer functions are replaced by increasingly powerful objects. For now, let us conjecture the following first steps, which could be solved independently.

▶ **Conjecture 30** (making the inner functions more powerful)**.** *$\textsc{Maj}_n$ cannot be represented as $h(g_1, \ldots, g_m)$, where $m = O(\sqrt{n})$ and each g_j is a depth-$O(\sqrt{n})$ decision tree.*

▶ **Conjecture 31** (making the outer function more powerful)**.** *$\textsc{Maj}_n$ cannot be represented as $h(g_1, \ldots, g_m)$, where h is a depth-$O(\sqrt{n})$ decision tree and each g_j depends on only $O(\sqrt{n})$ variables.*

In fact, we do not have to climb very far up the power ladder in order to reach the full power of depth-3 circuits. Indeed, if we were to replace "depth-$O(\sqrt{n})$ decision tree" by "size-$2^{O(\sqrt{n})}$ DNF" (resp. CNF) in Conjecture 31, this would already be equivalent to the conjecture that $\textsc{Maj}_n$ cannot be represented by Σ^3 (resp. Π^3) circuits of size $2^{O(\sqrt{n})}$.[8]

6.2 Further applications of our techniques

An information theoretic toolbox for circuit lower bounds?

In this paper, we showed (in Lemma 22 and Lemma 26) how we can trace the information flow within a circuit of arbitrary functions to prove tight lower bounds. Could one prove lower bounds against boolean circuits via the same technique? In particular, it would be interesting to prove a statement analogous to Lemma 22, but quantifying the mutual information between internal nodes of a boolean circuit.

One challenge of accomplishing this is in the difference in power between the inner functions and the AND/OR gates of boolean circuits. Since the boolean gates are so much weaker than the inner functions (which compute arbitrary boolean functions), we would need to prove much stronger statements about the information flow in order to get tight lower bounds. Nonetheless, we believe that challenges like Open Problem 1 require techniques that can precisely identify the weaknesses of small-depth circuits, and that information theory is a well-suited tool for this task.

The usefulness of multi-output functions

Multi-output functions played a key role in two steps of the proof. These steps can be distilled as a general plan for proving lower bounds for single-output functions as follows:

(1) Prove a lower bound for a multi-output function (in our case, HW_n).
(2) Show that the act of computing a desired single-output function (in our case, $\textsc{Maj}_n$) essentially entails computing the multi-output function (perhaps on a smaller set of inputs).

[8] In fact, this would still be true even if we replaced "each g_j depends only on $O(\sqrt{n})$ variables" by "each g_j is an OR of $O(\sqrt{n})$ variables".

We believe that this general approach could be applied to a much wider range of models. For example, consider the problem of improving the best lower bounds for depth-3 circuits. If one could prove a $2^{\omega(\sqrt{n})}$ lower bound against depth-3 circuits of HW_n, then this would *automatically* give us better lower bounds for an explicit function (we would get a lower bound of $2^{\omega(\sqrt{n})}/\log n = 2^{\omega(\sqrt{n})}$ for at least one of the output bits of HW_n). In all likelihood, this could be extended (perhaps by the approach in step (2)) to work for $\textsc{Maj}_n$, thereby resolving Open Problem 1.

We usually think of single-output functions as being components of multi-output functions, but this work shows that multi-output functions can be hidden within single-output functions as well. We hope that this will encourage the incorporation of multi-output functions into the lexicon of hard functions to prove lower bounds for.

Random linear codes

As a concrete example of functions that one might prove strong lower bounds for using our techniques, we propose random linear error-correcting codes.

In a celebrated work, Paturi et al. [23] showed that error-correcting codes[9] with large enough distance and enough codewords require size-$2^{1.282\sqrt{n}}$-size Σ^3 circuits, which remains to date the strongest lower bound proved for any explicit function. A natural question is, was this technique tight? Can we obtain $2^{\omega(\sqrt{n})}$ lower bounds for all "good enough" error-correcting codes?

Unfortunately, the answer is at least partially negative. Leffman, Pudlák, and Savický [19] showed that there exist linear codes of large distance which have *sparse* parity check matrices. More precisely, they show the existence of a linear code f of distance $n^{\Omega(1)}$ whose parity check matrix has $\sqrt{n}$ rows, each of which has only $O(\sqrt{n})$ non-zero entries. This means that we can write $f = h(g_1, \ldots, g_{\sqrt{n}})$, where each of the inner functions g_j computes a parity check over $O(\sqrt{n})$ variables, and h is the AND function. Using the notation of this paper, $\mathsf{CC}_{O(\sqrt{n})}(f) \leq \sqrt{n}$, and thus f has size-$2^{O(\sqrt{n})}$ Σ^3 circuits.

However, this code has small depth-3 circuits precisely because its parity check matrix is sparse. It is easy to see by a counting argument that most linear codes do *not* have sparse parity check matrices. So it is natural to wonder whether a random linear code (a code whose parity check matrix is chosen uniformly at random) might require large Σ^3 circuits. Concretely, we conjecture the following.

▶ **Conjecture 32.** *Let $\ell \ll k \ll n$. Let $H \sim \{0,1\}^{\ell \times n}$ be a random matrix of ℓ parity checks, and let*

$$f : \{0,1\} \to \{0,1\}^n : x \mapsto \begin{cases} 1 & \text{if } Hx = \vec{0} \\ 0 & \text{otherwise} \end{cases}$$

be the corresponding linear code. Then, with high probability,

(i) $\mathsf{CC}_k(f) \geq \Omega\left(\frac{n}{k} \cdot \frac{\ell}{\log(n/k)}\right)$*;*

(ii) *f requires Σ^3 circuits of size $2^{\Omega(\sqrt{n\ell/\log n})}$.*

The proposed bound $\Omega(\frac{n}{k} \cdot \frac{\ell}{\log(n/k)})$ in point (i) comes from the observation that one can verify any $\log(n/k)$ parity check using $O(n/k)$ inner functions: first split the variables into $O(n/k)$ sets according to their coefficients in each of the $\log(n/k)$ parity check, then compute the parity of each of those $O(n/k)$ sets. Point (i) simply conjectures that this observation

[9] More precisely, functions computing whether their input x belongs to a fixed error-correcting code.

gives the best upper bound on $\mathsf{CC}_k(f)$. The proposed bound $2^{\Omega(\sqrt{n\ell/\log n})}$ in point (ii) is obtained by conjecturing that the best Σ^3 circuit size for f will be obtained as $2^{O(\mathsf{CC}_k(f)+k)}$ through the connection with composition complexity (see Section 1.2.2), then balancing the sum by setting $k \coloneqq \sqrt{n\ell/\log n}$.

If Conjecture 32 is true, it would give lower bounds of the form $2^{n^{1-\epsilon}}$ against Σ^3 circuits. It seems likely to us that, depending on how the proof works, the randomness could then be lifted, and it could be extended to some explicit *pseudorandom* linear code.

We think the techniques in our paper would be particularly well-suited to proving Conjecture 32. Indeed, just like majority has a natural multi-output analog (the Hamming weight function), f has an even more obvious corresponding multi-output function: the function $\vec{f}(x) \coloneqq Hx$ which gives the outputs of all the parity checks. Therefore, it seems plausible that one could first prove a lower bound for $\vec{f}$ using techniques similar to the ones in Section 3, then extend it to f using our framework for bootstrapping lower bounds from multi-output functions to binary-output functions, which we presented in Section 4.

References

1. Miklós Ajtai. Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.
2. Noga Alon and Wolfgang Maass. Meanders, ramsey theory and lower bounds for branching programs. In *27th Annual Symposium on Foundations of Computer Science (SFCS 1986)*, pages 410–417. IEEE, 1986.
3. Kazuyuki Amano and Masafumi Yoshida. Depth two (n-2)-majority circuits for n-majority. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(9):1543–1545, 2018.
4. László Babai, Pavel Pudlák, Vojtech Rödl, and Endre Szemerédi. Lower bounds to the complexity of symmetric boolean functions. *Theoretical Computer Science*, 74(3):313–323, 1990.
5. Ravi B Boppana. The average sensitivity of bounded-depth circuits. *Information processing letters*, 63(5):257–261, 1997.
6. Christian Engels, Mohit Garg, Kazuhisa Makino, and Anup Rao. On expressing majority as a majority of majorities. *SIAM Journal on Discrete Mathematics*, 34(1):730–741, 2020.
7. Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *Proceedings of the 22nd IEEE Annual Symposium on Foundations of Computer Science*, pages 260–270, 1981.
8. 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.
9. Oded Goldreich and Avishay Tal. Matrix rigidity of random toeplitz matrices. *computational complexity*, 27(2):305–350, 2018.
10. Oded Goldreich and Avishay Tal. On constant-depth canonical boolean circuits for computing multilinear functions. In *Computational Complexity and Property Testing*, pages 306–325. Springer, 2020.
11. Oded Goldreich and Avi Wigderson. On the size of depth-three boolean circuits for computing multilinear functions. In *Computational Complexity and Property Testing*, pages 41–86. Springer, 2020.
12. Johan Håstad. *Computational Limitations for Small Depth Circuits*. MIT Press, Cambridge, MA, 1986.

13 Johan Hastad, Stasys Jukna, and Pavel Pudlák. Top-down lower bounds for depth 3 circuits. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 124–129. IEEE, 1993.

14 Pavel Hrubes, Sivaramakrishnan Natarajan Ramamoorthy, Anup Rao, and Amir Yehudayoff. Lower bounds on balancing sets and depth-2 threshold circuits. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

15 Pavel Hrubes and Anup Rao. Circuits with medium fan-in. In *30th Conference on Computational Complexity (CCC 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

16 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 5. Springer, 2012.

17 Maria Klawe, Wolfgang J Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 480–487, 1984.

18 Alexander S Kulikov and Vladimir V Podolskii. Computing majority by constant depth majority circuits with low fan-in gates. *Theory of Computing Systems*, 63(5):956–986, 2019.

19 Hanno Lefmann, Pavel Pudlák, and Petr Savicky. On sparse parity check matrices. *Designs, Codes and Cryptography*, 12(2):107–130, 1997.

20 Or Meir and Avi Wigderson. Prediction from partial information and hindsight, with application to circuit lower bounds. *computational complexity*, 28(2):145–183, 2019.

21 Edward I Nechiporuk. A boolean function. *Engl. transl. in Sov. Phys. Dokl.*, 10:591–593, 1966.

22 Ilan Newman and Avi Wigderson. Lower bounds on formula size of boolean functions using hypergraph entropy. *SIAM Journal on Discrete Mathematics*, 8(4):536–542, 1995.

23 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.

24 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 566–574. IEEE, 1997.

25 Gleb Posobin. Computing majority with low-fan-in majority queries. *arXiv preprint*, 2017. `arXiv:1711.10176`.

26 Alexander A Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *International Symposium on Fundamentals of Computation Theory*, pages 47–60. Springer, 1991.

27 Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

28 Benjamin Rossman. The average sensitivity of bounded-depth formulas. *computational complexity*, 27(2):209–223, 2018.

29 Srikanth Srinivasan. Personal communication, 2015.

30 Leslie G Valiant. Exponential lower bounds for restricted monotone circuits. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 110–117, 1983.

31 Guy Wolfovitz. The complexity of depth-3 circuits computing symmetric boolean functions. *Information Processing Letters*, 100(2):41–46, 2006.

32 Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 1–10, 1985.

The Acrobatics of BQP

Scott Aaronson
University of Texas at Austin, TX, USA

DeVon Ingram
University of Chicago, IL, USA

William Kretschmer
University of Texas at Austin, TX, USA

Abstract

One can fix the randomness used by a randomized algorithm, but there is no analogous notion of fixing the quantumness used by a quantum algorithm. Underscoring this fundamental difference, we show that, in the black-box setting, the behavior of quantum polynomial-time (BQP) can be remarkably decoupled from that of classical complexity classes like NP. Specifically:

- There exists an oracle relative to which $\mathsf{NP}^{\mathsf{BQP}} \not\subset \mathsf{BQP}^{\mathsf{PH}}$, resolving a 2005 problem of Fortnow. As a corollary, there exists an oracle relative to which $\mathsf{P} = \mathsf{NP}$ but $\mathsf{BQP} \neq \mathsf{QCMA}$.
- Conversely, there exists an oracle relative to which $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{BQP}}$.
- Relative to a random oracle, PP is not contained in the "QMA hierarchy" $\mathsf{QMA}^{\mathsf{QMA}^{\mathsf{QMA}^{\cdots}}}$.
- Relative to a random oracle, $\Sigma_{k+1}^{\mathsf{P}} \not\subset \mathsf{BQP}^{\Sigma_k^{\mathsf{P}}}$ for every k.
- There exists an oracle relative to which $\mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$ and yet PH is infinite. (By contrast, relative to all oracles, if $\mathsf{NP} \subseteq \mathsf{BPP}$, then PH collapses.)
- There exists an oracle relative to which $\mathsf{P} = \mathsf{NP} \neq \mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$.

To achieve these results, we build on the 2018 achievement by Raz and Tal of an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{PH}$, and associated results about the Forrelation problem. We also introduce new tools that might be of independent interest. These include a "quantum-aware" version of the random restriction method, a concentration theorem for the block sensitivity of AC^0 circuits, and a (provable) analogue of the Aaronson-Ambainis Conjecture for sparse oracles.

2012 ACM Subject Classification Theory of computation → Quantum complexity theory; Theory of computation → Complexity classes

Keywords and phrases BQP, Forrelation, oracle separations, Polynomial Hierarchy, query complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.20

Related Version *Full Version*: https://arxiv.org/abs/2111.10409 [8]

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.
DeVon Ingram: Supported by an NSF Graduate Research Fellowship.
William Kretschmer: Supported by an NDSEG Fellowship.

Acknowledgements We thank Lance Fortnow, Greg Kuperberg, Patrick Rall, and Avishay Tal for helpful conversations. We are especially grateful to Avishay Tal for helping us prove a tail bound on the block sensitivity of AC^0 circuits.

1 Introduction

The complexity-theoretic study of quantum computation is often dated from 1993, when Bernstein and Vazirani [15] defined BQP, or Bounded-Error Quantum Polynomial-Time: the class of languages that admit efficient quantum algorithms. Then as now, a central concern was how BQP relates to classical complexity classes, such as P, NP, and PH. Among the countless questions that one could raise here, let us single out three as especially fundamental:

(1) Can quantum computers efficiently solve any problems that classical computers cannot? In other words, does $\mathsf{BPP} = \mathsf{BQP}$?
(2) Can quantum computers solve NP-complete problems in polynomial time? In other words, is $\mathsf{NP} \subseteq \mathsf{BQP}$?
(3) What is the best classical upper bound on the power of quantum computation? Is $\mathsf{BQP} \subseteq \mathsf{NP}$? Is $\mathsf{BQP} \subseteq \mathsf{PH}$?

Three decades later, all three of these still stand as defining questions of the field. Nevertheless, from the early 2000s onwards, it became rare for work in quantum computing theory to address any of these questions directly, perhaps simply because it became too hard to say anything new about them. A major recent exception was the seminal work of Raz and Tal [38], who gave an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{PH}$, by completing a program proposed by one of us [2]. In this paper, we take the Raz-Tal breakthrough as a starting point. Using it, together with new tools that we develop, we manage to prove many new theorems about the power of BQP – at least in the black-box setting where much of our knowledge of quantum algorithms resides.

Before discussing the black-box setting or Raz-Tal, though, let's start by reviewing what is known in general about BQP. Bernstein and Vazirani [15] showed that $\mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{P}^{\#\mathsf{P}}$, and Adleman, DeMarrais, and Huang [10] improved the upper bound to $\mathsf{BQP} \subseteq \mathsf{PP}$, giving us the following chain of inclusions:

$$\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PP} \subseteq \mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}.$$

Fortnow and Rogers [21] slightly strengthened the inclusion $\mathsf{BQP} \subseteq \mathsf{PP}$, to show for example that $\mathsf{PP}^{\mathsf{BQP}} = \mathsf{PP}$. This complemented the result of Bennett, Bernstein, Brassard, and Vazirani [14] that $\mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$: that is, BQP is "self-low," or "the BQP hierarchy collapses to BQP."

1.1 The Contrast with BPP

Meanwhile, though, the relationships between BQP and complexity classes like NP, PH, and $\mathsf{P/poly}$ have remained mysterious. Besides the fundamental questions mentioned above – is $\mathsf{NP} \subseteq \mathsf{BQP}$? is $\mathsf{BQP} \subseteq \mathsf{NP}$? is $\mathsf{BQP} \subseteq \mathsf{PH}$? – one could ask other questions:

(i) In a 2005 blog post, Fortnow [20] raised the question of whether $\mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{NP}}$. Do we even have $\mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{PH}}$? I.e., when quantum computation is combined with classical nondeterminism, how does the order of combination matter?
(ii) What about the converse: is $\mathsf{BQP}^{\mathsf{NP}} \subseteq \mathsf{PH}^{\mathsf{BQP}}$?
(iii) Suppose $\mathsf{NP} \subseteq \mathsf{BQP}$. Does it follow that $\mathsf{PH} \subseteq \mathsf{BQP}$ as well?
(iv) Suppose $\mathsf{NP} \subseteq \mathsf{BQP}$. Does it follow that PH collapses?
(v) Is $\mathsf{BQP} \subset \mathsf{P/poly}$?
(vi) Suppose $\mathsf{P} = \mathsf{NP}$. Does it follow that BQP is "small" (say, not equal to EXP)?
(vii) Suppose $\mathsf{P} = \mathsf{NP}$. Does it follow that $\mathsf{BQP} = \mathsf{QCMA}$, where QCMA (Quantum Classical Merlin Arthur) is the analogue of NP with a BQP verifier?

What is particularly noteworthy about the questions above is that, if we replace BQP by BPP, then positive answers are known to all of them:

(i) $\mathsf{NP}^{\mathsf{BPP}} \subseteq \mathsf{AM} \subseteq \mathsf{BPP}^{\mathsf{NP}}$.
(ii) $\mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{PH} = \mathsf{PH}^{\mathsf{BPP}}$.
(iii) If $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\mathsf{PH} = \mathsf{BPP}$ – this is sometimes given as a homework exercise in complexity theory courses, and also follows from (i).

(iv) If $\mathsf{NP} \subseteq \mathsf{BPP}$, then $\mathsf{PH} = \Sigma_2^{\mathsf{P}}$ – this follows from (iii) and the Sipser-Lautemann Theorem [45, 34].
(v) $\mathsf{BPP} \subset \mathsf{P/poly}$ is Adleman's Theorem [9].
(vi) If $\mathsf{P} = \mathsf{NP}$, then $\mathsf{P} = \mathsf{BPP}$ and hence $\mathsf{BPP} \neq \mathsf{EXP}$, by the time hierarchy theorem.
(vii) If $\mathsf{P} = \mathsf{NP}$, then of course $\mathsf{BPP} = \mathsf{MA}$.

So what is it that distinguishes BPP from BQP in these cases? In all of the above examples, the answer turns out to be one of the fundamental properties of classical randomized algorithms: namely, that one can always "pull the randomness out" from such algorithms, viewing them as simply deterministic algorithms that take a uniform random string r as an auxiliary input, in addition to their "main" input x. This, in turn, enables one to play all sorts of tricks with such an algorithm $M(x, r)$ – from using approximate counting to estimate the fraction of r's that cause $M(x, r)$ to accept, to moving r from inside to outside a quantifier, to hardwiring r as advice. By contrast, there is no analogous notion of "pulling the randomness (or quantumness) out of a quantum algorithm." In quantum computation, randomness is just an intrinsic part of the model that rears its head at the *end* (rather than the beginning) of a computation, when we take the squared absolute values of amplitudes to get probabilities.

This difference between randomized and quantum algorithms is crucial to the analysis of the so-called "sampling-based quantum supremacy experiments" – for example, those recently carried out by Google [11] and USTC [49]. The theoretical foundations of these experiments were laid a decade ago, in the work of Aaronson and Arkhipov [5] on BosonSampling, and (independently) Bremner, Jozsa, and Shepherd [19] on the commuting Hamiltonians or IQP model. Roughly speaking, the idea is that, by using a quantum computer, one can efficiently sample a probability distribution $\mathcal{D}$ over n-bit strings such that even *estimating* the probabilities of the outcomes is a #P-hard problem. Meanwhile, though, if there were a polynomial-time classical randomized algorithm $M(x, r)$ to sample from the same distribution $\mathcal{D}$, then one could use the "pulling out r" trick to estimate the probabilities of M's outcomes in PH. But this would put $\mathsf{P}^{\#\mathsf{P}}$ into PH, thereby collapsing PH by Toda's Theorem [47].

More generally, with any of the apparent differences between quantum algorithms and classical randomized algorithms, the question is: how can we prove that the difference is genuine, that no trick will ever be discovered that makes BQP behave more like BPP? For questions like whether $\mathsf{NP} \subseteq \mathsf{BQP}$ or whether $\mathsf{BQP} \subseteq \mathsf{NP}$, the hard truth here is that not only have we been unable to resolve these questions in the unrelativized world, we've been able to say little more about them than certain "obvious" implications. For example, suppose $\mathsf{NP} \subseteq \mathsf{BQP}$ *and* $\mathsf{BQP} \subseteq \mathsf{AM}$. Then since BQP is closed under complement, we would also have $\mathsf{coNP} \subseteq \mathsf{BQP}$, and hence $\mathsf{coNP} \subseteq \mathsf{AM}$, which is known to imply a collapse of PH [17]. And thus, if PH is infinite, then either $\mathsf{NP} \not\subset \mathsf{BQP}$ *or* $\mathsf{BQP} \not\subset \mathsf{AM}$. How can we say anything more interesting and nontrivial?

1.2 Relativization

Since the work of Baker, Gill, and Solovay [13], whenever complexity theorists were faced with an impasse like the one above, a central tool has been *relativized* or *black-box* complexity: in other words, studying what happens when all the complexity classes one cares about are fed some specially-constructed oracle. Much like perturbation theory in physics, relativization lets us make well-defined progress even when the original questions we wanted to answer are out of reach. It is well-known that relativization is an imperfect tool – the $\mathsf{IP} = \mathsf{PSPACE}$ [42], $\mathsf{MIP} = \mathsf{NEXP}$ [12], and more recently, $\mathsf{MIP}^* = \mathsf{RE}$ [30] theorems provide famous examples

where complexity classes turned out to be equal, even in the teeth of oracles relative to which they were unequal. On the other hand, so far, almost all such examples have originated from a single source: namely, the use of algebraic techniques in interactive proof systems. And if, for example, we want to understand the consequences of $\mathsf{NP} \subseteq \mathsf{BQP}$, then arguably it makes little sense to search for nonrelativizing consequences if we don't even understand yet what the relativizing consequences (that is, the consequences that hold relative to all oracles) are or are not.

In quantum complexity theory, even more than in classical complexity theory, relativization has been an inextricable part of progress from the very beginning. The likely explanation is that, even when we just count queries to an oracle, in the quantum setting we need to consider algorithms that query all oracle bits in superposition – so that even in the most basic scenarios, it is already unintuitive what can and cannot be done, and so oracle results must do much more than formalize the obvious.

More concretely, Bernstein and Vazirani [15] introduced some of the basic techniques of quantum algorithms in order to prove, for the first time, that there exists an oracle A such that $\mathsf{BPP}^A \neq \mathsf{BQP}^A$. Shortly afterward, Simon [44] gave a quantitatively stronger oracle separation between BPP and BQP, and then Shor [43] gave a still stronger separation, along the way to his famous discovery that FACTORING is in BQP.

On the negative side, Bennett, Bernstein, Brassard, and Vazirani [14] showed that there exists an oracle relative to which $\mathsf{NP} \not\subset \mathsf{BQP}$: indeed, relative to which there are problems that take n time for an NP machine but $\Omega\left(2^{n/2}\right)$ time for a BQP machine. Following the discovery of Grover's algorithm [25], which quantumly searches any list of N items in $O\left(\sqrt{N}\right)$ queries, the result of Bennett, Bernstein, Brassard, and Vazirani gained the interpretation that *Grover's algorithm is optimal*. In other words, any quantum algorithm for NP-complete problems that gets more than the square-root speedup of Grover's algorithm must be "non-black-box." It must exploit the structure of a particular NP-complete problem much like a classical algorithm would have to, rather than treating the problem as just an abstract space of 2^n possible solutions.

Meanwhile, clearly there are oracles relative to which $\mathsf{P} = \mathsf{BQP}$ – for example, a PSPACE-complete oracle. But we can ask: would such oracles necessarily collapse the hierarchy of classical complexity classes as well? In a prescient result that provided an early example of the sort of thing we do in this paper, Fortnow and Rogers [21] showed that there exists an oracle relative to which $\mathsf{P} = \mathsf{BQP}$ and yet PH is infinite. In other words, if $\mathsf{P} = \mathsf{BQP}$ would imply a collapse of the polynomial hierarchy, then it cannot be for a relativizing reason. Aaronson and Chen [6] extended this to show that there exists an oracle relative to which *sampling-based quantum supremacy is impossible* – i.e., any probability distribution approximately samplable in quantum polynomial time is also approximately samplable in classical polynomial time – and yet PH is infinite. In other words, if it is possible to prove the central theoretical conjecture of quantum supremacy – namely, that there are noisy quantum sampling experiments that cannot be simulated in classical polynomial time unless PH collapses – then nonrelativizing techniques will be needed there as well.

What about showing the power of BQP, by giving oracle obstructions to containments like $\mathsf{BQP} \subseteq \mathsf{NP}$, or $\mathsf{BQP} \subseteq \mathsf{PH}$? There, until recently, the progress was much more limited. Watrous [48] showed that there exists an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{NP}$ and even $\mathsf{BQP} \not\subset \mathsf{MA}$ (these separations could also have been shown using the RECURSIVE FOURIER SAMPLING problem, introduced by Bernstein and Vazirani [15]). But extending this further, to get an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{PH}$ or even $\mathsf{BQP} \not\subset \mathsf{AM}$, remained an open problem for two decades. Aaronson [2] proposed a program for proving an oracle separation between BQP and PH, involving a new problem he introduced called FORRELATION:

▶ **Problem 1** (Forrelation). *Given black-box access to two Boolean functions* $f, g : \{0,1\}^n \to \{1,-1\}$*, and promised that either*

(i) *f and g are uniformly random and independent, or*

(ii) *f and g are uniformly random individually, but g has $\Omega(1)$ correlation with $\hat{f}$, the Boolean Fourier transform of f (i.e., f and g are "Forrelated"),*

decide which.

Aaronson [2] showed that Forrelation is solvable, with constant bias, using only a single quantum query to f and g (and $O(n)$ time). By contrast, he showed that any classical randomized algorithm for the problem needs $\Omega\left(2^{n/4}\right)$ queries – improved by Aaronson and Ambainis [4] to $\Omega\left(\frac{2^{n/2}}{n}\right)$ queries, which is essentially tight. The central conjecture, which Aaronson left open, said that Forrelation $\notin \mathsf{PH}$ – or equivalently, by the connection between PH machines and AC^0 circuits [22], that there are no AC^0 circuits for Forrelation of constant depth and $2^{\mathrm{poly}(n)}$ size.

Finally, Raz and Tal [38] managed to prove Aaronson's conjecture, and thereby obtain the long-sought oracle separation between BQP and PH.[1] Raz and Tal achieved this by introducing new techniques for constant-depth circuit lower bounds, involving Brownian motion and the L_1-weight of the low-order Fourier coefficients of AC^0 functions. Relevantly for us, Raz and Tal actually proved the following stronger result:

▶ **Theorem 2** ([38]). *A PH machine can guess whether f and g are uniform or Forrelated with bias at most $2^{-\Omega(n)}$.*

Recall that before Raz and Tal, we did not even have an oracle relative to which $\mathsf{BQP} \not\subseteq \mathsf{AM}$. Notice that, if $\mathsf{BQP} \subseteq \mathsf{AM}$, then many other conclusions would follow in a relativizing way. For example, we would have:

- $\mathsf{P} = \mathsf{NP}$ implies $\mathsf{P} = \mathsf{BQP}$,
- $\mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{NP}^{\mathsf{AM} \cap \mathsf{coAM}} \subseteq \mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{BQP}^{\mathsf{NP}}$,
- If $\mathsf{NP} \subseteq \mathsf{BQP}$, then $\mathsf{NP}^{\mathsf{NP}} \subseteq \mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{NP}} = \mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$, and
- If $\mathsf{NP} \subseteq \mathsf{BQP}$, then $\mathsf{NP} \subseteq \mathsf{coAM}$, which implies that PH collapses.

Looking at it a different way, our inability even to separate BQP from AM by an oracle served as an obstruction to numerous other oracle separations.

The starting point of this paper was the following question: in a "post-Raz-Tal world," can we at last completely "unshackle" BQP from P, NP, and PH, by showing that there are no relativizing obstructions to any possible answers to questions like the ones we asked in Section 1.1?

1.3 Our Results

We achieve new oracle separations that show an astonishing range of possible behaviors for BQP and related complexity classes – in at least one case, resolving a longstanding open problem in this topic. Our title, "The Acrobatics of BQP," comes from a unifying theme of the new results being "freedom." We will show that, as far as relativizing techniques can detect, collapses and separations of classical complexity classes place surprisingly few constraints on the power of quantum computation. In most cases, this can be understood as ultimately

[1] Strictly speaking, they did this for a variant of Forrelation where the correlation between g and $\hat{f}$ is only $\sim \frac{1}{n}$, and thus a quantum algorithm needs $\sim n$ queries to solve the problem, but this will not affect anything that follows.

stemming from the fact that one cannot "fix the randomness" (or quantumness) used by a quantum algorithm, similarly to how one fixes the randomness used by a randomized algorithm in many complexity-theoretic arguments.

As we alluded to earlier, many of our new results would not have been possible without Raz and Tal's analysis of Forrelation [38], which we rely on extensively. We will treat Forrelation no longer as just an isolated problem, but as a sort of cryptographic code, by which an oracle can systematically make certain information available to BQP machines while keeping the information hidden from classical machines.

Having said that, very few of our results will follow from Raz-Tal in any straightforward way. Most often we need to develop other lower bound tools, in addition to or instead of Raz-Tal. Our new tools, which seem likely to be of independent interest, include a random restriction lemma for quantum query algorithms, a concentration theorem for the block sensitivity of AC^0 functions, and a provable analogue of the Aaronson-Ambainis conjecture [3] for certain sparse oracles.

Perhaps our single most interesting result is the following.

▶ **Theorem 3.** *There exists an oracle relative to which* $\mathsf{NP}^{\mathsf{BQP}} \not\subset \mathsf{BQP}^{\mathsf{NP}}$*, and indeed* $\mathsf{NP}^{\mathsf{BQP}} \not\subset \mathsf{BQP}^{\mathsf{PH}}$*.*

As mentioned earlier, Theorem 3 resolves an open problem of Fortnow [20], and demonstrates a clear difference between BPP and BQP that exemplifies the impossibility of pulling the randomness out of a quantum algorithm. Indeed, Theorem 3 shows that there is no general, black-box way to move quantumness past an NP quantifier, like we can do for classical randomness.

As a straightforward byproduct of Theorem 3, we are also able to prove the following:

▶ **Theorem 4.** *There exists an oracle relative to which* $\mathsf{P} = \mathsf{NP}$ *but* $\mathsf{BQP} \neq \mathsf{QCMA}$*.*

Conversely, it will follow from one of our later results, Theorem 9, that there exists an oracle relative to which $\mathsf{P} \neq \mathsf{NP}$ and yet $\mathsf{BQP} = \mathsf{QCMA} = \mathsf{QMA}$. In other words, as far as relativizing techniques are concerned, the classical and quantum versions of the P vs. NP question are completely uncoupled from one another.

Theorem 3 also represents progress toward a proof of the following conjecture, which might be the most alluring open problem that we leave.

▶ **Conjecture 5.** *There exists an oracle relative to which* $\mathsf{NP} \subseteq \mathsf{BQP}$ *but* $\mathsf{PH} \not\subset \mathsf{BQP}$*. Indeed, for every* $k \in \mathbb{N}$*, there exists an oracle relative to which* $\Sigma_k^{\mathsf{P}} \subseteq \mathsf{BQP}$ *but* $\Sigma_{k+1}^{\mathsf{P}} \not\subset \mathsf{BQP}$*.*

Conjecture 5 would provide spectacularly fine control over the relationship between BQP and PH, going far beyond Raz-Tal to show how BQP could, e.g., swallow the first 18 levels of PH without swallowing the 19th. To see the connection between Theorem 3 and Conjecture 5, suppose $\mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{NP}}$, and suppose also that $\mathsf{NP} \subseteq \mathsf{BQP}$. Then, as observed by Fortnow [20], this would imply

$$\mathsf{NP}^{\mathsf{NP}} \subseteq \mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{NP}} \subseteq \mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP},$$

(and so on, for all higher levels of PH), so that $\mathsf{PH} \subseteq \mathsf{BQP}$ as well. Hence, any oracle that witnesses Conjecture 5 also witnesses Theorem 3, so our proof of Theorem 3 is indeed a prerequisite to Conjecture 5.

At a high level, we prove Theorem 3 by showing that no $\mathsf{BQP}^{\mathsf{PH}}$ machine can solve the $\mathrm{OR} \circ$ Forrelation problem, in which one is given a long list of Forrelation instances, and is tasked with distinguishing whether (1) all of the instances are uniformly random, or (2)

at least one of the instances is Forrelated. A first intuition is that PH machines should gain no useful information from the input, just because FORRELATION "looks random" (by Raz-Tal), and hence a $\mathsf{BQP}^{\mathsf{PH}}$ machine should have roughly the same power as a BQP machine at deciding OR $\circ$ FORRELATION. If one could show this, then completing the theorem would amount to showing that OR $\circ$ FORRELATION is hard for BQP machines, which easily follows from the BBBV Theorem [14].

Alas, initial attempts to formalize this intuition fail for a single, crucial reason: the possibility of homomorphic encryption! The Raz-Tal Theorem merely proves that FORRELATION is a strong form of encryption against PH algorithms. But to rule out a $\mathsf{BQP}^{\mathsf{PH}}$ algorithm for OR $\circ$ FORRELATION, we *also* have to show that one cannot take a collection of FORRELATION instances and transform them, by means computable in PH, into a single FORRELATION instance whose solution is the OR of the solutions to the input instances. Put another way, we must show that AC^0 circuits of constant depth and $2^{\mathrm{poly}(n)}$ size cannot homomorphically evaluate the OR function, when the encryption is done via the FORRELATION problem.

More generally, we even have to show that AC^0 circuits cannot transform the "ciphertext" into *any* string that could later be decoded by an efficient quantum algorithm. Theorem 3 accomplishes this with the help of an additional structural property of AC^0 circuits: our concentration theorem for block sensitivity. Loosely speaking, the concentration theorem implies that, with overwhelming probability, any small AC^0 circuit is insensitive to toggling between a yes-instance and a neighboring no-instance of the OR $\circ$ FORRELATION problem. This, together with the BBBV Theorem [14], then implies that such "homomorphic encryption" is impossible.

We also achieve the following converse to Theorem 3:

▶ **Theorem 6.** *There exists an oracle relative to which* $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{BQP}}$*, and even* $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{PromiseBQP}}$.

Note that an oracle relative to which $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{NP}^{\mathsf{BQP}}$ is almost trivial to achieve, for example by considering a problem in coNP. However, $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{BQP}}$ is much harder. At a high level, rather than considering the composed problem OR $\circ$ FORRELATION, we now need to consider the reverse composition: FORRELATION $\circ$ OR, a problem that's clearly in $\mathsf{BQP}^{\mathsf{NP}}$, but plausibly not in $\mathsf{PH}^{\mathsf{BQP}}$. The key step is to show that, when solving FORRELATION $\circ$ OR, *any* $\mathsf{PH}^{\mathsf{BQP}}$ *machine can be simulated by a* PH *machine*: the BQP oracle is completely superfluous! Once we've shown that, FORRELATION $\circ$ OR $\notin$ PH then follows immediately from Raz-Tal.

For our next result, recall that QMA, or *Quantum Merlin-Arthur*, is the class of problems for which a yes-answer can be witnessed by a polynomial-size quantum state. Perhaps our second most interesting result is this:

▶ **Theorem 7.** PP *is not contained in the* "QMA hierarchy"*, consisting of constant-depth towers of the form* $\mathsf{QMA}^{\mathsf{QMA}^{\mathsf{QMA}^{\cdots}}}$*, with probability* 1 *relative to a random oracle.*[2]

Note that PP = PostBQP, where PostBQP denotes BQP augmented with the power of postselection [1], and so Theorem 7 contrasts with the classical containment $\mathsf{PostBPP} \subseteq \mathsf{BPP}^{\mathsf{NP}} \subseteq \mathsf{PH}$ [26, 33]. Nevertheless, before this paper, to our knowledge, it was not even

[2] Actually, our formal definition of the QMA hierarchy is more general than the version given here, in order to accommodate recursive queries to QMA promise problems. This only makes our separation stronger.

known how to construct an oracle relative to which $\mathsf{PP} \not\subset \mathsf{BQP}^{\mathsf{NP}}$, let alone classes like $\mathsf{BQP}^{\mathsf{NP}^{\mathsf{BQP}^{\mathsf{NP}^{\cdots}}}}$ or $\mathsf{QCMA}^{\mathsf{QCMA}^{\mathsf{QCMA}^{\cdots}}}$, which are contained in the QMA hierarchy. The closest result we are aware of is due to Kretschmer [32], who gave a *quantum* oracle relative to which $\mathsf{BQP} = \mathsf{QMA} \neq \mathsf{PostBQP}$.

Perhaps shockingly, our proof of Theorem 7 can be extended even to show that PP is not in, say, $\mathsf{QMIP}^{\mathsf{QMIP}^{\mathsf{QMIP}^{\cdots}}}$ relative to a random oracle, where QMIP means Quantum Multi-prover Interactive Proofs with entangled provers. This is despite the breakthrough results of Reichardt, Unger, and Vazirani [39], and more recently Ji, Natarajan, Vidick, Wright, and Yuen [30], which showed that in the *unrelativized* world, $\mathsf{QMIP} = \mathsf{MIP}^* = \mathsf{RE}$ (where MIP^* means QMIP with classical communication only, and RE means Recursively Enumerable), so in particular, QMIP contains the halting problem. This underscores the dramatic extent to which results like $\mathsf{QMIP} = \mathsf{RE}$ are nonrelativizing!

Theorem 7 can also be understood as showing that in the black-box setting, there is no quantum analogue of Stockmeyer's approximate counting algorithm [46]. For a probabilistic algorithm M that runs in $\mathrm{poly}(n)$ time and an error bound $\varepsilon \geq \frac{1}{\mathrm{poly}(n)}$, the approximate counting problem is to estimate the acceptance probability of M up to a multiplicative factor of $1+\varepsilon$. Stockmeyer's algorithm [46] gives a relativizing $\mathrm{poly}(n)$-time reduction from the approximate counting problem to a problem in the third level of the polynomial hierarchy, and crucially relies on pulling the randomness out of M. In structural complexity terms, Stockmeyer's algorithm can be reinterpreted as showing that $\mathsf{SBP} \subseteq \mathsf{PH}$ relative to all oracles, where SBP is the complexity class defined in [16] that captures approximate counting.

One might wonder: is there a version of Stockmeyer's algorithm for the *quantum* approximate counting problem, where we instead wish to approximate the acceptance probability of a quantum algorithm? In particular, is SBQP, the complexity class that captures quantum approximate counting [33], contained in the QMA hierarchy?[3] Kuperberg [33] showed that $\mathsf{PP} \subseteq \mathsf{P}^{\mathsf{SBQP}}$, so it follows that $\mathsf{PP} \subseteq \mathsf{QMAH}$ if and only if $\mathsf{SBQP} \subseteq \mathsf{QMAH}$, where QMAH denotes the QMA hierarchy. Thus, Theorem 7 implies that $\mathsf{SBQP} \not\subset \mathsf{QMAH}$ relative to a random oracle, implying that such a quantum analogue of Stockmeyer's algorithm does not exist in the black-box setting.[4] This demonstrates yet another case where a classical complexity result that relies on fixing randomness cannot be generalized to the quantum setting.

Notably, our proof of Theorem 7 does not appeal to Raz-Tal at all, but instead relies on a new random restriction lemma for the acceptance probabilities of quantum query algorithms. Our random restriction lemma shows that if one randomly fixes most of the inputs to a quantum query algorithm, then the algorithm's behavior on the unrestricted inputs can be approximated by a "simple" function (say, a small decision tree or small DNF formula). We then use this random restriction lemma to generalize the usual random restriction proof that, for example, $\textsc{Parity} \notin \mathsf{AC}^0$ [27].

Here is another noteworthy result that we are able to obtain, by combining random restriction arguments with lower bounds on quantum query complexity:

---

[3] We thank Patrick Rall (personal communication) for bringing this question to our attention.

[4] Note that this is just one of many possible ways that we could ask whether there exists a quantum analogue of Stockmeyer's algorithm. For example, one might consider alternative definitions of the quantum approximate counting task, such as the problem defined in [18] of approximating the number of witness states accepted by a QMA verifier. One might also consider other definitions of the "quantum polynomial hierarchy," some of which are explored in [23].

▶ **Theorem 8.** *For every $k \in \mathbb{N}$, $\Sigma_{k+1}^{\mathsf{P}} \not\subset \mathsf{BQP}^{\Sigma_k^{\mathsf{P}}}$ with probability 1 relative to a random oracle.*

Theorem 8 extends the breakthrough of Håstad, Rossman, Servedio, and Tan [28], who (solving an open problem from the 1980s) showed that PH is infinite relative to a random oracle with probability 1. Our result shows, not only that a random oracle creates a gap between every two successive levels of PH, but that quantum computing fails to bridge that gap.

Again, Theorem 8 represents a necessary step toward a proof of Conjecture 5, because if we had $\Sigma_{k+1}^{\mathsf{P}} \subseteq \mathsf{BQP}^{\Sigma_k^{\mathsf{P}}}$, then clearly $\Sigma_k^{\mathsf{P}} \subseteq \mathsf{BQP}$ would imply $\Sigma_{k+1}^{\mathsf{P}} \subseteq \mathsf{BQP}^{\mathsf{BQP}} = \mathsf{BQP}$.

Our last two theorems return to the theme of the autonomy of BQP.

▶ **Theorem 9.** *There exists an oracle relative to which $\mathsf{NP} \subseteq \mathsf{BQP}$, and indeed $\mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$, and yet PH is infinite.*

Theorem 9 resolves an open problem of Aaronson [2]. As a simple corollary, we also obtain an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{NP/poly}$, resolving a question of Aaronson, Cojocaru, Gheorghiu, and Kashefi [7].

For three decades, one of the great questions of quantum computation has been whether it can solve NP-complete problems in polynomial time. Many experts guess that the answer is no, for similar reasons as they guess that $\mathsf{P} \neq \mathsf{NP}$ – say, the BBBV Theorem [14], combined with our failure to find any promising leads for evading that theorem's assumptions in the worst case. But the fact remains that we have no structural evidence connecting the $\mathsf{NP} \not\subset \mathsf{BQP}$ conjecture to any "pre-quantum" beliefs about complexity classes. No one has any idea how to show, for example, that if $\mathsf{NP} \subseteq \mathsf{BQP}$ then $\mathsf{P} = \mathsf{NP}$ as well, or anything even remotely in that direction.

Given the experience of classical complexity theory, it would be reasonable to hope for a theorem showing that, if $\mathsf{NP} \subseteq \mathsf{BQP}$, then PH collapses – analogous to the Karp-Lipton Theorem [31], that if $\mathsf{NP} \subset \mathsf{P/poly}$ then PH collapses, or the Boppana-Håstad-Zachos Theorem [17], that if $\mathsf{NP} \subseteq \mathsf{coAM}$ then PH collapses. No such result is known for $\mathsf{NP} \subseteq \mathsf{BQP}$, once again because of the difficulty that there is no known way to pull the randomness out of a BQP algorithm. Theorem 9 helps to explain this situation, by showing that any proof of such a conditional collapse would have to be nonrelativizing. The proof of Theorem 9 builds, again, on the Raz-Tal Theorem. And this is easily seen to be necessary, since as we pointed out earlier, if $\mathsf{BQP} \subseteq \mathsf{AM}$, then $\mathsf{NP} \subseteq \mathsf{BQP}$ really *would* imply a collapse of PH.

▶ **Theorem 10.** *There exists an oracle relative to which $\mathsf{P} = \mathsf{NP} \neq \mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$.*

Theorem 10 says, in effect, that there is no relativizing obstruction to BQP being inordinately powerful even while NP is inordinately weak. It substantially extends the Raz-Tal Theorem, that there is an oracle relative to which $\mathsf{BQP} \not\subset \mathsf{PH}$, to show that in some oracle worlds, BQP doesn't go just *slightly* beyond the power of PH (which, if $\mathsf{P} = \mathsf{NP}$, is simply the power of P), but *vastly* beyond it. Once again, this illustrates the difference between randomness and quantumness, because if $\mathsf{P} = \mathsf{NP}$, then $\mathsf{P} = \mathsf{BPP}$ for relativizing reasons.

We conjecture that Theorem 10 could be extended yet further, to give an oracle relative to which $\mathsf{P} = \mathsf{NP}$ and yet $\mathsf{BQP} = \mathsf{EXP}$, but we leave that problem to future work.

1.4 Proof Techniques

We now give rough sketches of the important ideas needed to prove our results. Here, in contrast to Section 1.3, we present the results in the order that they appear in the main text, which is roughly in order of increasing technical difficulty.

Our proofs of Theorem 9 and Theorem 10 serve as useful warm-ups, giving a flavor for how we use the Raz-Tal Theorem and oracle construction techniques in later proofs. In Theorem 9, to construct an oracle where $\mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$ but PH is infinite, we start by taking a random oracle, which by the work of Håstad, Rossman, Servedio, and Tan [28, 41] is known to make PH infinite. Then, for each $\mathsf{P}^{\#\mathsf{P}}$ machine M, we add to the oracle an instance of the Forrelation problem that encodes the behavior of M: if M accepts, we choose a Forrelated instance, while if M rejects, we choose a uniformly random instance. This gives a BQP machine the power to decide any $\mathsf{P}^{\#\mathsf{P}}$ language.[5]

It remains to argue that adding these Forrelation instances does not collapse PH. We want to show that relative to our oracle, for every k, there exists a language in $\Sigma_{k+1}^{\mathsf{P}}$ that is not in Σ_k^{P}. This is where we leverage the Raz-Tal Theorem: because the Forrelation instances look random to PH, we can show, by a hybrid argument, that a Σ_k^{P} algorithm's probability of correctly deciding a target function in $\Sigma_{k+1}^{\mathsf{P}}$ is roughly unchanged if we replace the Forrelation instances with uncorrelated, uniformly random bits. But auxiliary random bits cannot possibly improve the success probability, and so a simple appeal to [28] implies that the $\Sigma_{k+1}^{\mathsf{P}}$ language remains hard for Σ_k^{P}.

The proof of Theorem 10, giving an oracle where $\mathsf{P} = \mathsf{NP} \neq \mathsf{BQP} = \mathsf{P}^{\#\mathsf{P}}$, follows a similar recipe to the proof of Theorem 9. We start with a random oracle, which separates PH from $\mathsf{P}^{\#\mathsf{P}}$, and then we add a second region of the oracle that puts $\mathsf{P}^{\#\mathsf{P}}$ into BQP by encoding all $\mathsf{P}^{\#\mathsf{P}}$ queries in instances of the Forrelation problem. Next, we add a third region of the oracle that answers all NP queries, which has the effect of collapsing PH to P. Finally, we again leverage the Raz-Tal Theorem to argue that the Forrelation instances have no effect on the separation between PH and $\mathsf{P}^{\#\mathsf{P}}$, because the Forrelation instances look random to PH algorithms.

We next prove Theorem 8, that $\Sigma_{k+1}^{\mathsf{P}} \not\subset \mathsf{BQP}^{\Sigma_k^{\mathsf{P}}}$ relative to a random oracle. Our proof builds heavily on the proof by [28] that $\Sigma_{k+1}^{\mathsf{P}} \not\subset \Sigma_k^{\mathsf{P}}$ relative to a random oracle. Indeed, our proof is virtually identical, except for a single additional step.

[28]'s proof involves showing that there exists a function $\textsc{Sipser}_d$ that is computable by a small AC^0 circuit of depth d (which corresponds to a $\Sigma_{d-1}^{\mathsf{P}}$ algorithm), but such that any small AC^0 circuit of depth $d-1$ (which corresponds to a $\Sigma_{d-2}^{\mathsf{P}}$ algorithm) computes $\textsc{Sipser}_d$ on at most a $\frac{1}{2} + o(1)$ fraction of random inputs. This proof uses random restrictions, or more accurately, a generalization of random restrictions called *random projections* by [28]. Roughly speaking, the proof constructs a distribution $\mathcal{R}$ over random projections with the following properties:

[5] The careful reader might wonder: if we can encode the answers to $\mathsf{P}^{\#\mathsf{P}}$ machines, then what is to stop us from encoding the answers to some arbitrarily powerful class, such as EXP or RE, into the Forrelation instances? For a $\mathsf{P}^{\#\mathsf{P}}$ machine M, we exploit the fact that we can always choose Forrelation instances on oracle strings that cannot be queried by M. For example, if M runs in time t, then we can encode M's output into strings of length t^c for some $c > 1$, which remain accessible to a BQP machine with a larger polynomial running time. By contrast, if we tried to do the same for an EXP machine (say), we run into the problem that the machine whose behavior we are trying to encode could query the very encoding we are making of its output, and thus our oracle would be circularly defined.

(i) Any small AC^0 circuit C of depth $d-1$ "simplifies" with high probability under a random projection drawn from $\mathcal{R}$, say, by collapsing to a low-depth decision tree.

(ii) The target $\textsc{Sipser}_d$ function "retains structure" with high probability under a random projection drawn from $\mathcal{R}$.

(iii) The structure retained in (ii) implies that the original unrestricted circuit C fails to compute the $\textsc{Sipser}_d$ function on a large fraction of inputs.

To prove Theorem 8, we generalize step (i) above from $\Sigma^{\mathsf{P}}_{d-2}$ algorithms to $\mathsf{BQP}^{\Sigma^{\mathsf{P}}_{d-2}}$ algorithms. That is, if we have a quantum algorithm that queries arbitrary depth-$(d-1)$ AC^0 functions of the input, then we show that this algorithm's acceptance probability also "simplifies" under a random projection from $\mathcal{R}$. We prove this by combining the BBBV Theorem [14] with [28]'s proof of step (i).

We next move on to the proof of Theorem 3, where we construct an oracle relative to which $\mathsf{NP}^{\mathsf{BQP}} \not\subset \mathsf{BQP}^{\mathsf{PH}}$. Recall that we prove Theorem 3 by showing that no $\mathsf{BQP}^{\mathsf{PH}}$ machine can solve the OR $\circ$ Forrelation problem. To establish this, imagine that we fix a "no" instance x of the OR $\circ$ Forrelation problem, meaning that x consists of a list of $\sim 2^n$ Forrelation instances that are all uniformly random (i.e. non-Forrelated). We can turn x into an adjacent "yes" instance y by randomly choosing one of the Forrelation instances of x and changing it to be Forrelated.

Our proof amounts to showing that with high probability over x, an AC^0 circuit of size $2^{\mathrm{poly}(n)}$ is unlikely (over y) to distinguish x from y. Then, applying the BBBV Theorem [14], we can show that for most choices of x, a $\mathsf{BQP}^{\mathsf{PH}}$ algorithm is unlikely to distinguish x from y, implying that it could not have solved the OR $\circ$ Forrelation problem.

Next, we notice that it suffices to consider what happens when, instead of choosing y by randomly flipping one of the Forrelation instances of x from uniformly random to Forrelated, we instead choose a string z by randomly resampling one if the instances of x from the uniform distribution. This is because, as a straightforward consequence of the Raz-Tal Theorem (Theorem 2), if f is an AC^0 circuit of size $2^{\mathrm{poly}(n)}$, then $|\Pr_y[f(x) \neq f(y)] - \Pr_z[f(x) \neq f(z)]| \leq 2^{-\Omega(n)}$.

Our key observation is that the quantity $\Pr_z[f(x) \neq f(z)]$ is proportional to a sort of "block sensitivity" of f on x. More precisely, it is proportional to an appropriate averaged notion of block sensitivity, where the average is taken over collections of blocks that respect the partition into separate Forrelation instances. This is where our block sensitivity concentration theorem comes into play:

▶ **Theorem 11.** *Let $f : \{0,1\}^N \to \{0,1\}$ be an AC^0 circuit of size $\mathrm{quasipoly}(N)$ and depth $O(1)$, and let $B = \{B_1, B_2, \ldots, B_k\}$ be a collection of disjoint subsets of $[N]$. Then for any t,*

$$\Pr_{x \sim \{0,1\}^N}[\mathsf{bs}^x_B(f) \geq t] \leq 4N \cdot 2^{-\Omega\left(\frac{t}{\mathrm{polylog}(N)}\right)},$$

where $\mathsf{bs}^x_B(f)$ denotes the block sensitivity of f on x with respect to B.

Informally, Theorem 11 says that the probability that an AC^0 circuit has B-block sensitivity $t \gg \mathrm{polylog}(N)$ on a random input x decays exponentially in t. This generalizes the result of Linial, Mansour, and Nisan [35] that the *average* sensitivity of AC^0 circuits is at most $\mathrm{polylog}(N)$. It also generalizes a concentration theorem for the sensitivity of AC^0 circuits that appeared implicitly in the work of Gopalan, Servedio, Tal, and Wigderson [24], by

taking B to be the partition into singletons.[6] In fact, we derive Theorem 11 as a simple corollary of such a sensitivity tail bound for AC^0. For completeness, we will also prove our own sensitivity tail bound, rather than appealing to [24]. Our sensitivity tail bound follows from an AC^0 random restriction lemma due to Rossman [40].

To prove Theorem 4, which gives an oracle relative to which $\mathsf{P} = \mathsf{NP}$ but $\mathsf{BQP} \neq \mathsf{QCMA}$, we use a similar technique to the proof of Theorem 10. We first take the oracle constructed in Theorem 3 that contains instances of the $\text{OR} \circ \text{Forrelation}$ problem. Next, we add a second region of the oracle that answers all NP queries. This collapses PH to P. Finally, we use Theorem 3 to argue that these NP queries do not enable a BQP machine to solve the $\text{OR} \circ \text{Forrelation}$ problem, which is in QCMA.

We now move on to the proof of Theorem 6, that there exists an oracle relative to which $\mathsf{BQP}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{BQP}}$. Recall that our strategy is to show that no $\mathsf{PH}^{\mathsf{BQP}}$ machine can solve the $\text{Forrelation} \circ \text{OR}$ problem. We prove this by showing that with high probability, a $\mathsf{PH}^{\mathsf{BQP}}$ machine on a random instance of the $\text{Forrelation} \circ \text{OR}$ problem can be simulated by a PH machine, from which a lower bound easily follows from the Raz-Tal Theorem. This simulation hinges on the following theorem, which seems very likely to be of independent interest:

▶ **Theorem 12.** *Consider a quantum algorithm Q that makes T queries to an $M \times N$ array of bits x, where each length-N row of x contains a single uniformly random 1 and 0s everywhere else. Then for any $\varepsilon \gg \frac{T}{\sqrt{N}}$ and $\delta > 0$, there exists a deterministic classical algorithm that makes $O\left(\frac{T^5}{\varepsilon^4} \log \frac{T}{\delta}\right)$ queries to x, and approximates Q's acceptance probability to within additive error ε on a $1 - \delta$ fraction of such randomly chosen x's.*

Informally, Theorem 12 says that any fast enough quantum algorithm can be simulated by a deterministic classical algorithm, with at most a polynomial blowup in query complexity, on almost all sufficiently sparse oracles. The crucial point here is that the classical simulation still needs to work, even in most cases where the quantum algorithm is lucky enough to find many "1" bits. We prove Theorem 12 via a combination of tail bounds and the BBBV hybrid argument [14].

In the statement of Theorem 12, we do not know whether the exponent of 5 on T is tight, and suspect that it isn't. We only know that the exponent needs to be at least 2, because of Grover's algorithm [25].

We remark that Theorem 12 bears similarity to a well-known conjecture that involves simulation of quantum query algorithms by classical algorithms. A decade ago, motivated by the question of whether $\mathsf{P} = \mathsf{BQP}$ relative to a random oracle with probability 1, Aaronson and Ambainis [3] proposed the following conjecture:

▶ **Conjecture 13** ([3, Conjecture 1.5]; attributed to folklore)**.** *Consider a quantum algorithm Q that makes T queries to $x \in \{0,1\}^N$. Then for any $\varepsilon, \delta > 0$, there exists a deterministic classical algorithm that makes $\text{poly}\left(T, \frac{1}{\varepsilon}, \frac{1}{\delta}\right)$ queries to x, and approximates Q's acceptance probability to within additive error ε on a $1 - \delta$ fraction of uniformly randomly inputs x.*

[6] Interestingly, [24]'s goal, in proving their concentration theorem for the sensitivity of AC^0, was to make progress toward a proof of the famous *Sensitivity Conjecture* – a goal that Huang [29] achieved shortly afterward using completely different methods. One happy corollary of this work is that, nevertheless, [24]'s attempt on the problem was not entirely in vain.

While Conjecture 13 has become influential in Fourier analysis of Boolean functions,[7] it remains open to this day. Theorem 12 could be seen as *the analogue of Conjecture 13 for sparse oracles* – an analogue that, because of the sparseness, turns out to be much easier to prove.

We conclude with the proof of Theorem 7, showing that PP is not contained in the QMA hierarchy relative to a random oracle. This is arguably the most technically involved part of this work. Recall that our key contribution, and the most important step of our proof, is a random restriction lemma for quantum query algorithms. In fact, we even prove a random restriction lemma for functions with low *quantum Merlin-Arthur* (QMA) *query complexity*: that is, functions f where a verifier, given an arbitrarily long "witness state," can become convinced that $f(x) = 1$ by making few queries to x. Notably, our definition of QMA query complexity does not care about the length of the witness, but only on the number of queries made by the verifier. This property allows us to extend our results to complexity classes beyond QMA, such as QMIP.

An informal statement of our random restriction lemma is given below:

▶ **Theorem 14.** *Consider a partial function $f : \{0,1\}^N \to \{0,1,\perp\}$ with* QMA *query complexity at most* $\mathrm{polylog}(N)$*. For some $p = \frac{1}{\sqrt{N}\mathrm{polylog}(N)}$, let ρ be a random restriction that leaves each variable unrestricted with probability p. Then f_ρ is $\frac{1}{\mathrm{quasipoly}(N)}$-close, in expectation over ρ, to a* $\mathrm{polylog}(N)$*-width DNF formula.*[8]

An unusual feature of Theorem 14 is that we can only show that f_ρ is *close* to a simple function *in expectation*. By contrast, Håstad's switching lemma for DNF formulas [27] shows that the restricted function reduces to a simple function *with high probability*, so in some sense our result is weaker. Additionally, unlike the switching lemma, our result has a quantitative dependence on the number of inputs N. Whether this dependence can be removed (so that the bound depends only on the number of queries) remains an interesting problem for future work.

With Theorem 14 in hand, proving that $\mathsf{PP} \not\subset \mathsf{QMA}^{\mathsf{QMA}^{\mathsf{QMA}^{\cdots}}}$ relative to a random oracle is conceptually analogous to the proof that $\mathsf{PP} \not\subset \mathsf{PH}$ relative to a random oracle [27]. We first view a $\mathsf{QMA}^{\mathsf{QMA}^{\mathsf{QMA}^{\cdots}}}$ machine as a small constant-depth circuit in which the gates are functions of low QMA query complexity. Then we want to argue that the probability that such a circuit agrees with the Parity function on a random input is small. We accomplish this via repeated application of Theorem 14, interleaved with Håstad's switching lemma for DNF formulas [27].

To elaborate further, we first take a random restriction that, by Theorem 14, turns all of the bottom-layer QMA gates into DNF formulas. Next, we apply another random restriction and appeal to the switching lemma to argue that these DNFs reduce to functions of low decision tree complexity, which can be absorbed into the next layer of QMA gates. Finally, we repeat as many times as needed until the entire circuit collapses to a low-depth decision tree. Since the Parity function reduces to another Parity function under any random restriction,

[7] In the context of Fourier analysis, the Aaronson-Ambainis Conjecture usually refers to a closely-related conjecture about influences of bounded low-degree polynomials; see e.g. [36, 37]. Aaronson and Ambainis [3] showed that this related conjecture implies Conjecture 13.

[8] By saying that f_ρ is "close" to a DNF formula, we mean that there exists a DNF g depending on ρ such that the fraction of inputs on which f_ρ and g agree is $1 - \frac{1}{\mathrm{quasipoly}(N)}$, in expectation over ρ. In the full version [8], we introduce some additional notation and terminology that makes it easier to manipulate such expressions, but we will not use them in this exposition.

we conclude that this decision tree will disagree with the reduced PARITY function on a large fraction of inputs, and hence the original circuit must have disagreed with the PARITY function on a large fraction of inputs as well.

Of course, the actual proof of Theorem 7 is more complicated because of the accounting needed to bound the error introduced from Theorem 14, but all of the important concepts are captured above.

We end with a few remarks on the proof ideas needed for Theorem 14. Essentially, the first step involves proving that if we take a function f computed by a quantum query algorithm Q, a random restriction ρ, and a uniformly random input x to f_ρ, then x likely contains a small set K of "influential" variables. These influential variables have the property that for any string y that agrees with x on K, $|\Pr[Q(x) = 1] - \Pr[Q(y) = 1]|$ is bounded by a small constant. Hence, K serves as a certificate for f_ρ's behavior on x.

Proving that such a K usually exists amounts to a careful application of the BBBV Theorem [14]. Finally, we generalize from quantum query algorithms to arbitrary QMA query algorithms by observing that we only need to keep track of the certificates for inputs x such that $f_\rho(x) = 1$. The DNF we obtain in Theorem 14 is then simply the OR of all of these small 1-certificates.

Due to space constraints, we defer to the full version of our paper [8] for complete proofs and additional discussion.

References

1 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461:3473–3482, 2005. doi:10.1098/rspa.2005.1546.

2 Scott Aaronson. BQP and the polynomial hierarchy. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, pages 141–150, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1806689.1806711.

3 Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups. *Theory of Computing*, 10(6):133–166, 2014. doi:10.4086/toc.2014.v010a006.

4 Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. *SIAM Journal on Computing*, 47(3):982–1038, 2018. doi:10.1137/15M1050902.

5 Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013. doi:10.4086/toc.2013.v009a004.

6 Scott Aaronson and Lijie Chen. Complexity-Theoretic Foundations of Quantum Supremacy Experiments. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:67, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2017.22.

7 Scott Aaronson, Alexandru Cojocaru, Alexandru Gheorghiu, and Elham Kashefi. Complexity-Theoretic Limitations on Blind Delegated Quantum Computation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2019.6.

8 Scott Aaronson, DeVon Ingram, and William Kretschmer. The acrobatics of BQP, 2021. arXiv:2111.10409.

9 Leonard Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, pages 75–83, 1978. doi:10.1109/SFCS.1978.37.

10 Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997. doi:10.1137/S0097539795293639.

11 Frank Arute, Kunal Arya, Ryan Babbush, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. doi:10.1038/s41586-019-1666-5.

12 László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *computational complexity*, 1(1):3–40, 1991. doi:10.1007/BF01200056.

13 Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975. doi:10.1137/0204037.

14 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. doi:10.1137/S0097539796300933.

15 Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. doi:10.1137/S0097539796300921.

16 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.

17 Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, May 1987. doi:10.1016/0020-0190(87)90232-8.

18 Sergey Bravyi, Anirban Chowdhury, David Gosset, and Pawel Wocjan. On the complexity of quantum partition functions, 2021. arXiv:2110.15466.

19 Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A*, 467:459–472, 2010. doi:10.1098/rspa.2010.0301.

20 Lance Fortnow. Pulling out the quantumness [online]. December 2005. URL: https://blog.computationalcomplexity.org/2005/12/pulling-out-quantumness.html.

21 Lance Fortnow and John Rogers. Complexity limitations on quantum computation. In *Proceedings. Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference) (Cat. No.98CB36247)*, pages 202–209, 1998. doi:10.1109/CCC.1998.694606.

22 Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

23 Sevag Gharibian, Miklos Santha, Jamie Sikora, Aarthi Sundaram, and Justin Yirka. Quantum Generalizations of the Polynomial Hierarchy with Applications to QMA(2). In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.MFCS.2018.58.

24 Parikshit Gopalan, Rocco Servedio, Avishay Tal, and Avi Wigderson. Degree and sensitivity: tails of two distributions, 2016. Earlier version in CCC 2016. arXiv:1604.07432.

25 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 '96, pages 212–219, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/237814.237866.

26 Yenjo Han, Lane A. Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997. doi:10.1137/S0097539792240467.

27 Johan Håstad. *Computational Limitations of Small-Depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.

28 Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for Boolean circuits. *J. ACM*, 64(5), August 2017. doi:10.1145/3095799.

29 Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019. doi:10.4007/annals.2019.190.3.6.

30 Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP*=RE, 2020. arXiv:2001.04383.

31 Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, pages 302–309, New York, NY, USA, 1980. Association for Computing Machinery. doi:10.1145/800141.804678.

32 William Kretschmer. Quantum Pseudorandomness and Classical Complexity. In Min-Hsiu Hsieh, editor, *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*, volume 197 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TQC.2021.2.

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 Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983. doi:10.1016/0020-0190(83)90044-3.

35 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, July 1993. doi:10.1145/174130.174138.

36 Ashley Montanaro. Some applications of hypercontractive inequalities in quantum information theory. *Journal of Mathematical Physics*, 53(12):122206, 2012. doi:10.1063/1.4769269.

37 Ryan O'Donnell and Yu Zhao. Polynomial Bounds for Decoupling, with Applications. In Ran Raz, editor, *31st Conference on Computational Complexity (CCC 2016)*, volume 50 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2016.24.

38 Ran Raz and Avishay Tal. Oracle separation of BQP and PH. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 13–23, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316315.

39 Ben W. Reichardt, Falk Unger, and Umesh Vazirani. A classical leash for a quantum system: Command of quantum systems via rigidity of CHSH games. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 321–322, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2422436.2422473.

40 Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of AC0. Manuscript, 2017. URL: https://users.cs.duke.edu/~br148/logsize.pdf.

41 Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. Complexity theory column 89: The polynomial hierarchy, random oracles, and Boolean circuits. *SIGACT News*, 46(4):50–68, December 2015. doi:10.1145/2852040.2852052.

42 Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, October 1992. doi:10.1145/146585.146609.

43 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999. doi:10.1137/S0036144598347011.

44 Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997. doi:10.1137/S0097539796298637.

45 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 330–335, New York, NY, USA, 1983. Association for Computing Machinery. doi:10.1145/800061.808762.

46 Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 118–126, New York, NY, USA, 1983. Association for Computing Machinery. doi:10.1145/800061.808740.

47 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991. doi:10.1137/0220053.

48 John Watrous. Succinct quantum proofs for properties of finite groups. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 537–546. IEEE, 2000. doi:10.1109/SFCS.2000.892005.

49 Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020. doi:10.1126/science.abe8770.

Random Restrictions and PRGs for PTFs in Gaussian Space

Zander Kelley
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Raghu Meka
Department of Computer Science, University of California, Los Angeles, CA, USA

Abstract

A polynomial threshold function (PTF) $f : \mathbb{R}^n \to \mathbb{R}$ is a function of the form $f(x) = \mathsf{sign}(p(x))$ where p is a polynomial of degree at most d. PTFs are a classical and well-studied complexity class with applications across complexity theory, learning theory, approximation theory, quantum complexity and more. We address the question of designing pseudorandom generators (PRGs) for polynomial threshold functions (PTFs) in the gaussian space: design a PRG that takes a seed of few bits of randomness and outputs a n-dimensional vector whose distribution is indistinguishable from a standard multivariate gaussian by a degree d PTF.

Our main result is a PRG that takes a seed of $d^{O(1)} \log(n/\varepsilon) \log(1/\varepsilon)/\varepsilon^2$ random bits with output that cannot be distinguished from an n-dimensional gaussian distribution with advantage better than ε by degree d PTFs. The best previous generator due to O'Donnell, Servedio, and Tan (STOC'20) had a quasi-polynomial dependence (i.e., seed length of $d^{O(\log d)}$) in the degree d. Along the way we prove a few nearly-tight structural properties of *restrictions* of PTFs that may be of independent interest.

Similar results were obtained in [15] (independently and concurrently).[1]

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases polynomial threshold function, pseudorandom generator, multivariate gaussian

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.21

Related Version *Previous Version*: https://arxiv.org/abs/2103.14134

Funding *Zander Kelley*: Supported by NSF grants CCF-1755921 and CCF-1814788.
Raghu Meka: Supported by NSF Career Award 1553605 and NSF AF 2007682.

1 Introduction

Polynomial threshold functions (PTFs) are a classical and well-studied class of functions with several applications in complexity theory, learning theory, theory of approximation, and more. Here we study the question of designing *pseudorandom generators* (PRGs) that fool test functions that are PTFs. We first start with some standard definitions. Let $\mathsf{sign} : \mathbb{R} \to \mathbb{R}$ be defined as $\mathsf{sign}(z) = 1$ if $z \geq 0$ and 0 otherwise.

▶ **Definition 1.** *For an integer $d > 0$, a degree d PTF $f : \mathbb{R}^n \to \{0, 1\}$ is a function of the form $f(x) = \mathsf{sign}(p(x))$, where $p : \mathbb{R}^n \to \mathbb{R}$ is a polynomial of degree at most d.*

Our goal is to design a PRG that takes few random bits and outputs a high-dimensional vector whose distribution is indistinguishable from a standard multivariate gaussian by any low-degree PTF. Specifically:

[1] A version of our manuscript with the same proofs and results as in this submission appeared on arxiv on March 25, 2021. There are, however, differences in the technical overview and exposition (among other things, we've incorporated feedback from others within this submission).

▶ **Definition 2.** *A function $G : \{0,1\}^r \to \mathbb{R}^n$ is a pseudorandom generator for degree d PTFs with error ε if for every degree at most d PTF $f : \mathbb{R}^n \to \{0,1\}$,*

$$\left| \mathop{\mathbb{P}}_{y \in_u \{0,1\}^r} (f(G(y)) = 1) - \mathop{\mathbb{P}}_{x \sim N(0,1)^n} (f(x) = 1) \right| \leq \varepsilon.$$

We call r the seed length of the generator and say G ε-fools degree d PTFs with respect to the gaussian distribution [2]*. We say G is explicit if its output can be computed in time polynomial in n.*

Of particular interest is the *boolean case* where the target distribution is not gaussian but the uniform distribution on the hypercube $\{+1,-1\}^n$. The gaussian case is interesting by itself both from a complexity-theoretic view as well as a geometric one. For instance, a PRG as above can be used to get deterministic algorithms for approximating the gaussian volumes of polynomial surfaces. Further, the gaussian case is a necessary stepping-stone to obtaining PRGs in the Boolean case: a PRG for the latter implies a PRG for the gaussian case. Achieving similar parameters as we do for the boolean case would be a significant achievement: we do not even have non-trivial correlation lower bounds for NP against PTFs of degree $\omega(\log n)$ over the hypercube. A PRG would at the very least imply correlation lower bounds against a function in NP, resolving a longstanding bottleneck in circuit complexity.

Over the last several years, the question of designing PRGs for PTFs has received much attention. Non-explicitly (i.e., the generator is not necessarily efficiently computable), by the probabilistic method, it is known that there exists PRGs that ε-fool degree d PTFs with seed-length $O(d \log n + \log(1/\varepsilon))$. Meka and Zuckerman [12] gave the first non-trivial PRG for bounded degree PTFs with a seed length of $d^{O(d)} \log(n)/\varepsilon^2$ for the boolean and gaussian cases. Independent of [12], [2] showed that bounded independence fools degree-2 PTFs leading to seed length $O(\log(n)/\varepsilon^2)$. Since then, there have been several other works that make progress on the gaussian case [7, 6, 8, 10, 11]. The seed length in all of these works had an exponential dependence on the degree d of the PTF. In particular, until recently no non-trivial PRGs (i.e., seed length $o(n)$) were known for PTFs of degree $\omega(\log n)$. In a remarkable recent work, O'Donnell, Servedio, and Tan [14] got around this exponential dependence on the degree d, achieving a seed length of $(d/\varepsilon)^{O(\log d)} \log(n)$. Our work builds on their work (which in turn builds on a framework of [7]).

1.1 Main Results

Our main result is a PRG with seed length $(d/\varepsilon)^{O(1)} \log(n)$ that ε-fools n-variate degree-d PTFs:

▶ **Theorem 3** (PRG for PTFs)**.** *There exist constants c, C such that for all $\varepsilon > 0$ and $d \geq 1$, there exists an explicit PRG that ε-fools n-variate degree d PTFs with respect to the gaussian distribution with seed length $r(n,d,\varepsilon) = Cd^c \log(n/\varepsilon) \log(1/\varepsilon)/\varepsilon^2$.*

As remarked above, this is the first result with polynomial dependence on the degree for fooling PTFs against any distribution and gives the first non-trivial PRGs against PTFs of degree $n^{\Omega(1)}$. Previously, we could only handle degree at most $2^{O(\sqrt{\log n})}$.

Towards proving the above result, we develop several structural results on PTFs in the gaussian space that might be of independent interest. We expand on these later on. Briefly:

[2] Here, and henceforth, $y \in_u S$ denotes a uniformly random element from a multi-set S, and $N(0,1)$ represents the standard univariate gaussian distribution of variance 1.

- We show that the derivatives of a low-degree polynomial p, taken at a random point $x \sim N(0,1)^n$, are likely to have magnitudes $\|\nabla^k p(x)\|$ which grow slowly as k increases.
- We apply this fact to the study of random "gaussian restrictions" of a polynomial p,

$$p_{x,\lambda}(y) := p\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right),$$

  and conclude that for small enough λ, with high probability over $x \sim N(0,1)^n$, $p_{x,\lambda}(Y)$ becomes highly concentrated around its mean value μ when $Y \sim N(0,1)^n$, as quantified by a bound on the higher-moments $\mathbb{E}(p_{x,\lambda}(Y) - \mu)^R$.
- As this concentration result relies only on moment bounds, it extends easily to pseudorandom distributions Y over $\mathbb{R}^n$ which are k-moment-matching with $N(0,1)^n$, when $k \geq R \cdot \deg(p)$.

Note that the magnitudes of the derivatives $\nabla^k p_{x,\lambda}(0)$ (with respect to x) are the same as the magnitudes of the degree-k coefficients of $p_{x,\lambda}(y)$ (as a polynomial in y), up to a scaling factor of roughly $\lambda^{k/2}$. However, to obtain the moment bound, we must translate to the basis of Hermite polynomials and bound the degree-k coefficients with respect to this basis (rather than the standard basis). In contrast with our work, [14] derive coefficient-size bounds for the Hermite basis directly and work with it exclusively. However, there are some significant advantages in having the flexibility to work also within the standard basis which will become relevant later – mainly they are due to the fact that standard basis representations (or equivalently: derivatives) behave nicely under the scaling operator $p(t) \mapsto p(\gamma t)$. The Hermite-basis representation behaves poorly under scaling[3].

For an arbitrary fixed polynomial $p(t)$, a bound on the coefficient-sizes in one basis translates only to a fairly crude bound in the other basis[4]. Therefore, we come to the following rather technical contribution of our work which we would like to highlight: we find that, although it is rather painful to convert between bases while studying an arbitrary fixed polynomial, it is actually quite possible to do so when studying certain *average-case* behaviors of polynomials; for instance, to study the typical behavior of $p(x)$ in the neighborhood around a random point $x \sim N(0,1)^n$, or the typical moments of $p(\sqrt{1-\lambda}x + \sqrt{\lambda}Y)$, it is possible to pass freely between either polynomial basis, and we develop some simple tools for doing so. These tools appear to be new (at least with respect to the body of works on PTFs) and it seems likely that they could be helpful in future works.

Besides these structural results and technical contributions, we also manage to introduce some substantial simplifications to the analysis of the main PRG as compared to [14]. This is in part due to the flexibility we have to measure the *well-behavedness* of a polynomial p in the neighborhood around a point x directly via the derivatives at x, rather than indirectly by taking several Hermite expansions of p and other auxiliary polynomials (cf. *horizontal, diagonal mollifier checks* in [14]). We will expand on this in Section 2 when discussing our analysis, but we briefly summarize a few key points here.

- Following [7] and [14], the construction we analyze has the form $Z := \frac{1}{\sqrt{L}} \sum_{i=1}^{L} Y_i$, where each Y_i is a k-moment-matching gaussian. This can be thought of as the gaussian analogue of the boolean construction from [12], which pseudorandomly partitions the n input bits into L buckets, and then assigns the bits in each bucket using k-wise independence. This

[3] In contrast, the Hermite basis representation behaves nicely under the *noise operator*, $p(t) \mapsto \mathbb{E}_{x \sim N(0,1)^n} p(\sqrt{1-\lambda}x + \sqrt{\lambda}t)$.

[4] This is especially true in the current setting where we must control the *relative* sizes of the magnitudes of coefficients at degree k vs. $k+1$.

construction and its variants are by now the most widely-applied pseudorandom tool for fooling various "geometric" function classes including linear threshold functions and their generalizations (such as PTFs and intersections of halfspaces).

- A tempting first idea for analyzing Z is to apply a hybrid argument – this seems promising in light of the fact that for a low-degree polynomial, we know that $p\left(\sqrt{1-\frac{1}{L}}x+\sqrt{\frac{1}{L}}Y_i\right)$ should be highly-concentrated around its mean for typical x[5]. However, this naive idea fails quantitatively: The probability that we have good behavior at x is in general not smaller than $\sqrt{1/L}$, so we cannot afford a union-bound over L events as required by the standard hybrid argument. Remarkably in [7], Kane shows how to address this obstacle with a clever sandwiching argument which in some sense mimics the hybrid argument but manages to pay for the error caused by "bad points" x only once rather than L times.
- However, one drawback of Kane's analysis is that its implementation is highly elaborate. After the framework was extended by [14] to break the $\log(n)$-degree barrier, the complexity only increased and the details of the argument became only more specialized and technical[6]. Given the wide applicability of the aforementioned pseudorandom construction and its variants, it would be highly desirable to have a lean and more transparent analysis which might better serve as a flexible starting point for future adaptations. We propose that in this work, we do obtain such an analysis.

PTFs simplify under restrictions

As a byproduct of our analysis, we obtain a structural result on PTFs that is similar in spirit to the celebrated *switching lemmas* that show that certain classes of functions simplify significantly under random restrictions. Switching lemmas and random restrictions are a cornerstone in complexity theory, and are one of the main methods we have for proving lower bounds. We prove analogous results with nearly optimal parameters for the important class of PTFs in the continuous space.

In the *boolean case*, i.e., when studying distributions on the hypercube $\{+1,-1\}^n$, a *restriction* is a partial assignment of the form $\rho\in\{+1,-1,*\}^n$ with the understanding that the $*$-variables are left free. Typically, restrictions ρ as above are parametrized by some $\lambda>0$, the fraction of $*$'s.

Here, we study analogues of the above results in the continuous world, where the inputs are coming from the standard gaussian distribution. The first question however is what should the analogue of random restrictions be in the continuous space? As it turns out, adopting the usual interpretation (where some coordinates are fixed and some are free) is not a natural one to study in the continuous space especially for PTFs[7].

The answer comes from the work of [7] (further developed in [14]) who introduced the notion of a *zoom* of a polynomial. To draw a clearer parallel with random restrictions, we term these *gaussian restrictions*:

▶ **Definition 4.** *Given a function $p:\mathbb{R}^n\to\mathbb{R}$ and $x\in\mathbb{R}^n$, and a* restriction *parameter $\lambda\in(0,1)$, let $p_{x,\lambda}:\mathbb{R}^n\to\mathbb{R}$ be*[8] *the function $p_{x,\lambda}(y)=p(\sqrt{1-\lambda}x+\sqrt{\lambda}y)$.*

Intuitively, we can view $p_{x,\lambda}$ as a restriction where $(1-\lambda)$-fraction of the *variance* is already *fixed*. (Note that for independent $x,y\sim N(0,1)^n$, $\sqrt{1-\lambda}x+\sqrt{\lambda}y$ is distributed as $N(0,1)^n$.)

---

[5] For a quantitative version of this statement, see Lemma 13.
[6] Refer to [15], which fills in several details absent in [14], to see the full scope of the argument.
[7] One reason is that the class of PTFs is invariant under linear transformations, so it would be nice to have our notion of restrictions also have some symmetry under linear transformations.
[8] As the value of λ will often be clear, we will often in fact just use p_x for brevity.

We show that PTFs simplify significantly, i.e., become essentially constant, under *gaussian restrictions* for $\lambda \ll 1/d^6$.

▶ **Theorem 5.** *There is a constant $C > 0$ such that the following holds. For any $\delta, \varepsilon > 0$, if*
- *$f : \mathbb{R}^n \to \{0,1\}$ is a PTF of degree d, and*
- *$\lambda \leq C\frac{\delta^2}{d^6 \log(1/\varepsilon)}$,*

then with probability at least $1-\delta$ over $x \sim N(0,1)^n$, the gaussian restriction of the PTF $(f_{x,\lambda})$ is nearly fixed to a constant*: for some $b \in \{0,1\}$ we have*

$$\mathbb{P}_{y \sim N(0,1)^n}[f_{x,\lambda}(y) = b] > 1 - \varepsilon.$$

The work of [14] achieves a similar conclusion but when the restriction parameter is $\lambda = d^{-O(\log d)}$ as opposed to being polynomially small as above. This improved significantly on the work of [7] that implicitly shows a similar claim for $\lambda = 2^{-O(d)}$.

We remark that in a related line of work, [1, 4, 3, 5] study random restrictions of PTFs over the hypercube. Our focus here is on gaussian restrictions and obtaining stronger bounds quantitatively: these works had exponential dependence on the degree d.

Slow-growth of derivatives

The analysis of the PRG (Theorem 21) and the random restriction statement above (Theorem 5) rely crucially on a claim about the magnitude of the derivatives of a polynomial evaluated at random gaussian input which may itself be of independent interest (and can be stated in a self-contained way).

For a function $p : \mathbb{R}^n \to \mathbb{R}$, let $\|\nabla^k p(x)\|^2$ denote the sum of squares of all partial derivatives of p of order k at x. That is, $\|\nabla^k p(x)\|$ is the Frobenius norm of the tensor of k'th order partial derivatives of p. We show that for any degree d polynomial p, the Frobenius-norm of the k'th order derivatives are comparable to the $(k-1)$'th order derivatives on a random gaussian input with high probability:

▶ **Lemma 6.** *For any degree-d polynomial $p : \mathbb{R}^d \to \mathbb{R}$, and $x \sim N(0,1)^n$, the following holds with probability at least $1-\delta$:*

$$\|\nabla^k p(x)\| \leq O(d^3/\delta)\|\nabla^{k-1} p(x)\|, \quad \text{for all } 1 \leq k \leq d. \tag{1}$$

Note that the above lemma is tight up to the factor of $O(d^2)$: consider the example $p(x) = x_1^d$.

Independent and concurrent work

Independently and concurrent to our work, [15] (following up on [14]) also obtained similar results to Theorem 3. They first obtained an analogue of *hypervariance reduction* (cf., Lemma 11) as studied in [14] with better parameters and combined the improved hypervariance reduction lemma with the framework of [14] to yield a PRG with $d^{O(1)}$ dependence on the degree d.

Our approach differs in that we critically use our new bounds on the growth of derivatives of polynomials as in Lemma 6 (instead of Lemma 11 which follows from Lemma 6). Working with the derivatives directly allows us to get a substantially simpler analysis of the main PRG construction compared to [14, 15].

2 Proof Overview

We first describe the high-level ideas underlying our main PRG construction - the proof of Theorem 21. We then describe the main idea behind the proof of Lemma 6 which is critical in being able to handle PTFs of polynomially large degree. The proof of Lemma 6 is quite different from the approach taken in [7, 14] to prove analogous results in their analysis.

2.1 Analysis of the PRG

We will use the same generator as in [7], and the high-level strategy is similar in spirit to that of [7, 14]. However, we introduce several additional ingredients that exploit Lemma 6 and significantly simplify the analysis.

As in the works of [7] and [14], the PRG output will be

$$Z := \frac{1}{\sqrt{L}} \sum_{i=1}^{L} Y_i,$$

where each Y_i is an independent k-moment-matching gaussian vector with $k = d^{\Theta(1)}$. For the time being let us work under the idealized assumption that each Y_i is exactly k-moment-matching with a standard gaussian: i.e., for any polynomial $h : \mathbb{R}^n \to \mathbb{R}$ of degree at most k, $\mathbb{E}[h(Y_i)] = \mathbb{E}_{z \sim N(0,1)^n}[h(z)]$. We will later relax this condition without too much additional work as is now standard (see Section 3 for details), and ultimately output a discrete approximation to Z with finite support. For now, it is appropriate to imagine that the seed length required for generating each Y_i will be roughly $O(k \log n)$; the total seed length will thus be $L \cdot O(k \log n)$. We improve prior works by showing that it suffices to let $L = d^{\Theta(1)}$, rather than $L = 2^{\Theta(d)}$ as in [7] or $L = d^{\Theta(\log d)}$ as in [14].

For the rest of this section, fix a degree d polynomial $p : \mathbb{R}^n \to \mathbb{R}$ and let $f : \mathbb{R}^n \to \{0, 1\}$ defined as $f(x) = \mathsf{sign}(p(x))$ be the corresponding PTF we are trying to fool. For simplicity in this introduction, we consider the case where p is multi-linear. The general case is similar but is slightly more nuanced.

We wish to compare $\mathbb{E}_Z[f(Z)]$ to $\mathbb{E}_z[f(z)]$ where $z \sim N(0,1)^n$. Note that we can rewrite $z \sim N(0,1)^n$ as $z := \frac{1}{\sqrt{L}} \sum_{i=1}^{L} y_i$ where each y_i is an independent standard gaussian.

First attempt: A hybrid argument

A natural approach to analyze the PRG is to use a hybrid argument by replacing each y_i with a k-moment matching Gaussian vector Y_i as in our PRG output. That is, show the following sequence of inequalities:

$$\begin{aligned}\mathbb{E}\left[f\left(\frac{y_1}{\sqrt{L}} + \frac{y_2}{\sqrt{L}} + \cdots + \frac{y_L}{\sqrt{L}}\right)\right] &\approx \mathbb{E}\left[f\left(\frac{Y_1}{\sqrt{L}} + \frac{y_2}{\sqrt{L}} + \cdots + \frac{y_L}{\sqrt{L}}\right)\right] \\ &\approx \mathbb{E}\left[f\left(\frac{Y_1}{\sqrt{L}} + \frac{Y_2}{\sqrt{L}} + \cdots + \frac{y_L}{\sqrt{L}}\right)\right] \approx \cdots \approx \mathbb{E}\left[f\left(\frac{Y_1}{\sqrt{L}} + \frac{Y_2}{\sqrt{L}} + \cdots + \frac{Y_L}{\sqrt{L}}\right)\right]. \quad (2)\end{aligned}$$

Let $\lambda = 1/L$ and $y' = \sqrt{\lambda}(y_2 + \cdots + y_L)$. Note that $y' \sim N(0, 1-\lambda)^n$. The first inequality in the sequence above, corresponding to a single-step of the hybrid argument is, equivalent to showing:

$$\mathbb{E}\left[f(\sqrt{\lambda} y_1 + y')\right] \approx \mathbb{E}\left[f(\sqrt{\lambda} Y_1 + y')\right].$$

In other words, the above inequality is asking to show that

$$\mathbb{E}[f_{y'/\sqrt{1-\lambda}}(y_1)] \approx \mathbb{E}[f_{y'/\sqrt{1-\lambda}}(Y_1)].$$

Intuitively, this is equivalent to showing that k-moment matching gaussians fool gaussian restrictions of a PTF with high probability over the restriction. Indeed, such a claim follows from our bounds on the derivatives of polynomials at random evaluation points (Lemma 6).

We say that a polynomial p is *well-behaved* at a point x if

$$\|\nabla^{k+1} p(x)\| \leq (1/\varepsilon)\|\nabla^k p(x)\| \text{ for all } k = 0, 1, \ldots, d-1,$$

where ε is a parameter that will be set to be slightly larger than $\sqrt{\lambda}$. We say p is poorly-behaved at x if the above condition does not hold.

The starting point of the analysis is that if p is well-behaved at x, then $\mathsf{sign}(p(x+\sqrt{\lambda}Y))$ is fooled by a moment-matching Y with *very good* error:

▶ **Proposition 7** (Direct Corollary of Lemma 13)**.** *Let $q : \mathbb{R}^n \to \mathbb{R}$ be a degree d multi-linear polynomial and suppose that q is well-behaved at a point x. Let $R = \varepsilon^2/\lambda$. Then, for $y \sim N(0,1)^n$ and Y a dR-moment matching gaussian,*

$$\mathop{\mathbb{E}}_{y\sim N(0,1)^n}[\mathsf{sign}(q(x+\sqrt{\lambda}y)] - \mathop{\mathbb{E}}_{Y}[\mathsf{sign}(q(x+\sqrt{\lambda}Y)] \leq 2^{-\Omega(R)}.$$

This fact follows from the following argument. Since q is well-behaved at x, this in particular implies a non-negligible lower bound on the size of the constant term c of $h(t) := q(x+\sqrt{\lambda}t)$, relative to its other coefficients. In particular, $\mathsf{sign}(h(t))$ is *nearly fixed to a constant* in the sense of Theorem 5. Indeed, writing $h(t) = c + (h(t) - c)$, we see that $\mathsf{sign}(h(t))$ can only differ from $\mathsf{sign}(c)$ if we have a deviation with magnitude at least $|h(t) - c| \geq |c|$. We can use a concentration inequality to bound the probability that either $|h(y) - c| \geq |c|$ or $|h(Y) - c| \geq |c|$. In light of the bounds on $\|\nabla^k q(x)\|$, such a concentration inequality follows from moment bounds obtained from *hypercontractivity*.

The above lemma shows the first step of the hybrid argument and suggests the following strategy for analyzing the PRG. Define $Z_{-i} = Z - \sqrt{\lambda}Y_i$. We can now aim to show that the polynomial p is well-behaved at Z_{-i} with high probability. This indeed seems plausible as our Lemma 6 indeed shows that when Z is standard gaussian, the polynomial p is well-behaved at Z with high probability.

Immediately, there are two obstacles for this approach:

- First, Lemma 6 works only for truly random gaussian and not for our pseudorandom Z_{-i}.
- Second, even if we argue that p is likely to be well-behaved at Z_{-i}, we cannot apply a union bound over i. The error guarantee in Lemma 6, is $\gg \sqrt{\lambda}$; whereas, we have $L = 1/\lambda$ choices of i, so we cannot use such a straightforward union-bound argument to replace each Y_i with a y_i.

The second issue is especially problematic as the error probability in Lemma 6 cannot be improved, at least in that variant; the probability that the derivatives don't grow too fast is not small compared to $L = 1/\lambda$.

Beating the union bound

Roughly speaking, the main insight in going beyond the *union bound* obstacle mentioned above is as follows. There are two sources of error in the naive hybrid argument outlined above: (1) The probability of failure coming from p being poorly-behaved at the points Z_{-i}. (2) The error coming from applying Proposition 7 to replace a Y_i with y_i when p is well-behaved at Z_{-i}.

Note that we have very good control on the error of type (2) above: we could make it be much smaller than $1/L$ by increasing the amount of independence k. We will exploit this critically. We will complement this by showing that even though a naive union bound would be bad for errors of type (1) above, it turns out that we don't have to incur this loss: we (implicitly) show that $\mathbb{P}(\forall i,\ p \text{ is well-behaved at } Z_{-i}) \approx 1 - O(\varepsilon d^3)$. We do so by checking only that p is well-behaved at the single point Z (in a slightly stronger sense) and then we conclude that p is also highly-likely to be well-behaved at each of the "nearby" points Z_{-i}. Intuitively, this is what allows us to circumvent the union bound in the hybrid argument. However, it would be difficult to actually carry out the analysis as stated this way – we use a sandwiching argument to sidestep the complicated conditionings which would arise in this argument as stated.

We proceed to describe the sandwiching argument. We wish to lower-bound the PTF $\mathsf{sign}(p(x))$ by $\mathsf{sign}(p(x)) \cdot g(x)$, where $g(x)$ is some "mollifier" function taking values in $[0,1]$. The role of $g(x)$ is roughly to "test" whether p is well-behaved at x; we ideally want $g(x) = 1$ at points x where p is well-behaved and $g(x) = 0$ at points x where p is poorly-behaved. However, we also need $g(x)$ to be smooth[9], so there will be some intermediate region of points for which $g(x)$ yields a non-informative, non-boolean value.

We set $g(x)$ to be a smoothed version of the indicator function

$$g(x) \approx \prod_{k=0}^{d-1} \mathbb{1}\left(\|\nabla^{k+1} p(x)\| \le \frac{1}{\varepsilon} \|\nabla^k p(x)\| \right),$$

which tests whether the derivatives of p at x have controlled growth in the sense of Lemma 6. More specifically, we set

$$g(x) := \prod_{k=0}^{d-1} \rho\left(\log\left(\frac{1}{16\varepsilon^2} \frac{\|\nabla^k p(x)\|^2}{\|\nabla^{k+1} p(x)\|^2} \right) \right),$$

where $\rho(t) : \mathbb{R} \to [0,1]$ is some smooth univariate function with $\rho(t) = 0$ for $t \le 0$ and $\rho(t) = 1$ for $t \ge 1$.

Now, for every point $x \in \mathbb{R}^n$ we have

$$\mathsf{sign}(p(x)) \ge \mathsf{sign}(p(x)) g(x).$$

Furthermore, under truly-random gaussian inputs $z \sim N(0,1)^n$ we have

$$\mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) g(z) \ge \mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) - \mathop{\mathbb{E}}_z |g(z) - 1| \ge \mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) - O(\varepsilon d^3),$$

where the final inequality here follows from Lemma 6. Combining these, we get that

$$\mathop{\mathbb{E}}_Z \mathsf{sign}(p(Z)) \ge \mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) - O(\varepsilon d^3) - \left| \mathop{\mathbb{E}}_Z \mathsf{sign}(p(Z)) g(Z) - \mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) g(z) \right|.$$

Note that we can similarly obtain an upper-bound for $\mathbb{E}_Z \mathsf{sign}(p(Z))$ by repeating this argument on the polynomial $-p(x)$.

Thus, it suffices to bound $|\mathbb{E}_Z \mathsf{sign}(p(Z)) g(Z) - \mathbb{E}_z \mathsf{sign}(p(z)) g(z)|$. Having introduced the mollifier, we can now afford to do so by a standard hybrid argument. We represent z as $z := \frac{1}{\sqrt{L}} \sum_{i=1}^{L} y_i$ and recall that Z is of the form $Z = \frac{1}{\sqrt{L}} \sum_{i=1}^{L} Y_i$. We can replace each Y_i with y_i and get

$$\left| \mathop{\mathbb{E}}_Z \mathsf{sign}(p(Z)) g(Z) - \mathop{\mathbb{E}}_z \mathsf{sign}(p(z)) g(z) \right| \le \gamma L,$$

where γ is the (quite small) error coming from the following lemma.

[9] Ultimately we need g to be well approximated by some polynomial so that it can be fooled by limited independence. Smoothness will allow us to get such an approximation by truncating the Taylor series.

▶ **Lemma 8.** *There exists a constant c such that the following holds for $\lambda \le \varepsilon^2/Rd^c$. For any fixed vector $x \in \mathbb{R}^n$, Y a dR-moment-matching gaussian vector, and $y \sim N(0,1)^n$,*

$$|\mathop{\mathbb{E}}_{Y} \mathsf{sign}(p(x+\sqrt{\lambda}Y))g(x+\sqrt{\lambda}Y) - \mathop{\mathbb{E}}_{y} \mathsf{sign}(p(x+\sqrt{\lambda}y))g(x+\sqrt{\lambda}y)| \le \gamma = 2^{-\Omega(R)}.$$

Technically speaking, the above lemma is where our intuition on going around the union bound is quantified, allowing us to use the hybrid argument. We briefly outline our proof of this lemma, where for the purpose of illustration we continue with the simplifying assumption that the polynomial p is multilinear.

The proof is by a case analysis on the behavior of p at the the fixed point x. In the multilinear case it suffices to consider the derivatives $\nabla^k p(x)$; in the general case we need to consider something slightly different.

- Case 1: p is well-behaved at x, i.e., $\|\nabla^{k+1}p(x)\| \le (1/\varepsilon)\|\nabla^k p(x)\|$ for all k.
  - We can use Lemma 13 in this case to conclude that $\mathsf{sign}(p(x+\sqrt{\lambda}y))$, $\mathsf{sign}(p(x+\sqrt{\lambda}Y))$ are both almost constant with error $2^{-\Omega(R)}$.
  - So, it remains to show that Y fools $g(x+\sqrt{\lambda}y)$. We approximate g by a low-degree polynomial in y using a Taylor-truncation argument. Our assumption on the controlled growth of derivatives $\|\nabla^k p(x)\|$ allows us to bound the truncation error by bounding the higher-moments of the deviations $\|\nabla^k p(x+\sqrt{\lambda}Y)\| - \|\nabla^k p(x)\|$.
- Case 2: p is not well-behaved at x; let k_0 be the largest k such that $\|\nabla^{k_0+1}p(x)\| > (1/\varepsilon)\|\nabla^{k_0}p(x)\|$.
  - Intuitively, this says that the polynomial p is well behaved at degree above k_0, but not at degree k_0. This allows us to show, via an R-th moment bound, that both
    * $\|\nabla^{k_0}p(x+\sqrt{\lambda}Y)\| \le 2\varepsilon\|\nabla^{k_0+1}p(x)\|$
    * $\|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)\| \ge \frac{1}{2}\|\nabla^{k_0+1}p(x)\|$

    are highly likely. Thus, it is highly likely that

    $$\|\nabla^{k_0}p(x+\sqrt{\lambda}Y)\| \le 4\varepsilon\|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)\|.$$

    The latter means p is still sufficiently poorly-behaved at the point $x+\sqrt{\lambda}Y$ that the mollifier classifies it correctly as $g(x+\sqrt{\lambda}Y) = 0$.

2.2 Slow-growth of derivatives and simplification under restrictions

The proof of Lemma 6 is iterative and is relatively simple given Kane's *relative anti-concentration inequality* for degree d polynomials [9] developed in the context of studying the *Gotsman-Linial* conjecture for PTFs.

[9] shows that for any degree d polynomial, and $x, y \sim N(0,1)^n$ with probability at least $1-\delta$, we have $|\langle y, \nabla p(x)\rangle| \le (d^2/\delta)|p(x)|$. As y in the above statement is independent of x, for any x, $\langle y, \nabla p(x)\rangle$ is distributed as $N(0, \|\nabla p(x)\|^2)$. This says that the inequality is essentially equivalent to saying that with probability at least $1-\delta$ over x, we have $\|\nabla p(x)\|^2 \le O(d^2/\delta)|p(x)|$. The latter can be seen as the inequality corresponding to $k=1$ in the statement of Lemma 6. The full proof of the lemma is via iteratively applying the above argument using a vector-valued generalization of Kane's inequality.

Next, it is not too hard to prove Theorem 5 given Lemma 6. For illustration, suppose that we have a degree d multi-linear polynomial p, and write $f(t) := p(\sqrt{1-\lambda}t)$. Then, by elementary algebra[10], we have the identity

$$p_x(y) = p\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = \sum_{\alpha} \partial^\alpha f(x) \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} y^\alpha. \tag{3}$$

[10] If p is multi-linear, then the Hermite expansion (see Section 3) is just $p(x) = \sum_{\alpha\in\{0,1\}^n} \hat{p}(\alpha)h_\alpha(x) = \sum_{I\subseteq[n]} \hat{p}(I)\prod_{i\in I} x_i$. We can prove the identity for each monomial and use additivity.

Now, by Lemma 6, with probability $1-\delta$ over x, we have $\|\nabla^k f(x)\| \le O(d^3/\delta)\|\nabla^{k-1} f(x)\|$, for all k. Thus, if we take $\lambda \ll \delta^2/(R^2 d^6)$, the factor of λ will kill the growing derivatives leading to a bound on the higher-order moments of $p_x(y) - p_x(0)$ via hypercontractivity. These moment bounds in turn imply that $|p_x(y) - p_x(0)| < |p_x(0)|$ with high probability over y, and hence that $\mathsf{sign}(p_x(y)) = \mathsf{sign}(p_x(0))$ with high probability over y.

Notice that Equation (3) is essentially a Taylor expansion of p at $\sqrt{1-\lambda}x$: it expresses the function $p_x(y)$ as a polynomial in y in the standard basis, whose coefficients are determined by the derivatives of p at $\sqrt{1-\lambda}x$. We want to do something similar in the general case, but in the Hermite basis; for non-multi-linear polynomials these two bases no longer coincide. So, in the general case, we rely on the following identity, which we regard as an analogue of the Taylor expansion for the Hermite basis.

▶ **Lemma 9** (See Section 3). *Let $f(y) = \sum_\alpha \hat{f}(\alpha) h_\alpha(y)$. Then*

$$f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = \sum_\alpha \frac{\partial^\alpha g(x)}{\sqrt{\alpha!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} h_\alpha(y),$$

where $g(x) := U_{\sqrt{1-\lambda}} f(x) = \sum_\alpha \hat{f}(\alpha)(1-\lambda)^{|\alpha|/2} h_\alpha(x)$.

Hermite polynomials are such a ubiquitous tool used in such a wide range of fields that it seems unlikely that such an identity is new. However, we are not aware of any previous appearance of such an identity in the literature (at least in the body of work on PTFs) and we provide a proof.

Hypervariance reduction

We next remark on the relation between *slow-growth of derivatives* (as in Lemma 6) and *hypervariance reduction* as studied and introduced in [14]. The latter plays a similar role in their paper as the former does in this work. However, Lemma 6 importantly has only polynomial dependence on the degree d and is also much more conducive to our analysis of the PRG.

Recall the Hermite expansion (see Section 3) of polynomials: A degree d polynomial $p : \mathbb{R}^n \to \mathbb{R}$ can be uniquely expressed as

$$p(y) := \sum_{|\alpha| \le d} \hat{p}(\alpha) h_\alpha(y),$$

where $\alpha \in \mathbb{N}^n$ denotes a multi-index and $h_\alpha(y)$ is the α'th Hermite polynomial. The *hypervariance* and *normalized hypervariance* of a polynomial introduced in [14] are defined as follows:

▶ **Definition 10.** *For a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ of the form $p(y) := \sum_\alpha \hat{p}(\alpha) h_\alpha(y)$, define its* hypervariance*,* $\mathrm{HyperVar}_R(\,)$*, and* normalized hypervariance*, $H_R(\,)$, as*

$$\mathrm{HyperVar}_R(p) := \sum_{\alpha \ne 0} \hat{p}(\alpha)^2 R^{2|\alpha|}, \quad H_R(p) := \frac{\mathrm{HyperVar}_R(p)}{\hat{p}(0)^2}.$$

Intuitively, if the normalized hypervariance $H_R(p)$ of a polynomial is small for a large R, then it means that the *weights* of the higher-order Hermite coefficients of p have a geometric decay.

[14] showed that for any polynomial p, for a suitable $\lambda > 0$, a gaussian restriction of p will have small normalized hypervariance with high probability. Specifically, they showed that if $\lambda = d^{-O(\log d)}$, then $H_R(p_{x,\lambda})$ is bounded with high probability over $x \sim N(0,1)^n$. They also asked whether this property holds when $\lambda = d^{-O(1)}$ instead of being quasi-polynomially small in d. Lemma 6 implies this conjecture without too much difficulty:

▶ **Lemma 11.** *For any degree d polynomial p and $\lambda, \delta > 0$, the following holds. Except with probability δ over $x \sim N(0,1)^n$, the normalized hypervariance $H_R(p_{x,\lambda}) = O(\lambda d^6 R^2/\delta^2)$.*

The proof of the analogue of Lemma 11 for quasi-polynomially small λ (i.e. $\lambda = d^{-O(\log d)}$) in [14] was by an iterative process: Intuitively, if one sets $\lambda_0 = d^{-O(1)}$, and $\lambda = \lambda_0^{\log d}$, then the random restriction $p_{\lambda,x}$ is equivalent to $(\log d)$ independent random restrictions with restriction parameter λ_0. The authors in [14] show that each such λ_0-restriction (essentially) decreases the degree by a factor of 2. We instead take a different approach by drawing a connection between norms of derivatives and to *relative anti-concentration* as developed in the context of studying the *Gotsman-Linial* conjecture for PTFs.

3 Preliminaries

The pseudorandom generator construction: idealization vs. discretization. Following [7] and [14], we analyze the idealized pseudorandom distribution

$$Z = \frac{1}{\sqrt{L}} \sum_{i=1}^{L} Y_i,$$

where each $Y_i \in \mathbb{R}^n$ is a k-moment-matching gaussian (that is, $\mathbb{E}[p(Y_i)] = \mathbb{E}_{x \sim N(0,1)^n}[p(x)]$ for all polynomials $p : \mathbb{R}^n \to \mathbb{R}$ of degree at most k).

Suppose that, for any such Z with parameters (L, k), it is the case that Z fools degree-d PTFs with error $\varepsilon = \varepsilon(L, k, d)$. Then, it is shown in [7] how to obtain a small-seed length PRG (in the sense of Definition 2) by providing a specific instantiation and discretization of this construction.

▶ **Theorem 12** ([7], implicit in Section 6)**.** *Suppose a Z as above with parameters (L, k) fools degree d-PTFs with error $\varepsilon = \varepsilon(L, k, d)$. Then, there is an explicit, efficiently computable PRG with seed length $O(dkL \log(ndL/\varepsilon)$ that (2ε)-fools degree d PTFs.*

Hermite polynomials. To argue about polynomials which are not necessarily multilinear, we need some simple facts concerning Hermite polynomials. For our purposes, Hermite polynomials are simply a convenient choice of polynomial basis which have nice properties (in particular being *orthonormal*) with respect to gaussian inputs. For a more detailed background on Hermite polynomials and their use for analyzing functions over gaussian space, see [13, Ch. 11].

One concrete way to define the Hermite polynomials is the following:

- For the univariate polynomials, the degree-m "Probabilist's" Hermite polynomial is the m-th coefficient of the generating function

$$e^{st - \frac{1}{2}s^2} = \sum_{m \geq 0} H_m(t) s^m.$$

- We define the degree-m univariate Hermite polynomial by the normalization

$$h_m(t) := \frac{1}{\sqrt{m!}} H_m(t).$$

- For a multi-index $\alpha \in \mathbb{N}^n$, we define the multivariate Hermite polynomial $h_\alpha : \mathbb{R}^n \to \mathbb{R}$ via the product

$$h_\alpha(x) := \prod_{i=1}^{n} h_{a_i}(x_i).$$

We record some basic properties of this particular choice of polynomial basis. The final two properties say that the Hermite basis is orthonormal with respect to correlation under the standard gaussian distribution – this is the reason for our choice of normalization.

- The set $\{h_\alpha(x) : |\alpha| \leq d\}$ is a basis for real polynomials in n variables of degree $\leq d$.
- h_0 is the constant polynomial $h_0 \equiv 1$.
- For multi-indicies $\alpha \in \{0,1\}^n$, $h_\alpha(x)$ is simply the monomial $\prod_{i:a_i=1} x_i$.
- For $x \sim N(0,1)^n$, and distinct multi-indices $\alpha \neq \beta$, $\mathbb{E}_x\, h_\alpha(x)h_\beta(x) = 0$.
- For $x \sim N(0,1)^n$, and any multi-index α, $\mathbb{E}_x\, h_\alpha(x)^2 = 1$.

Gaussian noise operator. We recall the definition of the noise operator U_ρ, which here we regard as an operator on real polynomials in n variables (see [13, Ch. 11] for background and a more general viewpoint). For a polynomial $f : \mathbb{R}^n \to \mathbb{R}$ and a parameter $\rho \in [0,1]$, the action of U_ρ on f is specified by

$$(U_\rho f)(x) := \mathop{\mathbb{E}}_{Z \sim N(0,1)^n} f\left(\rho x + \sqrt{1-\rho^2} Z\right).$$

An important feature of the Hermite basis is that the noise operator acts on it *diagonally* (see [13, Ch. 11]):

$$U_\rho h_\alpha(x) = \rho^{|\alpha|} h_\alpha(x).$$

Thus, if f is a degree-d polynomial given in the Hermite basis as

$$f(x) = \sum_{|\alpha| \leq d} \hat{f}(\alpha) h_\alpha(x),$$

then we can express the result of the noise operator applied to f explicitly as

$$U_\rho f(x) = \sum_{|\alpha| \leq d} \hat{f}(\alpha) \rho^{|\alpha|} h_\alpha(x).$$

Higher moments and hypercontractivity. Fix a polynomial

$$f(x) := \sum_{|\alpha| \leq d} \hat{f}(\alpha) h_\alpha(x).$$

For an even natural number $q \geq 2$, we write the gaussian q-norm of f as

$$\|f\|_q := \left(\mathop{\mathbb{E}}_{x \sim N(0,1)^n} f(x)^q \right)^{1/q}.$$

We wish to be able to bound this quantity in terms of the magnitudes of the Hermite coefficients of f, $\hat{f}(\alpha)$. For this purpose, we extend the definition of U_ρ also to $\rho > 1$ by its action on the Hermite basis: $U_\rho h_\alpha(x) = \rho^{|\alpha|} h_\alpha(x)$. With this notation, we can express the well-known $(q,2)$-hypercontractive inequality [13, Ch. 9,11] as

$$\|f\|_q \leq \|U_{\sqrt{q-1}} f\|_2,$$

which is quite convenient for us, as we can use orthonormality of the Hermite basis to explicitly compute

$$\|U_{\sqrt{q-1}} f\|_2^2 = \sum_{|\alpha| \leq d} (q-1)^{|\alpha|} \hat{f}(\alpha)^2 \leq \sum_{|\alpha| \leq d} q^{|\alpha|} \hat{f}(\alpha)^2.$$

To get a feel for the utility of this bound, let's see how it can be used to prove the following concentration bound:

▶ **Lemma 13.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a degree d polynomial with normalized hypervariance $H_{\sqrt{q}}(f) \le \frac{1}{4}$, where q is an even natural number. Then,*

$$\mathop{\mathbb{P}}_{y\sim N(0,1)^n}\Big(\mathsf{sign}(f(y)) \neq \mathsf{sign}(\hat{f}(0))\Big) \le 2^{-q}.$$

Further, the same holds more generally for $y \sim Y$, as long as the distribution Y is dq-moment-matching.

Proof. Suppose that $f(y)$ is normalized so that

$$\mathop{\mathbb{E}}_{y\sim N(0,1)^n} f(y) = \hat{f}(0) = \pm 1.$$

We have the q-th moment bound

$$\|f(x) - \hat{f}(0)\|_q \le \|U_{\sqrt{q}}\left(f(y) - \hat{f}(0)\right)\|_2 \le \tfrac{1}{2}.$$

From the generic concentration inequality

$$\mathbb{P}\left(|X| \ge t\|X\|_q\right) \le t^{-q}$$

we obtain

$$\mathbb{P}\Big(\mathsf{sign}(f(y)) \neq \mathsf{sign}(\hat{f}(0))\Big) \le 2^{-q}.$$

Thus, we find that the PTF $\mathsf{sign}(f)$ almost always yields the value $\mathsf{sign}(\hat{f}(0))$ under random gaussian inputs. Crucially for us, this argument is also *easy to derandomize*: since the argument merely relies on a bound on the q-th moment $\mathbb{E}_{y\sim N(0,1)^n}(f(y) - \hat{f}(0))^q$, and for Y which is k-moment-matching for $k \ge dq$ we have

$$\mathop{\mathbb{E}}_{Y}(f(Y) - \hat{f}(0))^q = \mathop{\mathbb{E}}_{y\sim N(0,1)^n}(f(y) - \hat{f}(0))^q,$$

we conclude also that $\mathsf{sign}(f(Y))$ is typically equal to $\mathsf{sign}(\hat{f}(0))$. ◀

We remark that this lemma further implies that Y fools $\mathsf{sign}(f)$ when $H_{\sqrt{q}}(f)$ is small:

$$\mathop{\mathbb{E}}_{Y}\mathsf{sign}(f(Y)) = \mathop{\mathbb{E}}_{y\sim N(0,1)^n}\mathsf{sign}(f(y)) \pm O(2^{-q}).$$

Gaussian restrictions and derivatives on the Hermite basis. Besides the effect of the noise operator, it will also be important to understand the effect of two further operations on polynomials:

- The derivative map, $f(y) \mapsto \partial^\alpha f(y)$.
- The gaussian restriction at x, $f(y) \mapsto f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right)$.

In particular, we are concerned with how these operations affect the Hermite coefficients of a polynomial; ultimately, our goal will be to develop a "Hermite-basis analogue" of the Taylor expansion which can be applied to expand $f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right)$ as a function of y. We start by computing the effect of these two operations on univariate Hermite polynomials, and then on the full multivariate Hermite basis, and finally on a general polynomial $f(x)$ expressed in the Hermite basis.

▶ **Proposition 14.** *For univariate Hermite polynomials, we have the identities*

- $\frac{\partial^k}{\partial t^k} h_m(t) = \sqrt{\frac{m!}{(m-k)!}} h_{m-k}(t)$,
- $h_m\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = \sum_{k=0}^{m} \sqrt{\binom{m}{k}} (1-\lambda)^{(m-k)/2} \lambda^{k/2} h_{m-k}(x) h_k(y)$.

Proof. The first of these identities is standard (see e.g. [13, Ex. 11.10]); we provide a proof of the second.

The second identity can be proved by considering the generating function

$$e^{st - \frac{1}{2}s^2} = \sum_m \sqrt{m!} h_m(t) s^m,$$

and comparing the coefficient of s^m on both sides of

$$e^{s(\sqrt{1-\lambda}x + \sqrt{\lambda}y) - \frac{1}{2}s^2} = e^{(s\sqrt{1-\lambda})x - \frac{1}{2}(s\sqrt{1-\lambda})^2} \cdot e^{(s\sqrt{\lambda})y - \frac{1}{2}(s\sqrt{\lambda})^2}$$ ◀

The corresponding identities for multivariate Hermite polynomials follow easily from above.

▶ **Proposition 15.** *We have*

- $\partial^\alpha h_\beta(y) = \sqrt{\frac{\alpha!}{\gamma!}} h_\gamma(y)$, *where* $\gamma = \beta - \alpha$,
- $h_\beta\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = (1-\lambda)^{|\beta|/2} \sum_{\alpha \le \beta} \frac{\partial^\alpha h_\beta(x)}{\sqrt{\alpha!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} h_\alpha(y)$,
- $\partial^\alpha h_\beta\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = (1-\lambda)^{|\beta-\alpha|/2} \sum_{\gamma \le \beta-\alpha} \frac{\partial^{\alpha+\gamma} h_\beta(x)}{\sqrt{\gamma!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\gamma|/2} h_\gamma(y)$.

We conclude with a Taylor-like expansion in the Hermite basis that we use repeatedly.

▶ **Lemma 16.** *Let* $f(y) = \sum_\alpha \hat{f}(\alpha) h_\alpha(y)$. *Then*

$$f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = \sum_\alpha \frac{\partial^\alpha g(x)}{\sqrt{\alpha!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} h_\alpha(y),$$

where $g(x) := U_{\sqrt{1-\lambda}} f(x) = \sum_\alpha \hat{f}(\alpha)(1-\lambda)^{|\alpha|/2} h_\alpha(x)$.

Proof. We express

$$\begin{aligned} f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) &= \sum_\alpha \hat{f}(\alpha) h_\alpha\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) \\ &= \sum_\alpha \frac{h_\alpha(y)}{\sqrt{\alpha!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} \sum_{\beta \ge \alpha} \hat{f}(\beta)(1-\lambda)^{|\beta|/2} \partial^\alpha h_\beta(x) \\ &= \sum_\alpha \frac{h_\alpha(y)}{\sqrt{\alpha!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2} \partial^\alpha g(x). \end{aligned}$$ ◀

Lastly, we will also need an extension of this theorem which expresses $\partial^\alpha f$, at the point

$$\sqrt{1-\lambda}x + \sqrt{\lambda}y,$$

as a polynomial in y in the Hermite basis.

▶ **Theorem 17.** *Let* $f(y) = \sum_\alpha \hat{f}(\alpha) h_\alpha(y)$. *Then*

$$\partial^\alpha f\left(\sqrt{1-\lambda}x + \sqrt{\lambda}y\right) = (1-\lambda)^{-|\alpha|/2} \sum_{\beta \ge \alpha} \partial^\beta g(x) \sqrt{\frac{\alpha!}{\beta!}} \left(\frac{\lambda}{1-\lambda}\right)^{|\beta-\alpha|/2} h_{\beta-\alpha}(y),$$

where $g(x) := U_{\sqrt{1-\lambda}} f(x)$.

Proof. We express

$$\begin{aligned}\partial^\alpha f\left(\sqrt{1-\lambda}x+\sqrt{\lambda}y\right) &= \sum_\beta \hat{f}(\beta)\partial^\alpha h_\beta\left(\sqrt{1-\lambda}x+\sqrt{\lambda}y\right)\\ &= \sum_\gamma \frac{h_\gamma(y)}{\sqrt{\gamma!}}\left(\frac{\lambda}{1-\lambda}\right)^{|\gamma|/2}\sum_{\beta\geq\gamma+\alpha}(1-\lambda)^{|\beta-\alpha|/2}\partial^{\alpha+\gamma}h_\beta(x)\\ &= (1-\lambda)^{-|\alpha|/2}\sum_\gamma \frac{h_\gamma(y)}{\sqrt{\gamma!}}\left(\frac{\lambda}{1-\lambda}\right)^{|\gamma|/2}\partial^{\alpha+\gamma}g(x).\end{aligned}$$

◀

4 Gaussian restrictions of polynomials

Here we prove the structural properties of gaussian restrictions of polynomials: Theorem 5, Lemma 6, Lemma 11. Note that Theorem 5 follows immediately from Lemma 11 and Lemma 13. We next prove Lemma 11 from Lemma 6.

Proof of Lemma 11 from Lemma 6. Define $f(x) := U_{\sqrt{1-\lambda}}p(x)$. Then, by Lemma 16,

$$p_x(y) = f(x) + \sum_{\alpha\neq 0}\frac{\partial^\alpha f(x)}{\sqrt{\alpha!}}\left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|/2}h_\alpha(y).$$

Thus,

$$\begin{aligned}\mathrm{HyperVar}_R(p_x) = \sum_{\alpha\neq 0}\left(\frac{\partial^\alpha f(x)}{\sqrt{\alpha!}}\right)^2\left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|}R^{2|\alpha|} &\leq \sum_{\alpha\neq 0}\left(\partial^\alpha f(x)\right)^2\left(\frac{\lambda}{1-\lambda}\right)^{|\alpha|}R^{2|\alpha|}\\ &= \sum_{k=1}^d R^{2k}\left(\frac{\lambda}{1-\lambda}\right)^k\|\nabla^k f(x)\|^2,\end{aligned}$$

where the first inequality follows as $\sqrt{\alpha!}\geq 1$.

We now conclude by applying Lemma 6 to f. We have

$$H_R(p_x) = \frac{\sum_{k=1}^d R^{2k}\left(\frac{\lambda}{1-\lambda}\right)^k\|\nabla^k f(x)\|^2}{f(x)^2}.$$

Except with probability δ over $x\sim N(0,1)^n$, we can bound this by

$$\sum_{k=1}^d R^{2k}\left(\frac{\lambda}{1-\lambda}\right)^k\left(\frac{Cd^3}{\delta}\right)^{2k}\leq O\left(\frac{\lambda d^6R^2}{\delta^2}\right).$$

◀

4.1 Proof of Lemma 6

Our main tool will be Kane's relative-anticoncentration lemma for gaussian polynomials.

▶ **Lemma 18** ([9]). *For a degree d polynomial p, and independent standard gaussian vectors $x,y\in\mathbb{R}^n$,*

$$\mathbb{P}(|p(x)|\leq\varepsilon|\left\langle y,\nabla p(x)\right\rangle|)\leq O(\varepsilon d^2).$$

In fact, we will actually work with the following corollary which is essentially the first of the d inequalities in Lemma 6.

▶ **Corollary 19.** *For a degree d polynomial p, and independent standard gaussian vector $x \in \mathbb{R}^n$,*

$$\mathbb{P}(|p(x)| \le \varepsilon \|\nabla p(x)\|) \le O(\varepsilon d^2).$$

Proof. We note that for any fixed x, $\langle y, \nabla p(x)\rangle$ is identical in distribution to $Z\|\nabla p(x)\|$, where $Z \sim N(0,1)$ is a standard gaussian. So, we express

$$\begin{aligned}\mathbb{P}(|p(x)| \le \varepsilon |\langle y, \nabla p(x)\rangle|) &= \mathbb{P}(|p(x)| \le \varepsilon |Z| \|\nabla p(x)\|) \\ &\ge \mathbb{P}(|p(x)| \le \varepsilon \|\nabla p(x)\|) \cdot \mathbb{P}(|Z| \ge 1)\,.\end{aligned}$$

Since $\mathbb{P}(|Z| \ge 1) \ge \Omega(1)$, we conclude that

$$\mathbb{P}(|p(x)| \le \varepsilon \|\nabla p(x)\|) \le O(\varepsilon d^2).$$ ◀

The heart of the proof of Lemma 6 is a vector-valued variant of the above corollary:

▶ **Lemma 20.** *Let $\vec{f}(x) := (f_1(x), f_2(x), \ldots, f_m(x))$ be a collection of m degree-at-most d polynomials $f_j(x)$. If $x \in \mathbb{R}^n$ is a standard gaussian vector, then*

$$\mathbb{P}\left(\|\vec{f}(x)\|^2 \le \varepsilon^2 \sum_{j=1}^{m} \|\nabla f_j(x)\|^2 \right) \le O(\varepsilon d^2).$$

Proof of Lemma 6. We simply apply the above lemma d times and take a union bound. For $1 \le k \le d$, let $\vec{f}_k(x) := ((\partial^\alpha f(x) : |\alpha| = k))$. Note that $\|\vec{f}_k(x)\|^2 = \|\nabla^k f(x)\|^2$. Further, note that

$$\sum_{\alpha : |\alpha| = k} \|\nabla(\partial^\alpha f(x))\|^2 \ge \|\nabla^{k+1} f(x)\|^2,$$

where the inequality follows as each $(k+1)$'th order derivative would be counted at least once in the expression on the left hand side. Therefore, by the above lemma, for $x \sim N(0,1)^n$, we have

$$\mathbb{P}\big(\|\nabla^k f(x)\|^2 \le \varepsilon^2 \|\nabla^{k+1} f(x)\|^2\big) \le O(\varepsilon d^2)$$

Setting $\varepsilon = \delta/d^3$, and taking a union bound over all k, we get that for a constant $C > 0$,

$$\mathbb{P}\big(\forall k, \|\nabla^k f(x)\|^2 > C(\delta^2/d^6)\|\nabla^{k+1} f(x)\|^2\big) \ge 1 - \delta.$$

This proves Lemma 6. ◀

Proof of Lemma 20. Consider the auxiliary polynomial

$$h(x, y) := \sum_{j=1}^{m} f_j(x) y_j.$$

As a function of both x and y, we have

$$\nabla h(x, y) = \vec{f}(x) \circ M_x y,$$

where M_x is the matrix with columns $\nabla f_j(x)$ (that is, M_x has (i,j)-th entry $\frac{\partial}{\partial x_i} f_j(x)$). So, applying Corollary 19 to this auxiliary polynomial gives the probability bound

$$\begin{aligned} q &:= \mathbb{P}\big(h(x,y)^2 \le \varepsilon^2 \|\nabla g(x,y)\|^2\big) \\ &= \mathbb{P}\left(\left\langle y, \vec{f}(x)\right\rangle^2 \le \varepsilon^2 \left(\|\vec{f}(x)\|^2 + \|M_x y\|^2\right)\right) \\ &\le O(\varepsilon d^2). \end{aligned}$$

Now, for some constant $C \ge 2$ to be specified later, let E denote the event that

$$(C^2-1)\|\vec{f}(x)\|^2 \le \frac{\varepsilon^2}{2}\|M_x\|_F^2,$$

where $\|M_x\|_F$ is the Frobenius norm of M_x. We note that we can lower-bound the probability q by

$$q \ge \mathbb{P}(E) \cdot \mathbb{P}\left(\left|\left\langle y, \vec{f}(x)\right\rangle\right| \le C\|\vec{f}(x)\| \text{ and } \|M_x y\|^2 \ge \frac{1}{2}\|M_x\|_F^2 \mid E\right).$$

We claim that for large enough choice of constant C, this conditional probability can be lower-bounded by $\Omega(1)$. Indeed, we can argue for any fixed x:

- $\mathbb{P}\left(\left|\left\langle y, \vec{f}(x)\right\rangle\right| \ge C\|\vec{f}(x)\|\right) \le \frac{1}{C^2}$.
- $\mathbb{P}\big(\|M_x y\|^2 \ge \frac{1}{2}\|M_x\|_F^2\big) \ge \Omega(1)$.

The first item is just a Chebyshev inequality; the second item can be derived e.g. from the basic anticoncentration bound one obtains for degree-2 polynomials from the Paley-Zygmund bound together with hypercontractivity (since, for any fixed matrix M, the quadratic form $g(y) := \|My\|^2$ has second-moment $\mathbb{E}\, g(y)^2 \ge (\mathbb{E}\, g(y))^2 = \|M\|_F^2$).

Thus, by choosing C large enough, we can lower-bound this conditional probability by

$$\Omega(1) - \frac{1}{C^2} \ge \Omega(1).$$

We conclude that $\mathbb{P}(E) \le O(q) = O(\varepsilon d^2)$. This gives the desired conclusion

$$\mathbb{P}\left(\|\vec{f}(x)\| \le \Omega(\varepsilon)\|M_x\|_F\right) \le O(\varepsilon d^2). \qquad \blacktriangleleft$$

5 Pseudorandom Generator for PTFs

The following theorem gives quantitative bounds on the error of our main generator:

▶ **Theorem 21.** *Fix some parameters $\varepsilon > 0$ and $R \in \mathbb{N}$. Let z be a standard gaussian, and let $Z = \frac{1}{\sqrt{L}}\sum_{i=1}^{L} Y_i$, where each Y_i is dR-moment-matching. Then for some sufficiently large absolute constant c and any polynomial p of degree d,*

$$\mathop{\mathbb{E}}_{Z} \mathsf{sign}(p(Z)) \ge \mathop{\mathbb{E}}_{z\sim N(0,1)^n} \mathsf{sign}(p(z)) - O(\varepsilon d^3) - L \cdot 2^{-\Omega(R)},$$

as long as L is at least Rd^c/ε^2.

Combining the above with Theorem 12 immediately implies our main result Theorem 3.

Proof of Theorem 3. Given a target error ε', set $\varepsilon = \varepsilon'/Cd^3$, and $R = C\log(d/\varepsilon)$ for a sufficiently big constant so that the error in the above lemma is at most $\varepsilon'/2$ for $L = Rd^c/\varepsilon^2 = O(d^c \log(d/\varepsilon)/\varepsilon^2)$. While the above theorem only gives a lower bound, we can get an upper bound by applying the result to $-p$. Now, by applying Theorem 12 there exists an efficient PRG that fools degree d PTFs with error at most ε' and seed length $O(d^{O(1)} \log(nd/\varepsilon') \log(d/\varepsilon')/(\varepsilon')^2$ which can be simplified to the bound in the theorem. ◀

We now prove the above theorem by the lower-sandwiching argument outlined in Section 2.1. Fix a polynomial $p(x)$ of degree d. We remind the reader of our convention $\mathsf{sign}(t) := \mathbb{1}(t \geq 0)$.

We define the mollifier function

$$g(x) := \prod_{k=0}^{d-1} \rho\left(\log\left(\frac{1}{16\varepsilon^2}\frac{\|\nabla^k p(x)\|^2}{\|\nabla^{k+1} p(x)\|^2}\right)\right),$$

where $\rho : \mathbb{R} \to [0,1]$ is some smooth univariate function with $\rho(t) = 0$ for $t \leq 0$, $\rho(t) = 1$ for $t \geq 1$, and $\|\frac{\partial^k \rho}{\partial t^k}\|_\infty \leq k^{O(k)}$ for all k. [11]

Proof of Theorem 21. For every point $x \in \mathbb{R}^n$ we have

$$\mathsf{sign}(p(x)) \geq \mathsf{sign}(p(x))g(x).$$

Furthermore, under the truly-random gaussian inputs $z \sim N(0,1)^n$ we have

$$\mathop{\mathbb{E}}_{z} \mathsf{sign}(p(z))g(z) \geq \mathop{\mathbb{E}}_{z} \mathsf{sign}(p(z)) - \mathop{\mathbb{E}}_{z} |g(z) - 1| \geq \mathop{\mathbb{E}}_{z} \mathsf{sign}(p(z)) - O(\varepsilon d^3),$$

where the final inequality here follows from Lemma 6. Combining these, we get that

$$\mathop{\mathbb{E}}_{Z} \mathsf{sign}(p(Z)) \geq \mathop{\mathbb{E}}_{z\sim N(0,1)^n} \mathsf{sign}(p(z)) - O(\varepsilon d^3) - |\mathop{\mathbb{E}}_{Z} \mathsf{sign}(p(Z))g(Z) - \mathop{\mathbb{E}}_{z\sim N(0,1)^n} \mathsf{sign}(p(z))g(z)|.$$

Thus, it suffices to bound $|\mathbb{E}_Z \mathsf{sign}(p(Z))g(Z) - \mathbb{E}_{z\sim N(0,1)^n} \mathsf{sign}(p(z))g(z)|$, which we do by a hybrid argument. We first represent z as $z := \frac{1}{\sqrt{L}} \sum_{i=1}^{L} y_i$ where each y_i is an independent standard gaussian. We can replace each Y_i with y_i and get

$$|\mathop{\mathbb{E}}_{Z} \mathsf{sign}(p(Z))g(Z) - \mathop{\mathbb{E}}_{y} \mathsf{sign}(p(y))g(y)| \leq 2^{-\Omega(R)} L,$$

as a consequence of the following lemma (restatement of Lemma 8) that we prove in the next section. Theorem 21 now follows. ◀

▶ **Lemma 22** (Main hybrid-step)**.** *There exists a constant c such that the following holds for $\lambda \leq \varepsilon^2/Rd^c$. For any fixed vector $x \in \mathbb{R}^n$, Y a dR-moment-matching gaussian vector, and $y \sim N(0,1)^n$,*

$$|\mathop{\mathbb{E}}_{Y} \mathsf{sign}(p(x + \sqrt{\lambda}Y))g(x + \sqrt{\lambda}Y) - \mathop{\mathbb{E}}_{y} \mathsf{sign}(p(x + \sqrt{\lambda}y))g(x + \sqrt{\lambda}y)| \leq \gamma = 2^{-\Omega(R)}.$$

[11] For example, it suffices to let $\rho(t)$ be the standard mollifier $\rho(t) := 0$ for $t \leq 0$, $\rho(t) := 1$ for $t \geq 1$, and $\rho(t) := e \cdot \exp\left(\frac{1}{(t-1)^2 - 1}\right)$ for $t \in (0,1)$.

5.1 Analysis of the main hybrid-step

The proof of Lemma 22 is by a case-analysis as outlined in the introduction. Consider the setting as in the lemma and define

$$\phi(z) := U_{\sqrt{1-\lambda}}p\left(\frac{z}{\sqrt{1-\lambda}}\right).$$

The core argument will be a case-analysis on the derivatives of ϕ at the fixed point x and whether these are slow-growing. Note that if p were multi-linear, then we would simply have $\phi \equiv p$. The starting point is the following re-scaling of Lemma 16:

$$p\left(x+\sqrt{\lambda}y\right) = \sum_{|\alpha|\le d}\frac{\partial^\alpha\phi(x)}{\sqrt{\alpha!}}\lambda^{|\alpha|/2}h_\alpha(y). \tag{4}$$

Further, by a re-scaling of Theorem 17, we get the following identity which gives a nice nearly self-referential expression relating the derivatives of p to those of ϕ:

$$\partial^\alpha p\left(x+\sqrt{\lambda}y\right) = \sum_{\beta\ge\alpha}\sqrt{\frac{\alpha!}{\beta!}}\partial^\beta\phi(x)\lambda^{|\beta-\alpha|/2}h_{\beta-\alpha}(y). \tag{5}$$

Now, note that for a truly random gaussian y we have $\partial^\alpha\phi(x) = \mathbb{E}_y\,\partial^\alpha p(x+\sqrt{\lambda}y)$. Thus, it is reasonable to expect that for typical points x and small enough λ, $\partial^\alpha p(x+\sqrt{\lambda}y)$ will be strongly concentrated around $\partial^\alpha\phi(x)$. The following lemma gives quantitative bounds on how much the derivatives $\partial^\alpha p(x+\sqrt{\lambda}y)$ deviate from their expectations $\partial^\alpha\phi(x)$ for a random $y\sim N(0,1)^n$. As we will need such bounds even for k-moment-matching Y, we state the deviation bound in terms of moments:

▶ **Lemma 23.** *Suppose f is a degree-d polynomial, and let $\phi(z) = U_{\sqrt{1-\lambda}}f(\frac{z}{\sqrt{1-\lambda}})$. Consider the polynomial*

$$D(y) := \|\nabla^k f(x+\sqrt{\lambda}y) - \nabla^k\phi(x)\|^2,$$

which measures the euclidean distance between the k-th order derivatives $\nabla^k f(x+\sqrt{\lambda}y)$ and their expectations $\nabla^k\phi(x)$.

For $y\sim N(0,1)^n$, we have the moment bound

$$\|D(y)\|_{q/2} \le \sum_{t=k+1}^{d}(\lambda dq)^{t-k}\|\nabla^t\phi(x)\|^2.$$

That is,

$$\left(\mathop{\mathbb{E}}_{y\sim N(0,1)^n}\|\nabla^k f(x+\sqrt{\lambda}y) - \nabla^k\phi(x)\|^q\right)^{1/q} \le \sqrt{\sum_{t=k+1}^{d}(\lambda dq)^{t-k}\|\nabla^t\phi(x)\|^2}.$$

Proof. We express

$$D(y) = \sum_\alpha\left(\partial^\alpha f(x+\sqrt{\lambda}y) - \partial^\alpha\phi(x)\right)^2 = \sum_\alpha\left(\sum_{\beta>\alpha}\sqrt{\frac{\alpha!}{\beta!}}\partial^\beta\phi(x)\lambda^{|\beta-\alpha|/2}h_{\beta-\alpha}(y)\right)^2.$$

First, by triangle-inequality, we get

$$\|D(y)\|_{q/2} \leq \sum_{\alpha} \left\| \left(\sum_{\beta > \alpha} \sqrt{\frac{\alpha!}{\beta!}} \partial^{\beta} \phi(x) \lambda^{|\beta-\alpha|/2} h_{\beta-\alpha}(y) \right)^2 \right\|_{q/2}$$
$$= \sum_{\alpha} \left\| \sum_{\beta > \alpha} \sqrt{\frac{\alpha!}{\beta!}} \partial^{\beta} \phi(x) \lambda^{|\beta-\alpha|/2} h_{\beta-\alpha}(y) \right\|_{q}.$$

Applying hypercontractivity, we now get

$$\|D(y)\|_{q/2} \leq \sum_{\alpha} \left\| U_{\sqrt{q}} \sum_{\beta > \alpha} \sqrt{\frac{\alpha!}{\beta!}} \partial^{\beta} \phi(x) \lambda^{|\beta-\alpha|/2} h_{\beta-\alpha}(y) \right\|_{2}$$
$$= \sum_{\alpha} \sum_{\beta > \alpha} \frac{\alpha!}{\beta!} \partial^{\beta} \phi(x)^2 \lambda^{|\beta-\alpha|} q^{|\beta-\alpha|}$$
$$\leq \sum_{\alpha} \sum_{\beta > \alpha} \partial^{\beta} \phi(x)^2 \lambda^{|\beta-\alpha|} q^{|\beta-\alpha|}$$
$$= \sum_{t=k+1}^{d} \binom{t}{t-k} (\lambda q)^{t-k} \|\nabla^t \phi(x)\|^2$$
$$\leq \sum_{t=k+1}^{d} (\lambda d q)^{t-k} \|\nabla^t \phi(x)\|^2. \quad \blacktriangleleft$$

We are now ready to prove Lemma 22.

Proof of Lemma 22. We study two cases:

1. x is poorly-behaved for ϕ. In this case, we will show that $g(x+\sqrt{\lambda}Y) = 0$ with probability at least $1 - 2^{-\Omega(R)}$.
2. x is well-behaved for ϕ: In this case, we will exploit the fact that $\mathsf{sign}(p(x + \sqrt{\lambda Y}))$ will equal $\mathsf{sign}(\phi(x))$ with probability $1 - 2^{-\Omega(R)}$. We then have to show that Y fools the mollifier g which is a bit technically involved (hence we deal with this case second unlike in Section 2.1).

We begin with the first case.

Case 1: x is poorly-behaved for ϕ. Consider the case where the inequality $\|\nabla^k \phi(x)\| \geq \varepsilon \|\nabla^{k+1} \phi(x)\|$ is violated for some k, and indeed let k_0 be the largest k such that this inequality is violated. We will argue that with probability at least $1 - 2^{-\Omega(R)}$, over random choice of Y, that

$$\|\nabla^{k_0} p(x + \sqrt{\lambda} Y)\| \leq 4\varepsilon \|\nabla^{k_0+1} p(x + \sqrt{\lambda} Y)\|,$$

in which case $g(x + \sqrt{\lambda}Y) = 0$.

More specifically, we will show that it is highly likely that both

- $\|\nabla^{k_0} p(x + \sqrt{\lambda} Y)\| \leq 2\varepsilon \|\nabla^{k_0+1} \phi(x)\|$, and
- $\|\nabla^{k_0+1} p(x + \sqrt{\lambda} Y)\| \geq \frac{1}{2} \|\nabla^{k_0+1} \phi(x)\|$.

For this, we will use Equation (5) and Lemma 23. Supposing k_0 is the largest k such that

$$\|\nabla^k \phi(x)\| < \varepsilon \|\nabla^{k+1} \phi(x)\|,$$

we have

- $\|\nabla^{k_0}\phi(x)\| \le \varepsilon\|\nabla^{k_0+1}\phi(x)\|$ and
- $\|\nabla^{k_0+1}\phi(x)\| \ge \varepsilon^t\|\nabla^{k_0+1+t}\phi(x)\|$ for all $t \ge 0$.

Lemma 23 therefore gives the bounds

$$\left(\mathop{\mathbb{E}}_{Y}\|\nabla^{k_0}p(x+\sqrt{\lambda}Y)-\nabla^{k_0}\phi(x)\|^R\right)^{1/R} \le \varepsilon\|\nabla^{k_0+1}\phi(x)\|\sqrt{\sum_{t\ge1}(\lambda dR/\varepsilon^2)^t}$$

and

$$\left(\mathop{\mathbb{E}}_{Y}\|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)-\nabla^{k_0+1}\phi(x)\|^R\right)^{1/R} \le \|\nabla^{k_0+1}\phi(x)\|\sqrt{\sum_{t\ge1}(\lambda dR/\varepsilon^2)^t}.$$

So, as long as $\lambda dR/\varepsilon^2$ is at most a sufficiently small constant, we conclude that the following bounds hold with probability at least $1-2^{-R}$:

- $\|\nabla^{k_0}p(x+\sqrt{\lambda}Y)\| \le \|\nabla^{k_0}\phi(x)\| + \|\nabla^{k_0}p(x+\sqrt{\lambda}Y)-\nabla^{k_0}\phi(x)\| \le 2\varepsilon\|\nabla^{k_0+1}\phi(x)\|$, and
- $\|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)\| \ge \|\nabla^{k_0+1}\phi(x)\| - \|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)-\nabla^{k_0+1}\phi(x)\| \ge \frac{1}{2}\|\nabla^{k_0+1}\phi(x)\|$.

In the case that these bounds hold, we get

$$\|\nabla^{k_0}p(x+\sqrt{\lambda}Y)\| \le 4\varepsilon\|\nabla^{k_0+1}p(x+\sqrt{\lambda}Y)\|,$$

and so $g(x+\sqrt{\lambda}Y)=0$. As this holds with probability at least $1-2^{-\Omega(R)}$ for both $y\sim N(0,1)^n$ as well as Y, the conclusion of Lemma 22 follows. This finishes the proof of Case 1. ⌟

Case 2: x is well-behaved for ϕ. We now consider the complimentary case where

$$\|\nabla^k\phi(x)\| \ge \varepsilon\|\nabla^{k+1}\phi(x)\|$$

for all $k=0,1,\dots,d-1$. Consider the normalized polynomial

$$f(y) := \frac{p(x+\sqrt{\lambda}y)}{\phi(x)} = 1+\frac{1}{\phi(x)}\sum_{\alpha\neq0}\partial^\alpha\phi(x)\lambda^{|\alpha|/2}h_\alpha(y).$$

Using hypercontractivity, we bound the R-th moment of $f(y)-1$ by its $\sqrt{R}$-hypervariance:

$$\|f(y)-1\|_R \le \|U_{\sqrt{R}}(f(y)-1)\|_2 \le \sqrt{\sum_{k\ge1}\left(\frac{\lambda R}{\varepsilon^2}\right)^k} \le \frac{1}{2}.$$

So, by a Markov argument, we have

$$\mathbb{P}\Big(\mathsf{sign}(p(x+\sqrt{\lambda}Y)) \neq \mathsf{sign}(\phi(x))\Big) \le 2^{-R},$$

and this holds whenever Y is k-moment-matching for $k \ge dR$. So, $\mathsf{sign}(p(x+\sqrt{\lambda}Y))$ is nearly a constant for random Y; it remains to show that Y fools $g(x+\sqrt{\lambda}Y)$. We do this by (essentially) truncating the Taylor-series of g about x so that we are left with a degree dR polynomial, which is fooled by Y. The truncation-error will be small because our assumption,

$$\|\nabla^k\phi(x)\| \ge \varepsilon\|\nabla^{k+1}\phi(x)\| \text{ for all } k,$$

gives us good control on the R-th order moments of the deviations $\|\nabla^k\phi(x)\| - \|\nabla^k p(x+\sqrt{\lambda}Y)\|$. The exact calculations are somewhat cumbersome and are given below. We will show that Y fools the mollifier function

$$g(x+\sqrt{\lambda}y) = \prod_{k=0}^{d-1}\rho\left(\log\left(\frac{1}{16\varepsilon^2}\frac{\|\nabla^k p(x+\sqrt{\lambda}y)\|^2}{\|\nabla^{k+1}p(x+\sqrt{\lambda}y)\|^2}\right)\right).$$

To simplify notation we define the shifted function $\sigma(t) := \rho(t - \log(16\varepsilon^2))$, and express

$$g(x + \sqrt{\lambda}y) = \prod_{k=0}^{d-1} \sigma\left(\log \|\nabla^k p(x + \sqrt{\lambda}y)\|^2 - \log \|\nabla^{k+1} p(x + \sqrt{\lambda}y)\|^2\right).$$

It will be convenient to think of g (redundantly) as function of $2d$ auxiliary variables $s_1 \dots s_d$, $t_1, \dots t_d$, which we will eventually fix to

- $s_i := \|\nabla^{i-1} p(x + \sqrt{\lambda}y)\|^2$
- $t_i := \|\nabla^i p(x + \sqrt{\lambda}y)\|^2$,

so we write

$$g(s,t) := \prod_{i=1}^{d} \sigma\left(\log(s_i) - \log(t_i)\right).$$

We Taylor-expand $g(s,t)$ around the points

- $a_i := \|\nabla^{i-1}\phi(x)\|^2$
- $b_i := \|\nabla^i \phi(x)\|^2$,

which gives

$$g(s,t) = \ell(s,t) + h(s,t),$$

with low-degree part

$$\ell(s,t) := \sum_{\substack{\alpha,\beta \in \mathbb{N}^d \\ |\alpha|+|\beta|<R}} \frac{\partial_s^\alpha \partial_t^\beta g(a,b)}{\alpha!\beta!} (s-a)^\alpha (t-b)^\beta$$

and remainder

$$|h(s,t)| \le \sum_{\substack{\alpha,\beta \in \mathbb{N}^d \\ |\alpha|+|\beta|=R}} \frac{|\partial_s^\alpha \partial_t^\beta g(s^*,t^*)|}{\alpha!\beta!} |s-a|^\alpha |t-b|^\beta,$$

where "$|\partial_s^\alpha \partial_t^\beta g(s^*,t^*)|$" is notation for the maximum magnitude of $\partial_s^\alpha \partial_t^\beta g$ on any point on the line segment from (a,b) to (s,t). We need the following fact to bound the size of the derivatives of g,

▷ Claim 24. Suppose σ is a smooth univariate function with uniform derivative bounds

$$\|\sigma^{(n)}\|_\infty \le n^{O(n)}.$$

The bivariate function

$$r(u,v) := \sigma(\log(u) - \log(v))$$

has derivatives bounded in size by

$$\left|\frac{\partial^n}{\partial u^n}\frac{\partial^m}{\partial v^m} r(u,v)\right| \le \frac{n^{O(n)}}{|u|^n}\frac{m^{O(m)}}{|v|^m}.$$

This claim follows easily from the generalized chain rule (Faà di Bruno's formula). As a result, we get the derivative bounds

$$\left|\partial_s^\alpha \partial_t^\beta g(s,t)\right| \le \frac{|\alpha|^{O(|\alpha|)}}{|s^\alpha|}\frac{|\beta|^{O(|\beta|)}}{|t^\beta|}.$$

Using this, we bound the remainder

$$|h(s,t)| \le \sum_{\substack{\alpha,\beta\in\mathbb{N}^d \\ |\alpha|+|\beta|=R}} d^{O(R)} \prod_{i=1}^{d} \left(\frac{|1-\frac{s_i}{a_i}|}{1-|1-\frac{s_i}{a_i}|}\right)^{\alpha_i} \left(\frac{|1-\frac{t_i}{b_i}|}{1-|1-\frac{t_i}{b_i}|}\right)^{\beta_i}.$$

Now, consider the event E (which depends on y) that

$$(1-\delta)\|\nabla^i\phi(x)\|^2 \le \|\nabla^i p(x+\sqrt{\lambda}y)\|^2 \le (1+\delta)\|\nabla^i\phi(x)\|^2$$

holds for all i, where $\delta \le 1/2$ is a parameter we will set shortly. In the case that this indeed holds, we get

$$|h(s,t)| \le d^{O(R)}O(\delta)^R.$$

We set δ just small enough to ensure

$$|h(s,t)| \le 2^{-R}.$$

Now, we express g (which we now think of as a function of the underlying variable y) as

$$\begin{aligned} g &= g\cdot \mathbb{1}_E + g\cdot \mathbb{1}_{\bar{E}} \\ &= \ell\cdot \mathbb{1}_E + h\cdot \mathbb{1}_E + g\cdot \mathbb{1}_{\bar{E}} \\ &= \ell - \ell\cdot \mathbb{1}_{\bar{E}} + h\cdot \mathbb{1}_E + g\cdot \mathbb{1}_{\bar{E}}, \end{aligned}$$

and we obtain the pointwise bound

$$|g-\ell| \le 2^{-R} + \mathbb{1}_{\bar{E}} + |\ell|\cdot \mathbb{1}_{\bar{E}}.$$

On average over Y, we get truncation error

$$\begin{aligned} \mathop{\mathbb{E}}_{Y}\left|g(x+\sqrt{\lambda}Y)-\ell(Y)\right| &\le 2^{-R} + \mathop{\mathbb{E}}_{Y}\mathbb{1}_{\bar{E}}(Y) + \sqrt{\mathop{\mathbb{E}}_{Y}\ell^2(Y)}\sqrt{\mathop{\mathbb{E}}_{Y}\mathbb{1}_{\bar{E}}(Y)} \\ &\le 2^{-R} + O\left(\frac{d}{\delta}\right)^R \cdot \left(\frac{\lambda dR}{\varepsilon^2}\right)^{-\Omega(R)} \\ &\le 2^{-R} + d^{O(1)}\cdot \left(\frac{\lambda dR}{\varepsilon^2}\right)^{-\Omega(R)} \end{aligned}$$

where the second inequality here follows from the moment bounds in Lemma 23. As required by the conditions of Lemma 22, we insist that λ is small enough that this error is at most $2^{-\Omega(R)}$. Since this bound holds also for truly-random standard gaussian y, and $\mathbb{E}_Y \ell(Y) = \mathbb{E}_y \ell(y)$, we obtain the desired bound

$$|\mathop{\mathbb{E}}_{Y} g(x+\sqrt{\lambda}Y) - \mathop{\mathbb{E}}_{y} g(x+\sqrt{\lambda}y)| \le 2^{-\Omega(R)}.$$

This finishes the proof in Case 2 and hence of Lemma 22. ◀

References

1 Ido Ben-Eliezer, Shachar Lovett, and Ariel Yadin. Polynomial threshold functions: Structure, approximation and pseudorandomness. *CoRR*, abs/0911.3473, 2009. `arXiv:0911.3473`.

2 Ilias Diakonikolas, Daniel M Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2010.

3 Ilias Diakonikolas, Prasad Raghavendra, Rocco A. Servedio, and Li-Yang Tan. Average sensitivity and noise sensitivity of polynomial threshold functions. *SIAM J. Comput.*, 43(1):231–253, 2014. doi:10.1137/110855223.

4 Prahladh Harsha, Adam Klivans, and Raghu Meka. Bounding the sensitivity of polynomial threshold functions. *Theory of Computing*, 10(1):1–26, 2014. doi:10.4086/toc.2014.v010a001.

5 Valentine Kabanets, Daniel M Kane, and Zhenjian Lu. A polynomial restriction lemma with applications. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 615–628, 2017.

6 Daniel M Kane. k-independent gaussians fool polynomial threshold functions. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 252–261. IEEE Computer Society, 2011.

7 Daniel M Kane. A small PRG for polynomial threshold functions of gaussians. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 257–266. IEEE, 2011.

8 Daniel M Kane. A structure theorem for poorly anticoncentrated gaussian chaoses and applications to the study of polynomial threshold functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 91–100. IEEE, 2012.

9 Daniel M Kane. The correct exponent for the Gotsman-Linial conjecture. In *2013 IEEE Conference on Computational Complexity*, pages 56–64. IEEE, 2013.

10 Daniel M Kane. A pseudorandom generator for polynomial threshold functions of gaussian with subpolynomial seed length. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 217–228. IEEE, 2014.

11 Daniel M Kane. A polylogarithmic PRG for degree 2 threshold functions in the gaussian setting. In *Proceedings of the 30th Conference on Computational Complexity*, pages 567–581, 2015.

12 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.

13 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

14 Ryan O'Donnell, Rocco A Servedio, and Li-Yang Tan. Fooling gaussian PTFs via local hyperconcentration. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1170–1183, 2020.

15 Ryan O'Donnell, Rocco A. Servedio, Li-Yang Tan, and Daniel Kane. Fooling gaussian PTFs via local hyperconcentration, 2021. arXiv:2103.07809. arXiv:2103.07809.

Optimal-Degree Polynomial Approximations for Exponentials and Gaussian Kernel Density Estimation

Amol Aggarwal ✉
Department of Mathematics, Columbia University, New York, NY, USA
Institute for Advanced Study, Princeton, NJ, USA

Josh Alman ✉
Department of Computer Science, Columbia University, New York, NY, USA

Abstract

For any real numbers $B \geq 1$ and $\delta \in (0,1)$ and function $f : [0,B] \to \mathbb{R}$, let $d_{B;\delta}(f) \in \mathbb{Z}_{>0}$ denote the minimum degree of a polynomial $p(x)$ satisfying $\sup_{x \in [0,B]} \left| p(x) - f(x) \right| < \delta$. In this paper, we provide precise asymptotics for $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^{x})$ in terms of both B and δ, improving both the previously known upper bounds and lower bounds. In particular, we show

$$d_{B;\delta}(e^{-x}) = \Theta\left(\max\left\{ \sqrt{B \log(\delta^{-1})}, \frac{\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))} \right\} \right), \text{ and}$$

$$d_{B;\delta}(e^{x}) = \Theta\left(\max\left\{ B, \frac{\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))} \right\} \right),$$

and we explicitly determine the leading coefficients in most parameter regimes.

Polynomial approximations for e^{-x} and e^{x} have applications to the design of algorithms for many problems, including in scientific computing, graph algorithms, machine learning, and statistics. Our degree bounds show both the power and limitations of these algorithms.

We focus in particular on the Batch Gaussian Kernel Density Estimation problem for n sample points in $\Theta(\log n)$ dimensions with error $\delta = n^{-\Theta(1)}$. We show that the running time one can achieve depends on the square of the diameter of the point set, B, with a transition at $B = \Theta(\log n)$ mirroring the corresponding transition in $d_{B;\delta}(e^{-x})$:

- When $B = o(\log n)$, we give the first algorithm running in time $n^{1+o(1)}$.
- When $B = \kappa \log n$ for a small constant $\kappa > 0$, we give an algorithm running in time $n^{1+O(\log\log\kappa^{-1}/\log\kappa^{-1})}$. The $\log\log\kappa^{-1}/\log\kappa^{-1}$ term in the exponent comes from analyzing the behavior of the leading constant in our computation of $d_{B;\delta}(e^{-x})$.
- When $B = \omega(\log n)$, we show that time $n^{2-o(1)}$ is necessary assuming SETH.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Approximation algorithms analysis; Mathematics of computing → Mathematical analysis

Keywords and phrases polynomial approximation, kernel density estimation, Chebyshev polynomials

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.22

Related Version *Full Version*: https://arxiv.org/abs/2205.06249

Funding *Amol Aggarwal*: Partially supported by NSF grants DGE-1144152 and DMS-1664619, a Harvard Merit/Graduate Society Term-time Research Fellowship, and a Clay Research Fellowship. *Josh Alman*: Partially supported by a Harvard Michael O. Rabin postdoctoral fellowship and a grant from the Simons Foundation (Grant Number 825870 JA).

Acknowledgements We would like to thank Alexei Borodin, Lijie Chen, Andrei Martinez-Finkelshtein, Sushant Sachdeva, Paris Siminelakis, Roald Trigub, Ryan Williams, and anonymous reviewers for helpful discussions and advice throughout this project.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 22; pp. 22:1–22:23

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

1 Introduction

Polynomial approximations of important functions play a key role in many areas of computer science and mathematics. We measure the extent to which a function can be approximated by a degree d polynomial as follows.

▶ **Definition 1.** *For any real numbers $B \geq 1$ and $\delta \in (0,1)$, and function $f : [0,B] \to \mathbb{R}$, let $d_{B;\delta}(f) \in \mathbb{Z}_{>0}$ denote the minimum degree of a non-constant polynomial $p(x)$ satisfying*

$$\sup_{x\in[0,B]} \big|p(x) - f(x)\big| < \delta.$$

Past work in polynomial approximation theory has typically focused on the case when $B = O(1)$; see, for example, [31, Chapter 7]. However, recent computer science applications have motivated studying the setting where both B and δ^{-1} are growing simultaneously. Indeed, in algorithmic applications, both the magnitude of the input to the function f and the tolerance for error can scale with the size of the input to the problem.

In this paper, we focus specifically[1] on the functions e^x and e^{-x}. As we will discuss more shortly, polynomial approximations for these functions appear naturally in computational problems throughout scientific computing, graph algorithms, machine learning, statistics, and many other areas. Precisely determining $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$ is particularly important since in a number of algorithmic applications, such as the batch Gaussian Kernel Density Estimation that we discuss in Section 1.2 below, these quantities appear in the exponent of the input size in the running time. In these settings, logarithmic or even constant factors can be the difference between a fast or a trivially slow running time (see especially Sections 1.2.1 and 1.2.3 below). The standard framework of approximation theory (e.g., [26, 31, 32]) can be used to deduce bounds on $d_{B;\delta}$ that are typically suboptimal, often losing (at least) such logarithmic factors, especially in the regime when B is large.

Our main results are tight asymptotics, including the exact leading constant in most parameter regimes (see Remark 5 below), for both $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$.

In what follows, we define the function

$$G(x) = \sqrt{x^2+1} + x\log\big(\sqrt{x^2+1} - x\big), \tag{1}$$

for each $x \in \mathbb{R}_{\geq 0}$.

▶ **Theorem 2** (Approximate degree of e^{-x})**.** *Let $B \geq 1$ and $\delta \in (0,1)$. Then,*

$$d_{B;\delta}(e^{-x}) = \Theta\left(\max\left\{\sqrt{B\log(\delta^{-1})}, \frac{\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))}\right\}\right).$$

More precisely, we have the following asymptotics as $B + \delta^{-1}$ tends to ∞.

1. *If $B = o\big(\log(\delta^{-1})\big)$, then $d_{B;\delta}(e^{-x}) = \left(\dfrac{\log\left(\delta^{-1}\right)}{\log\left(B^{-1}\log(\delta^{-1})\right)}\right)\big(1 + o(1)\big)$.*
2. *If $B = 2r\log(\delta^{-1})$ for fixed $r > 0$, then $d_{B;\delta}(e^{-x}) = \big(\nu r + o(1)\big)\log(\delta^{-1})$, where $\nu = \nu(r) > 0$ is the unique positive solution[2] to the equation $G(\nu) = 1 - r^{-1}$.*

[1] We mention, however, that the method used in this paper is quite general and is expected to more broadly apply for functions f whose Taylor series coefficients decay sufficiently quickly.

[2] The uniqueness of this solution (and the ones to be mentioned below) follows from the facts that $G(0) = 1$, $\lim_{z\to\infty} G(z) = -\infty$, and $G'(z) < 0$ for $z > 0$.

3. *If* $B = \omega\big(\log(\delta^{-1})\big)$ *and* $B \leq \delta^{-o(1)}$, *then* $d_{B;\delta}(e^{-x}) = \big(1+o(1)\big)\sqrt{B\log(\delta^{-1})}$.
4. *If* $B \geq \delta^{-\Omega(1)}$, *then* $d_{B;\delta}(e^{-x}) = \Theta\left(\sqrt{B\log\left(\delta^{-1}\right)}\right)$.

▶ **Theorem 3** (Approximate degree of e^x)**.** *Let* $B \geq 1$ *and* $\delta \in (0,1)$. *Then,*

$$d_{B;\delta}(e^x) = \Theta\left(\max\left\{B, \frac{\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))}\right\}\right).$$

More precisely, we have the following asymptotics as $B + \delta^{-1}$ *tends to* ∞.

1. *If* $B = o\big(\log(\delta^{-1})\big)$, *then* $d_{B;\delta}(e^x) = \left(\dfrac{\log\left(\delta^{-1}\right)}{\log\left(B^{-1}\log(\delta^{-1})\right)}\right)\big(1+o(1)\big)$.
2. *If* $B = 2r\log(\delta^{-1})$ *for fixed* $r > 0$, *then* $d_{B;\delta}(e^x) = \big(\mu r + o(1)\big)\log(\delta^{-1})$, *where* $\mu = \mu(r) > 0$ *is the unique positive solution to the equation* $G(\mu) = -1 - r^{-1}$.
3. *If* $B = \omega\big(\log\left(\delta^{-1}\right)\big)$, *then* $d_{B;\delta}(e^x) = \dfrac{z_* B}{2}\big(1+o(1)\big)$, *where* $z_* \approx 2.2334$ *denotes the unique positive solution to the equation* $G(z_*) = -1$.

▶ Remark 4. Polynomials achieving the degree upper bounds stated in Theorems 2 and 3 can be constructed in $\mathrm{poly}(d)$ time, with coefficients which are rational numbers with $\mathrm{poly}(d)$-bit integer numerators and denominators, where d is the degree.

▶ Remark 5. In the fourth case of Theorem 2, we do not determine the leading constant $A = A(B;\delta)$ for which $d_{B;\delta}(e^{-x}) = \big(A+o(1)\big)\sqrt{B\log\left(\delta^{-1}\right)}$; as we will see below, when $\delta = o(1)$, we only bound it between $\frac{1}{2} \leq A \leq 1$. It is unclear to us whether or not this constant would admit a concise description in this parameter regime, especially in the case when δ is fixed as B tends to ∞. In all other parameter regimes of Theorem 2, and in every case of Theorem 3, we determine the exact leading constant.[3]

Previous bounds. The question of providing tight bounds on $d_{B;\delta}(e^{-x})$ was posed in works of Orecchia, Sachdeva, and Vishnoi [24, Sections 4 and 7], and Sachdeva and Vishnoi [28, Section 5]. They were motivated by algorithmic applications, as [24] showed how upper bounds on $d_{B;\delta}(e^{-x})$ can be used to design faster algorithms for the Balanced Separator problem from spectral graph theory. They gave an upper bound of $d_{B;\delta}(e^{-x}) \leq O(\sqrt{\max\{\log(\delta^{-1}), B\}} \cdot \log^{3/2}(\delta^{-1}))$, and a lower bound of $d_{B;\delta}(e^{-x}) \geq \frac{1}{2}\sqrt{B}$. Later, [28] improved the upper bound to $d_{B;\delta}(e^{-x}) \leq O(\sqrt{\max\{\log(\delta^{-1}), B\}} \cdot \log^{1/2}(\delta^{-1}))$ (noting that such a bound was also implicit in [16]). Theorem 2 provides precise asymptotics for $d_{B;\delta}(e^{-x})$, thereby answering the above question.

In particular, Theorem 2 shows that the prior upper bound could be improved by a logarithmic factor in some parameter regimes, but was otherwise asymptotically tight. For the Balanced Separator problem studied by [24], where the running time depends polynomially on $d_{B;\delta}(e^{-x})$, this rules out a big improvement without a new approach. For other applications where the running time has an exponential dependence on $d_{B;\delta}(e^{-x})$, our improvements have more significant implications. For instance, as we discuss below in Section 1.2.1, in some parameter regimes of the batch Gaussian Kernel Density Estimation problem, our Theorem 2 yields a near linear time algorithm, whereas applying instead the prior bound of [28] would only yield a trivial quadratic running time.

[3] It is quickly verified that the constants ν, μ, and z_* from Theorem 2 and Theorem 3 satisfy $2r^{-1/2} \leq \nu \leq \max\left\{r^{-1}, e\right\}$ and $z_* \leq \mu \leq \max\left\{r^{-1}, e\right\}$ for all $r > 0$.

We are unaware of prior work which specifically bounded $d_{B;\delta}(e^x)$, although one could apply standard results on Chebyshev interpolation (such as [32, Theorem 8.2]) with some work to yield a bound $d_{B;\delta}(e^x) \geq \Omega(\max\{B, \log(\delta^{-1})\})$.

Phase transitions. In fact, Theorem 2 and Theorem 3 indicate that the dependence of the optimal degrees $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$ on the parameters B and δ is quite intricate. First, their orders of magnitudes both exhibit transitions depending on the relative sizes of B and $\log(\delta^{-1})$. For example, when $B = \omega\big(\log(\delta^{-1})\big)$, Theorem 2 shows that $d_{B;\delta}(e^{-x})$ exhibits square root dependence on both $\log(\delta^{-1})$ and B, but when $B = o\big(\log(\delta^{-1})\big)$ it exhibits nearly linear dependence on $\log(\delta^{-1})$ and only logarithmic dependence on B. Second, in the "critical regime" $B = 2r\log(\delta^{-1})$, Theorem 2 shows that $d_{B;\delta}(e^{-x}) = \Theta\big(\log(\delta^{-1})\big)$, whose implicit constant is obtained by solving the transcendental equation $G(z) = 1 - r^{-1}$. A similar transition (with a transcendental leading constant in the critical regime) is shown for the approximating degrees of e^x in Theorem 3, but with the qualitative difference that $d_{B;\delta}(e^x)$ is linear in B (to leading order) and independent of δ, for $B = \omega\big(\log(\delta^{-1})\big)$.

To our knowledge, this is the first appearance of a transition arising when one simultaneously scales B and δ in the context of polynomial approximation theory. Indeed, as mentioned previously, prior works in this direction typically analyzed the case $B = O(1)$, where transitions like these are not visible. As we will explain in Section 1.1 below, these behaviors for $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$ will have algorithmic interpretations. For example, we will see that the estimates on $d_{B;\delta}(e^{-x})$ provided in Theorem 2 imply a fine-grained computational phase transition for Gaussian Kernel Density Estimation in certain parameter regimes.

Previous methods. In the theoretical computer science literature, proofs of upper and lower bounds on the approximate degree $d_{B;\delta}(f)$ of a function f had been typically based on two distinct arguments [23, 29, 1, 8, 24, 28, 10]. Upper bounds were often shown by providing an explicit polynomial approximation for f, usually given by (a truncation of) the expansion of f in the basis of Chebyshev polynomials. Lower bounds were typically shown by making use of an estimate, such as Markov Brothers' inequality, that constrains the maximum derivative of a bounded polynomial in terms of its degree. Both ideas are archetypes of classical approximation theory; see [31, Chapters 2 and 4].

Our methods. As above, to upper bound $d_{B;\delta}(f)$ we will explicitly provide an approximating polynomial for f, obtained from the Chebyshev expansion of its rescale $f_B(x) = f\big(\frac{B}{2}(1-x)\big)$ (whose domain is now $[-1, 1]$). However, derivative-degree estimates such as Markov's inequality that prior works used to lower bound $d_{B;\delta}(f)$ usually become insensitive to the tolerance parameter δ once it passes below a (typically non-optimal) threshold. Thus, they will not suffice for our purposes of pinpointing the precise asymptotic behavior of $d_{B;\delta}(f)$.

We therefore proceed differently, by instead again making use of the Chebyshev expansion of $f_B(x)$. In particular, we use the orthogonality of the Chebsyhev polynomials to lower bound the minimal distance from f_B to a polynomial of degree d in terms of the series coefficients of f_B when expanded in the Chebyshev basis; see Proposition 11 below. Thus, bounds on these series coefficients can be used to bound $d_{B;\delta}(f)$. This idea was also ubiquitous in the traditional theory and practice of approximating polynomials; for instance, it was very fruitful in proving the classical sharp estimates [31, Chapter 7.8 (22)] on $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^{-x})$ when $B = O(1)$.

However, to our understanding, this idea has not been implemented before in our context where B and δ scale jointly (either in the computer science or approximation theory literature). In this setting, we must study the limiting behaviors for the high-degree coefficients in the Chebyshev expansion f_B, simultaneously as the degree and as B tend to ∞; see Proposition 13 below. This analysis becomes more involved than in the case $B = O(1)$, as it should in order to give rise to the intricate asymptotic phenomena described in Theorem 2 and Theorem 3. In particular, the phase transitions observed in those results can be traced to corresponding phase transitions for these series coefficients, given in Lemma 15 below.

1.1 Algorithmic Applications

Polynomial approximations with low error have numerous applications throughout algorithm design and complexity theory; see, for instance, the introduction of the survey by Sachdeva and Vishnoi [28] for an overview. The quantities $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$, in particular, play a central role in many algorithms due to the prevalence of exponential functions. Some examples include:

- **Approximating matrix exponentials.** Given a matrix A and a vector v, approximate $e^A \cdot v$. One of the most common algorithms in theory and in practice for this problem is the Lanczos method [18], whose running time is bounded by $O(d \cdot m_A + d^2)$ [21], where m_A is the amount of time required to do a matrix-vector multiplication by A, and $d = d_{B;\delta}(e^x)$ is the approximate degree which we compute in Theorem 3 with $B = ||A||$ and δ is the desired approximation error parameter.
- **Finding balanced separators in graphs.** The aforementioned work by Orecchia, Sachdeva and Vishnoi [24] uses polynomial approximations of e^{-x} in a way similar to the Lanczos method to give fast, practical algorithms for the Balanced Separator problem.[4]
- **Estimating softmax.** Many "multinomial classification" problems in natural language processing and other areas make use of the *softmax* function to convert vectors representing the different classes into estimated probabilities. Given n vectors $w_1, \ldots, w_n \in \mathbb{R}^m_{\geq 0}$, an index $i \in [n]$ and a sample vector $h \in \mathbb{R}^m_{\geq 0}$, softmax is defined as

$$\text{softmax}(h, i, w_1, \ldots, w_n) := \frac{e^{\langle w_i, h \rangle}}{\sum_{j=1}^n e^{\langle w_j, h \rangle}}.$$

  Training models in these applications frequently requires many softmax computations, and so approximations of softmax which are faster to compute are often used [14]. Replacing the exponentials in softmax by the optimal polynomial approximations we give in Theorem 3 can be used to more quickly compute such approximations [17, 22].
- **Kernel methods.** Polynomial approximations for e^{-x} have been used to design faster sketching and estimation techniques for Gaussian kernels, including in a number of recent algorithms; see e.g. [33, 19, 6, 4]. In Section 1.2 below, we show a new application along these lines to batch Gaussian Kernel Density Estimation.

[4] They also give a faster algorithm in some special cases using *rational approximations* of e^{-x}.

1.2 Gaussian Kernel Density Estimation

Kernel Density Estimation (KDE) is one of the most common methods for non-parametric estimation of the density of an unknown distribution $\mathcal{D}$. Given a set $P \subset \mathbb{R}^m$ of samples from $\mathcal{D}$, along with a weight $w_y \in \mathbb{R}$ for each $y \in P$, the kernel density function (KDF) of P at a point $x \in \mathbb{R}^m$ is given by

$$KDF_P(x) := \sum_{y \in P} w_y \cdot k(x, y),$$

where $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a carefully chosen *kernel function*. In most applications, one would like to compute KDF_P at many points x. Perhaps the most commonly studied kernel function is the *Gaussian kernel* $k(x, y) = e^{-\|x-y\|_2^2}$. This motivates the question:

▶ **Problem 6** (Batch Gaussian KDE). *Given as input $2n$ points $x^{(1)}, \ldots, x^{(n)}, y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}^m$ which implicitly define the matrix $K \in \mathbb{R}^{n \times n}$ by $K[i, j] = e^{-\|x^{(i)} - y^{(j)}\|_2^2}$, as well as a vector $w \in \mathbb{R}^n$, and an error parameter $\delta > 0$, compute an approximation to $K \cdot w$, meaning, output a vector $v \in \mathbb{R}^n$ such that $\|K \cdot w - v\|_\infty \leq \delta \cdot \|w\|_1$.*

Polynomial method algorithm. This problem can be solved by using a polynomial approximation to e^{-x} in order to construct a low-rank approximation to the matrix K, as follows. Let $B \geq 1$ and $\delta \in (0, 1)$ denote real numbers, and suppose $p(z)$ is a univariate polynomial of degree $d \geq d_{B;\delta}(e^{-x})$ such that

$$\sup_{z \in [0,B]} \left| p(z) - e^{-z} \right| \leq \delta.$$

Thus, for $x, y \in \mathbb{R}^m$ with $\|x - y\|_2^2 \leq B$, the polynomial $p(\sum_{\ell=1}^m (x_\ell - y_\ell)^2)$ outputs a value within an additive δ of $e^{-\|x-y\|_2^2}$. Hence, to solve Batch Gaussian KDE, it suffices to output the vector $\tilde{K} \cdot w$, where $\tilde{K} \in \mathbb{R}^{n \times n}$ is the matrix given by $\tilde{K}[i, j] = p(\sum_{\ell=1}^m (x_\ell^{(i)} - y_\ell^{(j)})^2)$. By a standard argument, the rank of $\tilde{K}$ is at most the number of monomials in the expansion of $p(\sum_{\ell=1}^m (x_\ell - y_\ell)^2)$, which is bounded above by $M \leq \binom{2d+2m}{2d}$, and the corresponding low rank expression for $\tilde{K}$ can be found in time $O(n \cdot M \cdot m)$.

In other words, whenever $M < n^{o(1)}$, we can solve Batch Gaussian KDE in deterministic $n^{1+o(1)}$ time in this way (see also [6, Section 5.3] where this approach was previously laid out). Theorem 2 characterizes exactly when this is possible in terms of m (the dimension of the points), B (the square of the diameter of the point set), and δ (the error parameter):

▶ **Corollary 7.** *For any positive integer $m < n^{o(1)}$, and real numbers $B \geq 1$ and $\delta \in (0, 1)$, define $d = d_{B;\delta}(e^{-x})$ as in Theorem 2. Then, batch Gaussian KDE can be solved in deterministic time $n^{1+o(1)}$ whenever $\binom{2d+2m}{2d} < n^{o(1)}$. Similarly, if $\binom{2d+2m}{2d} < n^c$ for some constant $0 < c < 1$, then batch Gaussian KDE can be solved in truly subquadratic deterministic time $n^{1+c+o(1)}$.*

1.2.1 Comparison with prior work

The previous best known algorithm for Batch Gaussian KDE is due to recent work of Charikar and Siminelakis [13], which showed how to solve this problem in randomized time $\delta^{-2} n^{1+o(1)} \cdot (\log n)^{O(B^{2/3})}$ for any dimension $m < n^{o(1)}$. Their algorithm achieves randomized running time $n^{1+o(1)}$ whenever $\delta^{-1} < n^{o(1)}$, $B < o((\log n / \log\log n)^{3/2})$, and $m < n^{o(1)}$.

Focusing on the setting[5] where $m = O(\log n)$, Corollary 7 achieves deterministic running time $n^{1+o(1)}$ in all the same parameter settings as the previous algorithm, and also new settings including:

- When $B = o(\log^2 n)$ and $\delta^{-1} < n^{o(\log n/B)}$ (slightly improving the parameter B), or
- When $B = o(\log n)$ and $\delta^{-1} = n^{\Theta(1)}$ (considerably improving the parameter δ).

In particular, the latter setting enables us to take δ to depend polynomially in n, while still retaining an $n^{1+o(1)}$ running time.

The above parameter regimes are also where our upper bound on $d_{B;\delta}(e^{-x})$ from Theorem 2 logarithmically improves on the one given in [28]. This improvement was in fact necessary for our application to KDE, as the estimate from [28] was $d_{B;\delta}(e^{-x}) = \Omega(\log n)$ in these settings, which would give a running time of $n \cdot \binom{2d+2m}{2d} \geq n^{1+\Omega(1)}$, as opposed to our near-linear one.

Interestingly, our algorithm and that of [13] take approaches which rely on very different properties of the kernel function k. Charikar and Siminelakis's algorithm uses a clever Locality-Sensitive Hashing-based approach, and also works well for other kernels with efficient hash functions, whereas our approach instead requires k to have a low-degree polynomial approximation. Other popular algorithmic techniques for KDE, such as the Fast Multipole Method [15], or core-sets [3, 25], lead to $n^{1+\Omega(1)}$ running times in the high-dimensional $d = \Omega(\log n)$, low-error $\varepsilon < n^{-\Theta(1)}$ setting; see [30, Section 1.3.2] for an overview of these known approaches.

1.2.2 SETH lower bound

To complement Corollary 7, we also show a fine-grained lower bound, that assuming the Strong Exponential Time Hypothesis (SETH), when $m = \Theta(\log n)$ and $\delta^{-1} = n^{\Theta(1)}$, one cannot achieve running time $n^{1+o(1)}$ when $B = \Omega(\log n)$.

▶ **Proposition 8.** *Assuming SETH, for every $q > 0$, there are constants $\alpha, \beta, \kappa > 0$ such that Batch Gaussian KDE in dimension $m = \alpha \log n$ and error $\delta = n^{-\beta}$ for input points whose diameter squared is at most $B = \kappa \log n$ requires time $\Omega(n^{2-q})$.*

The proof of Proposition 8 is a slight modification of a similar lower bound of Backurs, Indyk, and Schmidt [9], which relates Gaussian KDE to nearest neighbor search (for which SETH lower bounds are already known [27]).

To summarize, in the natural setting where $m = \Theta(\log n)$ and $\delta^{-1} = n^{\Theta(1)}$:

- Our algorithm using the polynomial method achieves running time $n^{1+o(1)}$ when $B = o(\log n)$.
- Assume SETH. It is not possible to improve our algorithm to achieve running time $n^{1+o(1)}$ when $B = \Theta(\log n)$. Moreover, if $B = \omega(\log n)$, then no algorithm achieves running time faster than $n^{2-o(1)}$.

1.2.3 Critical regime behavior from the leading constant in Theorem 2

Thus, assuming SETH (and under the setting $m = \Theta(\log n)$ and $\delta^{-1} = n^{\Theta(1)}$), the complexity of Batch Gaussian KDE exhibits a transition mirroring the one exhibited by $d_{B;\delta}(e^{-x})$ in Theorem 2. More specifically, as B goes from $o(\log n)$ to $\omega(\log n)$, this complexity transitions

[5] Often, depending on the desired error guarantees, one can reduce to roughly this case using dimensionality reduction like the Johnson–Lindenstrauss lemma.

from $n^{1+o(1)}$ to $n^{2-o(1)}$. In the "critical regime" where $B = \kappa \log n$ for some fixed $\kappa > 0$, this suggests that its complexity should grow as $n^{1+\varphi(\kappa)}$, for some non-decreasing function $\varphi : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ satisfying $\lim_{\kappa\to 0} \varphi(\kappa) = 0$ and $\lim_{\kappa\to\infty} \varphi(\kappa) = 1$. It would be fascinating to better understand more precise behavior of this function φ. Does it continuously transition from 0 to 1 as κ increases, or does it admit a sudden "jump" at a specific threshold value for κ?

While these questions remain open, we can use our asymptotics for $d_{B;\delta}(e^{-x})$ to provide bounds on $\varphi(\kappa)$ for small κ. In particular, the below corollary implies that $\varphi(\kappa) = O\big(\log\log\kappa^{-1}/\log\kappa^{-1}\big)$. Our derivation of the term $\log\log\kappa^{-1}/\log\kappa^{-1}$ appearing in the exponent makes use of the leading constant ν (defined in the second part of Theorem 2) for the asymptotics of $d_{B;\delta}(e^{-x})$. Indeed, this is the case of Corollary 7 where $d = d_{B;\delta}(e^{-x}) = \Theta(\log n)$ and $m = O(\log n)$, and so $\binom{2d+2m}{2d} = n^{\Theta(1)}$, where the leading constant in $d_{B;\delta}(e^{-x})$ determines the value of the $\Theta(1)$.

▶ **Corollary 9.** *Fix constants $\alpha, \beta > 0$, and suppose that $m = \alpha \log n$ and $\delta = n^{-\beta}$. If $B = \kappa \log n$ for some $\kappa < \frac{1}{2}$, then Batch Gaussian KDE can be solved in time $O(n^{1+c\log\log\kappa^{-1}/\log\kappa^{-1}})$, where $c = c(\alpha,\beta) > 0$ only depends on α and β.*

Prior work has shown similar "critical regime" behavior for other problems with SETH-based lower bounds. The Orthogonal Vectors problem for n vectors in dimension $\kappa \log n$ for large κ can be solved in time $n^{2-1/O(\log\kappa)}$ [2, 12], whereas the problem in dimension $\omega(\log n)$ requires time $n^{2-o(1)}$ assuming SETH. The Batch Hamming Nearest Neighbors problem for n vectors in dimension $\kappa \log n$ for large κ can be solved in time $n^{2-1/\tilde{O}(\sqrt{\kappa})}$ [7, 5], whereas the problem in dimension $\omega(\log n)$ also requires time $n^{2-o(1)}$ assuming SETH. Interestingly, these algorithms make use of variants on the polynomial method using *probabilistic* polynomials, whereas we make use of approximate polynomials here.

The proofs of our results stated in Section 1.2 can be found in the full version of this paper.

2 Proof Overview

In this section we outline the proofs of Theorem 2 and Theorem 3, which will be established in detail in Section 3 below. To that end, we will use the Chebyshev polynomials, which are defined as follows; see Section 3 for a more thorough explanation of its properties.

▶ **Definition 10.** *Fix an integer $d \geq 0$. Let $\mathcal{P}_d \subset \mathbb{R}[x]$ denote the set of single-variable polynomials $p(x)$ with $\deg p \leq d$, and define the degree d* monic Chebyshev polynomial *$Q_d(x) \in \mathcal{P}_d$ as follows. Set $Q_0(x) = 1$ and, for each $d \geq 1$, define $Q_d(x)$ by imposing that*

$$Q_d(\cos\theta) = 2^{1-d}\cos(d\theta), \qquad \text{for each } \theta \in [0, 2\pi]. \tag{2}$$

It is well understood in the literature that smooth functions f are typically well-approximated by polynomials obtained by truncating the series expansion of f in the basis of Chebyshev polynomials; see, for instance, [32, (15.5), (15.8)] for more precise formulations of this statement.

In particular, the following proposition provides a version of this statement that will be more useful for our purposes. It provides upper and lower bounds on the optimal error of a polynomial approximation $p(x)$ of a function $f : [-1, 1] \to \mathbb{R}$ in terms of its Chebyshev expansion coefficients. Both bounds in this result are known; the lower bound follows from the L^2-orthogonality of the Chebyshev polynomials, and the upper bound follows from (2). Still, we provide a short and self-contained proof of the below proposition in Section 3.3.

▶ **Proposition 11.** *Let $a_0, a_1, \ldots \in \mathbb{R}$ satisfy $\sum_{j=0}^{\infty} |a_j| < \infty$. Then, the absolutely convergent series $f : [-1,1] \to \mathbb{R}$ defined by $f(x) = \sum_{j=0}^{\infty} 2^{j-1} a_j Q_j(x)$ satisfies*

$$\left(\frac{1}{2}\sum_{k=D}^{\infty}\frac{a_k^2}{k}\right)^{1/2} \leq \inf_{p\in\mathcal{P}_{D-1}} \sup_{x\in[-1,1]} \left|p(x) - f(x)\right| \leq \sum_{j=D}^{\infty} |a_j|, \tag{3}$$

for any integer $D \geq 1$.

In particular, (3) provides nearly matching upper and lower bounds (up to a factor of $(2D)^{1/2}$) if the coefficients $\{a_j\}$ decay sufficiently quickly. Indeed, then the left and right sides of that inequality are asymptotically governed by their leading terms a_D.

As stated, Proposition 11 only applies for approximating polynomials on the interval $[-1,1]$. However, we would like to approximate e^{-x} and e^x on $[0,B]$, for some $B \geq 1$. Therefore, we rescale by first setting $\lambda = \frac{B}{2}$, and then by observing that to approximate e^{-x} (or e^x) on $[0,B]$ it suffices to approximate $e^{\lambda x - \lambda}$ (or $e^{\lambda x + \lambda}$, respectively) on $[-1,1]$.

We will show that the coefficients of these latter functions, when written in the Chebyshev basis, indeed decay quickly (with an explicit rate, dependent on λ), and then apply Proposition 11. We can in fact compute these coefficients exactly, by first expressing e^{-x} and e^x through their Taylor series, and then by changing basis from the monomials x^n to the Chebyshev polynomials. This yields the following (known) lemma, whose short proof will be recalled in Section 4.1 below.

▶ **Lemma 12.** *For any real numbers $\lambda > 0$ and $x \in [-1,1]$, we have that*

$$e^{-\lambda x - \lambda} = \sum_{v=0}^{\infty} 2^{v-1} A_{v,\lambda} Q_v(x), \qquad e^{\lambda x + \lambda} = \sum_{v=0}^{\infty} 2^{v-1} B_{v,\lambda} Q_v(x), \tag{4}$$

where for any integer $v \geq 0$ we have set

$$A_{v,\lambda} = 2e^{-\lambda}(-1)^v \sum_{n-v\in 2\mathbb{Z}_{\geq 0}} \frac{\lambda^n}{2^n n!}\binom{n}{\frac{n-v}{2}}, \qquad B_{v,\lambda} = 2e^{\lambda} \sum_{n-v\in 2\mathbb{Z}_{\geq 0}} \frac{\lambda^n}{2^n n!}\binom{n}{\frac{n-v}{2}}. \tag{5}$$

We next apply a saddle point analysis to obtain precise asymptotics for $A_{v,\lambda}$ and $B_{v,\lambda}$, as $\lambda + v$ tends to ∞. In particular, the following proposition shows that these coefficients decay exponentially in v, with an explicit rate function given by $\Psi_{v,\lambda}$ in (6). This exact form of this rate function will eventually serve as the source of the phase transitions for $d_{B;\delta}(e^{-x})$ and $d_{B;\delta}(e^x)$ explained in Theorem 2 and Theorem 3, respectively. Indeed, one might already observe that $\Psi_{v,\lambda} = \lambda G\left(\frac{v}{\lambda}\right)$, where we recall the function $G(x)$ from those results. The below proposition will be stated a bit informally; we refer to Proposition 19 below for the more precise formulation needed for our purposes.

▶ **Proposition 13.** *Recall the quantities $A_{v,\lambda}$ and $B_{v,\lambda}$ from (5) for any integer $v \geq 0$ and real number $\lambda \geq \frac{1}{2}$. Denote*

$$\Psi_{v,\lambda} = \sqrt{v^2+\lambda^2} + v\log\left(\frac{\sqrt{v^2+\lambda^2} - v}{\lambda}\right). \tag{6}$$

As $v + \lambda$ tends to ∞, we have that

$$(-1)^v A_{v,\lambda} = (\lambda+v)^{O(1)} \exp\left(\Psi_{v,\lambda} - \lambda\right); \qquad B_{v,\lambda} = (\lambda+v)^{O(1)} \exp\left(\Psi_{v,\lambda} + \lambda\right).$$

Combining Proposition 11 and Proposition 13, we obtain the following corollary, which provides nearly sharp bounds on the error one can achieve for a degree d polynomial approximation of e^{-x} and e^{x} on $[0, B]$. Once again, the below proposition will be stated a bit informally, and we refer to Proposition 23 below for a more precise formulation.

▶ **Corollary 14.** *Let $d \geq 1$ be an integer, and let $B \geq 1$ be a real number. Set $\lambda = \frac{B}{2}$ and recall Ψ from (6). As $\lambda + d$ tends to ∞, we have that*

$$\inf_{p \in \mathcal{P}_d} \sup_{x \in [0,B]} \left|p(x) - e^{-x}\right| = (\lambda + d)^{O(1)} \exp(\Psi_{d,\lambda} - \lambda), \tag{7}$$

$$\inf_{p \in \mathcal{P}_d} \sup_{x \in [0,B]} \left|p(x) - e^{x}\right| = (\lambda + d)^{O(1)} \exp(\Psi_{d,\lambda} + \lambda). \tag{8}$$

Now Theorem 2 and Theorem 3 will follow from an explicit analysis of (7) and (8), respectively. For the purposes of this outline, we will omit the remaining details of analyzing the asymptotics of these expressions (referring to Section 4 for a more detailed exposition). However, let us briefly explain how the transitions from $B = \omega\big(\log(\delta^{-1})\big)$ to $B = o\big(\log(\delta^{-1})\big)$ arise, for example in Theorem 2.

They ultimately follow from the fact that the function $\Psi_{v,\lambda} - \lambda$ behaves differently depending on whether $v = O(\lambda)$ or $v = \Omega(\lambda)$, as different terms in the definition of $\Psi_{v,\lambda}$ are dominant in each of these settings; this is stated more precisely through the following lemma, which will be established in Section 4.2 below.

▶ **Lemma 15.** *Let $\lambda \geq \frac{1}{2}$ be a real number, $v \geq 0$ be an integer, denote $\kappa = \frac{v}{\lambda}$, and recall the function $\Psi_{v,\lambda}$ from (6).*

1. *If $v \leq 2\lambda$ (that is, $\kappa \leq 2$) then $\Psi_{v,\lambda} - \lambda = -\dfrac{v^2}{2\lambda}\big(1 + O(\kappa)\big)$.*
2. *If $v \geq 2\lambda$ (that is, $\kappa \geq 2$) then $\Psi_{v,\lambda} - \lambda = -v \log\left(\dfrac{v}{\lambda}\right)\Big(1 + O\big((\log \kappa)^{-1}\big)\Big)$.*

In particular, the first part of Lemma 15 gives rise to the first part of Theorem 2, and the second part of Lemma 15 gives rise to the third part of Theorem 2.

Finally, we need one last tool to prove the fourth part of Theorem 2. In Theorem 3 as well as the first three parts of Theorem 2, our degree lower bound ultimately followed by finding a large coefficient in the Chebyshev expansion of $e^{\lambda x + \lambda}$ or $e^{\lambda x - \lambda}$ and then applying Proposition 11. However, when B (and hence λ) is very large compared to the desired error, the Chebyshev expansion of $e^{\lambda x - \lambda}$ actually has no sufficiently large coefficients.

We instead take a different approach in this last case. We observe that any polynomial p satisfying the bound $\sup_{x \in [0,B]} \left|p(x) - e^{-x}\right| < \delta$ must have,

- $|p(x)| \leq 2\delta$ in the entire interval $x \in [\log(\delta^{-1}), B]$, and
- $p(0) \geq 1 - \delta$.

It is known (see Fact 16 below) that the polynomial of lowest degree achieving these two properties must in fact be a (rescaled) Chebyshev polynomial. We then show that a Chebyshev polynomial requires degree $\Omega(\sqrt{B \log(\delta^{-1})})$ to realize these properties, from which the desired result follows.

3 Preliminaries

3.1 Notation

For a nonnegative integer d, we write $\mathcal{P}_d$ to denote the set of polynomials $p : \mathbb{R} \to \mathbb{R}$ with real coefficients of degree at most d. For a Boolean predicate P, we write

$$\mathbf{1}_P = \begin{cases} 1 & \text{if } P \text{ is true,} \\ 0 & \text{if } P \text{ is false.} \end{cases}$$

All logarithms in this paper are assumed to have base e, and we similarly write $\exp(x) := e^x$.

3.2 Chebyshev Polynomials

In this paper we make heavy use of the Chebyshev polynomials. Chebyshev polynomials appear prominently throughout polynomial approximation theory, and have been used in numerous other areas of theoretical computer science, including in Boolean function analysis and quantum computing; see e.g., [11]. Here we define them and give some of their well-known properties which will be important in our proofs. We refer the reader to [20] for more details.

The degree d *monic Chebyshev polynomial* $Q_d(x) \in \mathcal{P}_d$ is defined in many equivalent ways:

Definition 1 Set $Q_0(x) = 1$ and, for each $d \geq 1$, define $Q_d(x)$ by imposing that, for each $\theta \in [0, 2\pi]$, we have $Q_d(\cos\theta) = 2^{1-d}\cos(d\theta)$.

Definition 2 $Q_0(x) = 1$, $Q_1(x) = x$, and for $d \geq 2$ we have $Q_d(x) = x \cdot Q_{d-1}(x) - \frac{1}{2}Q_{d-2}(x)$.

Definition 3 $Q_0(x) = 1$ and for each $d \geq 1$ we have $Q_d(x) = 2^{1-d} \cdot \sum_{k=0}^{\lfloor d/2 \rfloor} \binom{d}{2k}(x^2-1)^k x^{d-2k}$.

In particular, $Q_d(x)$ is an even function when d is even, and an odd function when d is odd. From Definition 1, one observes several simple properties of $Q_d(x)$ for all $d > 0$ for $x \in [-1, 1]$:

- For all $x \in [-1, 1]$, we have $2^{d-1} \cdot Q_d(x) \in [-1, 1]$.
- All d roots of $Q_d(x)$ lie in $[-1, 1]$, and they lie at the points $x = \cos\left(\pi(2k+1)/2d\right)$ for each integer $0 \leq k < d$.
- On the interval $[-1, 1]$, the extrema of Q_d are located at the points $x = \cos(\pi k/d)$ for each integer $0 \leq k \leq d$. $Q_d(x)$ alternates between the values 2^{1-d} and -2^{1-d} at these extrema, starting at $Q_d(\cos(0)) = Q_d(1) = 2^{1-d}$.

Outside of the interval $[-1, 1]$, it is well-known that the Chebyshev polynomials exhibit useful extremal properties.

▶ **Fact 16.** *For every integer $d > 0$, every polynomial $p(x) \in \mathcal{P}_d$ of degree d such that $2^{d-1} \cdot p(x) \in [-1, 1]$ for all $x \in [-1, 1]$, and every real $x' \notin [-1, 1]$, we have $|Q_d(x')| \geq |p(x')|$.*

Proof. Assume to the contrary that there is an $x' \notin [-1, 1]$ such that $|Q_d(x')| < |p(x')|$. By rescaling p by a factor in the range $(|Q_d(x')|/|p(x')|, 1)$, we can further assume that $2^{d-1} \cdot p(x) \in (-1, 1)$ for all $x \in [-1, 1]$. By the symmetry of $Q_d(x)$ (and by negating p if necessary), we may also assume without loss of generality that $x' > 1$ and that $p(x') > Q_d(x') > 0$ are positive.

Define the difference polynomial $g(x) = p(x) - Q_d(x)$, which has degree at most d. Consider the $d + 2$ points $x_{-1} > x_0 > x_1 > x_2 > \cdots > x_d \in \mathbb{R}$ given by

- $x_{-1} = x'$, and
- $x_k = \cos(\pi k/d)$ for each integer $0 \leq k \leq d$.

We have $g(x_{-1}) = p(x') - Q_d(x') > 0$ by assumption. For even $k \geq 0$ we have $Q_d(x_k) = 2^{1-d}$ and $|p(x_k)| < 2^{1-d}$, and so $g(x_k) < 0$. Similarly, for odd $k > 0$ we have $g(x_k) > 0$. Hence, $g(x)$ alternates signs at least $d+2$ times in the interval $[x_d, x_{-1}]$, meaning it has at least $d+1$ roots in that interval, a contradiction. ◀

In fact, $Q_d(1+\varepsilon)$ for small $\varepsilon > 0$ is very closely approximated by an exponential in $\sqrt{\varepsilon}$:

▶ **Fact 17.** *For any $\varepsilon > 0$ we have $Q_d(1+\varepsilon) = 2^{-d} \cdot e^{d\sqrt{2\varepsilon}(1+O(\sqrt{\varepsilon}))}$.*

Proof. Extending Definition 1 of $Q_d(x)$ to $x \notin [-1,1]$, we find that

$$Q_d(x) = 2^{-d}\left(\left(x - \sqrt{x^2-1}\right)^d + \left(x + \sqrt{x^2-1}\right)^d\right), \qquad \text{for each } x \text{ with } |x| > 1.$$

For $x = 1+\varepsilon$ with $\varepsilon > 0$, we thus get

$$Q_d(1+\varepsilon) = 2^{-d}\left(\left(1 - \sqrt{2\varepsilon} + O(\varepsilon)\right)^d + \left(1 + \sqrt{2\varepsilon} + O(\varepsilon)\right)^d\right) = 2^{-d}e^{d\sqrt{2\varepsilon}(1+O(\sqrt{\varepsilon}))},$$

as desired. ◀

3.3 Chebyshev Expansion Coefficients

In this section we prove Proposition 11, which shows how the coefficients of a function f written in the basis of Chebyshev polynomials can be used to bound how well f can be approximated by low-degree polynomials.

Proof of Proposition 11. Observe for any $d \in \mathbb{Z}_{\geq 0}$ and $x \in [-1,1]$ that $\left|Q_d(x)\right| \leq 2^{1-d}$, which follows from (2) after setting $x = \cos\theta$. This implies the absolute convergence of $f(x)$ for $x \in [-1,1]$, since $\sum_{j=0}^{\infty} |a_j| < \infty$. So, it remains to establish (3).

To establish the upper bound there, define

$$H_D(x) = \sum_{j=0}^{D-1} 2^{j-1} a_j Q_j(x). \tag{9}$$

Since $\left|Q_d(x)\right| \leq 2^{1-d}$ for each $x \in [-1,1]$, we have that

$$\sup_{x\in[-1,1]} \left|H_D(x) - f(x)\right| = \sup_{x\in[-1,1]} \left|\sum_{j=D}^{\infty} 2^{j-1} a_j Q_j(x)\right| \leq \sum_{j=D}^{\infty} |a_j|, \tag{10}$$

which proves the upper bound in (3).

To establish the lower bound, fix $p \in \mathcal{P}_{D-1}$, and let $c_0, c_1, \ldots \in \mathbb{R}$ satisfy

$$p(x) = \sum_{j=0}^{D-1} 2^{j-1} c_j Q_j(x), \quad \text{and } c_j = 0 \text{ for } j \geq D.$$

Then, applying (2), we obtain

$$p(\cos\theta) - f(\cos\theta) = \sum_{j=0}^{\infty} 2^{j-1}(a_j - c_j) Q_j(\cos\theta) = \frac{a_0 - c_0}{2} + \sum_{j=1}^{\infty} (a_j - c_j)\cos(j\theta). \tag{11}$$

Define $b_0, b_1, \ldots \in \mathbb{R}$ by setting $b_0 = \frac{a_0 - c_0}{2}$ and $b_j = a_j - c_j$ for $j > 0$, and observe that

$$\int_0^{2\pi} \cos(j\theta)\cos(k\theta)d\theta = 0, \qquad \text{for } j \neq k,$$

and

$$\int_0^{2\pi} \cos^2(k\theta)d\theta = \frac{1}{|k|}\int_0^{2\pi} \cos^2(\theta)d\theta = \pi|k|^{-1}\mathbf{1}_{k\neq 0} + (2\pi)\mathbf{1}_{k=0},$$

it follows that

$$\begin{aligned}\int_0^{2\pi} \left|p(\cos\theta) - f(\cos\theta)\right|^2 &= \int_0^{2\pi} \left(\sum_{j=0}^{\infty} b_j \cos(j\theta)\right)^2 d\theta \\ &= \sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\int_0^{2\pi} b_j b_k \cos(j\theta)\cos(k\theta)d\theta = 2\pi b_0^2 + \pi\sum_{k=1}^{\infty}\frac{b_k^2}{k}.\end{aligned}$$

Thus, since $b_j = a_j$ for $j \geq D$ (since $c_j = 0$ for $j \geq D$), we deduce

$$\begin{aligned}\sup_{x\in[-1,1]}\left|p(x) - f(x)\right|^2 = \sup_{\theta\in[0,2\pi]}\left|p(\cos\theta) - f(\cos\theta)\right|^2 &\geq \frac{1}{2\pi}\int_0^{2\pi}\left|p(\cos\theta) - f(\cos\theta)\right|^2 d\theta \\ &\geq \frac{1}{2}\sum_{k=D}^{\infty}\frac{a_k^2}{k},\end{aligned}$$

which yields the proposition. ◀

Finally, we recall a well-known [20] identity expressing a monomial as an explicit linear combination of Chebyshev polynomials.

▶ **Lemma 18** ([20], 2.14). *For any integer $n \geq 0$, we have that*

$$x^n = \sum_{k=0}^{\lfloor n/2\rfloor} 2^{-2k}\binom{n}{k}Q_{n-2k}(x).$$

4 Degree Bounds for Polynomial Approximations

4.1 Estimating $A_{v,\lambda}$ and $B_{v,\lambda}$

In this section we analyze $A_{v,\lambda}$ and $B_{v,\lambda}$ from (5). We begin with the proof of Lemma 12.

Proof of Lemma 12. We only establish the first statement in (4), as the proof of the second is entirely analogous. To that end, first using the series expansion for $e^{-z} = \sum_{n=0}^{\infty}\frac{(-z)^n}{n!}$ and then applying Lemma 18, yields

$$e^{-\lambda x-\lambda} = e^{-\lambda}\sum_{n=0}^{\infty}\frac{(-\lambda)^n}{n!}x^n = e^{-\lambda}\sum_{n=0}^{\infty}\frac{(-\lambda)^n}{n!}\sum_{k=0}^{\lfloor n/2\rfloor}2^{-2k}\binom{n}{k}Q_{n-2k}(x).$$

Then, by setting $v = n - 2k$, we obtain

$$e^{-\lambda x-\lambda} = e^{-\lambda}\sum_{v=0}^{\infty}Q_v(x)\sum_{k=0}^{\infty}\frac{(-\lambda)^{v+2k}}{(v+2k)!}2^{-2k}\binom{v+2k}{k} = \sum_{v=0}^{\infty}2^{v-1}A_{v,\lambda}Q_v(x),$$

from which we deduce the lemma. ◀

We next have the following proposition that more precisely formulates Proposition 13.

▶ **Proposition 19.** *There exist constants $C, c > 0$ such that the following holds. For any integer $v \geq 0$ and real number $\lambda \geq \frac{1}{2}$, recall the quantities $A_{v,\lambda}$ and $B_{v,\lambda}$ from* (5) *and $\Psi_{\lambda,v}$ from* (6)*.*

1. *For any v and λ as above, we have that*

$$c(v+\lambda)^{-1}\exp\big(\Psi_{v,\lambda}-\lambda\big) \leq (-1)^v A_{v,\lambda} \leq C(v+\lambda)\exp\big(\Psi_{v,\lambda}-\lambda\big);$$
$$c(v+\lambda)^{-1}\exp\big(\Psi_{v,\lambda}+\lambda\big) \leq B_{v,\lambda} \leq C(v+\lambda)\exp\big(\Psi_{v,\lambda}+\lambda\big).$$

2. *If $v \leq \lambda$, then*

$$c(v+\lambda)^{-1}\exp\big(\Psi_{v,\lambda}-\lambda\big) \leq (-1)^v A_{v,\lambda} \leq C\lambda^{-1/2}\exp\big(\Psi_{v,\lambda}-\lambda\big).$$

To to establish the above result, observe that the two quantities $A_{v,\lambda}$ and $B_{v,\lambda}$ are quite similar, in that they both involve certain sum given by

$$E_{v,\lambda} = \sum_{n-v\in 2\mathbb{Z}_{\geq 0}} \frac{\lambda^n}{2^n n!}\binom{n}{\frac{n-v}{2}}.a \tag{12}$$

We therefore require the following proposition that estimates $E_{v,\lambda}$. Observe its second statement slightly improves the upper bound in its first statement for λ large (which will be useful in analyzing $d_{B;\delta}$ in the regime of large B below).

▶ **Proposition 20.** *There exist constants $C, c > 0$ such that the following holds. For any integer $v \geq 0$ and real number $\lambda \geq \frac{1}{2}$, recall the quantities $E_{v,\lambda}$ and $\Psi_{v,\lambda}$ from* (12) *and* (6)*, respectively.*

1. *If $v \geq \lambda$, then $c(v+\lambda)^{-1}\exp\big(\Psi_{v,\lambda}\big) \leq E_{v,\lambda} \leq C(v+\lambda)\exp\big(\Psi_{v,\lambda}\big)$.*
2. *If $v \leq \lambda$, then $c(v+\lambda)^{-1}\exp\big(\Psi_{v,\lambda}\big) \leq E_{v,\lambda} \leq C\lambda^{-1/2}\exp\big(\Psi_{v,\lambda}\big)$.*

Proposition 13 now follows directly from Proposition 20.

Proof of Proposition 19 Assuming Proposition 20. Given Proposition 20, Proposition 19 follows from the facts that $(-1)^v A_{v,\lambda} = 2e^{-\lambda}E_{v,\lambda}$ and $B_{v,\lambda} = 2e^{\lambda}E_{v,\lambda}$. ◀

We must thus establish Proposition 20, to which end we begin with the following lemma.

▶ **Lemma 21.** *For any integer $v \geq 1$ and real number $\lambda \geq \frac{1}{2}$, we have that*

$$E_{v,\lambda} = \Theta\Bigg(\sum_{n-v\in 2\mathbb{Z}_{\geq 0}} \big(n^2 - v^2 + n\big)^{-1/2}\exp\big(F(n)\big)\Bigg), \tag{13}$$

where for any real number $n \geq v$ we have defined $F(n) = F(n)$ by

$$F(n) = n\log\lambda - n\log 2 + n - \Big(\frac{n-v}{2}\Big)\log\Big(\frac{n-v}{2}\Big) - \Big(\frac{n+v}{2}\Big)\log\Big(\frac{n+v}{2}\Big). \tag{14}$$

Proof. The explicit form (12) for $E_{v,\lambda}$ and the Stirling estimate $n! = \Theta\big((n+1)^{n+1/2}e^{-n}\big)$, which holds uniformly in $n \geq 0$, together imply

$$E_{v,\lambda} = \Theta\Bigg(\sum_{n-v\in 2\mathbb{Z}_{\geq 0}} \big((n+2)^2 - v^2\big)^{-1/2}\exp\big(F(n)\big)\Bigg).$$

From this, we deduce the lemma since $(n+2)^2 - v^2 = \Theta(n^2 - v^2 + n)$, uniformly in $n \geq v$. ◀

The right side of (13) will be dominated by the terms near which F is maximized, so we next perform a critical point analysis on F.

▶ **Lemma 22.** *Fix an integer $v \geq 1$ and a real number $\lambda \geq \frac{1}{2}$, and set $n_0 = n_0(v,\lambda) = \sqrt{v^2+\lambda^2}$. There exist constants $c, C > 0$ (independent of v and λ) such that the following holds.*

1. *The function $F(n)$ is maximized at $n = n_0$, and $F(n_0) = \Psi_{v,\lambda}$ (recall (6)).*
2. *For $z \geq 2\lambda$, we have that $F(n_0+z) \leq F(n_0) - cz$.*
3. *For at least one choice of $m \in \{\lfloor n_0 \rfloor, \lceil n_0 \rceil\}$, we have that $F(m) \geq F(n_0) - C$.*
4. *If $v \leq \lambda$, then for any $z \in [v - n_0, 2\lambda]$ we have that $F(n_0+z) \leq F(n_0) - c\lambda^{-1}z^2$.*

Proof. Using the explicit form (14) for $F(n)$, we deduce for $n \geq v$ that

$$F'(n) = \log\lambda - \frac{1}{2}\log(n^2-v^2), \quad \text{and} \quad F''(n) = \frac{n}{v^2-n^2} \in \left[-\frac{1}{n}, 0\right], \tag{15}$$

which implies that $F'(n_0) = 0$ and that F is maximized at n_0. Upon insertion into (14) (and recalling (6)), we also find that

$$F(n_0) = \sqrt{v^2+\lambda^2} + v\log\left(\frac{\sqrt{v^2+\lambda^2}-v}{\lambda}\right) = \Psi_{v,\lambda},$$

which verifies the first statement of the lemma.

To establish the second, first observe since $F''(n) \leq 0$ and $n_0^2 - v^2 = \lambda^2$ that

$$F'(n_0+\lambda) = \log\lambda - \frac{1}{2}\log\left((n_0+\lambda)^2 - v^2\right) \leq \log\lambda - \frac{1}{2}\log(n_0^2+\lambda^2-v^2) = -\frac{\log 2}{2}. \tag{16}$$

Thus, we deduce for $z \geq 2\lambda$ that

$$F(n_0+z) - F(n_0) \leq F(n_0+z) - F(n_0+\lambda) \leq (z-\lambda)F'(n_0+\lambda) \leq \frac{\log 2}{2}(\lambda - z) \leq -\frac{z\log 2}{4},$$

where in the first inequality we used the fact that F is maximized at n_0; in the second we used the fact that $F''(n) \leq 0$; in the third we used (16); and in the fourth we used the fact that $z - \lambda \geq \frac{z}{2}$. This verifies the second statement of the lemma.

To show the third part of the lemma, we separately consider the cases when $v \leq \lambda^2$ and $v \geq \lambda^2$. In the former situation $v \leq \lambda^2$, we select $m = \lceil n_0 \rceil$. Since $F'''(n) = (n^2+v^2)(n^2-v^2)^{-2} \geq 0$, we then have for for $n \in [n_0, m]$ that

$$F'(n) \geq (n-n_0)^2 F''(n_0) \geq -\frac{(n-n_0)^2}{n_0} \geq -\frac{1}{n_0} \geq \frac{n_0}{\lambda^2},$$

using the last second identity in (15). Since $n_0 \leq \lambda + v \leq 3\lambda^2$ (due to the facts that $\lambda \geq \frac{1}{2}$ and $v \leq \lambda^2$), it follows that $F'(n) \geq -\frac{1}{3}$ for $n \in [n_0, m]$, which implies that $F(m) \geq F(n_0) - \frac{1}{3}$ if $v \leq \lambda^2$.

Now instead suppose that $v \geq \lambda^2$, in which case we take $m = \lfloor n_0 \rfloor = v$. Then, using the second identity in (15), we obtain

$$F(n_0) - F(v) = \int_0^{n_0-v} F'(v+z)dz = -\frac{1}{2}\int_0^{n_0-v} \log\left(\frac{2vz+z^2}{\lambda^2}\right)dz \leq -\frac{1}{2}\int_0^{n_0-v} \log\left(\frac{vz}{\lambda^2}\right)dz.$$

Now set $r = \frac{\lambda^2}{v}$, and observe that $n_0 - v \leq r \leq 1$. This yields

$$F(m) \geq F(n_0) + \frac{1}{2}\int_0^r \log\left(\frac{z}{r}\right)dz = F(n_0) + \frac{r}{2}\int_0^1 (\log y)dy = F(n_0) - \frac{r}{2} \geq F(n_0) - 1,$$

where in the first equality we changed variables $z = ry$. This verifies the third part of the lemma.

To establish the fourth part of the lemma, let us first consider the case when $z \in [v-n_0, 0]$. Then, since $F'(n_0) = 0$ and $F'''(n) = (n^2+v^2)(n^2-v^2)^{-2} \geq 0$ for each $n \geq v$, we have that

$$F(n_0+z) \leq F(n_0) + \frac{z^2 F''(n_0)}{2} = F(n_0) - \frac{z^2(\lambda^2+v^2)^{1/2}}{2\lambda^2}, \quad \text{for } z \in [v-n_0, 0],$$

where in the last equality we used the second identity in (15) for $F'(n)$ (and the fact that $n_0 = \sqrt{\lambda^2+v^2}$). Thus, we deduce that

$$F(n_0+z) \leq F(n_0) - \frac{z^2}{2\lambda}, \quad \text{for } z \in [v-n_0, 0]. \tag{17}$$

Next we consider the case when $z \in [0, 2\lambda]$. In this case, the second identity in (15) implies for each $m \in [n_0, n_0+2\lambda]$ that

$$F''(m) = \frac{m}{v^2-m^2} \leq -\frac{\lambda}{m^2} \leq -\frac{1}{16\lambda},$$

where in the last inequality we used the fact that $m \leq n_0 + 2\lambda \leq 4\lambda$ (as $v \leq \lambda$ and $n_0 = \sqrt{v^2+\lambda^2}$). Thus, it follows from the fact that $F'(n_0) = 0$ that

$$F(n_0+z) \leq F(n_0) - \frac{z^2}{32\lambda}, \quad \text{for each } z \in [0, 2\lambda]. \tag{18}$$

Now the fourth statement of the lemma follows from (17) and (18). ◀

Now we can establish Proposition 20.

Proof of Proposition 20. We begin by establishing the lower bound on $E_{v,\lambda}$, simultaneously in both cases $v \geq \lambda$ and $v \leq \lambda$. To that end, we first apply Lemma 21 and then use the third part of Lemma 22 to bound the sum on the right side of (13) its summand corresponding to a suitable choice of index $m \in \{\lfloor n_0 \rfloor, \lceil n_0 \rceil\}$. This yields constants $c_1, c_2 > 0$ such that

$$E_{v,\lambda} \geq c_1\big(\lambda^2+v^2\big)^{-1/2} \exp\big(-F(m)\big) \geq c_1(\lambda+v)^{-1} \exp\big(F(n_0) - c_2\big),$$

which implies the lower bound on $E_{v,\lambda}$ (in either case $v \geq \lambda$ or $v \leq \lambda$), since $F(n_0) = \Psi_{\lambda,v}$.

To establish the upper bound in the case $v \geq \lambda$, observe that Lemma 21; the fact that $F(n) \leq F(n_0) = \Psi_{v,\lambda}$ for each $n \in [v, n_0+2\lambda]$ (by the first part of Lemma 22); and the existence of a constant $c > 0$ such that $F(n_0+z) \leq F(n_0) - cz$ for $z \geq 2\lambda$ (by the second part of Lemma 22) together yield a constant $C_1 > 0$ such that

$$E_{v,\lambda} \leq C_1(n_0+2\lambda)\exp(\Psi_{v,\lambda}) \sum_{z=2\lambda}^{\infty} e^{-cz} \leq 2c^{-1}C_1(n_0+2\lambda)\exp(\Psi_{v,\lambda}).$$

This establishes the upper bound on $E_{v,\lambda}$ when $v \geq \lambda$.

In the latter case $v \leq \lambda$, we proceed as above but additionally use the facts that $F(n_0+z) \leq F(n_0) - c\lambda^{-1}z^2$ for $z \in [v-n_0, 2\lambda]$ (by the fourth part of Lemma 22) to deduce for some constant $C_2 > 0$ that

$$\begin{aligned}
E_{v,\lambda} &\leq C_1 \exp(\Psi_{v,\lambda})\bigg(\sum_{z=v-n_0}^{2\lambda} \big((n_0+z)^2 - v^2 + n_0 + z\big)^{-1/2} e^{-cz^2/\lambda} + \sum_{z=2\lambda}^{\infty} e^{-cz} \bigg) \\
&\leq C_1 \exp(\Psi_{v,\lambda})\bigg(\sum_{|z|\leq\lambda/4} \big((n_0+z)^2 - v^2 + n_0 + z\big)^{-1/2} e^{-cz^2/\lambda} + \sum_{|z|\geq\lambda/4} e^{-cz^2/\lambda} + \sum_{z=2\lambda}^{\infty} e^{-cz} \bigg) \\
&\leq C_1 \exp(\Psi_{v,\lambda})\bigg(12\lambda^{-1} \sum_{|z|\leq\lambda/4} e^{-cz^2/\lambda} + 35c^{-1}e^{-2\lambda} \bigg) \leq C_2\lambda^{-1/2}\exp(\Psi_{v,\lambda}),
\end{aligned}$$

where in the third inequality we used the fact that $(n_0+z)^2 - v^2 \geq \frac{\lambda^2}{144}$ for $|z| \leq \frac{\lambda}{4}$ (as $n_0 = \sqrt{\lambda^2+v^2} \geq \frac{\lambda}{3} + v$ for $\lambda \geq v$). This establishes the upper bound on $E_{v,\lambda}$ when $v \leq \lambda$. ◀

4.2 Estimates for the Minimum Polynomial Approximation Error

In this section we establish the following proposition, which is the variant of Corollary 14 that will be useful for our purposes.

▶ **Proposition 23.** *There exist constants $C, c > 0$ such that the following holds. Let $d \geq 1$ be an integer, and let $B \geq 1$ be a real number. Set $\lambda = \frac{B}{2}$ and recall Ψ from (6).*

1. *For any B and d as above, we have*

$$c(d+\lambda)^{-3/2}\exp(\Psi_{d,\lambda}-\lambda) \leq \inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right| \leq C(d+\lambda)^2\exp(\Psi_{d,\lambda}-\lambda); \tag{19}$$

$$c(d+\lambda)^{-3/2}\exp(\Psi_{d,\lambda}+\lambda) \leq \inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{x}\right| \leq C(d+\lambda)^2\exp(\Psi_{d,\lambda}+\lambda). \tag{20}$$

2. *If $B \geq 2d$, then we have that*

$$c(d+\lambda)^{-3/2}\exp(\Psi_{d,\lambda}-\lambda) \leq \inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right| \leq C\exp(\Psi_{d,\lambda}-\lambda). \tag{21}$$

We will establish Proposition 23 as a consequence of Proposition 11 and Proposition 19. However, before doing so, it will be useful to obtain some properties for $\Psi_{v,\lambda}$. Therefore, we first prove Lemma 15.

Proof of Lemma 15. First observe that

$$\Psi_{v,\lambda}-\lambda = \lambda\Big(\sqrt{\kappa^2+1}-1+\kappa\log\left(\sqrt{\kappa^2+1}-\kappa\right)\Big).$$

In particular, if $\kappa \leq 2$ then using the series expansions

$$\sqrt{z^2+1} = 1+\frac{z^2}{2}+O(z^3), \quad \text{and} \quad \log(1+z) = z+O(z^2), \quad \text{valid for } z\in[0,2],$$

we obtain that

$$\Psi_{v,\lambda} = \lambda\left(\frac{\kappa^2}{2}+O(\kappa^3)+\kappa\log\left(1-\kappa+O(\kappa^2)\right)\right) = -\frac{\kappa^2\lambda}{2}\big(1+O(\kappa)\big).$$

If instead $\kappa \geq 2$ then using the series expansion

$$\sqrt{z^2+1} = z+\frac{1}{2z}+O(z^{-2}), \quad \text{and} \quad \log(z^{-1}+z^{-2}) = -\log z - O(z^{-1}), \quad \text{valid for } |z|\geq 2,$$

we deduce that

$$\Psi_{v,\lambda} = \lambda\left(\kappa+O\Big(\frac{1}{\kappa}\Big)+\kappa\log\left(\frac{1}{2\kappa}+O\Big(\frac{1}{\kappa^2}\Big)\right)\right) = -\lambda\kappa\log\kappa\Big(1+O\big((\log\kappa)^{-1}\big)\Big).$$

This establishes the lemma. ◀

Now we can establish Proposition 23.

Proof of Proposition 23. First observe that by rescaling (namely, replacing x with $\lambda(x+1)$ or $-\lambda(x+1)$), we have that

$$\begin{aligned}
\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{x}\right| &= \inf_{p\in\mathcal{P}_d}\sup_{x\in[-1,1]}\left|p(x)-e^{\lambda x+\lambda}\right|;\\
\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right| &= \inf_{p\in\mathcal{P}_d}\sup_{x\in[-1,1]}\left|p(x)-e^{-\lambda-\lambda x}\right|.
\end{aligned} \tag{22}$$

Therefore, Proposition 11 and the definitions of $A_{v,\lambda}$ and $B_{v,\lambda}$ from (5), together yield

$$\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right|\geq(2d)^{-1/2}\left|A_{v,\lambda}\right|;\qquad \inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{x}\right|\geq(2d)^{-1/2}\left|B_{v,\lambda}\right|. \tag{23}$$

Thus, the lower bounds on the minimal error for e^{-x} and e^{x} in both of the cases listed in the proposition follow from (23) and the lower bounds on $|A_{v,\lambda}|$ and $|B_{v,\lambda}|$ from the first part of Proposition 19.

Now let us establish the upper bounds in this proposition; in what follows, $C>0$ will denote a constant (uniform in d and λ) that might change between appearances. We first show (20), to which end, observe that (22), Proposition 11, and the upper bound for $B_{v,\lambda}$ from the first part of Proposition 19 together yield

$$\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right|\leq\sum_{v=d}^{\infty}B_{v,\lambda}\leq C\sum_{v=d}^{\infty}(v+\lambda)\exp(\Psi_{v,\lambda}+\lambda). \tag{24}$$

Next, observe from the second part of Lemma 15 that

$$\Psi_{v,\lambda}+\lambda\leq -v,\qquad \text{for } v>C\lambda. \tag{25}$$

Further observe that

$$\Psi_{v,\lambda}\text{ is decreasing in } v\geq 0 \text{ for fixed } \lambda, \tag{26}$$

since

$$\frac{\partial}{\partial v}\Psi_{v;\lambda}=\log\left(\sqrt{\kappa^2+1}-\kappa\right)\leq 0,\qquad \text{for } \kappa=\frac{v}{\lambda}\geq 0. \tag{27}$$

From (24), (25), and (26), it follows that

$$\begin{aligned}\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{x}\right|&\leq C(d+\lambda)\exp\left(\Psi_{d,\lambda}+\lambda\right)\left(d+C\lambda+\sum_{v\geq d+C\lambda}(v+\lambda)e^{-v}\right)\\&\leq C(d+\lambda)^2\exp\left(\Psi_{d,\lambda}+\lambda\right).\end{aligned}$$

This establishes (20); the proof of (19) is omitted as it is entirely analogous.

Now let us establish the improved upper bound on the minimum error in the case when $B\geq 2d$. To that end, we as before apply (22), Proposition 11, and the upper bound for $A_{v,\lambda}$ from the second part of Proposition 19 to obtain

$$\inf_{p\in\mathcal{P}_d}\sup_{x\in[0,B]}\left|p(x)-e^{-x}\right|\leq C\sum_{v=d}^{\infty}\lambda^{-1/2}\exp(\Psi_{v,\lambda}-\lambda). \tag{28}$$

By (27), we have that $\Psi_{v;\lambda}<0$ for $v>0$ and moreover that

$$\begin{aligned}\frac{\partial}{\partial v}\Psi_{v;\lambda}&=\log\left(1-\frac{v}{\lambda}+O\Big(\frac{v^2}{\lambda^2}\Big)\right)=O\left(\frac{v^2}{\lambda^2}\right)-\frac{v}{\lambda},\qquad \text{for } v\leq 10\lambda;\\ \frac{\partial}{\partial v}\Psi_{v;\lambda}&=\log\left(\frac{\lambda}{2v}+O\Big(\frac{\lambda^2}{v^2}\Big)\right)=\left(1+O\Big(\frac{\lambda}{v}\Big)\right)\log\left(\frac{\lambda}{2v}\right),\qquad \text{for } v\geq 10\lambda.\end{aligned}$$

In particular, there exist constants $c_1,c_2>0$ such that for $v,\lambda\geq\frac{1}{2}$ we have $\frac{\partial}{\partial v}\Psi_{v;\lambda}\leq -c_1$ if $v\geq c_2\lambda$ and $\frac{\partial}{\partial v}\Psi_{v;\lambda}\leq -\frac{v}{2\lambda}$ for $v\leq c_2\lambda$. Thus,

$$\Psi_{v;\lambda}-\Psi_{d;\lambda}\leq -\frac{c_2}{4\lambda}(v-d)^2,\quad \text{for } v\leq c_2\lambda;\qquad \Psi_{v;\lambda}-\Psi_{d;\lambda}\leq C^{-1}(d-v),\quad \text{for } v\geq c_2\lambda,$$

and so

$$\sum_{v=d}^{\infty}\exp(\Psi_{v,\lambda}-\lambda)\leq C\lambda^{1/2}\exp(\Psi_{d,\lambda}-\lambda). \tag{29}$$

The upper bound in (21) now follows from (28) and (29). ◀

4.3 Proofs of Theorem 2 and Theorem 3

In this section we establish Theorem 2 and Theorem 3. Recalling the function $G(x)$ and the quantity $\Psi_{v,\lambda}$ from (6) from these statements, both of these proofs will use the fact that

$$\Psi_{v,\lambda} = \lambda G\Big(\frac{v}{\lambda}\Big). \tag{30}$$

We begin with the proof of Theorem 3.

Proof of Theorem 3. Set $\lambda = \frac{B}{2}$. Observe that there exists a constant $c_1 > 0$ such that $G'(z) < -c_1$ whenever $z \in [z_* - c_1, z_* + c_1]$. Thus, $G(x) + 1 > c_1(z - x_*)$, and so (30) yields for some constant $c_2 > 0$ that

$$(d+\lambda)^{-1}\exp(\Psi_{d,\lambda} + \lambda) > (d+\lambda)^{-1}\exp(c_2\lambda^{1/2}) > 10,$$

if $d < z_*\lambda - \lambda^{1/2}$ and λ is sufficiently large. Thus, by the lower bound in Proposition 23, for any λ sufficiently large and $\delta \leq \frac{1}{2}$ we have

$$d_{B;\delta} \geq \big(z_* + o(1)\big)\lambda = \big(z_* + o(1)\big)\frac{B}{2}. \tag{31}$$

Now, assume first that $B = \omega\big(\log(\delta^{-1})\big)$. Then, the upper bound in Proposition 23 implies that $d(B;\delta) \leq d$ if d satisfies $(d+\lambda)^2\exp\big(\Psi_{d,\lambda} + \lambda\big) < \delta$. Since $G(z_*) = 0$ and $G'(z_*) < 0$, there exists a constant $C > 0$ such that

$$(d+\lambda)^2\exp\big(\Psi_{d,\lambda} + \lambda\big) \leq (d+\lambda)^2 e^{-C(d - z_*\lambda)}.$$

Since $\lambda = \frac{B}{2} = \omega\big(\log(\delta^{-1})\big)$ implies that $(d+\lambda)^2\exp\big(\Psi_{d,\lambda} + \lambda\big) < \delta$ for $d = \big(z_* + o(1)\big)\lambda = \big(z_* + o(1)\big)\frac{B}{2}$. Hence, in this case $d_{B;\delta}(e^x) \leq \big(z_* + o(1)\big)\frac{B}{2}$, which by (31) implies that $d_{B;\delta}(e^x) = \big(z_* + o(1)\big)\frac{B}{2}$.

Now assume that $B = \big(2r + o(1)\big)\log(\delta^{-1})$ for some fixed $r > 0$, so that $\lambda = \big(r + o(1)\big)\log(\delta^{-1})$. Suppose that $d < \mu'\lambda$ for some $\mu' < \mu(r)$. Then, $G(\mu') + 1 > -r^{-1}$ and so we have again using (30) (and the fact that G is decreasing) that there would exist a constant $c_3 > 0$ such that

$$\begin{aligned}(d+\lambda)^{-3/2}\exp\big(\Psi_{d,\lambda} + \lambda\big) &= (d+\lambda)^{-3/2}\exp\Big(\lambda G\Big(\frac{d}{\lambda} + \lambda\Big)\Big)\\ &\geq (d+\lambda)^{-3/2}\exp\Big(\lambda\big(G(\mu') + 1\big)\Big)\\ &\geq (d+\lambda)^{-3/2}\exp\big((c_3 - r^{-1}\lambda\big) = \delta(d+\lambda)^{-1}e^{(c_3 - o(1))\lambda} > \delta.\end{aligned}$$

Thus, the lower bound in Proposition 23 implies that $d_{B;\delta}(e^x) \geq \big(\mu + o(1)\big)\lambda = \big(\mu r + o(1)\big)\log(\delta^{-1})$.

Similarly, if $d > \mu''\lambda$ for some $\mu'' > \mu(r)$, then there exists some constant $c_4 > 0$ such that

$$\begin{aligned}(d+\lambda)^2\exp\big(\Psi_{d,\lambda} + \lambda\big) &\geq (d+\lambda)^2\exp\Big(\lambda\big(G(\mu'') + 1\big)\Big)\\ &\geq (d+\lambda)^{-1}\exp\big(-\lambda(r^{-1} + c_4)\big) = \delta(d+\lambda)^2 e^{(o(1) - c_4)\lambda} < \delta,\end{aligned}$$

which implies by the upper bound in Proposition 23 that $d_{B;\delta}(e^x) \leq \big(\mu r + o(1)\big)\log(\delta^{-1})$. Hence, $d_{B;\delta}(e^x) = \big(\mu r + o(1)\big)\log(\delta^{-1})$.

Now let us consider the final case $B = o\big(\log(\delta^{-1})\big)$. Suppose that

$$d = \frac{\gamma \log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))},$$

for some $\gamma \in (0,\infty)$ (bounded above and below). Then, $\frac{d}{\lambda} = \omega(1)$, and so the second part of Lemma 15 implies that

$$\Psi_{d,\lambda} = -d\log\Big(\frac{d}{\lambda}\Big)\big(1+o(1)\big).$$

Hence, if $\gamma < 1$, then

$$\begin{aligned}
&(d+\lambda)^{-3/2}\exp\big(\Psi_{d,\lambda}+\lambda\big)\\
&\qquad = (d+\lambda)^{-3/2}\exp\bigg(-d\log\Big(\frac{d}{\lambda}\Big)\big(1+o(1)\big)\bigg)\\
&\qquad = (d+\lambda)^{-3/2}\exp\bigg(-\frac{\gamma\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))}\log\Big(\frac{2\gamma B^{-1}\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))}\Big)\big(1+o(1)\big)\bigg)\\
&\qquad = (d+\lambda)^{-3/2}\exp\Big(\big(\gamma+o(1)\big)\log(\delta^{-1})\Big) \geq \delta^{\gamma+o(1)}\big(\log(\delta^{-1})\big)^{-3/2} > \delta.
\end{aligned}$$

Hence, the lower bound in (23) implies that

$$d_{B;\delta}(e^x) \geq \frac{\big(1+o(1)\big)\log(\delta^{-1})}{\log(B^{-1}\log(\delta^{-1}))},$$

The proof of the matching upper bound is entirely analogous and is therefore omitted. ◀

Next we establish Theorem 2. To that end, we begin with the following lemma that addresses the last part of that theorem, when $B \geq \delta^{-\Omega(1)}$.

▶ **Lemma 24.** *For every real $\delta \in (0,1/4)$ and $B \geq 1$ with $B > \omega(\log(\delta^{-1}))$ we have $d_{B;\delta}(e^{-x}) \geq (1/2+o(1))\sqrt{B\log((2\delta)^{-1})}$.*

Proof. Let $p(x)$ be any polynomial satisfying $\sup_{x\in[0,B]}\big|p(x)-e^{-x}\big| < \delta$, and set $d = \deg p$. Let $x_0 = 0$, $x_1 = \log(\delta^{-1})$, and $x_2 = B$. It follows that $p(x_0) \geq 1-\delta$, and that $p(x) \in [-\delta, 2\delta]$ for all $x \in [x_1, x_2]$. Let $a : \mathbb{R} \to \mathbb{R}$ be the linear function satisfying $a(1) = x_1$ and $a(-1) = x_2$, and let

$$x_0' := a^{-1}(x_0) = 1 + \frac{2(x_1-x_0)}{x_2-x_1} = 1 + \frac{2\log(\delta^{-1})}{B-\log(\delta^{-1})}.$$

Finally, define the polynomial $q(x) = \frac{p(a(x))}{2\delta}$, which also has degree d. It follows that $q(x) \in [-1,1]$ for all $x \in [-1,1]$, and that $q(x_0') \geq \frac{1-\delta}{2\delta}$.

Applying Fact 16 to q, we see that $|Q_d(x_0')| \geq 2^{1-d}q(x_0') \geq \frac{1-\delta}{2^d\delta}$. Furthermore, by Fact 17 we have that $|Q_d(x_0')| \leq 2^{-d}e^{(\sqrt{2}+o(1))d\sqrt{x_0'-1}}$. Combining the two bounds yields:

$$\frac{1-\delta}{2^d\delta} \leq 2^{-d}e^{(\sqrt{2}+o(1))d\sqrt{x_0'-1}} = 2^{-d}e^{(2+o(1))d\sqrt{\log(\delta^{-1})/(B-\log(\delta^{-1}))}}.$$

Taking logs of both sides and rearranging gives the desired result. ◀

Now we can establish Theorem 2.

Proof of Theorem 2. The proofs of the estimates on $d_{B;\delta}(e^{-x})$ in the first and second cases, when either $B = o\big(\log(\delta^{-1})\big)$ and $B = \Theta\Big(\log(\delta^{-1})\Big)$ are entirely analogous to those for $d_{B;\delta}(e^{x})$ shown in Theorem 3 above. Therefore, they are omitted.

So, let us assume that $B = \omega\big(\log(\delta^{-1})\big)$, and let $d > 0$ be some integer with

$$d = \sqrt{\gamma B \log(\delta^{-1})} = \sqrt{2\gamma\lambda \log(\delta^{-1})},$$

where γ is uniformly bounded above and below. Observe since $B = \omega\big(\log(\delta^{-1})\big)$ that $d = o(B)$, and so Lemma 15 implies that

$$\Psi_{d,\lambda} - \lambda = -\frac{d^2}{2\lambda}\big(1 + o(1)\big).$$

Now, let us first approximate $d_{B;\delta}(e^{-x})$ by $(1 + o(1))\sqrt{B\log(\delta^{-1})}$ in the regime where $B \leq \delta^{-o(1)}$. To lower bound it, suppose that $\gamma < 1$ (and is uniformly bounded away from 1). Then,

$$\begin{aligned}(\lambda + d)^{-3/2}\exp(\Psi_{d,\lambda} - \lambda) &\geq (\lambda + d)^{-3/2}\exp\Big(-\frac{d^2}{2\lambda}\big(1 + o(1)\big)\Big)\\ &\geq (\lambda + d)^{-3/2}\exp\Big(-\gamma\log(\delta^{-1})\Big) \geq \delta^{\gamma + o(1)},\end{aligned}$$

where in the last bound we used the fact that $d = o(\lambda)$ and that $\lambda = \frac{B}{2} \leq \delta^{-o(1)}$. Hence, by the lower bound in the second part of Proposition 23, we find that $d_{B;\delta}(e^{-x}) \geq \big(1 + o(1)\big)\sqrt{B\log(\delta^{-1})}$.

To upper bound $d_{B;\delta}(e^{-x})$ for $B \leq \delta^{-o(1)}$, assume that $\gamma > 1$ (and is uniformly bounded away from 1). Then,

$$\begin{aligned}\exp(\Psi_{d,\lambda} - \lambda) &\leq (\lambda + d)^2\exp\Big(-\frac{d^2}{2\lambda}\big(1 + o(1)\big)\Big)\\ &\leq (\lambda + d)^2\exp\Big(-\log(\delta^{-1})\big(\gamma - o(1)\big)\Big) \leq (\lambda + d)^2\delta^{\gamma - o(1)} < \delta,\end{aligned}$$

and so again by Proposition 23 we deduce that $d_{B;\delta}(e^{-x}) \leq \big(1 + o(1)\big)\sqrt{B\log(\delta^{-1})}$. Together, these upper and lower bounds imply that $d_{B;\delta}(e^{-x}) = \big(1 + o(1)\big)\sqrt{B\log(\delta^{-1})}$ when $B = \omega\big(\log(\delta^{-1})\big)$ and $B \leq \delta^{-o(1)}$.

It remains to show that $d_{B;\delta}(e^{-x}) = \Theta\big(\sqrt{B\log(\delta^{-1})}\big)$ when $B \geq \delta^{-\Omega(1)}$. The lower bound (with implicit constant $\frac{1}{2} + o(1)$) was shown by Lemma 24, so we must verify the upper bound. To that end, we assume $\gamma > 1$ (uniformly bounded away from 1) and observe that $B \geq 2d$ for $B \geq \delta^{-\Omega(1)}$. Then, the upper bound from (21) applies; since the first part of Lemma 15

$$\exp(\Psi_{d,\lambda} - \lambda) \leq \exp\Big(-\frac{d^2}{2\lambda}\big(1 + o(1)\big)\Big) \leq \exp\Big(-\log(\delta^{-1})\big(\gamma - o(1)\big)\Big) \leq \delta^{\gamma - o(1)} \leq \delta,$$

we deduce that $d_{B;\delta}(e^{-x}) \leq \big(1 + o(1)\big)\sqrt{B\log(\delta^{-1})}$, which establishes the theorem. ◀

References

1 Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM)*, 51(4):595–605, 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*, pages 218–230. SIAM, 2014.

3 Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52(1), 2005.

4 Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 141–160. SIAM, 2020.

5 Josh Alman, Timothy M Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 467–476. IEEE, 2016.

6 Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–552, 2020. doi:10.1109/FOCS46700.2020.00057.

7 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 136–150. IEEE, 2015.

8 Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(1):37–46, 2005.

9 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.

10 Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov–bernstein inequalities. *Information and Computation*, 243:2–25, 2015.

11 Mark Bun and Justin Thaler. Guest column: Approximate degree in classical and quantum computing. *ACM SIGACT News*, 51(4):48–72, 2021.

12 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. SIAM, 2016.

13 Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

14 Welin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. *arXiv preprint*, 2015. arXiv:1512.04906.

15 Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

16 Marlis Hochbruck and Christian Lubich. On krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):1911–1925, 1997.

17 Armand Joulin, Moustapha Cissé, David Grangier, Hervé Jégou, et al. Efficient softmax approximation for gpus. In *International Conference on Machine Learning*, pages 1302–1310. PMLR, 2017.

18 Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

19 Jasper CH Lee, Jerry Li, Christopher Musco, Jeff M Phillips, and Wai Ming Tai. Finding the mode of a kernel density estimate. *arXiv preprint*, 2019. arXiv:1912.07673.

20 J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. Chapman & Hall/CRC, Boca Raton, FL, 2003.

21 Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the lanczos method for matrix function approximation. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1605–1624. SIAM, 2018.

22 Peter Nilsson, Ateeq Ur Rahman Shaik, Rakesh Gangarajaiah, and Erik Hertz. Hardware implementation of the exponential function using taylor series. In *2014 NORCHIP*, pages 1–4. IEEE, 2014.

23 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994.

24 Lorenzo Orecchia, Sushant Sachdeva, and Nisheeth K Vishnoi. Approximating the exponential, the lanczos method and an o (m)-time spectral algorithm for balanced separator. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1141–1160, 2012.

25 Jeff M Phillips. ε-samples for kernels. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1622–1632. SIAM, 2013.

26 Michael JD Powell. On the maximum errors of polynomial approximations defined by interpolation and by least squares criteria. *The Computer Journal*, 9(4):404–407, 1967.

27 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268, 2018.

28 Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9(2):125–210, 2014.

29 Yaoyun Shi. Approximating linear restrictions of boolean functions. In *Manuscript*. Citeseer, 2002.

30 Paraskevas Syminelakis. *Fast Kernel Evaluation in High Dimensions: Importance Sampling and near Neighbor Search*. Stanford University, 2019.

31 A. F. Timan. *Theory of approximation of functions of a real variable*. Dover Publications, Inc., New York, 1994. Translated from the Russian by J. Berry, Translation edited and with a preface by J. Cossar, Reprint of the 1963 English translation.

32 Lloyd N. Trefethen. *Approximation theory and approximation practice*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013.

33 Changjiang Yang, Ramani Duraiswami, Nail A Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density estimation. In *Computer Vision, IEEE International Conference on*, volume 2, pages 464–464. IEEE Computer Society, 2003.

Extremely Efficient Constructions of Hash Functions, with Applications to Hardness Magnification and PRFs

Lijie Chen
CSAIL, MIT, Cambridge, MA, USA

Jiatu Li
IIIS, Tsinghua University, Beijing, China

Tianqi Yang
IIIS, Tsinghua University, Beijing, China

Abstract

In a recent work, Fan, Li, and Yang (STOC 2022) constructed a family of almost-universal hash functions such that each function in the family is computable by $(2n + o(n))$-gate circuits of fan-in 2 over the B_2 basis. Applying this family, they established the existence of pseudorandom functions computable by circuits of the same complexity, under the standard assumption that OWFs exist. However, a major disadvantage of the hash family construction by Fan, Li, and Yang (STOC 2022) is that it requires a seed length of $\mathsf{poly}(n)$, which limits its potential applications.

We address this issue by giving an improved construction of almost-universal hash functions with seed length $\mathsf{polylog}(n)$, such that each function in the family is computable with POLYLOGTIME-uniform $(2n + o(n))$-gate circuits. Our new construction has the following applications in both complexity theory and cryptography.

- (**Hardness magnification**). Let $\alpha : \mathbb{N} \to \mathbb{N}$ be any function such that $\alpha(n) \leq \log n / \log\log n$. We show that if there is an $n^{\alpha(n)}$-sparse NP language that does not have probabilistic circuits of $2n + O(n/\log\log n)$ gates, then we have (1) $\mathsf{NTIME}[2^n] \not\subseteq \mathsf{SIZE}\left[2^{n^{1/5}}\right]$ and (2) $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$ for every constant k. Complementing this magnification phenomenon, we present an $O(n)$-sparse language in P which requires probabilistic circuits of size at least $2n - 2$. This is the first result in hardness magnification showing that even a *sub-linear additive* improvement on known circuit size lower bounds would imply $\mathsf{NEXP} \not\subset \mathsf{P}_{/\mathsf{poly}}$.
  Following Chen, Jin, and Williams (STOC 2020), we also establish a sharp threshold for **explicit obstructions**: we give an explict obstruction against $(2n - 2)$-size circuits, and prove that a sub-linear additive improvement on the circuit size would imply (1) $\mathsf{DTIME}[2^n] \not\subseteq \mathsf{SIZE}\left[2^{n^{1/5}}\right]$ and (2) $\mathsf{P} \not\subseteq \mathsf{SIZE}[n^k]$ for every constant k.
- (**Extremely efficient construction of pseudorandom functions**). Assuming that one of integer factoring, decisional Diffie-Hellman, or ring learning-with-errors is sub-exponentially hard, we show the existence of pseudorandom functions computable by POLYLOGTIME-uniform $\mathsf{AC}^0[2]$ circuits with $2n + o(n)$ wires, with key length $\mathsf{polylog}(n)$. We also show that PRFs computable by POLYLOGTIME-uniform B_2 circuits of $2n + o(n)$ gates follows from the existence of sub-exponentially secure one-way functions.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Almost universal hash functions, hardness magnification, pseudorandom functions

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.23

Funding Lijie Chen is supported by NSF CCF-2127597 and an IBM Fellowship.

Acknowledgements We are grateful for Ryan Williams for insightful discussions during this project and many helpful comments on a draft of this paper. We would also like to thank Ce Jin for discussions during the early stage of this research project and anonymous reviewers for their comments.

1 Introduction

Background and motivation. *Universal hash families*[1], introduced by Carter and Wegman [6], are among the most useful and well-studied objects in computer science, with applications to both real-world systems and the theory of computation (e.g., algorithm design, complexity theory, and cryptography). Its wide applications motivate the problem of constructing universal hash functions with the smallest computational overhead. After decades of research, optimal constructions (up to a constant factor) have been obtained in both the RAM model [36] and the Boolean circuit model [23].

In particular, the investigation of hash functions with small circuit complexity was motivated by the program of constructing low-complexity cryptographic primitives. Ishai, Kushilevitz, Ostrovsky, and Sahai [23] showed that universal hash functions can be constructed with linear-size circuits, which, combined with Levin's trick of domain extension (see, e.g., [5]), led to linear-size constructions of pseudorandom functions (PRFs) and many other cryptographic primitives, under the standard assumption that one-way functions (OWFs) exist.

In a recent work, Fan, Li, and Yang [13] proved that *almost* universal hash functions[2] are sufficient for constructing extremely efficient PRFs. They then constructed almost universal hash functions with $2n + o(n)$ circuit complexity in both general and $\mathsf{CC}^0[2]$ circuits[3], and proved the optimality of the constant factor 2 (i.e., there is no $2n - O(1)$-size almost-universal hash construction). As a consequence of their almost-universal hash construction, they presented $2n + o(n)$-size constructions of PRFs assuming OWFs exist. However, a *major disadvantage* of the hash constructions in [13] is that they require a seed of length $O(n \log n)$ to sample a circuit from the hash family, which greatly limits their applications.

Overview of our results. The main technical contribution of this paper is to improve the hash constructions in [13] for both general and $\mathsf{CC}^0[2]$ circuits by reducing the seed length from $O(n \log n)$ to $\mathsf{polylog}(n)$. Moreover, we observe that the improvement of the seed length makes the hash family explicit in a strong sense: the local topology of the circuit computing the hash function can be obtained with a uniform algorithm (e.g., given a gate index, compute its gate type and which gates' outputs are fed into this gate) in poly-logarithmic time given a seed of poly-logarithmic length. We call such hash family POLYLOGTIME-uniform; see Theorem 1.1 for the details.

Despite being weaker than universal hash functions, our new randomness-efficient low-complexity hash constructions allow us to obtain two important consequences in complexity theory and cryptography.

- (**Hardness magnification**) Following the kernelization method developed by Chen, Jin, and Williams [9, 10], we present extremely sharp bootstrapping results for hardness magnification and explicit obstruction. In particular, we show that a $2.01n$ lower bound

[1] Recall that a universal hash family has the property that for a function drawn from the family, the hash values of any distinct pair of inputs collide with probability 2^{-m}, where m is the bit length of the hash value.

[2] In an almost universal hash family, the hash collision probability is a negligible function (e.g., $1/2^{\log^2 n}$) instead of $1/2^m$ for m-bit hash values. It is weaker than a universal hash family, but is still useful in many applications.

[3] For general circuits we mean B_2 circuits, in which each gate is of fan-in 2 and can compute an arbitrary Boolean function in $B_2 \triangleq \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2$. $\mathsf{CC}^0[2]$ is a sub-class of $\mathsf{AC}^0[2]$ representing the constant-depth circuits consisting of only unbounded fan-in XOR gates. The complexity of $\mathsf{CC}^0[2]$ circuits are measured in the number of wires instead of gates.

for any sparse NP language against probabilistic B_2 circuits would imply a major breakthrough: NP does not have n^k-size circuits for every fixed constant k. We also obtain stronger consequences with the same lower bound for MCSP using the explicitness of our hash family.[4]

- (**Extremely efficient PRFs**) Following [23, 13], our new hash constructions imply extremely efficient PRFs as well. Under the sub-exponential decisional Diffie-Hellman assumption (see, e.g., [5]), we can construct a PRF computable by $\mathsf{AC}^0[2]$ circuits of wire complexity $2n + o(n)$ with $\mathsf{polylog}(n)$ seed length (instead of $\mathsf{poly}(n)$ in [13]).[5] Furthermore, the $\mathsf{AC}^0[2]$-computable PRF is POLYLOGTIME-uniform, which implies a parallel algorithm to print the $\mathsf{AC}^0[2]$ circuit. Similar results also hold for general circuits assuming sub-exponentially secure one-way function exists.

Both of the above consequences crucially rely on our improvement upon [13] on seed length and explicitness; we believe that our new hash construction will find further applications in other areas of computer science.

1.1 Randomness-efficient and strongly-explicit almost universal hash functions

Before stating our main theorem, we first define the notion of an almost universal hash family and its properties. A family of hash functions is defined as $\mathcal{H} = \{\mathcal{H}_n\}_{n\geq 1}$, where each $\mathcal{H}_n$ is a distribution of functions from $\{0,1\}^n$ to $\{0,1\}^m$ for output length $m = m(n)$. For a function $\varepsilon\colon \mathbb{N} \to \mathbb{R}$, we say $\mathcal{H}$ is *ε-almost universal* if for all sufficiently large $n \in \mathbb{N}$, and for every two distinct inputs $x, y \in \{0,1\}^n$, it holds that

$$\Pr_{h\leftarrow\mathcal{H}_n}[h(x) = h(y)] \leq \varepsilon(n).$$

We say $\mathcal{H}$ is *linear* if every function in the family is linear over the field $\mathbb{F}_2$.

We are now ready to present our construction of almost universal hash functions. The theorem is formally proved as Theorem 3.9.

▶ **Theorem 1.1** (Unconditional construction of extremely efficient almost universal hash)**.** *Let $\ell \triangleq \log^2 n/\log\log n$. There is a family of linear $\Theta(\ell)$-output $\exp(-\Omega(\ell))$-almost universal hash functions $\mathcal{H} = \{\mathcal{H}_n\}_{n\geq 1}$ satisfying the following properties.*

(Low complexity) *Each function $h \in \mathsf{supp}(\mathcal{H}_n)$ is computable by a $\mathsf{CC}^0[2]$ circuit with at most $2n + o(n)$ wires.*

(Randomness efficient) *$\mathcal{H}_n$ is samplable with seed length $r = O(\ell \log^2 n)$ in polynomial time. More formally, there exists a polynomial-time algorithm $\mathcal{G}$ that takes 1^n and a seed v of length r and outputs (the description of) a function $h_v \in \mathsf{supp}(\mathcal{H}_n)$, such that $\mathcal{H}_n$ and $\mathcal{G}(1^n, \mathcal{U}_r)$ are identical distributions (if we identify a function with its description), where $\mathcal{U}_r$ is the uniform distribution over $\{0,1\}^r$. The function h_v is said to be corresponding to the seed v.*

---

[4] The Minimum Circuit Size Problem with size parameter $s(n)$ ($\mathsf{MCSP}[s(n)]$) is defined as follows: given a string x of length $N = 2^n$, determine whether there exists an n-input $s(n)$ size circuit with x as its truth table.

[5] The existence of PRFs computable by $\mathsf{AC}^0[2]$ circuits might sound surprising, given that there exists a natural property against sub-exponential-size $\mathsf{AC}^0[2]$ and natural properties can be used to break PRFs [38]. However, a closer look at the arguments in [38] shows that the natural property against $\mathsf{AC}^0[2]$ only gives a large quasi-polynomial-time (i.e., $2^{\log^k n}$ for some large constant k) adversary breaking PRFs computable by $\mathsf{AC}^0[2]$. This does not contradict our PRF construction in $\mathsf{AC}^0[2]$, as our construction is only secure against $\exp(\log^2 n/\log\log n)$-time adversaries; see Theorem 1.8 for details.

*(**POLYLOGTIME-uniform**) There exists a polynomial-time algorithm $\mathcal{A}$ which takes a tuple (n, v, i, j) as the input[6], and outputs the source of the j^{th} in-wire of the i^{th} gate in the $\mathsf{CC}^0[2]$-circuit (with $2n+o(n)$ wires) computing the function $h_v \in \mathsf{supp}(\mathcal{H}_n)$ corresponding to the seed v.*

*(**Strongly explicit**) There exists a polynomial-time algorithm $\mathcal{B}(n, v, i, j)$ satisfying the following: Let v be a seed, $h_v \in \mathsf{supp}(\mathcal{H}_n)$ be the $\mathbb{F}_2$-linear function corresponding to v, and $\boldsymbol{M}$ be the $m \times n$ $\mathbb{F}_2$-matrix such that $h_v(x)_i = \sum_{j=1}^{n} \boldsymbol{M}_{i,j} x_j$ for any $i \in [m]$. It holds that $\mathcal{B}(n, v, i, j)$ outputs $\boldsymbol{M}_{i,j}$.*

The POLYLOGTIME-uniformity and strongly explicitness characterize the explicitness of our hash construction in two different senses. The former one captures the explicitness of the $2n + o(n)$-size circuit computing the hash function, while the latter one focuses on the linear transformation corresponding to the hash function.

Here we briefly discuss why the notion of POLYLOGTIME-uniformity is important for our (and other potential) applications. POLYLOGTIME-uniformity is the most natural definition of parallel uniformity. It says that our hash function is parallel-efficient not only in the evaluation phase (since it is in sparse $\mathsf{CC}^0[2]$) but also in the pre-processing phase, i.e., to construct the circuit according to the seed. With a POLYLOGTIME-uniform hash family, we are able to construct POLYLOGTIME-uniform low-complexity PRFs under standard assumptions, which is beneficial to the overall efficiency of other cryptographic systems involving PRFs (see, e.g., [5]). The "strongly explicitness" of our construction will be useful in proving hardness magnification theorems for MCSP, see Section 4.2.

Table 1 summarizes the differences between our new construction and known low-complexity constructions in [23, 9, 13].

Table 1 Comparison of known constructions of almost universal hash functions. Note that the collision probability of these hash function are $\exp(-\Omega(m))$ for output length m. The construction in [23] is in addition a pairwise-independent hash function.

	Output size m	Seed Length	Circuit Class	Uniformity/Explicitness
Folklore[7]	$m \leq n$	$\log n + O(m)$	linear-size NC^1	P-uniform[8]
[23]	$m \leq n$	$O(n)$	linear-size NC^1	P-uniform
[13]	$m = O\left(\frac{\log^2 n}{\log\log n}\right)$	$O(n \log n)$	$2n + o(n)$ size $\mathsf{CC}^0[2]$	P-uniform
Ours	$m = O\left(\frac{\log^2 n}{\log\log n}\right)$	$\mathsf{polylog}(n)$	$2n + o(n)$ size $\mathsf{CC}^0[2]$	POLYLOGTIME-uniform and strongly explicit

1.2 Implications on sharp bootstrapping results

Prior works in Hardness Magnification. Proving strong circuit lower bounds against explicit functions is one of the most significant challenges in complexity theory. However, despite decades of efforts, it remains unknown whether NP has linear-size circuits. The

[6] The input lengths to $\mathcal{A}$ and the algorithm $\mathcal{B}$ in the next bullet are both $O(\log n) + r = \mathsf{polylog}(n)$, so their running times are actually polylogarithmic in n.

[7] This is done by sampling over the output bits of Spielman's ECC [40] with an random sampling based on an expander walk (see Lemma 3.2 of [9]).

[8] A hash construction is P-uniform if there is a polynomial time algorithm that prints the circuit computing a hash function given its seed v.

strongest explicit circuit lower bounds against general fan-in 2 circuits (known as B_2 circuits) is $3.1n - o(n)$ [28], following [12, 14]. A $5n - o(n)$ lower bound is known against U_2 circuits [26, 24][9].

A recently discovered phenomena in computational complexity called *hardness magnification* (see, e.g., [35, 34, 30, 9, 8]) provides new insights in bridging the gap between what we can prove and what we want to prove regarding circuit lower bounds. It says that a relatively weak circuit lower bound (e.g., $n \cdot \mathsf{polylog}(n)$ size against B_2 circuits) for some special problems would imply major breakthroughs like $\mathsf{NP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$.

Most interestingly, the required lower bounds for hardness magnification are only slightly stronger than provable lower bounds. For instance, Chen, Jin, and Williams [10] showed that an $n^{2+\varepsilon}$ lower bound against probabilistic De Morgan formulas for MCSP would imply $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$ for all k, while an $n^{2-o(1)}$ lower bound for the same problem can be proved using a variant of the random restriction method in [18, 41, 22, 21, 34]. For more related works on hardness magnification, see [8] for a comprehensive summary.

Still, there are asymptotically significant gaps between what are provable and what suffice to bootstrap in all known results. For example, [30] says that proving a slightly super-linear circuit lower bound for a sparse version of MCSP would already imply $\mathsf{NP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$. However, the best unconditional lower bound for any language in NP is only $3.1n - o(n)$ [28], which is still far from the superlinear threshold for bootstrapping. This is formally discussed in [17], which shows that the current technique for proving unconditional circuit lower bounds is not capable of breaking the linear barrier. Hence, even proving a super-linear lower bound for MCSP seems out of reach.

Overview of results in this section. By slightly generalizing the computation model to *probabilistic circuits*, we observe that even a *sub-linear* improvement over the known lower bounds would be enough to imply breakthroughs in complexity theory. For example, we show that improving a known $2n - O(1)$ probabilistic circuit lower bound for particular sparse languages to $2n + O(n/\log\log n)$ would imply $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$ for every constant k. Regarding circuit lower bounds for deterministic time classes, we also obtain a similar threshold phenomenon by studying *explicit obstructions*; see Section 1.2.2 for details.

1.2.1 Sharp magnification threshold for probabilistic circuits

Notation. Formally, a *probabilistic circuit deciding a language* L is a distribution $\mathcal{D}$ over circuits that outputs the wrong answer with probability at most $1/\mathsf{poly}(n)$ on any input x, i.e., for all $x \in \{0,1\}^n$ it holds that $\Pr_{D \leftarrow \mathcal{D}}[D(x) \neq L(x)] < 1/\mathsf{poly}(n)$.[10] We say a language L is $s(n)$-sparse if for all sufficiently large n, $L \cap \{0,1\}^n$ has size at most $s(n)$.

We first present the most standard form of our result, which gives fixed polynomial lower bounds for NP. This theorem is a special case of our general theorem of hardness magnification, which is stated and proved as Theorem 4.1.

[9] A U_2 circuit consists of fan-in 2 gates computing all binary functions except for XOR and its complement.

[10] Note that in contrast to robust classes like BPP, our result is actually sensitive to the error probability in the definition of probabilistic circuits, since the complexity overhead of error reduction is costly in the linear-size setting. For simplicity, we stick to the definition with error $1/\mathsf{poly}(n)$.

▶ **Theorem 1.2** (Hardness magnification, high-end). *Let* $\alpha(n) \triangleq \log n / \log\log n$. *If there exists an* $n^{\alpha(n)}$*-sparse language in* NP *that does not have probabilistic circuits*[11] *of size* $2n + O(n/\log\log n)$, *then we have (1)* $\mathsf{NTIME}[2^n] \not\subseteq \mathsf{SIZE}\left[2^{n^{1/5}}\right]$ *and (2)* $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$ *for every constant* k.[12]

Let us compare this theorem with Theorem 1.2 of [9]. They proved that $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$ follows from the non-existence of $O(n)$-time randomized algorithms with n^ε bits of advice and $O(\log n)$ bits of randomness for $2^{n^{o(1)}}$-sparse NP languages. In our theorem, we trade the sparsity for the exact constants in the running time of the randomized algorithm, showing that even a $2n + o(n)$ probabilistic circuit lower bound is enough.

Theorem 1.2 allows us to bootstrap a non-trivial lower bound for sparse languages in NP to strong sub-exponential lower bounds. If we only want a super-polynomial circuit lower bound for NEXP, then the assumption can be considerably weakened. The following theorem is another extreme case of Theorem 4.1.

▶ **Theorem 1.3** (Hardness magnification, low-end). *Let* $\alpha(n) \triangleq \log n / \log\log n$. *If there is an* $n^{\alpha(n)}$*-sparse language in* $\mathsf{NTIME}\left[2^{n^{o(1)}}\right]$ *that is not computable by probabilistic circuits of size* $2n + O(n/\log\log n)$, *then* $\mathsf{NEXP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$.

We show that a nearly-matching lower bound is provable, from the proof of the $2n - O(1)$ lower bound for pseudorandom functions in [13]. This complements the bootstrapping above, and shows that there is only a gap of an *additive sub-linear term* towards breakthrough circuit lower bounds. The following theorem is formally proved as Corollary 4.11.

▶ **Theorem 1.4.** *There is an explicit* $O(n)$*-sparse language* L *computable in* P, *such that every probabilistic circuit deciding* L *must have size at least* $2n - 2$.

Comparison with known lower bounds. We remark that a better-than-$2n$ circuit lower bound against probabilistic B_2 circuits can indeed be proved for certain non-sparse languages in P from known average-case lower bounds. This is because given a probabilistic circuit with small error probability, we can guarantee the existence of a particular circuit that approximates the language well by an averaging argument. In particular, the following lower bound of Chen and Kabanets [11] implies that no probabilistic circuits of size $2.49n$ can decide an explicit language in P.

▶ **Theorem 1.5** (Theorem 4.8, [11]). *There is a language* L *in* P *such that for any* B_2 *circuit* C *of size* $2.49n$,

$$\Pr_{x \leftarrow \{0,1\}^n}[C(x) = L(x)] \leq \frac{1}{2} + \frac{1}{2^{\Omega(n)}}.$$

Note that every sparse language must be *easy* on average as a trivial circuit outputting 0 can approximate it well. As a result, it is impossible to obtain a breakthrough directly from our hardness magnification result and an average-case lower bound. Nevertheless, linear-size

[11] Although it is not explicitly stated for simplicity, we mention that the lower bound required to obtain a breakthrough in Theorem 1.2, Theorem 1.3, and Theorem 1.6 can be weakened to the one-sided error case. That is, we only need to show that for any sparse language $L \in \mathsf{NP}$, no probabilistic circuit of small size could output 1 with certainty for any $x \in L$ and output 0 with high probability otherwise.

[12] We remark that we can indeed obtain a conclusion that is stronger than both (1) and (2), albeit it is more technical: for all $c > 0$ there exists some $c' > c$ such that $\mathsf{NTIME}\left[2^{c'n^{1/5}}\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$.

lower bounds against general circuits of size greater than $2n$ is a widely-studied problem for decades, so this magnification phenomenon may give us more insights on the setting of small linear size circuits. Following [8], our results achieve a "hardness magnification frontier".

Better magnification results for meta-complexity problems. We also show that weak circuit lower bounds for specific meta-complexity problems such as MCSP imply consequences stronger than that of Theorem 1.2. Formally, we define $\mathsf{MCSP}[s(n)]$ as the language taking a truth table of length $N = 2^n$ as input, and decides whether it admits a circuit of size at most $s(n)$. The following theorem is formally proved as Theorem 4.3.

▶ **Theorem 1.6.** *Let $N = 2^n$ be the truth table length of n-input Boolean functions and $n \leq s(n) \leq n^2/\log n$ be any size function. If $\mathsf{MCSP}[s(n)]$ does not have probabilistic circuits of size $2N + O(N/\log\log N)$, then there exists some $c > 0$ such that $\oplus\mathsf{P} \not\subseteq \mathsf{SIZE}\left[2^{N^c}\right]$.*

Unfortunately, our lower bound technique in proving Theorem 1.4 cannot be used to derive a lower bound for MCSP. We leave proving an unconditional $2n - o(n)$ circuit size lower bound for MCSP, even against deterministic circuits, as an interesting open problem.

1.2.2 Sharp magnification thresholds for explicit obstruction

A drawback of Theorem 1.2 and Theorem 1.3 is that the conclusion only gives circuit lower bounds for nondeterministic time classes such as $\mathsf{NTIME}[2^n]$. By studying a stronger notion called explicit obstruction, we are able to obtain a bootstrapping theorem with tight threshold, which gives circuit lower bounds for deterministic time classes such as $\mathsf{TIME}[2^n]$.

Explicit obstruction. An *explicit obstruction of size $S(n)$ against $\mathcal{C}$* is an algorithm $\mathcal{A}$ running in $\mathsf{poly}(n, S(n))$ time, such that on input 1^n, A outputs a set $E_n = \{(x_i, y_i)\}$ of size $S(n)$ such that $x_i \in \{0,1\}^n$ and $y_i \in \{0,1\}$ for every i, and $x_i \neq x_j$ for every $i \neq j$. The set has the property that, for all circuit $C \in \mathcal{C}$, there exists some $i \in [S(n)]$ such that $C(x_i) \neq y_i$. This can be viewed as an explicit proof of the hardness of C computing any n-bit function f that is consistent with E_n, since one can always efficiently find a counter-example from the explicit obstruction. Indeed, giving an explicit obstruction of polynomial size would directly imply $\mathsf{P} \not\subseteq \mathcal{C}$. This concept is first suggested by Mulmuley [32] in the context of geometric complexity theory, where he argued that explicit obstructions might be essential in proving arithmetic circuit lower bounds. For more discussions on explicit obstruction, see [10].

Our results. Our theorem below shows that in the case of general Boolean circuits, even presenting such an obstruction for very small linear-size circuits would imply breakthrough circuit lower bounds. It is proved formally as Corollary 4.9 and Theorem 4.5.

▶ **Theorem 1.7.** *Let $\alpha(n) = \log n/\log\log n$. The following holds.*

- *There is an explicit obstruction of $\mathsf{poly}(n)$-size against $(2n-2)$-size B_2 circuits.*
- *If for some $\beta(n) \geq \omega(n/\log\log n)$, there is an explicit obstruction of size $n^{\alpha(n)}$ against $2n + \beta(n)$-size B_2 circuits, then we have (1) $\mathsf{DTIME}[2^n] \not\subseteq \mathsf{SIZE}\left[2^{n^{1/5}}\right]$ and (2) $\mathsf{P} \not\subseteq \mathsf{SIZE}[n^k]$ for every k.*[13]

[13] Similar to Theorem 1.2, we can indeed obtain a stronger conclusion that for all $c > 0$ there exists a $c' > c$ such that $\mathsf{DTIME}\left[2^{c'n^{1/5}}\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$.

1.3 Strongly uniform pseudorandom functions

Utilizing the POLYLOGTIME-uniformity and short seed length of our hash constructions, we can apply them to the framework in [23, 13] to obtain extremely uniform low-complexity pseudorandom functions (PRFs) with short key length. Recall that a PRF is a family of distributions over functions that are indistinguishable from uniformly random functions by efficient adversaries. We show, from standard cryptography assumptions, that each member of a PRF can be constructed by circuits of size $2n + o(n)$; moreover, the construction itself can be POLYLOGTIME-uniform as well.

Formally, an m-output t-secure pseudorandom function is a family $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq 1}$ of distributions $\mathcal{F}_n$ on n-input m-output Boolean functions that fools every probabilistic $t(n)$-time adversary $\mathcal{A}$, i.e.,

$$\left| \Pr_{f\leftarrow\mathcal{F}_n}\left[\mathcal{A}^f(1^n) \text{ accepts}\right] - \Pr_{g:\{0,1\}^n\to\{0,1\}^m}\left[\mathcal{A}^g(1^n) \text{ accepts}\right] \right| < \frac{1}{t}.$$

We say that $\mathcal{C}$ has *key length* $s(n)$ if $\mathcal{F}_n$ is samplable (by a $\mathsf{poly}(n)$-time algorithm) with $s(n)$ random bits. Similar to the hash functions, we say that $\mathcal{F}$ is computable by POLYLOGTIME-*uniform* $\mathcal{C}$-circuits, if $s(n) \leq \mathsf{polylog}(n)$ and there exists an algorithm $\mathcal{A}(n, v, i, j)$ running in $\mathsf{polylog}(n)$ time (hence polynomial in its input length), that outputs the type of the i^{th} gate and the j^{th} descendant of the i^{th} gate in the $\mathcal{C}$-circuit computing the function $f_v \in \mathsf{supp}(\mathcal{F}_n)$ keyed by v.

Since our hash functions can be implemented in $\mathsf{CC}^0[2]$, we can obtain uniform PRFs in different circuit models based on different assumptions.

▶ **Theorem 1.8** (Uniform low-complexity PRFs). *There is a* $t = t(n) = \exp\left(\Omega\left(\frac{\log^2 n}{\log\log n}\right)\right)$ *and the following candidates of* t*-secure PRFs. Let* ε *be an arbitrarily small constant.*

B_2 circuits. *Assuming OWFs against* $\exp(n^\varepsilon)$*-time adversaries exist, there exists a family of* t*-secure PRFs with key length* $\mathsf{polylog}(n)$ *computable by* POLYLOGTIME*-uniform* B_2 *circuits of size* $2n + o(n)$ *and depth* $\mathsf{polylog}(n)$ *simultaneously.*

NC^1 circuits. *Assuming* $\exp(n^\varepsilon)$*-secure PRFs in* NC^1 *exist, there exists a family of* t*-secure PRFs with key length* $\mathsf{polylog}(n)$ *computable by* POLYLOGTIME*-uniform* B_2 *circuits of size* $2n + o(n)$ *and depth* $\log n + O(\log\log n)$ *simultaneously.*

$\mathsf{AC}^0[2]$ circuits. *Assuming* $\exp(n^\varepsilon)$*-secure PRFs in* NC^1 *exist, there exists a family of* t*-secure PRFs with key length* $\mathsf{polylog}(n)$ *computable by* POLYLOGTIME*-uniform* $\mathsf{AC}^0[2]$ *circuits of wire complexity* $2n + o(n)$.

Note that the assumption for our NC^1 and $\mathsf{AC}^0[2]$ constructions can be derived from standard assumptions such as sub-exponentially hard decisional Diffie-Hellman [33] or Ring Learning-with-Error [4], as well as other constructions like [31, 2]. We also mention that quasi-polynomial security is known to be optimal for $\mathsf{AC}^0[2]$ PRFs by [38] (see, e.g., [5]).

Our PRF constructions are the first to achieve extremely low-complexity (efficient evaluation), short key length, and POLYLOGTIME-uniformity (parallel pre-processing) simultaneously from standard assumptions. Prior to our result, only PRF candidates with $O(n \log n)$ key length computable by P-uniform circuits of the same sizes were known [13] in these three circuit classes, improved on the linear-size NC^1 PRFs by Ishai, Kushilevitz, Ostrovsky, and Sahai [23] and the $\mathsf{AC}^0[2]$ PRF due to Viola [43] with $n \cdot \mathsf{polylog}(n)$ wire complexity and key length. Our improvement could also improve the efficiency of other cryptographic primitives involving PRFs such as message authentication code (MAC) and CCA-secure encryption (see, e.g., [23, 5]). See Section 1.3 [13] for more discussions on low-complexity PRFs.

▶ Remark 1.9. As pointed out by an anonymous reviewer, assuming a PRF against $\exp(n^{\varepsilon})$-time adversaries (as we did in Theorem 1.8), we can reduce the key length to $\mathsf{polylog}(n)$ generically by directly applying such a PRF to the keys (here we are actually using the PRF as a PRG of sub-exponential stretch). Particularly, let $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq 1}$ be the low-complexity PRF in [13], $m = m(n) \leq \mathsf{poly}(n)$ be the key length of $\mathcal{F}_n$, and $\mathcal{G} = \{\mathcal{G}_n\}_{n\geq 1}$ be a PRF against $\exp(n^{\varepsilon})$-time adversaries. Let $f_{n,k}$ be the function in $\mathcal{F}_n$ with key k, and $\mathtt{tt}_m(f)$ be the first m bits in the truth table of the function f. We can define $\mathcal{H} = \{\mathcal{H}_n\}_{n\geq 1}$ as

$$\mathcal{H}_n = \left\{ f_{n,\mathtt{tt}_m(g)} \mid g \in \mathcal{G}_{\lceil \log^{2/\varepsilon} m \rceil} \right\}.$$

Then $\mathcal{H}$ is a PRF with the same circuit complexity as $\mathcal{F}$, while the key length is only $\mathsf{poly}(\lceil \log^{2/\varepsilon} m \rceil) = \mathsf{polylog}(n)$. However, it is not clear how to directly reduce the uniformity to POLYLOGTIME-uniform in this way.

1.4 Intuition

We now briefly discuss the ideas behind our new results. Since our hash construction is based on the previous construction by Fan, Li, and Yang [13], we will first recall their original construction and then discuss how to reduce the key length and make the construction POLYLOGTIME-uniform and strongly explicit. After that, we will explain how to derive sharp bootstrapping results and construct POLYLOGTIME-uniform PRFs from the new hash construction.

1.4.1 Construction of randomness efficient almost-universal hash functions

The $2n + o(n)$ almost universal hash in [13]. Our starting point is the low-complexity hash function $\mathcal{H}$ in [13] from high-girth graphs[14]. The constructed $\mathcal{H}$ is linear over $\mathbb{F}_2$. Hence, in order to make it almost universal, we only need to guarantee that for every non-zero input $x \in \{0,1\}^n \setminus 0^n$, $\mathcal{H}(x) \neq 0$ with high probability.

Their hash function $\mathcal{H}$ is the concatenation of two "hash functions" $\mathcal{H}^{\mathsf{light}}$ and $\mathcal{H}^{\mathsf{heavy}}$. The former one ensures that $\mathcal{H}^{\mathsf{light}}(x) \neq 0$ with high probability for any non-zero x with small Hamming weights ($0 < |x| \leq n/2$), and the latter one ensures that $\mathcal{H}^{\mathsf{heavy}}(x) \neq 0$ for any x with large Hamming weights ($|x| > n/2$). The construction of $\mathcal{H}^{\mathsf{heavy}}$ is quite simple: for any non-zero input with Hamming weight larger than $n/2$, a random sampling of $n/\log n$ positions over all input bits includes a 1 with high probability; the hash $\mathcal{H}^{\mathsf{light}}$ that deals with non-zero inputs with Hamming weights smaller than $n/2$ is in fact a combinatorial primitive called 1-*detector*: a distribution $\mathcal{L}$ over n-bit functions such that for any non-zero x with Hamming weight at most r (which is called the *range* of the 1-detector), $\mathcal{L}(x) \neq 0$ with high probability.

Fan, Li, and Yang [13] presented a construction of low-complexity 1-detectors using high-girth graphs. Consider an undirected graph with m vertices and n edges, the *girth* of G is the length of the shortest cycle in it. Let G be a graph with $n = \Theta(m \log m)$ and girth $g = \Omega\left(\frac{\log n}{\log\log n}\right)$ (see, e.g., [27, 7]), then the required 1-detector (with range $r = n/2$) is a depth-1 $\mathsf{CC}^0[2]$ circuit whose topology is the vertex-edge incident graph of G, with input bits being randomly permuted. More formally, the edges of G are randomly permuted and

[14] The *girth* of an undirected graph is the length of the shortest cycles in it. A high-girth graph usually means a graph $G = (V, E)$ with girth $\Omega(\log_k n)$, where $k = 2|E|/|V|$ is the average degree of vertices.

assigned to the n input bits of the circuit, and each of the m nodes in G is assigned to an output gate computing the XOR of the input bits corresponding to the adjacent edges of the node.

The crucial observation leading to the analysis of their construction is that every input x with an all-zero output would imply a subset S of edges in G of size $|x|$ such that every vertex is adjacent to even number of edges in S, which further leads to a cycle in G of size at most $|x|$ that does not likely to exist in a high-girth graph.

Now we present the analysis more formally. We call a subset S of edges *good* if at least one of the vertices is adjacent to an odd number of edges in S. It is easy to see that an input x with non-zero output corresponds to a good subset of size $|x|$. Therefore, the 1-detector above has range r if and only if for all $0 < \ell \leq r$, a randomly chosen subset of size ℓ is good with high probability. Consider the cases for $\ell < g$ and $g \leq \ell \leq n/2$ separately.

1. Note that the graph has girth at least $g = \Omega\left(\frac{\log n}{\log\log n}\right)$. Every subset S of size $\ell < g$ is good, since the induced subgraph of a bad subset S would contain a cycle of size at most $|S|$.
2. Otherwise let $\ell \in [g, n/2]$. We claim that if we have chosen all but the first $\lceil g/3 \rceil$ edges, there will be *at most one* bad subset containing the $\ell - \lceil g/3 \rceil$ chosen edges: if both S_1 and S_2 are two distinct bad subsets containing them, their symmetric difference $S_1 \oplus S_2$ would also be a bad subset of size at most $2\lceil g/3 \rceil < g$, which is impossible as discussed above. This means that if a subset of edges of size ℓ is randomly chosen, it will not hit a bad subset with high probability.

Derandomizing the construction. The original hash in [13] requires $O(n \log n)$ bits of seed in $\mathcal{H}^{\mathsf{light}}$ to permute the input bits and another $O(n)$ bits of seed in $\mathcal{H}^{\mathsf{heavy}}$ to randomly sample $n/\log n$ input bits. To reduce the overall seed length to $\mathsf{polylog}(n)$, we need to derandomize both of these two parts.

To derandomize $\mathcal{H}^{\mathsf{light}}$ (i.e., the 1-detector $\mathcal{L}$ based on high-girth graphs), we need to look into the correctness proof of it. Let x be an input with Hamming weight smaller than the range r of the 1-detector. It can be verified that the analysis of the 1-detector is correct as long as the 1-indices of x are randomly permuted (i.e., the 1-indices are randomly assigned to distinct edges in G). So if we restrict ourselves to the cases $r = \log^3 n$ (instead of $n/2$ in [13]), then a $\log^3 n$-wise (almost) independent permutation[15] would already be sufficient. In particular, we use the k-wise ε-dependent permutation by Kaplan, Naor, and Reingold [25] with seed length $O(k \log n + \log(1/\varepsilon)) = O(\log^4 n)$, where $k = \log^3 n$ and $\varepsilon = \exp(-\log^2 n)$ in our setting.

Apart from the 1-detector, we also need to derandomize the random sampling part in $\mathcal{H}^{\mathsf{heavy}}$. We need to carefully choose the parameters so that we can handle all x with Hamming weights greater than $\log^3 n$ (since our 1-detector can only handle those below this threshold). In order to achieve negligible collision probability, we can sample $n/\log n$ number of bits. This is done with a folklore sampling trick via k-wise independent hash functions, which requires $O(\log^4 n)$ bits of randomness using [1].

Note that the collision probability of our derandomized hash function is $\exp(-\Omega(\frac{\log^2 n}{\log\log n}))$, which is the same as the original construction in [13] up to the constant hidden in $\Omega(\cdot)$.

[15] A k-wise almost independent permutation is a distribution $\mathcal{F}$ of permutations over $[n]$ such that for all $0 \leq i_0 < i_1 < \cdots < i_{k-1} < n$, the distribution $(\mathcal{F}(i_0), \mathcal{F}(i_1), \ldots, \mathcal{F}(i_{k-1}))$ is statistically close to $(\mathcal{R}(i_0), \mathcal{R}(i_1), \ldots, \mathcal{R}(i_{k-1}))$, where $\mathcal{R}$ is a truly random permutation.

Reducing the output length. The hash function above has $\Theta(n/\log n)$ output bits, whereas an ordinary universal hash with the same collision probability has output length only $\mathsf{polylog}(n)$. Fortunately, the output length can be reduced to $\mathsf{polylog}(n)$ with little overhead in seed length and circuit complexity. Since the composition of two almost universal hash functions is still almost universal, we compose the hash above with itself to reduce the output length to $\Theta(n/\log^2 n)$. We can then compose the resulting hash function with an almost universal hash with output length $\mathsf{polylog}(n)$ and $(\mathsf{CC}^0[2])$ circuit complexity $o(n\log^2 n)$ in [9] based on ε-biased sets [1] and expander walks, the overall circuit complexity can still be bounded by $2n + o(n)$.

In fact, there is a trade-off between circuit depth and seed length in our hash construction by setting up the parameters more carefully. The approach described above can be computed by depth-3 $\mathsf{CC}^0[2]$ circuits and requires a seed length $O(\log^4 n)$. If we allow the depth to be 300, the seed length can be reduced to $\log^{3.01} n$.

Making the construction explicit and uniform. There are several main components in our hash construction that are not obviously strongly explicit and POLYLOGTIME-uniform. The first one is the high-girth graph construction of 1-detectors in $\mathcal{H}^{\mathsf{light}}$. We observe that the local topology around an edge or vertex in the high-girth graphs constructed by [27] can be obtained within poly-logarithmic time; see Appendix A for a formal argument. The k-wise almost-independent random permutation used in our 1-detectors for $\mathcal{H}^{\mathsf{light}}$ is also strongly explicit according to [25]. Therefore $\mathcal{H}^{\mathsf{light}}$ is strongly explicit. It is also easy to verify that $\mathcal{H}^{\mathsf{heavy}}$ is strongly explicit given the k-wise ε-dependent distribution in [1]. Finally, the hash function in [9] for shrinkage reduction is strongly explicit, since both of its two components (i.e., ε-biased sets and expander graphs) are strongly explicit (see [25, 3]).

1.4.2 Applications to hardness magnification and construction of PRFs

Sharp hardness magnification. We first show how to apply our new hash construction to obtain an extremely sharp hardness magnification result for all sparse NP languages (i.e., Theorem 1.2 and Theorem 1.3), following [9, 10]. Recall that [9, Theorem 1.2] proved that if there is a $2^{n^{o(1)}}$-sparse NP language L that cannot be computed by cn-size probabilistic circuits for some big constant $c \gg 1$, then $\mathsf{NP} \nsubseteq \mathsf{SIZE}[n^k]$ for all constant k.

[9, Theorem 1.2] is proved by taking the contrapositive: assuming that $\mathsf{NP} \subseteq \mathsf{SIZE}[n^k]$ for some k, we first sample h from a proper hash function family to *kernelize* any sparse language L in NP (i.e., a randomized reduction from the sparse language L to another (non-sparse) NP problem L' with much smaller input size $m \ll n^{1/k}$), then use the $\mathsf{SIZE}[n^k]$ circuit D for L' (which has size $m^k \ll n$) to solve L. Now, composing the hash function h together with D gives a linear-size probabilistic circuit computing L. This proves the contrapositive of our desired theorem. Due to technical reasons, in the proof above one also has to combine the hash function h with an error-correcting code. While there is an efficient construction of linear-size error-correcting codes [40], it has size cn for some constant $c \gg 2$. Hence, the size of encoding circuits for error-correcting codes become the bottleneck which prevents further improvement.

We are able to avoid using error-correcting codes by utilizing properties of the new almost-universal hash construction (the construction of [9] works for all k-perfect hash). In more details, let $H(v,x) : \{0,1\}^{O(\ell\log^2 n)} \times \{0,1\}^n \to \{0,1\}^{m(n)}$ be the hash function given by Theorem 1.1 with $\ell = \log^2 n/\log\log n$. For any sparse language L, we define an intermediate problem as follows: $L' = \{(v,h) \mid \exists x \in \{0,1\}^n \cap L \text{ s.t. } H(v,x) = h\}$. By the definition, for any no-instance $x \in \{0,1\}^n \setminus L$, the probability that there exists

an yes-instance $x' \in L \cap \{0,1\}^n$ having the same hash value with x can be bounded by $\exp(-\Omega(\ell)) \cdot |L \cap \{0,1\}^n| = n^{-\omega(1)}$. Hence by checking whether $(v, H(v,x)) \in L'$ for a random seed v, we can determine with high probability whether $x \in L$. This gives us a simple probabilistic circuit to decide L that only needs one evaluation of the hash function H (which is of circuit complexity $2n + o(n)$) and one oracle query to L'. Given the assumption that $\mathsf{NP} \subseteq \mathsf{SIZE}[n^k]$, we can show that L' can be decided by circuits of size $o(n)$, hence the overall circuit complexity would be $2n + o(n)$.

Uniform pseudorandom functions. We now briefly describe how our hash function can be used to obtain POLYLOGTIME-uniform pseudorandom functions with $\mathsf{polylog}(n)$ key length. Following the framework established in [23, 13], we use Levin's trick, which says that the composition of a PRF and an almost universal hash function is also pseudorandom. More formally, let $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq 1}$ be a family of PRF secure against any $\exp(n^\varepsilon)$-time adversary, and $\mathcal{H} = \{\mathcal{H}_n\}_{n\geq 1}$ be an almost universal hash function with output length $\log^c n$, then the composition $\mathcal{F} \circ \mathcal{H}$ is also a PRF against any polynomial-time adversary as long as $c > \varepsilon^{-1}$. Let $c = 2\varepsilon^{-1}$ and $\mathcal{C}$ be any circuit class. By assuming a PRF (secure against any $\exp(n^\varepsilon)$-time adversary) with polynomial key length computable by P-uniform $\mathcal{C}$-circuits of size $o(\exp(n^\varepsilon))$, we can then obtain a PRF (secure against any polynomial-time adversary) computable by POLYLOGTIME-uniform $2n + o(n)$ size $\mathcal{C}$-circuits with $\mathsf{polylog}(n)$ key length, using our POLYLOGTIME-uniform efficient hash with $\mathsf{polylog}(n)$ seed length. Note that for $\mathcal{C} \in \{B_2, \mathsf{NC}^1, \mathsf{AC}^0[2]\}$, we can implement such PRF candidates by plugging in standard constructions based on the existence of OWFs against any sub-exponential adversary for B_2 circuits, and decisional Diffie-Hellman [33] or Ring Learning-with-Errors [4] for NC^1 and $\mathsf{AC}^0[2]$ circuits.

2 Preliminaries

Notation. We define $[n] \triangleq \{0, 1, \ldots, n-1\}$, $B_{n,m}$ as the set of all functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ and $B_n \triangleq B_{n,1}$. The Hamming weight of $x \in \mathbb{F}_2^n$, denoted by $|x|$, is defined as the number of 1's in x; the Hamming distance $\Delta(x,y)$ of x and y from $\mathbb{F}_2^n$ is defined as the Hamming weight of the bitwise XOR of them. The concatenation of two strings x and y is denoted by $x\|y$, and the i^{th} bit of x (0-indexed) is denoted by x_i. Graphs are undirected by default. We assume that all functions used as parameters of our constructions are poly-logarithmic time constructible, i.e., for any function $\ell(n)$ that is used as a parameter in our results, there is a polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}(n)$ outputs the binary representation of $\ell(n)$.

We use $x \leftarrow \mathcal{D}$ to denote a random element x sampled from a distribution $\mathcal{D}$, and $\mathcal{D}(x_0) \triangleq \Pr_{x\leftarrow\mathcal{D}}[x = x_0]$. Natural numbers are represented in binary when being fed into Turing machines or circuits; and 1^n represents the unary representation of n. We assume basic familiarity with cryptographic primitives like one-way functions and pseudorandom functions, and typical complexity classes like P, NP, and $\oplus$P (see, e.g., [3]).

2.1 Probability Theory

Let $\mathcal{U}(S)$ be the uniform distribution supported on a set S, and $\mathcal{U}_\ell \triangleq \mathcal{U}(\{0,1\}^\ell)$. A family $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ of distributions is said to be *samplable with seed length* $\ell(n)$ (or $\ell(n)$-*samplable*) if there is a polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}(1^n, \mathcal{U}_{\ell(n)})$ samples $\mathcal{D}_n$. For every $s \in \{0,1\}^\ell$, we say $\mathcal{A}(1^n, s)$ is the element corresponding to the seed s.

The *statistical distance* between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over a set S, denoted by $\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2)$, is defined as

$$\mathsf{SD}(\mathcal{D}_1, \mathcal{D}_2) \triangleq \frac{1}{2} \sum_{u \in S} |\mathcal{D}_1(u) - \mathcal{D}_2(u)| .$$

A distribution $\mathcal{D}_2$ is said to be δ-*close* to $\mathcal{D}_1$ if their statistical distance is at most δ.

Statistical distance characterizes the intractability of distinguishing two distributions by any test (even computationally unbounded): if $\mathcal{D}_1$ and $\mathcal{D}_2$ have statistical distance at most δ, then for any stochastic process T, $|\Pr_{u \leftarrow \mathcal{D}_1, T}[T(u) = 1] - \Pr_{u \leftarrow \mathcal{D}_2, T}[T(u) = 1]| \leq \delta$.

2.2 Circuit Classes

In this paper we will work with various circuit classes. In general, a circuit is an acyclic graph where each of its nodes can be an input variable, a constant $c \in \{0, 1\}$, or a gate. One or more nodes are marked as output nodes (together with an index i denoting the corresponding output bit). The *depth* of a circuit is the number of edges on the longest path from any input variable to an output node. An n-input m-output circuit computes a function in $B_{n,m}$.

B_2 circuits. A B_2 circuit (or general circuit) contains fan-in-2 gates that can compute any binary function $f \in B_2$. The *size* of a B_2 circuit refers to the number of gates involved.

NC^1 circuits. An NC^1 circuit is a B_2 circuit with $O(\log n)$ depth. Note that a single-output NC^1 circuit of depth d can be converted into a formula of size $O(2^d)$.

$\mathsf{CC}^0[2]$ circuits. A $\mathsf{CC}^0[2]$ circuit is a constant-depth circuit with only XOR gates of unbounded fan-in. It is easy to see that $\mathsf{CC}^0[2]$ circuits can only compute linear functions over $\mathbb{F}_2$. The complexity of a $\mathsf{CC}^0[2]$ circuit is usually measured by the number of wires.

$\mathsf{AC}^0[m]$ circuits. An $\mathsf{AC}^0[m]$ circuit is a constant-depth circuit with fan-in-1 NOT gates and unbounded fan-in gates over $\{\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m\}$, where $\mathsf{MOD}_m(x_1, \ldots, x_k) = 1$ if and only if $x_1 + x_2 + \cdots + x_k \equiv 0 \pmod m$. Similar to $\mathsf{CC}^0[2]$ circuits, the complexity of an $\mathsf{AC}^0[m]$ circuit is measured by the number of wires.

A *probabilistic B_2 circuit* (or simply a probabilistic circuit) with input size n and output size m is a distribution $\mathcal{C}_n$ over n-input m-output B_2 circuits. The circuit complexity of a probabilistic circuit is defined as the maximum complexity of functions in its support. A family $\mathcal{C} = \{\mathcal{C}_n\}_{n \geq 1}$ of n-input 1-output probabilistic circuit is said to *decide a language L with error probability $\varepsilon = \varepsilon(n)$* if for sufficiently large n and all $x \in \mathbb{F}_2^n$, $\Pr_{C \leftarrow \mathcal{C}_n}[C(x) \neq L(x)] \leq \varepsilon$.

2.3 Hash and 1-detector

▶ **Definition 2.1** (Almost universal hash function). *Let $m = m(n)$ and $\varepsilon = \varepsilon(n)$ be two parameters. An* m-output ε-almost universal hash function *is a family of distributions $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$, where $\mathcal{H}_n$ is supported on $B_{n,m}$, such that $\Pr_{h \leftarrow \mathcal{H}_n}[h(x) = h(y)] \leq \varepsilon(n)$ for all $x \neq y$ and sufficiently large n. The parameter ε is called its* collision probability. *It is* linear *if every function in the support of $\mathcal{H}_n$ is linear. It is said to be $s(n)$-*samplable *if the family $\mathcal{H}$ of distributions is samplable with seed length $s(n)$.*

We will make heavy use of the notion of 1-detectors in [13], as parts of our almost universal hash construction.

▶ **Definition 2.2** (1-detector). *An* m-output (randomized) 1-detector with range r and error ε *is a family of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \geq 1}$, where $\mathcal{D}_n$ is supported on linear functions in $B_{n,m}$, such that for sufficiently large n, $\Pr_{L \leftarrow \mathcal{D}_n}[L(x) \neq 0] \leq \varepsilon(n)$ for all $x \in \mathbb{F}_2^n$ satisfying $0 < |x| \leq r$. It is said to be $s(n)$-*samplable *if the family $\mathcal{D}$ of distributions is samplable with seed length $s(n)$.*

We emphasize that the definition requires the functions in the family of distributions to be linear. This ensure that, for any randomized 1-detector $\mathcal{D}_n$ with range r and error ε, and any $x_1 \neq x_2$ with Hamming distance at most r, it holds that $\Pr_{L \leftarrow \mathcal{D}_n}[L(x_1) \neq L(x_2)] \leq \varepsilon(n)$.

The *shrinkage* of a hash function (or 1-detector) is defined as the input length n divided by the output length m (e.g., poly-logarithmic shrinkage means $m = n/\mathsf{polylog}(n)$).

2.4 ε-biased set and k-wise independence

To derandomize the low-complexity hash function in [13], we need several standard derandomization tools including strongly explicit ε-biased sets, k-wise independent distributions, and k-wise independent permutations.

▶ **Definition 2.3** (ε-biased set)**.** *For any $n \geq 1$ and $\varepsilon \in (0, 1/2)$, a set $S \subseteq \mathbb{F}_2^n$ is said to be ε-biased if for all non-zero $v \in \mathbb{F}_2^n$, $\Pr_{w \leftarrow S}[\langle w, v\rangle = 0] \in [1/2 - \varepsilon, 1/2 + \varepsilon]$.*

▶ **Theorem 2.4** (Alon, Goldreich, Håstad, and Peralta [1])**.** *For any constant $\varepsilon \in (0, 1/2)$, there is a family $\{S_n \subseteq \mathbb{F}_2^n\}_{n \geq 1}$ of ε-biased sets such that $|S_n| = \tilde{O}(n^2)$. Moreover, there is an algorithm $\mathcal{A}(n, i, j)$ running in time $\mathsf{poly}(\log n)$ that computes the j^{th} bit of the i^{th} vector in S_n.*

▶ **Definition 2.5.** *A k-wise ε-dependent distribution with length n and alphabet size p is a distribution $\mathcal{D}$ over $[p]^n$ such that for all $0 \leq i_1 < i_2 < \cdots < i_k < n$, the distribution over $[p]^k$ obtained by restricting $\mathcal{D}$ to the $i_1^{\text{th}}, i_2^{\text{th}}, \ldots, i_k^{\text{th}}$ coordinates is ε-close to the uniformly distribution over $[p]^k$. It is said to be a k-wise independent distribution if $\varepsilon = 0$.*

▶ **Theorem 2.6** (Alon, Goldreich, Håstad, and Peralta [1])**.** *Let $\varepsilon = \varepsilon(n) > 0$ be a parameter. There is an algorithm $\mathcal{A}$ such that $\mathcal{A}(1^n, k, \mathcal{U}_r)$ runs in time $\mathsf{poly}(n, \log(1/\varepsilon))$ and samples a k-wise ε-dependent distribution with length n and alphabet size 2 for every $1 \leq k \leq n$, where $r = O(k + \log\log n + \log(1/\varepsilon))$. Moreover, there is an algorithm $\mathcal{B}$ such that $\mathcal{B}(n, k, v, i)$ runs in $\mathsf{poly}(\log n, k, \log(1/\varepsilon))$ time and computes the i^{th} coordinate of the string corresponding to the seed v.*

▶ **Definition 2.7.** *Let $n \geq 1$ be the number of elements, and P_n be the set of all permutations over $[n]$. A k-wise ε-dependent permutation is a distribution $\mathcal{D}$ over P_n such that for all $0 \leq i_1 < i_2 < \cdots < i_k < n$, the distribution of $(f(i_1), f(i_2), \ldots, f(i_k))$ with $f \leftarrow \mathcal{D}$ is ε-close to the uniform distribution over $\{(j_1, j_2, \ldots, j_k) \mid j_1, \ldots, j_k \in [n]$ are pairwisely distinct$\}$.*

▶ **Theorem 2.8** (Kaplan, Naor, and Reingold [25], Theorem 5.9)**.** *Let n be a power of 2, $2 \leq k \leq n$, and $\varepsilon > 2^{-n}$. There is an n-element k-wise ε-dependent permutation that is samplable with seed length $O(k \log n + \log(1/\varepsilon))$. Moreover, there is an algorithm $\mathcal{A}(n, v, i)$ that runs in $\mathsf{poly}(\log n, k, \log(1/\varepsilon))$ time and outputs $\rho_v(i)$ for the permutation ρ_v corresponding to the seed v.*

2.5 Expander Graphs

We will need strongly explicit expander graphs in our proofs. We first recall the definition of expander graphs.

▶ **Definition 2.9.** *An n-vertex d-regular graph G is called an (n, d, λ)* expander graph *(or (n, d, λ)-graph) if $\lambda_2(G) \leq \lambda$, where $\lambda_2(G)$ denotes the second largest eigenvalue of normalized adjacency matrix (i.e., adjacency matrix divided by d) of G. A family of graphs $\{G_n\}_{n \geq 1}$ is an* expander graph family *if there exists constants d and $\lambda < 1$ such that for all sufficiently large n, G_n is an (n, d, λ)-graph.*

We will make use of the following construction of strongly explicit expander graphs.

▶ **Theorem 2.10** (Strongly explicit expander; see [3, Theorem 21.19]). *There exists an expander graph family $\{G_n\}_{n\geq 1}$ and an algorithm $\mathcal{A}(n, v, i)$ that runs in* $\mathsf{polylog}(n)$ *time (i.e., polynomial in input length) and outputs the i^{th} neighbor of the node v in G_n, where $v \in [n]$ and $i \in [d]$.*

Performing random walk on expanders is a standard derandomization technique. Consider the task to find a good element from n elements in which there is a constant fraction of them being good. A trivial approach is to sample ℓ independently random elements, which has $\exp(-\Omega(\ell))$ error probability but requires $\ell \log n$ random bits. By applying the following lemma, we can reduce the randomness complexity to $O(\log n + \ell)$ while keeping the error probability to be exponentially small.

▶ **Lemma 2.11** (Expander walk; see [3, Theorem 21.12]). *Let G be an (n, d, λ)-graph and S be a subset of vertices of size at most βn for some $\beta \in (0, 1)$. Let $X_1, X_2, \ldots, X_k$ be random variables denoting a random walk in G (where X_1 is uniformly chosen), then*

$$\Pr[\forall 1 \leq i \leq k, X_i \in S] \leq \left((1-\lambda)\sqrt{\beta} + \lambda\right)^{k-1}.$$

2.6 Graph with large girth

The *girth* of an undirected graph is the length of the minimum cycle in it. We need the following construction of strongly explicit graphs with large girth.

▶ **Theorem 2.12** (Adapted from [27]; see Appendix A). *Let $r = r(n) = n^{o(1)}$ be a parameter. For every sufficiently large n, there exists an $m = \Theta(\frac{n}{r})$ and a regular graph $G_{m,n}$ with m vertices, n edges, and girth $\Omega(\frac{\log n}{\log r})$. Moreover, there exists a* $\mathsf{polylog}(n)$*-time algorithm $\mathcal{A}(n, i)$ for $i \in [n]$ that outputs the indices of the two endpoints of the i^{th} edge in $G_{m,n}$, and a* $\mathsf{polylog}(n)$*-time algorithm $\mathcal{B}(n, i, j)$ for $i \in [m]$ that outputs the j^{th} edge attaching to the i^{th} vertex.*

3 Randomness-efficient low-complexity hash functions

In this section, we present constructions of randomness-efficient low-complexity hash functions with various parameters and properties.

In Section 3.1, we show almost universal hash functions can be constructed from 1-detectors (Lemma 3.2). In Section 3.2 we give a derandomized version of construction of 1-detectors from [13] based on high-girth graphs (Lemma 3.3), and use that to construct a low-complexity hash function with poly-logarithmic seed-length and $n/\mathsf{polylog}(n)$ output length (Theorem 3.4).

In Section 3.3, we show how to reduce the output length from $n/\mathsf{polylog}(n)$ to $\mathsf{polylog}(n)$, by composing almost universal hash families. Finally, in Section 3.4 and 3.5, we establish the uniformity and explicitness of our constructions, which are essential for our applications to hardness magnification and PRF constructions.

Notation. In this section, for a construction $\mathcal{H}$ with parameters a, b, c, we will use $\mathcal{H}_{\hat{a},\hat{b},\hat{c}}$ to denote the specific construction $\mathcal{H}$ with the parameters specified to $\hat{a}, \hat{b}$ and $\hat{c}$. When the parameters are obvious from the context, we often omit the subscripts and simply write it as $\mathcal{H}$.

3.1 General construction from 1-detectors

We first give a general construction of almost universal hash functions from 1-detectors, using the following sampling procedure.

▶ **Lemma 3.1.** *For all integers n, b and r such that $b \le n$ and $\max\{10n/b, 10\log\log n\} \le r \le n$, there is a distribution $\mathcal{D}^{\mathsf{samp}}_{n,b,r}$ supported on $[n]^b$ samplable by $O(r\log(n/b))$ bits, such that the following conditions hold:*

1. *(**Hitting Condition**). For all $x \in \mathbb{F}_2^n$ with $|x| \ge r$, it holds that*

$$\Pr_{(w_0,\ldots,w_{b-1})\leftarrow\mathcal{D}^{\mathsf{samp}}_{n,b,r}}\left[\bigwedge_{j\in[b]}[x_{w_j}=0]\right] \le 2\exp\left(-\frac{br}{2n}\right). \tag{1}$$

2. *(**Ordering**) For every $(w_0,\ldots,w_{b-1}) \in \mathsf{supp}(\mathcal{D}^{\mathsf{samp}}_{n,b,r})$, it holds that $w_0 < w_1 < \cdots < w_{b-1}$.*
3. *(**Explicitness**) There are algorithms $\mathcal{A}^{\mathsf{samp}}_{n,b,r}(v,i)$ and $\mathcal{B}^{\mathsf{samp}}_{n,b,r}(v,j)$ running in $\mathsf{poly}(\log n, r)$ time such that*

$$\mathcal{A}^{\mathsf{samp}}_{n,b,r}(v,i) \text{ outputs } \begin{cases} j & \text{if } w_j = i \text{ for some } j\in[b],\\ \perp & \text{otherwise;}\end{cases}$$

$$\mathcal{B}^{\mathsf{samp}}_{n,b,r}(v,j) \text{ outputs } w_j.$$

where $(w_0,\ldots,w_{b-1}) \in \mathsf{supp}(\mathcal{D}^{\mathsf{samp}}_{n,b,r})$ is the vector corresponding to the seed v.

Proof. We firstly describe a sampling procedure that satisfies Equation (1) but has a long seed length and then reduce the seed length to $O(r\log(n/b))$ bits using the explicit k-wise ε-dependent distribution from Theorem 2.6. The sampling procedure is simple: we first partition $[n]$ into b consecutive groups $g_0, g_1, \ldots, g_{b-1}$ of size either $\lfloor n/b\rfloor$ or $\lceil n/b\rceil$; we then uniformly choose an $i_j \in g_j$ for each $j \in [b]$; finally we output the tuple $(i_0, i_1, \ldots, i_{b-1})$.

For any $x \in \mathbb{F}_2^n$ with Hamming weight at least r, for some $\ell \le r$, there are ℓ groups such that there are at least r 1-indices (i.e., the corresponding bits of x are 1) in these groups. Assume that there are a_j 1-indices in the j^{th} among these ℓ groups. The probability that none of the 1-indices is sampled is at most

$$\prod_{j=1}^{\ell}\left(1-\frac{a_j b}{2n}\right) \le \exp\left(\sum_{j=1}^{\ell}\log\left(1-\frac{a_j b}{2n}\right)\right) \le \exp\left(-\sum_{j=1}^{\ell}\frac{a_j b}{2n}\right) \le \exp\left(-\frac{br}{2n}\right). \tag{2}$$

A trivial implementation of the sampling procedure above needs seed length $O(b\log(n/b))$: for each group, we sample a random binary string of length $t \triangleq \lceil\log(n/b)\rceil$ indicating the index we want to choose. Therefore, we would need to sample bt bits in total. The observation here is that the argument in (2) only involves $\ell \le r$ groups. Hence, the analysis still works if we sample those required bt bits from an rt-wise $\exp(-br/(2n))$-dependent distribution with alphabet 2. Note that this incurs an additional error of $\exp(-br/(2n))$.

By Theorem 2.6, the distribution is samplable with seed length

$$O(rt + \log\log(bt) + br/(2n)) = O(r\log(n/b)),$$

and the algorithms $\mathcal{A}^{\mathsf{samp}}_{n,b,r}$ and $\mathcal{B}^{\mathsf{samp}}_{n,b,r}$ can be constructed straightforwardly from the algorithm $\mathcal{B}$ in Theorem 2.6.[16] ◀

[16] To compute $\mathcal{A}^{\mathsf{samp}}_{n,b,r}(v,i)$ for some i belonging to the group g_a, we only need to check if i is the selected element of the group g_a, and then output a if the answer is yes, and output $\perp$ otherwise.

Now we are ready to give the construction of our hash function from 1-detectors. Recall that their definitions are given in Section 2.3.

▶ **Construction 1** (Hash function $\mathcal{H}^{\mathcal{L}}$ from 1-detectors $\mathcal{L}$)**.** *Let n be the input length and $b(n) = o(n)$ be a parameter. Given a randomized 1-detector $\mathcal{L} = \{\mathcal{L}_n\}_{n\geq 1}$ with range $\hat{r}$, we construct a hash function $\mathcal{H}_b^{\mathcal{L}} = \{\mathcal{H}_n\}_{n\geq 1}$ as follows.*

- *Let $\mathcal{D}^{\mathsf{samp}}$ be the distribution in Lemma 3.1 with parameters n, b and $r = \hat{r}$. For each L in the support of $\mathcal{L}$ and each $D = (j_0, \ldots, j_{b-1})$ in the support of $\mathcal{D}^{\mathsf{samp}}$, we define a function $h_{L,D}$ as*

$$h_{L,D}(x) \triangleq L(x) \| x_{j_0} \| x_{j_1} \| \ldots \| x_{j_{b-1}}.$$

- *$\mathcal{H}_n$ is then defined to be the distribution generated as follows: sample $L \leftarrow \mathcal{L}_n$ and $D \leftarrow \mathcal{D}^{\mathsf{samp}}$, and then output $h_{L,D}$.*

We show that Construction 1 indeed gives almost universal hash functions.

▶ **Lemma 3.2** ($\mathcal{H}^{\mathcal{L}}$ is a linear almost universal hash)**.** *Let $\mathcal{L} = \{\mathcal{L}_n\}_{n\geq 1}$ be an $\hat{m}$-output $\hat{s}$-samplable randomized 1-detector with error $\hat{\varepsilon}$ and range $\hat{r}$, such that $\hat{m} = o(n)$ and $\hat{r} = \omega(\log n)$. Let $b = b(n) = o(n)$ be a parameter and $r = r(n) = \hat{r}(n)$. The followings hold for $\mathcal{H}_b^{\mathcal{L}}$:*

1. *$\mathcal{H}_b^{\mathcal{L}}$ has output length $m = \hat{m} + b$.*
2. *$\mathcal{H}_b^{\mathcal{L}}$ is $(\hat{s} + \hat{r}\log(n/b))$-samplable.*
3. *$\mathcal{H}_b^{\mathcal{L}}$ is a linear ε-almost universal hash function, where $\varepsilon \triangleq \hat{\varepsilon} + \exp(-\Omega(br/n))$.*

Proof. The first two items follow directly from the definition of $\mathcal{H}^{\mathcal{L}}$ from Construction 1. So we will only establish the last item. We need to show that for any two inputs $x_1 \neq x_2$ from $\{0,1\}^n$, their hash values collide with probability at most ε.

Consider the following two cases depending on whether the Hamming distance between x_1 and x_2 is small or large:

- If $0 < \Delta(x_1, x_2) \leq r$, then by the definition of randomized 1-detector (Definition 2.2), $L(x_1)$ equals to $L(x_2)$ with probability at most $\hat{\varepsilon}$ for $L \leftarrow \mathcal{L}_n$ (note that L is a linear function over $\mathbb{F}_2$). Since the hash values contain the output of the 1-detector as the first $\hat{m}$ bits, the hash values collide with probability at most $\hat{\varepsilon}$.
- If $\Delta(x_1, x_2) > r$, then $y \triangleq x_1 \oplus x_2$ has Hamming weight at least r. By Lemma 3.1, a random $D \leftarrow \mathcal{D}^{\mathsf{samp}}_{n,b,r}$ fails to contain all 1-indices of y with probability at most $\exp(-\Omega(br/n))$, which means that the last b bits of the hash values of x_1 and x_2 collide with probability at most $\exp(-\Omega(br/n))$.

It follows immediately that the collision probability is at most $\varepsilon = \hat{\varepsilon} + \exp(-\Omega(br/n))$. ◀

3.2 Randomness-efficient low-complexity 1-detectors

Now we will present the construction of randomness-efficient low-complexity 1-detectors. Applying Lemma 3.2, this construction yields a hash function with inverse-super-polynomial collision probability, short seed length, and moderately large shrinkage.

Intuition. Our construction is essentially a derandomized version of the randomized 1-detector based on high-girth graphs in [13]. The randomized 1-detector supports on depth-1 $\mathsf{CC}^0[2]$ circuits with $o(n)$ gates and wire-complexity $2n$. The topology of any depth-1 $\mathsf{CC}^0[2]$ circuit can be considered as the edge-vertex incident graph of an undirected graph $G_{m,n} = (V, E)$, that is, we identify the m gates with the vertices, and the n variables with

the edges in the graph. One can check that if $G_{m,n}$ has girth g, the corresponding circuit would directly make a (deterministic) m-output 1-detector with range $g-1$: assume that it is not the case, the bad input x with Hamming weight $w < g$ specifies a subset of edges $T \subseteq E$ which forms a Eulerian cycle of size $w < g$. By a similar argument, [13] boosts the range to $n/2$ by randomly permuting the input bits.

The construction of [13] does not suffice for our application because we need $\Theta(n \log n)$ bits to sample a random permutation. Fortunately, it can be derandomized. By looking into its correctness proof, we can see that if we replace the random permutation by an r-wise almost independent permutation (see Theorem 2.8), it can still achieve 1-detection for range r and roughly quasi-polynomial error probability.

Now we specify the construction of our 1-detector $\mathcal{L}^{\mathsf{hg}}$ (hg stands for high-girth graphs).

▶ **Construction 2** (Construction of the 1-detector $\mathcal{L}^{\mathsf{hg}}$ from high-girth graphs)**.** *Let $k = k(n) = n^{o(1)}$ be the shrinkage parameter, $\log n \le r(n) \le n/2$ be the range. Let $\gamma = \gamma(n) \triangleq \log^2 n / \log k$. $\mathcal{L}^{\mathsf{hg}}_{k,r} = \{\mathcal{L}_n\}_{n\ge 1}$ is constructed as follows.*

- *Let n' be the smallest power of two that is larger than n. Let $G_{m,n'}$ be the regular graph with $m = \Theta(n'/k)$ vertices, n' edges, and girth $g = \Theta(\log n / \log k)$ from Theorem 2.12. Let $\mathcal{D} = \{\mathcal{D}_{n'}\}$ be the explicit family of r-wise $2^{-\gamma}$-dependent permutation in Theorem 2.8.*
- *Let $\Gamma(i)$ be the set of indices of edges incident to the i^{th} vertex in $G_{m,n'}$. For each permutation $\sigma \in \mathsf{supp}(\mathcal{D}_{n'})$, we define a function $L_\sigma : \{0,1\}^n \to \{0,1\}^m$ as*

$$L_\sigma(x)_i \triangleq \bigoplus_{j\in[n] \ s.t. \ \sigma(j)\in\Gamma(i)} x_j \qquad \forall i \in [m].$$

- *$\mathcal{L}_n$ is then defined to be the distribution generated as follows: sample $\sigma \leftarrow \mathcal{D}_{n'}$ and then output L_σ.*

▶ **Lemma 3.3** ($\mathcal{L}^{\mathsf{hg}}$ is a 1-detector)**.** *Let $k = k(n) = n^{o(1)}$ be the shrinkage parameter, $\log n \le r(n) \le n/2$ be the range. The followings hold for $\mathcal{L}^{\mathsf{hg}}_{k,r} = \{\mathcal{L}_n\}_{n\ge 1}$:*

1. *$\mathcal{L}^{\mathsf{hg}}_{k,r}$ has output size $\Theta(n/k)$ and seed length $O(r \log n)$.*
2. *Every $L \in \mathcal{L}_n$ can be computed by depth-1 $\mathsf{CC}^0[2]$ circuits of wire complexity at most $2n$, or B_2 circuits of size $2n$ and depth $\log k + O(1)$.*
3. *$\mathcal{L}^{\mathsf{hg}}_{k,r}$ is a randomized 1-detector with range r and error $\exp(-\Omega(\log^2 n / \log k))$.*

Proof. Recall that $\gamma = \log^2 n / \log k$ from Construction 2. For the first item, the output size follows directly from the definition of $\mathcal{L}^{\mathsf{hg}}_{k,r}$. The seed length of $\mathcal{L}^{\mathsf{hg}}_{k,r}$ equals the seed length of the required r-wise $2^{-\gamma}$-dependent permutation, which is $O(r \log n + \gamma) = O(r \log n)$ from Theorem 2.8 (note that $r \ge \log n$). The second item follows directly from the definition of L_σ in Construction 2 together with the fact that $G_{m,n'}$ has maximum degree $O(k)$ (since it is regular).

In the rest of the proof we establish the third item. We need to show that for any non-zero $x \in \mathbb{F}_2^n$, the probability that $L(x) = 0$ with $L \leftarrow \mathcal{L}_n$ is at most $\exp(-\Omega(\gamma))$. For any input x with Hamming weight $0 < |x| < g$ (where g is the girth of $G_{m,n}$), we must have $L_\sigma(x) \neq 0$ for all $\sigma \in D_{n'}$, since otherwise we can extract from the edges $\{\sigma(i) \mid x_i = 1 \wedge i \in [n]\}$ a cycle of size length less than g. Now we consider the case when $g \le |x| \le r$. Let $i_1, i_2, \ldots, i_{|x|}$ be the indices corresponding to the 1's in x (i.e., $x_{i_j} = 1$ for all j). Since our construction is linear, only these bits can influence the output. Also, by the r-wise $2^{-\gamma}$-dependence of $D_{n'}$, we have

$$\begin{aligned}\Pr_{\sigma\leftarrow D_{n'}}[L_\sigma(x)=0] &= \Pr_{\sigma\leftarrow D_{n'}}\left[\bigwedge_{i'\in[m]}[|\{i_j \mid \sigma(i_j)\in\Gamma(i')\}| \text{ is even}]\right]\\ &\le \Pr_{S\subseteq[n'],|S|=|x|}\left[\bigwedge_{i'\in[m]}[|S\cap\Gamma(i')| \text{ is even}]\right]+2^{-\gamma}\\ &= \mathop{\mathbb{E}}_{\substack{S\subseteq[n']\\|S|=|x|-\alpha}}\left[\Pr_{\substack{S'\subseteq[n']\setminus S\\|S'|=\alpha}}\left[\bigwedge_{i'\in[m]}[|(S\cup S')\cap\Gamma(i')| \text{ is even}]\right]\right]+2^{-\gamma}.\\ &\qquad\qquad\left(\alpha\triangleq\lceil\tfrac{g}{3}\rceil\right)\end{aligned}$$

The final observation is that for any fixed S, there should be at most one S' satisfying the condition in the summation. Indeed, if two sets S'_1 and S'_2 satisfy the condition at the same time, then the symmetric difference of them contains a cycle in $G_{m,n'}$ of length $2\alpha < g$ for sufficiently large n, contracting the fact that $G_{m,n'}$ has girth g. In particular, it means for every fixed S, we have

$$\Pr_{\substack{S'\subseteq[n']\setminus S\\|S'|=\alpha}}\left[\bigwedge_{i'\in[m]}[|(S\cup S')\cap\Gamma(i')| \text{ is even}]\right] \le \frac{1}{\binom{n'-|x|+\alpha}{\alpha}}.$$

Since $\alpha=\lceil g/3\rceil=\Theta(\log n/\log k)$, putting everything together, for sufficiently large n we have

$$\Pr_{\sigma\leftarrow D_n}[L_\sigma(x)=0] \le \frac{1}{\binom{n'-|x|+\alpha}{\alpha}}+2^{-\gamma}=\exp(-\Omega(\gamma)).$$ ◀

Plugging $\mathcal{L}^{\mathsf{hg}}$ into Construction 1 with appropriately chosen parameters, we immediately obtain a linear hash $\mathcal{H}^{\mathsf{md}}$ with poly-logarithmic seed length and shrinkage as follows. (Here md stands for moderate, meaning that the hash has moderate (poly-logarithmic) shrinkage.)

▶ **Construction 3** (Construction of hash $\mathcal{H}^{\mathsf{md}}$ with poly-logarithmic shrinkage). *Let $\beta\in(0,2]$ be a constant and $\omega(\log n)\le\delta(n)\le O(\frac{\log^2 n}{\log\log n})$ be the error parameter. $\mathcal{H}^{\mathsf{md}}_{\beta,\delta}=\{\mathcal{H}_n\}_{n\ge1}$ is constructed as follows.*

- *Setting $k=\log^\beta n$ and $r=\delta\log^\beta n$, $\mathcal{L}^{\mathsf{hg}}=\mathcal{L}^{\mathsf{hg}}_{k,r}$ (from Construction 2) is an $O(\delta\log^{\beta+1}n)$-samplable $\Theta(n/\log^\beta n)$-output randomized 1-detector with range r and error $\exp(-\Omega(\delta))$.*
- *Setting $b=n/\log^\beta n$, $\mathcal{H}^{\mathsf{md}}$ is now defined to be $\mathcal{H}^{\mathcal{L}^{\mathsf{hg}}}_b$ (from Construction 1).*

The following theorem follows directly from the Construction 3, Lemma 3.2, and Lemma 3.3.

▶ **Theorem 3.4** (Properties of the intermediate hash construction $\mathcal{H}^{\mathsf{md}}$). *Let $\beta\in(0,2]$ be the shrinkage parameter and $\omega(\log n)\le\delta(n)\le O(\frac{\log^2 n}{\log\log n})$ be the error parameter. The followings hold for $\mathcal{H}^{\mathsf{md}}_{\beta,\delta}=\{\mathcal{H}_n\}_{n\ge1}$:*

1. *$\mathcal{H}^{\mathsf{md}}_{\beta,\delta}$ is a linear $O(\delta\log^{\beta+1}n)$-samplable $\Theta(n/\log^\beta n)$-output $\exp(-\Omega(\delta))$-almost universal hash function.*
2. *Every $H\in\mathcal{H}_n$ can be computed either by a depth-1 $\mathsf{CC}^0[2]$ circuit of wire complexity $2n$, or by a B_2 circuit of size $2n$ and depth $\beta\log\log n+O(1)$.*

3.3 Shrinkage reduction of hash function

Now we reduce the output length of the hash function $\mathcal{H}^{\mathsf{md}}$ in Construction 3 from $n/\mathsf{polylog}(n)$ to $\mathsf{polylog}(n)$ with little overhead in its circuit complexity and seed length, which is crucial for our applications. The idea is simple: we composite it with a hash with large shrinkage, short seed length, and relatively larger circuit complexity. In particular, we can use the following hash construction $\mathcal{H}^{\mathsf{expw}}$ of Chen, Jin, and Williams [9]. (Here expw stands for expander walk.)

▶ **Construction 4** (Hash function $\mathcal{H}^{\mathsf{expw}}$ from expander walk [9])**.** *Let $\ell = \ell(n)$ be the output length. $\mathcal{H}^{\mathsf{expw}}_\ell = \{\mathcal{H}_n\}_{\geq n}$ is constructed as follows.*

- *Let $\{S_n\}_{n\geq 1}$ be a family of 0.1-biased set from Theorem 2.4 and $\{G_n\}_{n\geq 1}$ be the family of strongly explicit expanders from Theorem 2.10. Assume that $S_n = \{w_0, w_1, \ldots, w_{t-1}\}$ for $t = \tilde{O}(n^2)$.*
- *For each walk $v = (v_0, v_1, \ldots, v_{\ell-1}) \in [t]^\ell$ of length ℓ on G_t, we define a hash function $h_v \colon \{0,1\}^n \to \{0,1\}^\ell$ as*

$$h_v(x) \triangleq \langle w_{v_0}, x\rangle \,\|\, \langle w_{v_1}, x\rangle \,\|\, \ldots \,\|\, \langle w_{v_{\ell-1}}, x\rangle\,,$$

  where $\langle a, b\rangle$ denotes the inner product over $\mathbb{F}_2$.
- *$\mathcal{H}_n$ is then defined to be the distribution generated as follows: samples a random walk v of length ℓ on G_t uniformly at random, then outputs h_v.*

▶ **Lemma 3.5** (Properties of the hash construction $\mathcal{H}^{\mathsf{expw}}$ from [9])**.** *Let $\ell = \ell(n)$ be the output length. The followings hold for $\mathcal{H}^{\mathsf{expw}}_\ell = \{\mathcal{H}_n\}_{n\geq 1}$:*

1. *Every $H \in \mathsf{supp}(\mathcal{H}_n)$ can implemented by a depth-1 $\mathsf{CC}^0[2]$ circuit of wire complexity $n\ell$ or a B_2 circuit of size $n\ell$ and depth $\log n + O(1)$.*
2. *$\mathcal{H}^{\mathsf{expw}}_\ell$ is a linear $O(\log n + \ell)$-samplable ℓ-output $\exp(-\Omega(\ell))$-almost universal hash function.*

Proof. The first item follows directly from the definition of h_v from Construction 4. In the following we establish the second item.

The linearity, seed length, and output length are straightforward to verify, thus we only analyze the collision probability. Since the hash function is linear, we only need to show that for any non-zero $x \in \mathbb{F}_2^n$, $\Pr_{h\leftarrow\mathcal{H}_n}[h(x) = 0] \leq \exp(-\Omega(\ell))$. By Theorem 2.4, we know that for any non-zero x, at least 0.4 fraction of vectors $w \in S_n$ satisfies $\langle w, x\rangle = 1$. Then by Theorem 2.11, a random walk of length ℓ will find one of such w with probability $1 - \exp(-\Omega(\ell))$. This immediately implies that $h(x) \neq 0$ with probability at most $\exp(-\Omega(\ell))$ for $h \leftarrow \mathcal{H}_n$. ◀

Next we formally define the composition of two hash function families.

▶ **Definition 3.6** (Composition of hash families)**.** *Let $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq 1}$ and $\mathcal{G} = \{\mathcal{G}_n\}_{n\geq 1}$ be two families of hash functions. The composition of $\mathcal{F}$ and $\mathcal{G}$ is defined as $\mathcal{F}\circ\mathcal{G} = \{(\mathcal{F}\circ\mathcal{G})_n\}_{n\geq 1}$, where $(\mathcal{F}\circ\mathcal{G})_n$ is the following distribution: let $m = m(n)$ be the output length of $\mathcal{G}$, we sample $f \leftarrow \mathcal{F}_m$ and $g \leftarrow \mathcal{G}_n$, and then output $f \circ g$.*

The following proposition is crucial for the analysis of our final hash construction.

▶ **Proposition 3.7.** *The composition $\mathcal{H}'\circ\mathcal{H}$ of an ε_1-almost universal hash function $\mathcal{H}'$ and an ε_2-almost universal hash function $\mathcal{H}$ is an $(\varepsilon_2(n) + \varepsilon_1(m(n)))$-almost universal hash function, where $m(n)$ is the output length of $\mathcal{H}$. Moreover, $\mathcal{H}'\circ\mathcal{H}$ is linear if both of $\mathcal{H}'$ and $\mathcal{H}$ are linear.*

The following is a simple corollary of Proposition 3.7.

▶ **Corollary 3.8.** *For any $t \geq 2$, non-increasing $\varepsilon = \varepsilon(n)$, and output length parameter $\ell(n) \leq n$, the t^{th} order composition of an ℓ-output ε-almost universal hash $\mathcal{H}$ with itself, denoted by $\mathcal{H}^{\circ t}$, is also a hash function with collision probability $t \cdot \varepsilon(\hat{\ell})$, where $\hat{\ell} = \ell \circ \ell \circ \cdots \circ \ell(n)$ ($t-1$ times) is the input of the outer-most hash. Moreover, $\mathcal{H}^{\circ t}$ is linear if $\mathcal{H}$ is linear.*

We are now ready to specify our final hash construction $\mathcal{H}^{\mathsf{final}}$ with $\mathsf{polylog}(n)$ output length.

▶ **Construction 5** (Construction of $\mathcal{H}^{\mathsf{final}}$ with $\mathsf{polylog}(n)$ output length)**.** *Let $\beta \in (0, 2]$ be a parameter and $\omega(\log n) \leq \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. We define*

$$\mathcal{H}^{\mathsf{final}}_{\beta,\ell} \triangleq \mathcal{H}^{\mathsf{expw}}_{\ell} \circ \left(\mathcal{H}^{\mathsf{md}}_{\beta,\ell}\right)^{\circ \lceil \frac{2}{\beta} \rceil}.$$

We analyze the properties of $\mathcal{H}^{\mathsf{final}}$ constructed above by combining the composition proposition (Proposition 3.7 and Corollary 3.8) together with properties of $\mathcal{H}^{\mathsf{expw}}$ (Lemma 3.5) and $\mathcal{H}^{\mathsf{md}}$ (Theorem 3.4).

▶ **Theorem 3.9** (Properties of the final hash construction $\mathcal{H}^{\mathsf{final}}$)**.** *Let $\beta \in (0, 2]$ be a constant and $\omega(\log n) \leq \ell(n) \leq O(\frac{\log^2 n}{\log \log n})$ be a non-decreasing function. The followings hold for $\mathcal{H}^{\mathsf{final}}_{\beta,\ell} = \{\mathcal{H}_n\}_{n \geq 1}$:*

1. *$\mathcal{H}^{\mathsf{final}}_{\beta,\ell}$ is a linear $O(\ell \log^{\beta+1} n)$-samplable $\Theta(\hat{\ell})$-output $\exp(-\Omega(\hat{\ell}))$-almost universal hash function, where $\hat{\ell} = \ell(\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n))$.*
2. *Every $H \in \mathcal{H}_n$ can be computed by depth-$(1 + \lceil \frac{2}{\beta} \rceil)$ $\mathsf{CC}^0[2]$ circuits of wire complexity $2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right)$, or by B_2 circuits of size $2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right)$ and depth $\log n + O(1)$.*

Proof. Let $\mathcal{H}^{\mathsf{md}} = \mathcal{H}^{\mathsf{md}}_{\beta,\ell}$ and $\mathcal{H}^{\mathsf{expw}} = \mathcal{H}^{\mathsf{expw}}_{\ell}$ be the hash functions described in Construction 5. We also use $\mathcal{H}^{\mathsf{md}}_n$ and $\mathcal{H}^{\mathsf{expw}}_n$ to denote the distributions over n-input hash functions in $\mathcal{H}^{\mathsf{md}}$ and $\mathcal{H}^{\mathsf{expw}}$, respectively.

$\mathcal{H}^{\mathsf{final}}$ is almost universal with $\mathsf{polylog}(n)$ output length. Let $\hat{m}$ be the output length of $(\mathcal{H}^{\mathsf{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$, and $\hat{\ell} \triangleq \ell(\hat{m}) = \ell(\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n))$. By a simple induction and Theorem 3.4 we know that for any $1 \leq d \leq \lceil \frac{2}{\beta} \rceil$, $(\mathcal{H}^{\mathsf{md}})^{\circ d}$ is a linear $O(\ell \log^{\beta+1} n)$-samplable $\Theta(n/\log^{\beta d} n)$-output $\exp(-\Omega(\hat{\ell}))$-almost universal hash function. In particular, when $d = \lceil \frac{2}{\beta} \rceil$, we know that $(\mathcal{H}^{\mathsf{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is a linear $O(\ell \log^{\beta+1} n)$-samplable $\Theta(n/\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n)$-output $\exp(-\Omega(\hat{\ell}))$-almost universal hash function.

Now apply Proposition 3.7 and Lemma 3.5, we know that $\mathcal{H}^{\mathsf{final}} = \mathcal{H}^{\mathsf{expw}} \circ (\mathcal{H}^{\mathsf{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is a linear $O(\ell \log^{\beta+1} n)$-samplable $\Theta(\hat{\ell})$-output $\exp(-\Omega(\hat{\ell}))$-almost universal hash function.

Complexity with $\mathsf{CC}^0[2]$ circuits. By Theorem 3.4, every $H \in \mathsf{supp}(\mathcal{H}^{\mathsf{md}}_n)$ can be computed by a depth-1 $\mathsf{CC}^0[2]$ circuit with wire complexity $2n$. Also, by Lemma 3.5, every $H \in \mathsf{supp}(\mathcal{H}^{\mathsf{expw}}_n)$ can be computed by a depth-1 $\mathsf{CC}^0[2]$ circuit with wire complexity $n\ell$. Since the output length of $(\mathcal{H}^{\mathsf{md}})^{\circ d}$ is $\Theta(n/\log^{\beta d})$, the total wire complexity of a hash function from the support of $\mathcal{H}^{\mathsf{expw}} \circ (\mathcal{H}^{\mathsf{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ can be bounded by

$$2n + O\left(\sum_{d=1}^{\lceil \frac{2}{\beta} \rceil - 1} \frac{n}{\log^{\beta d} n}\right) + \frac{n}{\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n} \cdot \hat{\ell} = 2n + O\left(\frac{n\hat{\ell}}{\log^2 n}\right).$$

The depth complexity can be verified straightforwardly.

Complexity with B_2 circuits. We only analyze the depth since the analysis of the size complexity is similar to the analysis above. By Theorem 3.4 and Lemma 3.5, every $H \in \mathsf{supp}(\mathcal{H}_n^{\mathsf{md}})$ can be computed by a B_2 circuit with depth $\beta \log\log n + O(1)$, and every $H \in \mathsf{supp}(\mathcal{H}_n^{\mathsf{expw}})$ can be computed by a B_2 circuit with depth $\log n + O(1)$. Therefore the total depth of a hash function from the support of $\mathcal{H}^{\mathsf{expw}} \circ (\mathcal{H}^{\mathsf{md}})^{\circ \lceil \frac{2}{\beta} \rceil}$ is at most

$$\sum_{d=0}^{\lceil \frac{2}{\beta} \rceil - 1} \left[\beta \log\log \left(\Theta \left(\frac{n}{\log^{\beta d} n} \right) \right) + O(1) \right] + \log \left(\Theta \left(\frac{n}{\log^{\beta \cdot \lceil \frac{2}{\beta} \rceil} n} \right) \right) = \log n + O(1). \quad \blacktriangleleft$$

3.4 Explicitness of our construction

Apart from the seed length, shrinkage and collision probability, the *explicitness* of the a hash function is also critical for many applications. That is, given a seed v (which is usually much shorter than the input, say $|v| = \mathsf{polylog}(n)$), whether we can obtain information about the hash function corresponding to the seed v efficiently. In this section we show that our hash constructions are indeed explicit in a strong sense, which is crucial for our application to hardness magnification.

▶ **Definition 3.10** (Locally explicit hash function)**.** *A family of* $\mathsf{polylog}(n)$*-samplable distributions over* m*-output linear functions (e.g., linear hash functions or 1-detectors)* $\mathcal{H} = \{\mathcal{H}_n\}_{n \geq 1}$ *is said to be* locally explicit *if each input bit only influences* $\mathsf{polylog}(n)$ *output bits, and there exists an algorithm* $\mathcal{A}(n, v, i)$ *running in* $\mathsf{polylog}(n)$ *time that returns the list of output bits influenced by the* i^{th} *input bit in the hash function corresponding to the seed* v*.*

Note that any linear function can be realized as $x \mapsto \mathbf{M}x$ for some transformation matrix $\mathbf{M}$ over $\mathbb{F}_2$. Clearly, a locally explicit linear hash has a sparse transformation matrix, and we can efficiently list all 1-entries in any column of it. By performing a sparse matrix multiplication, we can immediately obtain the following proposition.

▶ **Proposition 3.11.** *The composition of two locally explicit hash families is locally explicit.*

Now we verify that our constructions of the randomness-efficient low-complexity hash functions are indeed locally explicit.

▶ **Theorem 3.12.** *Let* $k = k(n) = n^{o(1)}$ *be the shrinkage parameter and* $\log n \leq r(n) \leq n/2$ *be the range. The 1-detector* $\mathcal{L}_{k,r}^{\mathsf{hg}}$ *in Construction 2 is locally explicit.*

Proof. Recall that the 1-detector in Construction 2 consists of two components: a depth-1 $\mathsf{CC}^0[2]$ circuit whose topology is determined by the high-girth graph from Theorem 2.12, and a random permutation of the input bits according to the k-wise almost independent permutation from Theorem 2.8. Given the seed v of the 1-detector (i.e., the seed for the permutation) and an index i. According to Theorem 2.8, we can obtain $\sigma_v(i)$ in $\mathsf{poly}(\log n, |v|)$ time, where σ_v is the permutation corresponding to the seed v. To find all the output bits influenced by the i^{th} input bit, we only need to find the two endpoints of the $\sigma_v(i)$-th edge in the high-girth graph, which can be done in $\mathsf{polylog}(n)$ time by Theorem 2.12. $\blacktriangleleft$

▶ **Theorem 3.13.** *Let* $\beta > 0$ *be the shrinkage parameter and* $\delta = \delta(n) \leq O(\frac{\log^2 n}{\log\log n})$ *be the error parameter. The hash family* $\mathcal{H}_{\beta,\delta}^{\mathsf{md}}$ *in Construction 3 is locally explicit.*

Proof. Recall that the hash function $\mathcal{H}$ in Construction 3 is obtained by the 1-detector $\mathcal{L}^{\mathsf{hg}}$ in Construction 2 and the reduction in Construction 1. The output of the hash $\mathcal{H}_{\beta,\delta}^{\mathsf{md}}$ consists of two parts: the output of the 1-detector $\mathcal{L}^{\mathsf{hg}}$, and several randomly sampled bits from the

input. Since $\mathcal{L}^{\mathsf{hg}}$ is locally explicit by Theorem 3.12, it is sufficient to show that the sampling procedure is locally explicit. More precisely, given a seed v of the sampling procedure and an index i, we need to compute efficiently all the indices j such that $w_j = i$ for the vector $(w_0, \ldots, w_{b-1})$ corresponding to the seed v. This can be done using the algorithm $\mathcal{A}^{\mathsf{samp}}$ from Lemma 3.1. ◀

▶ **Theorem 3.14.** *Let $\ell = \ell(n) = \mathsf{polylog}(n)$. The hash family $\mathcal{H}_\ell^{\mathsf{expw}}$ in Construction 4 is locally explicit.*

Proof. Let $S_n = \{w_0, \ldots, w_{t-1}\}$ with $t = \tilde{O}(n^2)$ is a 0.1-biased set from Theorem 2.4 and G_t is the expander in Theorem 2.10. The hash function in Construction 4 is defined as

$$h_v(x) \triangleq \langle w_{v_0}, x\rangle \,\|\, \langle w_{v_1}, x\rangle \,\|\, \ldots \,\|\, \left\langle w_{v_{\ell-1}}, x\right\rangle,$$

where $v = (v_0, \ldots, v_{\ell-1}) \in [t]^\ell$ is a random walk on G_t. Given the seed v (i.e., the seed of a random walk), we can produce $v_0, v_1, \ldots, v_{\ell-1}$ in $\mathsf{polylog}(n, \ell) = \mathsf{polylog}(n)$ time by Theorem 2.10. Then using the algorithm $\mathcal{A}$ in Theorem 2.4 we can check for each j whether the i^{th} bit of w_{v_j} is 1, which indicates whether the j^{th} output bit is influenced by the i^{th} input bit. Enumerating all $j \in [\ell]$ gives a required $\mathsf{polylog}(n)$ time algorithm listing all the outputs influenced by a particular input bit. ◀

Using Theorem 3.13, Theorem 3.14, and Proposition 3.11, we immediately obtain the explicitness of Construction 5.

▶ **Corollary 3.15.** *Let $\beta \in (0, 2]$ be a parameter and $\ell = \ell(n) \le O(\frac{\log^2 n}{\log\log n})$ be a non-decreasing function. The hash family $\mathcal{H}_{\beta,\ell}^{\mathsf{final}}$ in Construction 5 is locally explicit.*

3.5 Uniformity of our construction

We have shown in Theorem 3.9 that our hash function can be computed by extremely sparse $\mathsf{CC}^0[2]$ circuit or B_2 circuits with small size and depth simultaneously. It remains to clarify the *uniformity* of our hash construction, i.e., the complexity we need to construct the circuit given the input length and the seed.

The definitions of uniformity vary with respect to the complexity measures, which can be chosen according to the applications. Typical choices include space complexity (e.g., LOGSPACE-uniform, see Section 6.2.1 of [3]), time complexity (e.g., polynomial-time uniform), parallel time (see, e.g., Section 3 of [29]), and descriptive complexity (see, e.g., [20]). If we choose the time complexity, for instance, we can define P-uniformity as the existence of the polynomial-time algorithm that prints the circuit given 1^n and the seed v. Indeed, it is easy to check that the $2n + o(n)$ size B_2 and $\mathsf{CC}^0[2]$ circuits for Construction 5 in Theorem 3.9 is P-uniform.

Since our hash function can be evaluated in low-depth circuit models such as NC^1 and $\mathsf{CC}^0[2]$, the parallel time to generate the circuit also seems to be crucial for its further applications. Therefore in this section we will discuss the POLYLOGTIME-uniformity of our hash functions, defined as follows.

▶ **Definition 3.16.** *Let $\mathcal{C}$ be a circuit class. A family of $\mathsf{polylog}(n)$-samplable distributions over m-output functions (e.g., hash functions or 1-detectors) $\mathcal{H} = \{\mathcal{H}_n\}_{n\ge1}$ is said to be computable with POLYLOGTIME-uniform $S(n)$-size (measured in size, depth, wire complexity, etc) $\mathcal{C}$-circuits if each of the functions $h_v \in \mathsf{supp}(\mathcal{H}_n)$ corresponding to the seed v is computable by an $S(n)$-size $\mathcal{C}$-circuit, supplemented with the following $\mathsf{polylog}(S(n))$-time algorithms:*

SIZE(n, v) *returns the size of the* $\mathcal{C}$*-circuit* C_v *computing* h_v*;*

TYPE(n, v, i) *returns the type of the* i^{th} *node in the circuit* C_v*, which indicates (1) whether it is a gate, an input variable or a constant; (2) whether it is an output node; and (3) the type (if it is a gate or a constant) or index (if it is an input variable and/or output node) of it;*

EDGE(n, v, i, j) *returns the* j^{th} *input of the* i^{th} *node if the* i^{th} *node is a gate in* C_v*.*

OUT(n, v, i) *returns the number of the* i^{th} *output node.*

It is guaranteed that the nodes are numbered in topological orders, and in particular, the input variables are numbered from 1 *to* n*.*

We claim that all our results of circuit complexity for the constructions in the previous parts of this section are in fact POLYLOGTIME-uniform. It is straightforward to check the claim, so we only sketch the proof and left the details to the readers.

▶ **Theorem 3.17** (Uniformity of Theorem 3.9)**.** *Let* $\beta \in (0, 2]$ *be a parameter and* $\ell = \ell(n) \leq O(\frac{\log^2 n}{\log\log n})$ *be a non-decreasing function. The hash function* $\mathcal{H}^{\mathsf{final}}_{\beta,\ell}$ *in Construction 5 can be computed either by* POLYLOGTIME*-uniform depth-*$(1+\lceil\frac{2}{\beta}\rceil)$ $\mathsf{CC}^0[2]$ *circuits of wire complexity* $2n + O(n/\log\log n)$*, or by* POLYLOGTIME*-uniform* B_2 *circuits of size* $2n + O(n/\log\log n)$ *and depth* $\log n + O(\beta^{-1})$ *simultaneously.*

Sketch. Since the POLYLOGTIME-uniformity is close under composition, we only need to verify that Construction 3 and Construction 4 are POLYLOGTIME-uniform. The uniformity of Construction 3 directly follows from the algorithm $\mathcal{B}$ in Theorem 2.12, the algorithm $\mathcal{A}$ in Theorem 2.8, and the algorithm $\mathcal{A}$ in Lemma 3.1. The uniformity of Construction 4 follows from the algorithm $\mathcal{A}$ in Theorem 2.4 and the algorithm $\mathcal{A}$ in Theorem 2.10. Note that the circuit for Construction 4 should be slightly modified to make $\mathsf{EDGE}(n, \cdot, \cdot)$ $\mathsf{polylog}(n)$-time computable. Take the $\mathsf{CC}^0[2]$ case for example: each of the output gates has fan-in *exactly* n (instead of $|w_{v_i}|$ for the i^{th} output bit, see Construction 4 for the notation); the j^{th} input wire of the i^{th} output gate is connected to the j^{th} input or a constant 0 according to the j^{th} bit of w_{v_i}. ◀

4 Sharp bootstrapping results from hash functions

We now show the extremely sharp bootstrapping results for small linear size circuits based on the almost universal hash function constructed above, using a refined *kernelization method* of Chen, Jin, and Williams [9, 10].

We will first prove the general hardness magnification theorem for all sparse NP languages in Section 4.1. In Section 4.2, we will present stronger hardness magnification results for MCSP, which will utilize the explicitness of our hash construction. Then we will show in Section 4.3 that similar techniques can be applied to obtain a bootstrapping result for *explicit obstructions* (see, e.g., [10]), which formalizes the *explicit* proofs of circuit lower bounds. In Section 4.4, we construct explicit obstructions and prove circuit lower bounds that tightly match these bootstrapping results.

4.1 Hardness magnification for all sparse NP languages

We first prove the most general version of the hardness magnification result. More discussions are presented after the proof.

▶ **Theorem 4.1.** *Let $s = s(n)$ and $T = T(n)$ be two functions such that*

- *$\omega(\log n) \le s(n) \le O(\log^2 n/\log\log n)$, $s(\Theta(n/\log^2 n)) = \Theta(s(n))$, s is non-decreasing;*
- *$n^\gamma \le T(n) \le 2^{O(n)}$, where $\gamma > 1$ is an absolute large constant.*

Then, if there is a $2^{s(n)}$-sparse language L in $\mathsf{NTIME}[T(n)]$, such that L cannot be computed by probabilistic circuits of size $2n + O(ns/\log^2 n)$ within error $\exp(-\Omega(s))$, it then follows that $\mathsf{NTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$ for all $c > 1$.

Proof. Towards a contradiction we assume that $\mathsf{NTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$ for some $c > 1$. Let L be a $2^{s(n)}$-sparse language in $\mathsf{NTIME}[T(n)]$. We now show that L can be computed by probabilistic circuits of size $2n + O(ns/\log^2 n)$ within error $\exp(-\Omega(s))$.

Let $\mathcal{H} = \mathcal{H}^{\mathsf{final}}_{\beta,\ell} = \{\mathcal{H}_n\}_{n\ge 1}$ be the linear $O(\ell \log^{1+\beta} n)$-samplable $\Theta(\ell)$-output $\exp(-\Omega(\ell))$-almost universal hash function in Theorem 3.9 with $\ell(n) = O(s)$ and $\beta = 2$, such that the collision probability is at most 2^{-2s}[17]. The circuit complexity of each $h \in \mathsf{supp}(\mathcal{H}_n)$ is bounded by $2n + O(ns/\log^2 n)$. Since $\mathcal{H}$ is efficiently samplable, there is a polynomial time algorithm $M(1^n, v, \cdot)$ computing the hash function corresponding to the seed v, where $|v| = O(\ell \log^3 n) = O(s \log^3 n)$.[18]

We now define an intermediate language $L' = \{(n, v, h) \mid \exists x \in \{0,1\}^n \cap L, M(1^n, v, x) = h\}$. For any sufficiently large n, we pad the language to make the corresponding length of the tuple in L' has length exactly $m = m(n) = \lfloor (\log(n)/(2c))^5 \rfloor$ to obtain L''. That is, for sufficiently large n and v, h of appropriate length, we define

$$z_{n,v,h} \triangleq (n, v, h) \| 1 \| 0^{m(n)-|(n,v,h)|-1}$$

and we have

$$z_{n,v,h} \in L'' \Leftrightarrow (n, v, h) \in L'.$$

Using the straightforward non-deterministic algorithm for L' (guessing $x \in \{0,1\}^n$ and determine whether $M(1^n, v, x) = h$), we can show that $L'' \in \mathsf{NTIME}\left[T\left(2^{O(m^{1/5})}\right)\right] \subseteq \mathsf{SIZE}\left[2^{cm^{1/5}}\right]$, the second containment follows from our assumption.

The key in our argument is an algorithm for the sparse language L with an oracle access to this intermediate problem L'', and then replace the oracle with the small size circuit by assumption. This is formalized in the following lemma.

▶ **Lemma 4.2.** *There is a uniform family $\mathcal{D} = \{\mathcal{D}_n\}_{n\ge 1}$ of probabilistic oracle circuits of size $2n + O(ns/\log^2 n)$, such that for every input length n, the followings hold:*

1. *Every $D \in \mathsf{supp}(\mathcal{D}_n)$ contains at most one L'' oracle gate of fan-in $\lfloor \log^5(n)/(32c^5) \rfloor$.*
2. *$\mathcal{D}$ decides the language L with error at most 2^{-s}.*

Proof. Consider the following probabilistic circuit family $\mathcal{D} = \{\mathcal{D}_n\}_{n\ge 1}$. Given input x of length n, $\mathcal{D}_n$ is constructed as follows: we randomly choose a seed v to sample a hash function h_v from $\mathcal{H}_n$; and then query whether $z_{n,v,h_v(x)}$ is in L'' via the oracle gate, where h_v is the hash function corresponding to the seed v. The circuit complexity of $\mathcal{D}_n$ equals the circuit complexity of the hash function h_v, which is at most $2n + O(ns/\log^2 n)$ by

[17] In particular, assume that α is a constant such that $\mathcal{H}$ is $\exp(-\alpha\ell)$-almost universal hash function, then we take $\ell(n) = \lceil 2s/\alpha \rceil$.

[18] That is, for every $x \in \{0,1\}^n$ and every seed v, we have $M(1^n, v, x) = h_v(x)$, where h_v is the hash function corresponding to the seed v.

Theorem 3.9, and every $D \in \mathsf{supp}(\mathcal{D}_n)$ only needs to call the L'' oracle once with input length $m = \lfloor \log^5(n)/(32c^5) \rfloor$. Clearly, by the definition of L'', for any $x \in L$, $\mathcal{D}_n$ accepts x with probability 1. For any $x \notin L$, we have

$$\begin{aligned}
&\Pr_v[\mathcal{D}_n \text{ accepts } x] \\
=&\Pr_{h \leftarrow \mathcal{H}_n}[\exists x' \in L \cap \{0,1\}^n \text{ s.t. } h(x) = h(x')] \\
\leq&\sum_{x' \in L \cap \{0,1\}^n} \Pr_{h \leftarrow \mathcal{H}_n}[h(x) = h(x')] \\
=&2^s \cdot 2^{-2s} \\
\leq&2^{-s}.
\end{aligned}$$

◀

Now applying Lemma 4.2 and replacing the oracle query by circuits using the fact that $L'' \in \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$, we finish the proof of Theorem 4.1. ◀

We now discuss some typical choices of parameters in the above theorem. Let the sparsity parameter $s(n) = \frac{\log^2 n}{\log\log n}$.[19]

- A standard form of hardness magnification result similar to the one in [9] but quantitatively stronger can be obtained by choosing $T(n) = \mathsf{poly}(n)$. Then the theorem says that, if there exists an $n^{\log n/\log\log n}$-sparse language in NP that does not have probabilistic circuits of size $2n + O(n/\log\log n)$, then $\mathsf{NTIME}\left[2^{O\left(n^{1/5}\right)}\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$ for all $c > 0$. By a padding argument, this implies that $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^c]$ for all $c > 0$. This establishes Theorem 1.2.
- Being less ambitious, we can let $T(n) = 2^{n^{o(1)}}$ and show that, even the existence of an $n^{\log n/\log\log n}$-sparse language in $\mathsf{NTIME}\left[2^{n^{o(1)}}\right]$ that does not have probabilistic circuits of size $2n + O(n/\log\log n)$, would be enough to imply that $\mathsf{NTIME}\left[2^{2^{o\left(n^{1/5}\right)}}\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$, and hence $\mathsf{NTIME}\left[2^{n^{o(1)}}\right] \not\subseteq \mathsf{SIZE}[n^c]$ for all $c > 0$ and $\mathsf{NEXP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$, which would already be a major breakthrough in circuit complexity. This establishes Theorem 1.3.

4.2 Hardness magnification for MCSP

We now utilized the strongly explicit hash function constructed above to obtain a stronger hardness magnification theorem for MCSP. Intuitively, this is possible since the yes-instances of MCSP can be efficiently encoded in a much shorter string. Indeed, for $\mathsf{MCSP}[s(n)]$, one only need $O(s(n)\log s(n))$ bits to encode a small circuit, instead of 2^n bits for a whole truth table.

We assume a paddable encoding of circuits, i.e., any string x and $x\|0$ encode the same circuit. This can be done with only a constant overhead. For a circuit C, we use the notation $\langle C \rangle$ to denote its encoding.

▶ **Theorem 4.3.** *Let $n \leq s(n) \leq O(n^2/\log^2 n)$ be a non-decreasing size parameter that satisfies $s(\log(\Theta(n/\log^2 n))) = \Theta(s(\log n))$. Let $N = 2^n$ be the truth table length. Let $g = g(n) = s(n)\log s(n)$. If $\mathsf{MCSP}[s(n)]$ cannot be computed by probabilistic circuits of size $2N + O(Ng/\log^2 N)$ within error $\exp(-\Omega(g))$, then there is some $c \in (0,1)$ such that $\oplus\mathsf{P} \not\subseteq \mathsf{SIZE}\left[2^{N^c}\right]$.*

[19] For a typical $2^{s(n)}$-sparse language in NP, consider $\mathsf{MCSP}[n^{1.9}]$ on input length $N = 2^n$.

Intuition. The proof is similar to the one for Theorem 4.1, but we need to make the oracle from Lemma 3.1 computable in $\oplus\mathsf{P}$ instead of in super-polynomial non-deterministic time. Concretely, we need to slightly modify the intermediate problem so that we can make use of the strongly explicitness of our hash construction $\mathcal{H}^{\mathsf{final}}$.

Proof. Towards a contradiction, suppose that $\oplus\mathsf{P} \subseteq \mathsf{SIZE}\left[2^{N^c}\right]$ for all $c \in (0,1)$, we now prove that $\mathsf{MCSP}[s(n)]$ can be computed by probabilistic circuits of size $2N + O(Ng/\log^2 N)$ within error $2^{-\Omega(g(n))}$.

Let $N = 2^n$ be the truth table length of n-input functions. Take $\mathcal{H} = \mathcal{H}^{\mathsf{final}}_{\beta,\ell} = \{\mathcal{H}_N\}_{N\geq 1}$ to be the linear $O(\ell \log^{1+\beta} N)$-samplable $\Theta(\ell)$-output $\exp(-\Omega(\ell))$-almost universal hash function in Theorem 3.9 with $\ell(N) = O(g(n)) = O(g(\log N))$ and $\beta = 2$, such that the collision probability is at most 2^{-2g}.[20]

We denote the hash function of input length N corresponding to the seed v as $H_{N,v}(\cdot)$. By the strongly explicitness (Corollary 3.15), there exists a algorithm $M(N, v, i, j)$ running in $\mathsf{polylog}(N) = \mathsf{poly}(n)$ time that decides whether the i^{th} output bit depends on the j^{th} input bit in $H_{N,v}$. Note that for a circuit C of input length n,[21]

$$H_{N,v}(\mathsf{tt}(C))_i = \sum_{j=0}^{N-1} M(N, v, i, j) \cdot C(j) \pmod 2,$$

and $M(N, v, i, j) \cdot C(j)$ can be computed in $\mathsf{poly}(n, |\langle C\rangle|)$ time, so the language $L^h = \{(N, i, v, \langle C\rangle) \mid H_{N,v}(\mathsf{tt}(C))_i = 1\}$ is decidable in $\oplus\mathsf{P}$. Hence if we define an intermediate language $L' = \{(N, v, h) \mid \exists C \in \mathsf{SIZE}[s(n)], H_{N,v}(\mathsf{tt}(C)) = h\}$ similar to the proof of Theorem 4.1, it is now decidable in NP given oracle access to $\oplus\mathsf{P}$.

Let $m = O(\ell \log^3 n)$ be the input length of L'. Note that $L' \in \mathsf{NP}^{\oplus\mathsf{P}} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}} \subseteq \mathsf{P}^{\oplus\mathsf{P}}_{/\mathsf{poly}} \subseteq \oplus\mathsf{P}_{/\mathsf{poly}}$, where $\mathsf{NP}^{\oplus\mathsf{P}} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}}$ follows from [42] (see also [15]) and $\mathsf{P}^{\oplus\mathsf{P}}_{/\mathsf{poly}} \subseteq \oplus\mathsf{P}_{/\mathsf{poly}}$ follows from $\oplus\mathsf{P}^{\oplus\mathsf{P}} \subseteq \oplus\mathsf{P}$ [37][22] Since $\oplus\mathsf{P} \subseteq \left[2^{N^c}\right]$ for all $c \in (0,1)$ by the assumption, we know that the evaluation of $\oplus\mathsf{P}_{/\mathsf{poly}}$ circuits can be computable in $\mathsf{SIZE}\left[2^{N^c}\right]$ for all $c \in (0,1)$, which implies that $L' \in \mathsf{SIZE}\left[\sqrt{N}\right]$. The rest of the proof follows similar to Theorem 4.1 and the fact that the $m = O(\ell \log^3 N) = o(\log^5 N)$. ◀

4.3 Explicit obstruction

We now formally state and prove our results regarding explicit obstructions. We begin by formally defining the notion of *explicit obstructions*.

▶ **Definition 4.4** (Explicit obstruction). *An* explicit obstruction of size $S(n)$ computable in $\mathcal{C}$ against $\mathcal{D}$ *is a family of lists of input-output pairs* $\mathcal{O} = \{\mathcal{O}_n\}_{n\geq 1}$ *satisfying the following.*

- $|\mathcal{O}_n| \leq S(n)$ *for all sufficiently large* n.
- *There is a machine in* $\mathcal{C}$ *that prints the set* $\mathcal{O}_n$ *given input* 1^n.
- *For every* n, $\mathcal{O}_n = \{(x_i, y_i)\}$ *satisfies that* $x_i \neq x_j$ *for all* $i \neq j$.
- *For all sufficiently large* n, *and for every* n*-input* $\mathcal{D}$ *circuit* f, *there is a pair* $(x_i, y_i) \in \mathcal{O}_n$ *such that* $f(x_i) \neq y_i$.

[20] Note that since $s(n) \leq O(n^2/\log^2 n)$, we have $\ell(N) = O(\log^2 N/\log\log N)$ and ℓ is non-decreasing. Hence, ℓ satisfies the requirements in Theorem 3.9.

[21] Here we use the notation $\mathsf{tt}(C)$ to represent the truth table of C. Particularly, if C is a single-output circuit of n inputs, then $\mathsf{tt}(C)$ is a string of length 2^n such that $\mathsf{tt}(C)_i = C(i)$.

[22] Since $\oplus\mathsf{P}^{\oplus\mathsf{P}} \subseteq \oplus\mathsf{P}$ implies that $\mathsf{P}^{\oplus\mathsf{P}} \subseteq \oplus\mathsf{P}$, it follows that the evaluation of polynomial-size circuits with $\oplus\mathsf{P}$ oracles can be computable in $\oplus\mathsf{P}$, which further implies that $\mathsf{P}^{\oplus\mathsf{P}}_{/\mathsf{poly}} \subseteq \oplus\mathsf{P}_{/\mathsf{poly}}$.

▶ **Theorem 4.5.** *There is an absolute constant $\gamma > 0$ such that the following holds. Let $n^\gamma \le T(n) \le 2^n$ and $\log n \le s(n) \le \min\{O(\log^2 n/\log\log n), \log T(n)\}$ being non-decreasing and satisfying $s(\Theta(n/\log^2 n)) = \Theta(s(n))$. If there is an explicit obstruction of size $2^{s(n)}$ computable in $\mathsf{DTIME}[T(n)]$ against $2n + O(n/\log\log n)$-size circuits, then $\mathsf{DTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \not\subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$ for all $c > 1$.*

Proof. Let $t(n) = 2^{s(n)}$. Towards a contradiction we assume that there is some constant $c > 1$, such that $\mathsf{DTIME}\left[T\left(2^{O(n^{1/5})}\right)\right] \subseteq \mathsf{SIZE}\left[2^{cn^{1/5}}\right]$. Suppose that $\mathcal{O} = \{\mathcal{O}_n\}_{n\ge1}$ is an explicit obstruction against $2n + O(n/\log\log n)$ size circuits. For any sufficiently large input length n, we suppose that $\mathcal{O}_n = \{(x_{n,1}, y_{n,1}), (x_{n,2}, y_{n,2}), \ldots, (x_{n,t(n)}, y_{n,t(n)})\}$. Our goal is to design a circuit with extremely small size, but agree with $\mathcal{O}_n$ on all its input-output pairs. Following the similar proof outline as for Theorem 4.1, we begin by using an almost universal hash function to kernalize the inputs from $\mathcal{O}_n$.

Let $\mathcal{H} = \mathcal{H}^{\mathsf{final}}_{\ell,\beta} = \{\mathcal{H}_n\}_{n\ge1}$ be the linear $O(\ell \log^{1+\beta} n)$-samplable $\Theta(\ell)$-output $\exp(-\Omega(\ell))$-almost universal hash function in Theorem 3.9 with $\ell(n) = O(s)$ and $\beta = 2$ such that the collision probability is at most 2^{-3s}. Let n be a sufficiently large input length. We call a hash function h *good* if it is perfect on the inputs in $\mathcal{O}_n$, i.e., any two distinct inputs in $\mathcal{O}_n$ have different hash values assigned by h. For a randomly chosen function $h \leftarrow \mathcal{H}_n$, the probability of it not being good is bounded by

$$\begin{aligned}
&\Pr_{h\leftarrow\mathcal{H}_n}[\exists 1 \le i < j \le t(n) \text{ s.t. } h(x_{n,i}) = h(x_{n,j})] \\
\le &\sum_{1\le i<j\le t(n)} \Pr_{h\leftarrow\mathcal{H}_n}[h(x_{n,i}) = h(x_{n,j})] \\
\le &\binom{2^s}{2} 2^{-3s} \le 2^{-s}.
\end{aligned}$$

So a good hash function always exists in the support of $\mathcal{H}_n$ for all sufficiently large n. For any large input length n, we arbitrarily fix such a good hash and denote its seed by v_n^{good}.

Let h_v be the hash function corresponding to the seed v. We define an intermediate language $L' = \{(n, v, h) \mid \exists 1 \le i \le t(n), y_{n,i} = 1 \wedge h_v(x_{n,i}) = h\}$. We again pad its input to have length exactly $m = \lfloor(\log(n)/(2c))^5\rfloor$, and form a padded language L''. Then $L'' \in \mathsf{DTIME}\left[T\left(2^{O(m^{1/5})}\right)\right] \subseteq \mathsf{SIZE}\left[2^{cm^{1/5}}\right]$. We also note that $2^{cm^{1/5}} \le \sqrt{n}$.

Then we consider the function $f_n(x) \triangleq L'(n, v_n^{\mathrm{good}}, H_n(v_n^{\mathrm{good}}, x))$, where $H_n(v, \cdot)$ is the hash function in $\mathcal{H}_n$ corresponding to the seed v. By an argument similar to that of Theorem 4.1, we can show that f_n can be decided by a circuit of size $2n + O(n/\log\log n) + \sqrt{n} = 2n + O(n/\log\log n)$, but totally agrees with $\mathcal{O}_n$, contradicting to the assumption that $\mathcal{O}$ is an explicit obstruction against circuits of such size. ◀

4.4 Unconditional lower bounds for sparse languages

Now we complement the results mentioned in previous subsections by constructing an explicit obstruction against B_2 circuits of size $2n - O(1)$ and a corresponding sparse language in P with a $2n - O(1)$ probabilistic circuit lower bound. They form sharp bootstrapping thresholds together with Theorem 4.1 and Theorem 4.5.

The main idea behind our explicit obstruction is the investigation of a combinatorial structure in Boolean circuits called *critical path* introduced in [13], which was used to prove $2n - O(1)$ circuit lower bounds for PRFs and hash functions.

For the simplicity of presentation, we assume without loss of generality that our circuits are *normalized*, in the sense that there is no non-output gates with out-degree 0. This is without loss of generality since redundant gates with out-degree 0 can be removed.

▶ **Definition 4.6** (Critical path). *Let C be a circuit, and u be an input variable of it. The* critical path of u in C *is a sequence of vertices $v_0, v_1, \ldots, v_k$ satisfying the following conditions:*

1. *$v_0 = u$, and v_i is a descendent of v_{i-1} for all $i \geq 1$, and*
2. *out-degree$(v_i) = 1$ for all $0 \leq i < k$, and out-degree$(v_k) \neq 1$.*

Fix a circuit with n input bits, we can obtain a total of n critical paths. The crucial observation in [13] is that, if all critical paths do not intersect with one another, then the circuit must contain at least $2n - O(1)$ gates. This is formalized below.

▶ **Lemma 4.7** ([13], Lemma 6.4). *For any normalized n-input single-output circuit C with no intersecting critical paths and no input variables with out-degree 0, the number of gates in the circuit is at least $2n - 2$.*

Let $\mathcal{O} = \{\mathcal{O}_n \subseteq \{0,1\}^n \times \{0,1\}\}_{n \geq 1}$, where

$$\mathcal{O}_n = \{(x, 0) \mid |x| \in \{0, 2, n-2, n-1\}\} \cup \{(x, 1) \mid |x| \in \{1, n\}\}.$$

In the remaining part of the section, we will first prove that $\mathcal{O}$ is an explicit obstruction against $2n-2$ size circuits, and then present a general connection between explicit obstruction and probabilistic circuit lower bounds.

Explicit obstruction. According to Lemma 4.7, we only need to prove that any circuit with intersecting critical paths or input variables with out-degree 0 does not fully agree with $\mathcal{O}$. For circuits with input variables of out-degree 0, this can be verified straightforwardly. Now we prove the case for circuits with intersecting critical paths.

▶ **Lemma 4.8.** *For any circuit $C_n : \{0,1\}^n \to \{0,1\}$ with intersecting critical paths, there must exist a pair $(x, b) \in \mathcal{O}_n$ such that $C_n(x) \neq b$.*

Proof. Suppose that the critical paths of u and v in C_n intersect. Let G be the first gate on the intersection. Let f_G be the function computed by G. Assume that we take a restriction ρ to all variables except u and v, and consider the restricted function $C_n\restriction_\rho$. The key observation is that, for any ρ, there are functions $\phi_\rho, \psi_\rho, \chi_\rho \colon \{0,1\} \to \{0,1\}$ such that

$$C_n\restriction_\rho(u, v) = \chi_\rho(f_G(\phi_\rho(u), \psi_\rho(v))).$$

Based on the characterization of different functions in B_2, we call a function f *quadratic* if it has the form $f(u, v) = ((u \oplus c_1) \wedge (v \oplus c_2)) \oplus c_3$. We call a function *linear* if it has the form $f(u, v) = u \oplus v \oplus c$. The pivotal point in [13] is that for a fixed pair of u and v, $C_n\restriction_\rho$ cannot be quadratic under some restriction ρ_1, then become linear under another restriction ρ_2. However, by checking the truth table it can be verified that our construction of $\mathcal{O}_n$ forces $C_n\restriction_\rho$ to be the linear function $(u, v) \mapsto u \oplus v$ for all-zero restriction, and the quadratic function $(u, v) \mapsto u \wedge v$ for all-one restriction, which leads to contradiction. ◀

▶ **Corollary 4.9.** *$\mathcal{O}$ is an explicit obstruction against $2n - 2$ size B_2 circuits.*

Probabilistic circuit lower bound. We now present a general reduction from explicit obstruction to probabilistic circuit lower bounds for sparse languages in P.

▶ **Lemma 4.10.** *For any circuit class $\mathcal{C}$, if there exists an explicit obstruction $\mathcal{O}$ of size $S(n)$ against $\mathcal{C}$, then every language agreeing with $\mathcal{O}$ cannot be computed by probabilistic $\mathcal{C}$-circuits with error probability less than $\frac{1}{S(n)}$, even infinitely often.*

Proof. Towards contradiction, let L be a language agreeing with $\mathcal{O}$, and assume that there exists a probabilistic $\mathcal{C}$-circuit C computing L with error probability smaller than $\frac{1}{S(n)}$, for infinitely many n. Then for those n, we have

$$\mathop{\mathbb{E}}_{C}[|\{(x,b) \in \mathcal{O}_n \mid C(x) \neq b|\}] < 1.$$

By the averaging argument, there exists a deterministic circuit that agrees with all pairs in $\mathcal{O}_n$. This guarantees the existence of a family of circuit in $\mathcal{C}$ that agrees with $\mathcal{O}$ on those n, which leads to a contradiction. ◀

In particular, let $\mathcal{L} = \{\mathcal{L}_n\}_{n\geq 1}$ be a language such that $\mathcal{L}_n \triangleq \{x \in \{0,1\}^n \mid (x,1) \in \mathcal{O}_n\}$, then it is a sparse language in P that agrees with $\mathcal{O}$. Together with Corollary 4.9, we obtain the following lower bound.

▶ **Corollary 4.11.** *There exists a $O(n)$-sparse language in P that cannot be computed (even infinitely often) by probabilistic B_2 circuits of size $2n-2$ with error probability smaller than $\frac{1}{2n^2}$.*

▶ Remark 4.12. We note that the error probability in the lower bound cannot be trivially boosted to 1/3, since the complexity overhead of error reduction is not affordable when we are dealing with small linear-size circuits. Therefore it might be significantly more difficult to prove a similar circuit lower bound with constant error probability.

5 Low-complexity PRFs from hash functions

In this section, we will present the consequences of our hash constructions for low-complexity constructions of pseudorandom functions. We will discuss the Levin's trick for PRF construction in Section 5.1, and then present our constructions of PRFs for B_2 and low-depth circuits in Section 5.2 and 5.3, respectively.

5.1 PRF and Levin's trick

▶ **Definition 5.1** (Pseudorandom functions)**.** *Let $s = s(n)$, $m = m(n)$, and $\varepsilon = \varepsilon(n)$. An (s,ε)-secure m-output* pseudorandom function *(PRF) is a family $\mathcal{F} = \{\mathcal{F}_n\}_{n\geq 1}$ of distributions $\mathcal{F}_n$ over $B_{n,m}$ such that no probabilistic $s(n)$-time adversary could distinguish $f \leftarrow \mathcal{F}_n$ from a truly random function $g \leftarrow \mathcal{U}(B_{n,m})$ with advantage $\varepsilon(n)$ given oracle access to the functions, i.e.,*

$$\left| \Pr_{f\leftarrow\mathcal{F}_n}[\mathcal{A}^f(1^n) \text{ accepts}] - \Pr_{g\leftarrow\mathcal{U}(B_{n,m})}[\mathcal{A}^g(1^n) \text{ accepts}] \right| < \varepsilon.$$

for all probabilistic $s(n)$-time algorithm $\mathcal{A}$ and sufficiently large n.

A PRF is said to be *s-secure* if it is $(s, 1/s)$-secure. In particular, a PRF is said to be polynomially secure if it is n^k-secure for all constants $k \geq 1$, and it is said to be sub-exponentially secure if it is $\exp(n^\varepsilon)$-secure for some constant $\varepsilon \in (0,1)$. Similar to the hash functions, we define the key (seed) length, circuit complexity, composition, P-uniformity, and POLYLOGTIME-uniformity of PRFs.

Following [23, 13], the key to construct low-complexity PRFs is Levin's trick: the composition of a PRF and an almost universal hash function is still a PRF.

▶ **Lemma 5.2** (Levin's trick, see, e.g., [5, 13]). *Let $s = s(n)$, $m = m(n)$, $\varepsilon = \varepsilon(n)$, and $\delta = \delta(n)$. The composition $\mathcal{F} \circ \mathcal{H}$ of an (s, ε)-secure PRF $\mathcal{F}$ and a polynomial-time computable m-output δ-almost universal hash $\mathcal{H}$ is an $(\hat{s}, \hat{\varepsilon})$-secure PRF if $\hat{s}(n) \leq s(m(n)) - \mathsf{poly}(n)$ and $\hat{\varepsilon}(n) \geq \varepsilon(m(n)) + \hat{s}(n)^2 \cdot \delta(n)$ for sufficiently large n.*

Note that the circuit complexity of the resulting PRF mostly depends on the complexity of the hash function, since we can reduce the complexity of the "pseudorandom kernel" by reducing the length of the hash value. Indeed, we need to balance the complexity and security by tuning the output length of the hash function: it should be sufficiently small to reduce the complexity of the original PRF, while it has to be moderately large to guarantee the security of the constructed PRF.

5.2 Low-complexity PRFs in B_2 circuit

Now we discuss the low-complexity PRF construction in B_2 circuits. We first introduce the assumption for the security of our PRF construction.

▶ **Assumption 5.3.** *There exists a sub-exponentially secure polynomial-time computable pseudorandom function with polynomial key length.*

Note that by the celebrated works of Goldreich, Goldwasser, and Micali [16], the existence of sub-exponentially secure PRF is equivalent to the existence of sub-exponentially secure pseudorandom generator, which is further equivalent to the existence of sub-exponentially secure one-way function [19].

▶ **Theorem 5.4.** *Assuming Assumption 5.3, there exists an $\exp\left(\Omega\left(\frac{\log^2 n}{\log\log n}\right)\right)$-secure PRF with key length $\mathsf{polylog}(n)$ computable by POLYLOGTIME-uniform B_2 circuits of size $2n + O(n/\log\log n)$ and depth $\mathsf{polylog}(n)$ simultaneously*

Proof. Since any polynomial-time computable function can be computed by a P-uniform polynomial-size B_2 circuit, the existence of polynomial-time PRF implies the existence of PRF computable by P-uniform polynomial-size B_2 circuits. Under Assumption 5.3 we can obtain an $(\exp(n^\varepsilon), \exp(n^\varepsilon))$-secure PRF $\mathcal{F}$ computable by P-uniform polynomial-size B_2 circuits for an $\varepsilon \in (0,1)$. From Construction 5, Theorem 3.9, and Theorem 3.17 with $\ell = \frac{\log^2 n}{\log\log n}$ and $\beta = 1$, we can construct a $\mathsf{polylog}(n)$-samplable $\Theta(\ell)$-output $\exp(-\Omega(\ell))$-almost universal hash function $\mathcal{H}$ computable by POLYLOGTIME-uniform $2n + O(n/\log\log n)$ size B_2 circuits. We increase the output length of $\mathcal{H}$ to $m = \lceil \log^{2/\varepsilon}(n) \rceil$ to obtain $\mathcal{H}'$ by padding constant outputs. We now prove that $\mathcal{F} \circ \mathcal{H}'$ is the desired $\exp(-\Omega(\ell))$-secure PRF.

Recall that $\mathcal{H}'$ is an m-output $\exp(-c\ell)$-almost universal hash function for some constant $c \in (0,1)$. According to Lemma 5.2, we know that for $\hat{s}(n) \triangleq \exp(c\ell/4)$, $\mathcal{F} \circ \mathcal{H}'$ is an $\hat{s}$-secure PRF, since for sufficiently large n,

$$\exp(m^\varepsilon) - \mathsf{poly}(n) \geq \exp(\log^2 n) - \mathsf{poly}(n) \geq \hat{s},$$

$$\exp(-m^\varepsilon) + \hat{s}^2 \cdot \exp(-c\ell) \leq \exp(-\log^2 n) + \exp(-c\ell/2) \leq \frac{1}{\hat{s}}.$$

Because the key of $\mathcal{F} \circ \mathcal{H}'$ consists of the seed of $\mathcal{H}'$ (with input length n) and the key of $\mathcal{F}$ (with input length m), the key length would be $\mathsf{polylog}(n) + \mathsf{poly}(m) = \mathsf{polylog}(n)$.

It is straightforward to verify that $\mathcal{F} \circ \mathcal{H}'$ can be computed by B_2 circuits of size $2n + O(n/\log\log n)$ and depth $\mathsf{polylog}(n) + \mathsf{poly}(m) = \mathsf{polylog}(n)$, so it remains to show that the circuit computing it is $\mathsf{POLYLOGTIME}$-uniform. We only demonstrate the evaluation of function $\mathsf{EDGE}(n, v, i, j)$ (i.e., the j^{th} input node of the i^{th} node in the circuit computing function keyed by v) and remark that the evaluations of other functions (SIZE, TYPE, and OUT) can be done in a similar way.

Given a key $v = v_1 \| v_2$ where v_1 is the seed for $\mathcal{H}'$ and v_2 is the key for $\mathcal{F}$, we firstly decide whether the i^{th} node is inside the hash $\mathcal{H}'$ or the PRF $\mathcal{F}$ (which can be done since we can compute the number of nodes in $\mathcal{H}$ in $\mathsf{polylog}(n)$ time given v_1). In the former case, we can use the $\mathsf{EDGE}(n, v_1, \cdot, \cdot)$ function for the hash function $\mathcal{H}'$, since it is $\mathsf{POLYLOGTIME}$-uniform. In the latter case, we can draw the circuit computing $\mathcal{F}$ given key v_2 in $\mathsf{polylog}(n)$ time (since $\mathcal{F}$ is P-uniform, and the input length is $m = \mathsf{polylog}(n)$), and then find out the j^{th} input node of the i^{th} node. ◀

▶ Remark 5.5. It can be easily verified that if we do not require the PRF to be computable within $\mathsf{polylog}(n)$ depth, we can in fact rely on the following (possibly) weaker assumption: the existence of $\exp(n^{\varepsilon/4})$-time computable $\exp(n^{\varepsilon})$-secure PRFs with polynomial key length for an $\varepsilon > 0$[23]. By [16, 19] (implicitly), this assumption follows from the existence of $\exp(n^{\varepsilon/8})$-time computable OWFs against any $\exp(n^{\varepsilon})$-time adversary for an $\varepsilon > 0$. We stick to Assumption 5.3 since it has been a quite standard assumption.

5.3 Low-complexity PRFs in low-depth circuits

To construct efficient PRFs in low-depth circuit classes such as NC^1 and $\mathsf{AC}^0[2]$ using our low-complexity hash functions, we need to rely on the existence of low-depth PRFs. In particular, we will need the existence of (sub-exponentially secure) NC^1 PRFs to construct efficient NC^1 and $\mathsf{AC}^0[2]$ PRFs.

▶ **Assumption 5.6** (NC^1 PRF)**.** *There exists a sub-exponentially secure PRF with polynomial key length computable by* P*-uniform polynomial-size* NC^1 *circuits.*

Note that it is unknown whether such assumption can be reduced to more elementary ones such as the existence of certain kinds of one-way functions. Nevertheless, it follows from standard cryptographic assumptions such as sub-exponential decisional Diffie-Hellman [33] or sub-exponential Ring Learning-with-Error [4].

▶ **Theorem 5.7.** *There exists a* $\exp\left(\Omega\left(\frac{\log^2 n}{\log\log n}\right)\right)$*-secure PRF with* $\mathsf{polylog}(n)$ *key length computable by* $\mathsf{POLYLOGTIME}$*-uniform* B_2 *circuits of size* $2n + O(n/\log\log n)$ *and depth* $\log n + O(\log\log n)$ *simultaneously under Assumption 5.6.*

Proof. We will use the construction in the proof of Theorem 5.4 by replacing the PRF $\mathcal{F}$ with a PRF computable by P-uniform NC^1 circuits. We only check the circuit depth of the construction since other properties can be established similarly as in the proof of Theorem 5.4. By Theorem 3.17, the B_2 circuit computing the hash function $\mathcal{H}$ (and $\mathcal{H}'$) in

[23] The $\mathsf{POLYLOGTIME}$-uniformity follows from the folklore simulation of P algorithms by $\mathsf{POLYLOGTIME}$-uniform circuits (see, e.g., [29]).

the proof of Theorem 5.4 has depth $\log n + O(1)$. Since the output length of $\mathcal{H}'$ (i.e., the input length of the original PRF) is $\mathsf{polylog}(n)$, the depth of the original PRF would be $\log(\mathsf{polylog}(n)) = O(\log\log n)$. Therefore the total depth would be $\log n + O(\log\log n)$. ◀

To construct $\mathsf{AC}^0[2]$ PRFs from Assumption 5.6, we need a folklore reduction from logarithmic depth circuits to AC^0 circuits. In particular, we show how to transform a polynomial-size NC^1 circuit to an AC^0 circuit of size $2^{O(n^\varepsilon)}$ and depth $O(1/\varepsilon)$.

▶ **Lemma 5.8.** *Let f be a function computable by* P*-uniform* NC^1 *circuits. For any constant $\varepsilon > 0$, f is computable by* POLYLOGTIME*-uniform* AC^0 *circuits of $2^{O(n^\varepsilon)}$ size and $O(1/\varepsilon)$ depth.*

Proof. Suppose that f is computable by an NC^1 circuit C of depth $d\log n$. Let $k = d/\varepsilon$. We partition C into k layers of chunks, each of depth $\varepsilon\log n$. In such case, each chunk only depends on $O(2^{\varepsilon\log n}) = O(n^\varepsilon)$ number of gates, so we can expand it into a CNF of size $2^{O(n^\varepsilon)}$. After expanding all the chunks, we get an AC^0 circuit C' of size $2^{O(n^\varepsilon)}$ and depth $2k = O(1/\varepsilon)$.

Let $N = 2^{O(n^\varepsilon)}$ be the size of C'. To show the uniformity, we observe that to compute the local structure of a gate, we only need to evaluate one of the chunks on a given input, which can be done in $\mathsf{poly}(n) = \mathsf{polylog}(N)$ time. ◀

▶ **Theorem 5.9.** *There exists a* $\exp\left(\Omega\left(\frac{\log^2 n}{\log\log n}\right)\right)$*-secure PRF with* $\mathsf{polylog}(n)$ *key length computable by* POLYLOGTIME*-uniform* $\mathsf{AC}^0[2]$ *circuits with $2n + O(n/\log\log n)$ wires under Assumption 5.6.*

Proof. We will use the construction in the proof of Theorem 5.4 by replacing the PRF $\mathcal{F}$ with a PRF computable by P-uniform NC^1 circuits. Suppose that the parameter m in the proof of Theorem 5.4 is at most $\log^c n$. Initiating Lemma 5.8 with $\varepsilon = 1/(2c)$, we can compute the PRF $\mathcal{F}$ with an POLYLOGTIME-uniform AC^0 circuit with $2^{O(\sqrt{\log n})} = n^{o(1)}$ wires. It immediately follows that our PRF construction is computable by POLYLOGTIME-uniform $\mathsf{AC}^0[2]$ circuits with $2n + O(n/\log\log n)$ wires. The other parts are the same as the proof of Theorem 5.4. ◀

References

1. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 544–553. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89575.
2. Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 27–56, 2016. doi:10.1007/978-3-662-53641-4_2.
3. Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
4. 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*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012. doi:10.1007/978-3-642-29011-4_42.

5 Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography*, pages 79–158. Springer International Publishing, 2017. doi:10.1007/978-3-319-57048-8_3.

6 Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. doi:10.1016/0022-0000(79)90044-8.

7 L. Sunil Chandran. A high girth graph construction. *SIAM J. Discret. Math.*, 16(3):366–370, 2003. doi:10.1137/S0895480101387893.

8 Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.70.

9 Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1240–1255. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00077.

10 Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1335–1348. ACM, 2020. doi:10.1145/3357713.3384283.

11 Ruiwen Chen and Valentine Kabanets. Correlation bounds and #SAT algorithms for small linear-size circuits. *Theor. Comput. Sci.*, 654:2–10, 2016. doi:10.1016/j.tcs.2016.05.005.

12 Evgeny Demenkov and Alexander S. Kulikov. An elementary proof of a 3n - o(n) lower bound on the circuit complexity of affine dispersers. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 256–265. Springer, 2011. doi:10.1007/978-3-642-22993-0_25.

13 Zhiyuan Fan, Jiatu Li, and Tianqi Yang. The exact complexity of pseudorandom functions and the black-box natural proof barrier for bootstrapping results in computational complexity. *Electron. Colloquium Comput. Complex.*, page 125, 2021. To appear in STOC 2022. URL: https://eccc.weizmann.ac.il/report/2021/125.

14 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 *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. doi:10.1109/FOCS.2016.19.

15 Lance Fortnow. A simple proof of Toda's theorem. *Theory Comput.*, 5(1):135–140, 2009. doi:10.4086/toc.2009.v005a007.

16 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

17 Alexander Golovnev, Edward A. Hirsch, Alexander Knop, and Alexander S. Kulikov. On the limits of gate elimination. *J. Comput. Syst. Sci.*, 96:107–119, 2018. doi:10.1016/j.jcss.2018.04.005.

18 Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998. doi:10.1137/S0097539794261556.

19 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.

20 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. doi:10.1016/S0022-0000(02)00025-9.

21 Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

22 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019. doi:10.1145/3230630.

23 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442. ACM, 2008. doi:10.1145/1374376.1374438.

24 Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of 5n - o(n) for boolean circuits. In *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. doi:10.1007/3-540-45687-2_29.

25 Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k-wise (almost) independent permutations. *Algorithmica*, 55(1):113–133, 2009. doi:10.1007/s00453-008-9267-y.

26 Oded Lachish and Ran Raz. Explicit lower bound of *4.5n - o(n)* for boolena 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. doi:10.1145/380752.380832.

27 Felix Lazebnik and Vasiliy A. Ustimenko. Explicit construction of graphs with an arbitrary large girth and of large size. *Discret. Appl. Math.*, 60(1-3):275–284, 1995. doi:10.1016/0166-218X(94)00058-L.

28 Jiatu Li and Tianqi Yang. 3.1n - o(n) circuit lower bounds for explicit functions. *Electron. Colloqu ium Comput. Complex.*, page 23, 2021. To appear in STOC 2022. URL: https://eccc.weizmann.ac.il/report/2021/023.

29 Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Comput. Complex.*, 22(2):311–343, 2013. doi:10.1007/s00037-013-0069-5.

30 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1215–1225. ACM, 2019. doi:10.1145/3313276.3316396.

31 Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. *J. ACM*, 62(6):46:1–46:29, 2015. doi:10.1145/2792978.

32 Ketan Mulmuley. On P vs. NP and geometric complexity theory: Dedicated to sri ramakrishna. *J. ACM*, 58(2):5:1–5:26, 2011. doi:10.1145/1944345.1944346.

33 Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. doi:10.1145/972639.972643.

34 Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Theory Comput.*, 17:1–38, 2021.

35 Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 65–76, 2018. doi:10.1109/FOCS.2018.00016.

36 Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008. doi:10.1137/060658400.

37 Christos H. Papadimitriou and Stathis Zachos. Two remarks on the power of counting. In *Theoretical Computer Science, 6th GI-Conference, Dortmund, Germany, January 5-7, 1983, Proceedings*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276. Springer, 1983. doi:10.1007/BFb0009651.

38 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

39 Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 283–290. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21944.

40 Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996. doi:10.1109/18.556668.

41 Avishay Tal. Shrinkage of de Morgan formulae by spectral techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.65.

42 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.

43 Emanuele Viola. The communication complexity of addition. *Comb.*, 35(6):703–747, 2015. doi:10.1007/s00493-014-3078-3.

A Strongly explicit high-girth graphs

In this section, we will examine the construction of bipartite high-girth graphs of Lazebnik and Ustimenko [27], and verify that it is indeed strongly explicit, thereby proving Theorem 2.12.

Let q be an odd prime power. Let P and L be two infinite sequences of elements from $\mathbb{F}_q$ indexed as follows:

$$P = \langle p_1, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{3,2}, \dots, p_{i,i+1}, p_{i+1,i}, p_{i+1,i+1}, p'_{i+1,i+1}, p_{i+1,i+2}, \dots \rangle$$
$$L = \langle l_1, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{3,2}, \dots, l_{i,i+1}, l_{i+1,i}, l_{i+1,i+1}, l'_{i+1,i+1}, l_{i+1,i+2}, \dots \rangle .$$

The names P and L mean points and lines, respectively, due to certain geometric intuition of the construction. We say P is incident with L if they satisfy the following set of equations

$$E_1 = \begin{cases} l_{1,1} - p_{1,1} = l_1 p_1 \\ l_{1,2} - p_{1,2} = l_{1,1} p_1 \\ l_{2,1} - p_{2,1} = l_1 p_{1,1} \end{cases}, \quad E_i = \begin{cases} l_{i,i} - p_{i,i} = l_1 p_{i-1,i} \\ l'_{i,i} - p'_{i,i} = l_{i,i-1} p_1 \\ l_{i,i+1} - p_{i,i+1} = l_{i,i} p_1 \\ l_{i+1,i} - p_{i+1,i} = l_1 p'_{i,i} \end{cases} (\forall i \geq 2) \tag{3}$$

One can verify that for every point P, a line L incident with it is uniquely determined by l_1, since all other coordinates of L can be computed from the equations above iteratively. Similarly for every line L, a point P incident with it is uniquely determined by p_1.

Let $P_k \in \mathbb{F}_q^k$ be the length-k prefix of P, and $L_k \in \mathbb{F}_q^k$ be the length-k prefix of L. We define $D(k,q) = (V_1, V_2, E \subseteq V_1 \times V_2)$ to be the following bipartite graph:

$V_1 = V_2 = \mathbb{F}_q^k$;
$(u,v) \in E \iff \exists (P,L), P$ is incident with $L, u = P_k$, and $v = L_k$.

Equivalently, u and v are connected if and only if they satisfy the first k equations of $\cup_i E_i$ in (3). Note that $|V_1| = |V_2| = q^k$, $|E| = q^{k+1}$, and the graph is q-regular.

▶ **Theorem A.1** (Lazebnik and Ustimenko [27]). *Let $k \geq 3$ be an odd integer and q be an odd prime power. Then the girth of $D(k,q)$ is at least $k+5$.*

For our purpose, it is sufficient to consider $q = p^r$ for prime $p = O(1)$. To construct the graph explicitly, we need to evaluation field operations over $\mathbb{F}_q$ for a prime power q in $\mathsf{polylog}(q)$ time, for which we need a representation of $\mathbb{F}_q$. Recall that $\mathbb{F}_q$ is isomorphic to $\mathbb{F}_p[x]/(Q(x))$ for any irreducible degree-r polynomial $Q \in \mathbb{F}_p[x]$. So the explicit representation of $\mathbb{F}_q$ follows from the construction of irreducible polynomials by Shoup [39].

▶ **Theorem A.2** (Shoup [39]). *There is a deterministic algorithm that constructs a degree-n irreducible polynomial over $\mathbb{F}_p$ given an integer n and a prime power p in $\mathsf{poly}(n, p)$ time.*

Now we check that $D(k, q)$ is strongly explicit, in the sense that given the index i of an edge, we can obtain the indices j_1 and j_2 in $\mathsf{poly}(k, \log q)$ time such that the i^{th} edge connects the j_1^{th} vertex in V_1 and the j_2^{th} vertex in V_2. Let $q = p^r$ for a prime $p = O(1)$. We number the vertices and edges as follows.

1. We identify the elements in the finite field $\mathbb{F}_q$ as length-r vectors from $[p]^r$, and number all the elements in the lexicographic order.
2. We identify the vertices in both V_1 and V_2 (that are length k sequences of elements in $\mathbb{F}_q$) as length-kr vectors from $[p]^{kr}$, and number them in the lexicographic order.
3. For any $i \in [q^k]$ and $j \in [q]$, the $(iq + j)^{\text{th}}$ edge connects the i^{th} vertex P in V_1 and the unique vertex $L = (l_1, \dots)$ in V_2 connected to P such that l_1 is the j^{th} element in $\mathbb{F}_q$.

Under such a numbering scheme, given the index $iq + j$ of an edge with $i \in [q^k]$ and $j \in [q]$, we can easily determine its two endpoints in $\mathsf{poly}(k, \log q)$ time.

Now we are ready to prove Theorem 2.12.

▶ **Remainder of Theorem 2.12.** Let $r = r(n) = n^{o(1)}$ be a parameter. For every sufficiently large n, there exists an $m = \Theta(\frac{n}{r})$ and a regular graph $G_{m,n}$ with m vertices, n edges, and girth $\Omega(\frac{\log n}{\log r})$. Moreover, there exists a $\mathsf{polylog}(n)$-time algorithm $\mathcal{A}(n, i)$ for $i \in [n]$ that outputs the indices of the two endpoints of the i^{th} edge in $G_{m,n}$, and a $\mathsf{polylog}(n)$-time algorithm $\mathcal{B}(n, i, j)$ for $i \in [m]$ that outputs the j^{th} edge attaching to the i^{th} vertex.

Proof. Let q be the power of 3 in the interval $[r, 3r)$. Applying Theorem A.2, we construct a fixed representation of $\mathbb{F}_q$ in $\mathsf{polylog}(q)$ time so that the field operations over $\mathbb{F}_q$ can be evaluated in $\mathsf{polylog}(q)$ time. Let $k \triangleq \lfloor \log_q n \rfloor - 1$ (i.e., k is the largest integer such that $q^{k+1} \le n$), $\ell \triangleq \lceil n/q^{k+1} \rceil$, and $m \triangleq 2q^k \ell$. Note that

$$m \le 2q^k \left(\frac{n}{q^{k+1}} + 1 \right) = \frac{2n}{q} + \frac{2q^{k+1}}{q} \le O\left(\frac{n}{r}\right),$$
$$m \ge 2q^k \cdot \frac{n}{q^{k+1}} \ge \Omega\left(\frac{n}{r}\right),$$

and therefore $m = \Theta(n/r)$. Now we construct a graph $G_{m,n}$ with ℓ connected components as follows. Each of the first $\ell - 1$ connected components is a copy of $D(q, k)$ with $2q^k$ vertices and q^{k+1} edges. The last connected component is a subgraph of $D(q, k)$ with $2q^k$ vertices but only the first $n - q^{k+1}(\ell - 1)$ edges.

By Theorem A.1, it is easy to see that the girth of $G_{m,n}$ is at least $k = \Omega(\log n / \log q) = \Omega(\frac{\log n}{\log r})$. According to our numbering scheme and related discussions, it is also easy to verify that given an edge index i, we can compute the two endpoints of the i^{th} edge in $G_{m,n}$ in $\mathsf{polylog}(n)$ time. Furthermore, the j^{th} edge attaching to the i^{th} vertex can be easily computed since each of the q edges attaching to the i^{th} vertex is uniquely determined by $l_1 \in \mathbb{F}_q$ or $p_1 \in \mathbb{F}_q$ (see Equation 3). ◀

Hardness of Approximation for Stochastic Problems via Interactive Oracle Proofs

Gal Arnon
Weizmann Institute of Science, Rehovot, Israel

Alessandro Chiesa
EPFL, Lausanne, Switzerland

Eylon Yogev
Bar-Ilan University, Ramat-Gan, Israel

Abstract

Hardness of approximation aims to establish lower bounds on the approximability of optimization problems in NP and beyond. We continue the study of hardness of approximation for problems beyond NP, specifically for *stochastic* constraint satisfaction problems (SCSPs). An SCSP with k alternations is a list of constraints over variables grouped into $2\mathsf{k}$ blocks, where each constraint has constant arity. An assignment to the SCSP is defined by two players who alternate in setting values to a designated block of variables, with one player choosing their assignments uniformly at random and the other player trying to maximize the number of satisfied constraints.

In this paper, we establish hardness of approximation for SCSPs based on interactive proofs. For $\mathsf{k} \leq O(\log n)$, we prove that it is AM[k]-hard to approximate, to within a constant, the value of SCSPs with k alternations and constant arity. Before, this was known only for $\mathsf{k} = O(1)$.

Furthermore, we introduce a natural class of k-round interactive proofs, denoted IR[k] (for *interactive reducibility*), and show that several protocols (e.g., the sumcheck protocol) are in IR[k]. Using this notion, we extend our inapproximability to all values of k: we show that for every k, approximating an SCSP instance with $O(\mathsf{k})$ alternations and constant arity is IR[k]-hard.

While hardness of approximation for CSPs is achieved by constructing suitable PCPs, our results for SCSPs are achieved by constructing suitable IOPs (interactive oracle proofs). We show that every language in AM[$\mathsf{k} \leq O(\log n)$] or in IR[$\mathsf{k}$] has an $O(\mathsf{k})$-round IOP whose verifier has *constant* query complexity (*regardless* of the number of rounds k). In particular, we derive a "sumcheck protocol" whose verifier reads $O(1)$ bits from the entire interaction transcript.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Interactive proof systems

Keywords and phrases hardness of approximation, interactive oracle proofs, stochastic satisfaction problems

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.24

Related Version *Full Version*: https://eprint.iacr.org/2022/168

Funding *Gal Arnon*: Supported in part by a grant from the Israel Science Foundation (no. 2686/20) and by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness.
Alessandro Chiesa: Funded by the Ethereum Foundation.
Eylon Yogev: Supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the Alter Family Foundation.

1 Introduction

Many combinatorial optimization problems are NP-hard, and so there is little hope to solve them in polynomial time. This has motivated the study of polynomial-time *approximation algorithms* to solve optimization problems, which has revealed a surprising landscape. While

they are equivalent as *decision* problems (under polynomial-time reductions), NP-complete problems behave radically different from the point of view of approximability. Specifically, known approximation algorithms for NP-complete problems sometimes achieve approximations to within any constant, sometimes only to within a certain constant, and sometimes do not even achieve a constant approximation.

This differing behavior is justified via results in the area of *hardness of approximation*. For a given NP-complete problem, the goal is to prove that it is NP-hard to approximate the problem to better than a certain approximation ratio (e.g., better than 1/2). Ideally, this ratio would match the best known approximation algorithm, thereby ruling out better approximation algorithms.

The key tool used to establish hardness of approximation for NP-complete problems are probabilistic proofs [13]. For example, the PCP theorem [6, 5] says that every language in NP can be decided via a verifier that reads $O(1)$ bits from a polynomial-length proof, and in turn this implies, e.g., that the value of 3SAT cannot be approximated to within an arbitrary constant.

More generally, improvements in PCP constructions imply hardness results for corresponding *constraint satisfaction problems* (CSPs). Yet, there are numerous problems of interest that are *not* CSPs, and for which we wish to understand their behavior with respect to inapproximability.

Hardness of approximation beyond NP

Prior work has investigated hardness of approximation for natural problems in other complexity classes. In one direction, PCP-like theorems for fine-grained complexity have been used to establish hardness results for problems within the complexity class P (see [1, 3, 2, 9, 8]). In the other direction, several works study the inapproximability of two-player CSPs. A CSP can be viewed as a one-player game where the player wishes to maximize the number of satisfied constraints; this view naturally leads to two-player CSPs played in moves. Ko and Lin [17] proved the inapproximability of two-player CSPs with k moves, based on the hardness of the k-th level of the Polynomial Hierarchy. Haviv, Regev, and Ta-Shma [16] proved that this inapproximability result holds even when each variable occurs $O(1)$ times.

SCSPs and their hardness

In this paper, we study the hardness of approximating a natural class of problems known as *stochastic constraint satisfaction problems* (SCSPs), also known as *games against nature* [19]. Informally, they are two-player CSPs where one player is an adversary and the other player is a (public-coin) referee that plays random moves.

▶ **Definition 1** (informal). *An* **SCSP** Φ *with* k *alternations is a list of constraints* $C_1, \ldots, C_{\mathsf{m}}$ *over variables that are grouped into* $2\mathsf{k}$ *blocks and take on values over an alphabet* Σ*. The SCSP has* arity q *if each constraint depends on at most* q *variables. An assignment for* Φ *is defined by two players who alternate in setting values to a designated block of variables with one player choosing their assignments uniformly at random and the other player trying to maximize the number of satisfied clauses. The* value *of* Φ *is the expected fraction of clauses satisfied in this process.*

The hardness of approximating the value of SCSPs, to within a constant factor, has been studied in a line of works. Let AM[k] be the class of languages that have a k-round public-coin IP with constant soundness error. Drucker [12] extended the PCP theorem to the

stochastic setting, showing that it is AM[1]-complete to approximate the value of SCSPs with one alternation ($\mathsf{k} = 1$) and arity $\mathsf{q} = O(1)$. Subsequently, [4] showed that, for every k, it is AM[k]-complete to approximate the value of SCSPs with k alternations and arity $\mathsf{q} = O(\mathsf{k})$. In the regime of many alternations, Condon, Feigenbaum, Lund, and Shor [10] showed that it is PSPACE-complete to approximate the value of SCSPs with $\mathsf{k} = \text{poly}$ alternations and arity $\mathsf{q} = O(1)$. This leaves open the approximation hardness of SCSPs with k alternations and *constant* arity, for general values of k:

How hard is approximating the value of SCSPs with k *alternations and arity* $\mathsf{q} = O(1)$*?*

It seems reasonable to hypothesize that it is AM[k]-hard to approximate SCSPs with k alternations.

Note that Goldreich, Vadhan, and Wigderson [14] showed that AM[k] $\neq$ AM[$o(\mathsf{k})$] (under reasonable hardness assumptions), meaning that increasing the round complexity k adds more power to the complexity class AM[k]. For sufficiently many rounds, we know that IP = PSPACE [18, 21]. Thus, it is imperative to understand the approximation hardness of SCSPs with k alternations while respecting the different regimes for k.

SCSP hardness from IOPs

The above results are derived (implicitly or explicitly) by leveraging the connection between SCSPs and the PCP analog of interactive proofs, called *interactive oracle proofs* (IOPs) [7, 20]. A k-round (public-coin) IOP is a k-round (public-coin) IP where the verifier has PCP-like access to each prover message: after the interaction, the verifier probabilistically reads a small number of locations from the interaction transcript and then accepts or rejects.

A k-round public-coin IOP with query complexity q can be viewed as an SCSP with k alternations and arity q (see Section 2.1). Therefore, constructions of IOPs for hard languages imply corresponding hardness of approximation results for the resulting SCSPs. This leads us to ask:

Does every language in AM[k] *have a* k*-round IOP with constant query complexity?*

1.1 Our results

In this paper, we establish hardness of approximation for SCSPs from (general) interactive proofs. Moreover, we also prove that tighter hardness results can be achieved for specific languages that are *interactively reducible* (a notion that we introduce).

1.1.1 On the AM hardness of SCSPs

We prove that it is AM[k]-hard to approximate the value of binary-alphabet SCSPs with k alternations and arity $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$ ($\mathbb{x}$ is the instance).

▶ **Theorem 2** (informal). *Let* $L \in$ AM[k] *be a language. There exists a deterministic polynomial-time reduction that maps an instance* $\mathbb{x}$ *for* L *to an* SCSP *instance* Φ *with binary alphabet,* k *alternations, and arity* $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$ *such that:*

- *if* $\mathbb{x} \in L$ *then the value of* Φ *is* 1*;*
- *if* $\mathbb{x} \notin L$ *then the value of* Φ *is at most* 1/2*.*

Our improvement in arity is particularly meaningful for logarithmic round complexity: Theorem 2 establishes that, for $\mathsf{k}(|\mathbb{x}|) = O(\log|\mathbb{x}|)$, approximating the value of an SCSP instance with k alternations and arity $O(1)$ is as hard as deciding all of AM[k]. Previously, this was known only for constant k [12, 4].

Many results on the hardness of approximating the value of (standard) CSPs are achieved by constructing suitable PCPs. Similarly (as was noted in [4]), by constructing a suitable k-round IOP for a language L, one can show that approximating the value of SCSPs with k alternations up to a constant factor is as hard as deciding L. We establish Theorem 2 using this framework by providing a transformation that maps a k-round IP into a k-round IOP with small query complexity.

▶ **Lemma 3** (informal). *Let L be a language with a k-round public-coin IP. Then L has a k-round public-coin binary IOP where, on input $\mathbb{x}$, the verifier reads $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$ bits of the interaction transcript. All other parameters are polynomially related.*

Lemma 3 is surprising in light of the work of Goldreich, Vadhan, and Wigderson [14], which shows a separation between AM[k] and AM[$o(\mathsf{k})$] (under relatively weak hardness assumptions). Since the query complexity is smaller than the round complexity, Lemma 3 implies that the IOP verifier does not make queries to every round of the protocol. This can be viewed as saying that the power of AM[k] is unchanged even when the verifier only accesses $O(\mathsf{k}/\log|\mathbb{x}|)$ of the k rounds. In other words, reducing round complexity of public-coin protocols reduces their power, but it is nevertheless possible to not read every round of interaction while preserving the power.

1.1.2 Hardness of SCSPs from interactive reducibility

Theorem 2 establishes the hardness of SCSPs with $\mathsf{k} = O(\log|\mathbb{x}|)$ alternations and arity $O(1)$ (over the binary alphabet), but does not work for $O(1)$-arity SCSPs with $\mathsf{k} = \omega(\log|\mathbb{x}|)$ alternations.

We extend this by showing that approximating the value of $O(1)$-arity SCSPs with k alternations is as hard as solving $\#\mathsf{SAT}_{\mathsf{k}}$.[1] In fact, we show this for a more general class of languages that are *interactively reducible*, a notion that we introduce in this work. Informally, the notion requires that it is possible to reduce, via an interactive protocol, multiple transcripts of an IP for the relation into a single transcript. We require that the probability of the verifier accepting conditioned on the reduced transcript be (roughly) the minimum probability of the verifier accepting conditioned on any of the original transcripts. See Section 2.3 for further details and discussion.

Interactive reducibility is a natural property; we show that general interactive proofs can be seen as interactive reductions (albeit ones with bad parameters), and show interactive reductions for the sumcheck protocol [18] and for Shamir's protocol [21].

The notion of interactive reducibility allows us to get an optimal version of Lemma 3 for additional languages. Let IR[k] be the class of languages that have a (1-round) interactive reduction with k predicates (these predicates roughly align with rounds of an IP).

▶ **Lemma 4** (informal). *Let L be a language in* IR[k]. *Then L has an $O(\mathsf{k})$-round public-coin IOP, with polynomial proof length, where the verifier reads $O(1)$ bits of the interaction transcript.*

In particular, applying Lemma 4 to the sumcheck protocol yields the following (perhaps surprising) conclusion: *any k-round sumcheck protocol can be transformed to a $O(\mathsf{k})$-round IOP where the verifier has $O(1)$ query complexity (over the binary alphabet).* Notice that using standard PCP techniques it is only known how to achieve a similar (non-interactive) result with *exponential* proof length.

[1] $\#\mathsf{SAT}_{\mathsf{k}}$ is the restriction of $\#\mathsf{SAT}$ to instances of size n and $\mathsf{k}(n)$ variables.

Using this improved lemma, we immediately get that for every k, deciding whether a binary-alphabet SCSP instance with $O(\mathsf{k})$ alternations and arity $O(1)$ has value 1 or value $1/2$ is IR[k]-hard:

▶ **Theorem 5** (informal). *Let $L \in$ IR[k] be a language. There exists a deterministic polynomial-time reduction that maps an instance $\mathbb{x}$ for L to an* SCSP *instance Φ with binary alphabet, $O(\mathsf{k})$ alternations, and arity $O(1)$ such that:*

- *if $\mathbb{x} \in L$ then the value of Φ is 1;*
- *if $\mathbb{x} \notin L$ then the value of Φ is at most $1/2$.*

Using our interactive reduction for sumcheck, we know that $\#\mathsf{SAT}_{\mathsf{k}} \in$ IR[k]. Thus, by Theorem 5, we establish that approximating the value of SCSPs with $O(\mathsf{k})$ alternations and constant arity is $\#\mathsf{SAT}_{\mathsf{k}}$-hard. Similarly, using our interactive reduction for Shamir's protocol, we recover the result of [10], showing that approximating the value of SCSPs with polynomially-many alternations and constant arity is PSPACE-hard.

1.1.3 Summary of results and open questions

We construct $O(1)$-query IOPs for every language in AM[$\mathsf{k} \leq O(\log|\mathbb{x}|)$] or in IR[$\mathsf{k}$]. Moreover, we construct $O(\mathsf{k}/\log|\mathbb{x}|)$-query IOPs for every language in AM[k]. These results establish approximation hardness for SCSPs as follows: (a) for $\mathsf{k} = O(\log|\mathbb{x}|)$, it is AM[$\mathsf{k}$]-hard to approximate the value of SCSPs with k alternations and constant arity; and (b) for $\mathsf{k} = \omega(\log|\mathbb{x}|)$, it is IR[$\mathsf{k}$]-hard to approximate the value of SCSPs with $O(\mathsf{k})$ alternations and constant arity, and AM[k]-hard when the SCSPs have arity $O(\mathsf{k}/\log|\mathbb{x}|)$. Our results are summarized in Figure 1 together with previously known results.

Our work leaves open the AM-hardness of approximating the value of SCSPs with $\mathsf{k} = \omega(\log|\mathbb{x}|)$ alternations and constant arity. From the perspective of IOPs, we also leave open the basic question that we raised in the introduction: *Does every language in* AM[k] *have a k-round (public-coin) IOP with constant query complexity over the binary alphabet?* Our results contribute notable progress towards resolving this question (see paragraph above), but answering the question for every regime of k remains a fascinating challenge in the theory of probabilistic proofs.

	hardness	**alternations**	**arity**
[10]	PSPACE	unbounded	$O(1)$
[this work]	IR[k]	$O(\mathsf{k})$	$O(1)$
[this work]	$\#\mathsf{SAT}_{\mathsf{k}}$	$O(\mathsf{k})$	$O(1)$
[4]	AM[k]	k	$O(\mathsf{k})$
[this work]	AM[k]	k	$\max\{O(1), O(\mathsf{k}/\log\|\mathbb{x}\|)\}$
[12]	AM[1]	1	$O(1)$
[5, 6, 11]	NP	n/a	$O(1)$

Figure 1 Summary of results for approximating the value of binary-alphabet SCSPs to within a constant factor. AM[k] denotes the class of languages with k-round public-coin interactive proofs. IR[k] denotes the class of languages with (1-round) interactive reductions with k predicates. $\#\mathsf{SAT}_{\mathsf{k}}$ is the restriction of $\#\mathsf{SAT}$ to instances of size n and $\mathsf{k}(n)$ variables.

2 Techniques

We outline the main ideas behind our results. In Section 2.1 we explain a generic connection between SCSPs and IOPs: in order to establish the hardness of approximating SCSPs, it suffices to construct IOPs with certain properties. This will be our goal in the remaining sections. In Section 2.2 we show how to transform k-round IPs into k-round IOPs with query complexity $\max\{O(1), O(\mathsf{k}/\log|\mathbb{x}|)\}$. In Section 2.3 we show that for relations that are *interactively reducible* we can construct $O(1)$-query IOPs even when those relations are only known to have IPs with round complexity $\omega(\log|\mathbb{x}|)$.

2.1 On the hardness of approximating SCSPs via IOPs

We review the generic connection between CSPs and PCPs, and then describe the analogous connection between SCSPs and IOPs. In both cases, efficient constructions of PCPs/IOPs imply hardness of approximation results for corresponding CSPs/SCSPs.

2.1.1 CSP hardness from PCPs

A CSP Φ is a list of boolean functions $C_1, \ldots, C_{\mathsf{m}}$ over variables from a bounded alphabet Σ. The CSP has arity q if each constraint depends on at most q variables. The goal is to determine the maximum fraction of constraints that can be satisfied by any assignment.

We can map a non-adaptive PCP verifier $\mathbf{V}$ for a language L and an instance $\mathbb{x}$ into a CSP. The variables represent locations of the PCP string. Each choice of PCP verifier randomness induces a corresponding constraint, whose variables are those that the PCP verifier would have read from the PCP string. The constraint is satisfied if and only if the PCP verifier accepts when it receives the assignment of the variables as its query answers. The CSP's arity equals the PCP's query complexity.

By completeness of the PCP system, if $\mathbb{x} \in L$ then there exists a PCP string that makes the PCP verifier accept with probability 1, which in turn means that there is an assignment that simultaneously satisfies every constraint in the CSP. By the soundness of the PCP system, if $\mathbb{x} \notin L$ then every PCP string makes the PCP verifier accept with at most probability $1/2$, which in turn means that no assignment can satisfy more than half of the constraints of the CSP.

Thus, distinguishing whether the CSP's value is 1 or at most $1/2$ is as hard as deciding L.

2.1.2 SCSP hardness from IOPs

The general connection between SCSPs (Definition 1) and IOPs is stated in the lemma below. Recall that a k-round IOP is a k-round IP where the verifier has PCP-like access to each prover message: the prover and verifier interact for k rounds, and after the interaction, the verifier probabilistically reads a small number of bits from each prover message and decides to accept or reject based on the examined locations. The randomness used in the final phase is called *decision randomness* (which we distinguish from the random messages that the verifier sends to the prover during the interaction and may be queried at only few locations).

▶ **Lemma 6.** *Let L be a language with a non-adaptive k-round public-coin IOP with alphabet Σ, polynomial proof length, query complexity q, decision randomness r_{dc}, and soundness error β.*

Then there exists a deterministic polynomial-time reduction that maps an instance $\mathbb{x}$ *for* L *to an SCSP instance* Φ *with alphabet* Σ, k *alternations, arity* q, $2^{\mathsf{r_{dc}}}$ *constraints and a polynomial number of variables such that:*

- *if* $\mathbb{x} \in L$ *then the value of* Φ *is* 1*;*
- *if* $\mathbb{x} \notin L$ *then the value of* Φ *is at most* β*.*

Proof sketch. Let $\mathbf{V}_{\mathsf{IOP}}$ be the (non-adaptive) IOP verifier for L and let l be the (per-round) proof length of the IOP. Given an instance $\mathbb{x}$, we construct the SCSP instance Φ as follows. The SCSP has $2\mathsf{k}$ blocks of l variables that align with the interaction transcript between the IOP prover and IOP verifier (the i-th variable of the j-th block corresponds to the i-th symbol of the j-th message of the protocol). The SCSP has a constraint C_ρ for each $\rho \in \{0,1\}^{\mathsf{r_{dc}}}$, whose input variables correspond to the locations of the transcript that $\mathbf{V}_{\mathsf{IOP}}$ queries given instance $\mathbb{x}$ and randomness ρ; the constraint C_ρ is satisfied if and only if $\mathbf{V}_{\mathsf{IOP}}(\mathbb{x};\rho)$ accepts if it reads the symbols assigned to the variables of C_ρ.

The SCSP instance Φ has k alternations (corresponding to the rounds of the IOP) and $\mathsf{l} = \mathrm{poly}(|\mathbb{x}|)$ variables per alternation (corresponding to the message length) that are assigned values in the alphabet Σ (the alphabet of the IOP). Each of its $2^{\mathsf{r_{dc}}}$ constraints has arity q since each constraint has as many inputs as queries made by the IOP verifier $\mathbf{V}_{\mathsf{IOP}}$.

Finally, we analyze the value of the SCSP. By construction, there exists an IOP prover strategy that causes the IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ to accept with probability δ if and only if there exists a strategy for the existential player in the SCSP such that the expected fraction of constraints that are satisfied is δ (i.e., the value of Φ is at least δ). Therefore, by perfect completeness of the IOP, if $\mathbb{x} \in L$ then the value of Φ is 1. Conversely, by soundness of the IOP, if $\mathbb{x} \notin L$ then the value of Φ is at most β. ◀

2.2 Transforming IPs into IOPs

We outline the proof of Lemma 3 (transforming an IP into an IOP with small query complexity). In Section 2.2.1, we show how to transform a logarithmic-round IP into a $O(1)$-query IOP. Then in Section 2.2.2 we extend this idea to transform a k-round IP into a $O(\mathsf{k}/\log|\mathbb{x}|)$-query IOP.

2.2.1 From $O(\log|\mathbb{x}|)$-round IP to $O(1)$-query IOP

We show how to transform a k-round public-coin IP where $\mathsf{k} = O(\log|\mathbb{x}|)$ into an $O(\mathsf{k})$-round public-coin IOP with the following efficiency: polynomial proof length over the binary alphabet; constant query complexity; and logarithmic decision randomness.

First, we sketch how to transform a k-round public-coin IP $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ into an $O(\mathsf{k})$-round public-coin IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ where the verifier reads $O(1)$ rounds (in their entirety) from the interaction transcript. Then we explain how to ensure that the verifier queries $O(1)$ bits in total.

2.2.1.1 A strawman protocol

We describe a natural strategy for transforming the IP into an IOP where the verifier reads $O(1)$ rounds, albeit with high soundness error. The IOP prover $\mathbf{P}_{\mathsf{IOP}}$ and IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ respectively simulate the IP prover and verifier $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$, inducing an interaction transcript $\mathsf{tr} = (\rho_1, a_1, \ldots, \rho_{\mathsf{k}}, a_{\mathsf{k}})$. At this time, however, $\mathbf{V}_{\mathsf{IOP}}$ *does not read any messages from* tr. After this interaction, $\mathbf{P}_{\mathsf{IOP}}$ sends a transcript tr', which is allegedly equal to tr, as a single message. Then $\mathbf{V}_{\mathsf{IOP}}$ reads tr' and checks that tr' is an accepting transcript for the IP verifier

$\mathbf{V}_{\mathsf{IP}}$. Moreover, $\mathbf{V}_{\mathsf{IOP}}$ tests consistency between tr (the real interaction) and tr' (the alleged copy of the interaction sent as a single message): $\mathbf{V}_{\mathsf{IOP}}$ samples a random $i \in [\mathsf{k}]$, reads the i-th prover message and i-th verifier message in tr, and checks that these equal the corresponding messages in tr'.

In this IOP, $\mathbf{V}_{\mathsf{IOP}}$ reads $O(1)$ messages from the interaction transcript, but the soundness error of the IOP is large (even when discounting the soundness error of the underlying IP). Indeed, it may be that a cheating IOP prover sends a malicious transcript tr' that is accepting but differs from the real transcript tr in one round only. In this case, $\mathbf{V}_{\mathsf{IOP}}$ catches the inconsistency only with probability $1/\mathsf{k}$, which means that the soundness error could be as large as $1 - 1/\mathsf{k}$.

Note that reducing this soundness error via parallel repetition would increase the number of rounds queried by $\mathbf{V}_{\mathsf{IOP}}$. Achieving constant soundness error would require $O(\mathsf{k})$ repetitions, resulting in an IOP verifier that reads $O(\mathsf{k})$ rounds, taking us back to where we started.

2.2.1.2 Our transformation

We present a transformation that improves on the above strawman protocol, achieving a constant soundness error for any IP that has a logarithmic number of rounds.

A malicious IOP prover in the strawman protocol has two strategies: the transcript tr' sent as the last message either agrees with the real transcript tr on more than half of the rounds, or it does not. If tr' agrees with tr in less than half of the rounds, then the IOP verifier catches this inconsistency with probability at least $1/2$. Intuitively, the transformation that we sketch below ensures that if tr' is consistent with tr on at least half of the rounds, then the consistent rounds must contain within them a full execution of the underlying IP. Then, since this consistent part was generated interactively in tr, by the soundness property of the IP, this contained transcript will be rejected with high probability. We now describe our transformation in more detail.

Suppose that our public-coin IP has $\mathsf{k}(|\mathbb{x}|) = O(\log |\mathbb{x}|)$ rounds and that the verifier message in each round is r bits long. The IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ on a given instance $\mathbb{x}$ works as follows.

1. $\mathbf{P}_{\mathsf{IOP}}$ sets $S_0 := \{\emptyset\}$ (i.e., S_0 consists of the empty transcript).
2. For $i = 1, \ldots, 2\mathsf{k}$:
   - $\mathbf{P}_{\mathsf{IOP}}$ sends S_{i-1}.
   - $\mathbf{V}_{\mathsf{IOP}}$ sends a random $\rho_i \in \{0,1\}^{\mathsf{r}}$ (corresponding to a message of $\mathbf{V}_{\mathsf{IP}}$).
   - $\mathbf{P}_{\mathsf{IOP}}$ sets $S_i := S_{i-1} \cup \{(\mathsf{tr}||\rho_i||a_{\mathsf{tr},i})\}_{\mathsf{tr} \in S_{i-1}}$ where $a_{\mathsf{tr},i} := \mathbf{P}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}||\rho_i)$ for each $\mathsf{tr} \in S_{i-1}$.
3. $\mathbf{P}_{\mathsf{IOP}}$ sends $S_{2\mathsf{k}}$ and, for every $i \in [2\mathsf{k}]$, also sends $T_i := S_i$.
4. In the decision phase, $\mathbf{V}_{\mathsf{IOP}}$ performs the checks below.
   a. *Subset consistency:* For every $i \in [2\mathsf{k}]$, check that $T_{i-1} \subseteq T_i$.
   b. *Transcript consistency:* Choose a random $i \in [2\mathsf{k}]$. Check that $S_{i-1} = T_{i-1}$ and $S_i = T_i$. Additionally, check that for every $\mathsf{tr} \in S_{i-1}$ there is a message $a_{\mathsf{tr},i}$ such that $(\mathsf{tr}||\rho_i||a_{\mathsf{tr},i}) \in S_i$, where ρ_i is the verifier message sent during the i-th round of interaction.
   c. *Membership:* Check that for every transcript $\mathsf{tr} \in T_{2\mathsf{k}}$ that is complete (i.e., contains messages for all k rounds of the IP) it holds that $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \mathsf{tr}) = 1$.

2.2.1.3 Efficiency

We briefly discuss the main efficiency measures of the transformation.

- *Query complexity.* The IOP verifier reads $O(1)$ rounds from the transcript.
- *Communication complexity.* We argue that all messages in the protocol have length $\text{poly}(|\mathbb{x}|)$. For every i, $|S_i| \leq 2|S_{i-1}|$ since S_i contains all transcripts in S_{i-1} and continuations of each of these transcripts. Since $\mathsf{k} = O(\log|\mathbb{x}|)$, $|S_i| = \text{poly}(|\mathbb{x}|)$ for every i. Each transcript within S_i has polynomial length, so the length of these messages is $\text{poly}(|\mathbb{x}|)$. Finally, the sets $T_1, \ldots, T_{2\mathsf{k}}$ have the same sizes as $S_1, \ldots, S_{2\mathsf{k}}$ (respectively) and so the prover's final message has length $\text{poly}(|\mathbb{x}|)$.
- *Decision randomness.* The IOP verifier uses $O(\log \mathsf{k}) = O(\log|\mathbb{x}|)$ bits of decision randomness.

2.2.1.4 Analysis

In this overview, we discuss soundness only, as completeness follows straightforwardly from the construction. Let β be the soundness error of $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$. We show that the IOP $(\mathbf{P}_{\mathsf{IOP}}, \mathbf{V}_{\mathsf{IOP}})$ has soundness error

$$\beta_{\mathsf{IOP}} = \max\left\{\frac{1}{2}, \binom{2\mathsf{k}}{\mathsf{k}} \cdot \beta\right\} \ .$$

The above expression can be made constant by applying $\text{poly}(|\mathbb{x}|)$ parallel repetitions to $(\mathbf{P}_{\mathsf{IP}}, \mathbf{V}_{\mathsf{IP}})$ prior to applying our transformations, until it has soundness error $\beta' \leq \left(2 \cdot \binom{2\mathsf{k}}{\mathsf{k}}\right)^{-1}$.

Fix $\mathbb{x} \notin L$ and a cheating IOP prover $\tilde{\mathbf{P}}_{\mathsf{IOP}}$. A fixed transcript of the IOP has the structure:

$$\left(S_0, \rho_1, S_1, \ldots, \rho_{2\mathsf{k}_{\mathsf{IP}}}, S_{2\mathsf{k}}, (T_0, \ldots, T_{2\mathsf{k}})\right) \ .$$

Given such a transcript, we say that an index i is *consistent* if: (a) $S_{i-1} = T_{i-1}$ and $S_i = T_i$; and (b) for every $\mathsf{tr} \in S_{i-1}$ there is a message $a_{\mathsf{tr},i}$ such that $(\mathsf{tr}||\rho_i||a_{\mathsf{tr},i}) \in S_i$.

Conditioned on the event that the transcript generated during the interaction has less than k consistent indices i, $\mathbf{V}_{\mathsf{IOP}}$ rejects with probability at least $1/2$ due to its check in Item 4b. We are thus left to analyze the probability that $\mathbf{V}_{\mathsf{IOP}}$ rejects conditioned on the event that the generated transcript has at least k consistent indices.

Fix indices $i_1 < \cdots < i_{\mathsf{k}}$ and suppose that all these indices are consistent with respect to the transcript. By the definition of consistency, for every $j \in [\mathsf{k}]$, $S_{i_j-1} = T_{i_j-1}$, $S_{i_j} = T_{i_j}$ and $(\mathsf{tr}||\rho_{i_j}||a_{\mathsf{tr},i_j}) \in S_{i_j}$ for every $\mathsf{tr} \in S_{i_j-1}$. This implies that there exist $a_{i_1}, \ldots, a_{i_{\mathsf{k}}}$ such that $(\rho_{i_1}||a_{i_1}||\cdots||\rho_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) \in T_{i_{\mathsf{k}}}$. By the subset consistency check, $\mathbf{V}_{\mathsf{IOP}}$ accepts only if $T_{i_{\mathsf{k}}} \subseteq T_{2\mathsf{k}}$, in which case $(\rho_{i_1}||a_{i_1}||\cdots||\rho_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) \in T_{2\mathsf{k}}$. This transcript was generated interactively by the prover and verifier and hence, by the soundness of the IP, $\mathbf{V}_{\mathsf{IP}}(\mathbb{x}, \rho_{i_1}||a_{i_1}||\cdots||\rho_{i_{\mathsf{k}}}||a_{i_{\mathsf{k}}}) = 1$ with probability at most β. If the transcript is rejecting, then this is detected by $\mathbf{V}_{\mathsf{IOP}}$ in its membership check.

The previous analysis holds for fixed indices $i_1 < \cdots < i_{\mathsf{k}}$. By applying the union bound to all choices of indices, we have that, conditioned on the transcript generated having at least k consistent rounds, $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\binom{2\mathsf{k}}{\mathsf{k}} \cdot \beta$.

Putting together both (non-intersecting) events of the number of rounds consistent with the generated transcript, we conclude that $\mathbf{V}_{\mathsf{IOP}}$ accepts with probability at most $\max\{\frac{1}{2}, \binom{2\mathsf{k}}{\mathsf{k}} \cdot \beta\}$.

2.2.1.5 Achieving query complexity $O(1)$ over the binary alphabet

The verifier in the IOP described above reads $O(1)$ rounds (in their entirety) from the interaction transcript, rather than $O(1)$ bits in total. We additionally achieve this latter goal by building on a result in [4].

In more detail, [4] transforms a k-round public-coin IP into an $O(\mathsf{k})$-round public-coin IOP with polynomial proof length over the binary alphabet and where the IOP verifier reads $O(1)$ bits from each round. We extend this to transform a k-round IOP whose verifier reads q of the k rounds into a $O(\mathsf{k})$-round IOP whose verifier reads $O(1)$ bits from each of $O(\mathsf{q})$ rounds. See the full version of this paper for more details.

2.2.2 IOPs from general IPs

The transformation described in the previous section works for $O(\log |\mathbb{x}|)$-round IPs. However, it cannot be directly applied to IPs with more rounds because the proof length (and thus also the verifier running time) would be more than polynomial.

Nevertheless, we extend the transformation to work for any public-coin IP while achieving a moderate improvement on the number of read rounds. In the main loop of the IOP, rather than advancing each IP transcript in S_{i-1} by one round, advance it by $O(\mathsf{k}/\log |\mathbb{x}|)$ rounds before inserting the resulting transcripts into the set S_i. During its decision phase, the IOP verifier chooses an index i and reads the entire $O(\mathsf{k}/\log |\mathbb{x}|)$-round interaction done during this iteration of the IOP. Completeness and soundness of this new transformation are similar to the one presented in the previous section, but now the verifier reads $O(\mathsf{k}/\log |\mathbb{x}|)$ rounds. The rest of the efficiency parameters are similar to the IOP described in the previous section, except that proof length is polynomial regardless of k.

After applying the (adapted) transformation of [4], this process yields a $O(\mathsf{k})$-round IOP with query complexity $O(\mathsf{k}/\log |\mathbb{x}|)$ (over the binary alphabet). This concludes the proof sketch of Lemma 3.

Can we do better?

The construction described in this section doubles the number of transcripts stored in the set S_i relative to S_{i-1}. This causes a blow-up in parameters and is the reason why this approach fails in constructing $O(1)$-query IOPs from IPs with super-logarithmic round complexity. Intuitively, if we could reduce this doubling then we may be able to modify the transformation to get $O(1)$-query IOPs. While we do not achieve this for *general* IP, we show that, using this intuition, we can construct $O(1)$-query IOPs for a rich class of relations that are *interactively reducible*. We discuss this notion, and corresponding new IOP constructions, in the following section.

2.3 Interactive reducibility

In Section 2.2.1 we described how to transform an IP into an IOP with $O(1)$ query complexity. The new protocol kept track of a set containing all partial transcripts of the IP generated so far in the protocol. In every round, every partial transcript in the set was advanced by one round, and the newly advanced transcripts were added to the set held previously. This meant that in every round, the set contains twice as many transcripts as in the previous round. As we require polynomial proof length and verifier running time, this technique was unable to allow reading of $O(1)$ rounds for IPs with greater than $O(\log |\mathbb{x}|)$ rounds.

Intuitively, suppose it were possible to take multiple transcripts and reduce them into a single transcript that in some sense preserves soundness of all of the transcripts combined. In that case, this issue could be bypassed, and the protocol would work for IPs with super-logarithmic round complexity. In more detail, suppose we want to reduce transcripts $\mathsf{tr}_1, \ldots, \mathsf{tr}_\mathsf{t}$ into a new transcript tr'. The *acceptance probability* of a transcript prefix tr_i is the maximum over all prover strategies of the probability that the verifier will end up accepting

when continuing interaction with the prover from transcript tr_i. Roughly, we require that: (a) if the acceptance probability of every tr_i is 1, then so is the acceptance probability of tr'; and (b) if there exists some transcript tr_i with small acceptance probability, then (with high probability) the acceptance probability of tr' is also small.

In this section, we introduce the concept of *interactive reducibility*, which formally captures this intuition. We exemplify this in Section 2.3.1 by describing how to reduce multiple transcripts of the sumcheck protocol into a single transcript. Then, in Section 2.3.2, we formally define interactive reducibility. In Section 2.3.3, we show how to adapt the protocol described in Section 2.2.1 to work with interactive reductions and bypass the blow-up in the original protocol. Finally, in Section 2.3.4, we discuss relations known to have interactive reducibility.

2.3.1 An interactive reduction for sumcheck

We exemplify the notion of interactive reducibility in the case of the sumcheck protocol [18]. Below we review this protocol, and then explain how to reduce multiple transcripts into one transcript via an interactive reduction.

2.3.1.1 The sumcheck protocol

The verifier has query access to a n-variate polynomial p of individual degree d over some field $\mathbb{F}$. The goal of the verifier is to test, for a given field element γ, whether

$$\sum_{\alpha_1,\ldots,\alpha_n\in\{0,1\}} p(\alpha_1,\ldots,\alpha_n) = \gamma \ .$$

The protocol begins with the prover sending a polynomial $\tilde{p}_1$ of degree d, claimed to equal $p_1(X) := \sum_{\alpha_2,\ldots,\alpha_n\in\{0,1\}} p(X,\alpha_2,\ldots,\alpha_n)$. The verifier checks that $\tilde{p}_1(0)+\tilde{p}_1(1)=\gamma$ (rejecting if not), samples a random field element r_1, and sends it to the prover. Both parties define $\gamma_1 := \tilde{p}_1(r_1)$.

This one-round interaction leads to a new sumcheck claim

$$\sum_{\alpha_2,\ldots,\alpha_n\in\{0,1\}} p(r_1,\alpha_2,\ldots,\alpha_n) = \gamma_1 \ ,$$

that has the following properties: (a) if the original claim is true then the new claim is also true; and (b) if the original claim is false then with high probability the new claim is also false.

Next, the prover sends $\tilde{p}_2$ claimed to equal $p_2(X) := \sum_{\alpha_3,\ldots,\alpha_n\in\{0,1\}} p(r_1,X,\alpha_3,\ldots,\alpha_n)$ and the protocol repeats as before. This process continues until the n variables are fixed to some field elements $(r_1,\ldots,r_n)$, and the problem has been reduced to checking that $p(r_1,\ldots,r_n)=\gamma_n$, which the verifier can check via one query to the polynomial p.

One can associate a round j of the protocol with a list of field elements $(r_1,\ldots,r_j)$ and a claimed sum γ_j, and think of that round as "reducing" a claim $\mathbb{z}=((r_1,\ldots,r_j),\gamma_j)$ that $\sum_{\alpha_{j+1},\ldots,\alpha_n\in\{0,1\}} p(r_1,\ldots,r_j,\alpha_{j+1},\ldots,\alpha_n)=\gamma_j$ into a new claim $\mathbb{z}'=((r_1,\ldots,r_{j+1}),\gamma_{j+1})$ that $\sum_{\alpha_{j+2},\ldots,\alpha_n\in\{0,1\}} p(r_1,\ldots,r_{j+1},\alpha_{j+2},\ldots,\alpha_n)=\gamma_{j+1}$.

2.3.1.2 Reducing multiple sumcheck claims

We are given claims $\mathbb{z}_1,\ldots,\mathbb{z}_{\mathsf{t}}$ where each $\mathbb{z}_i$ consists of $(r_{i,1},\ldots,r_{i,j})$ and a claimed sum $\gamma_{i,j}$. We seek to reduce these t claims into a single claim $\mathbb{z}'=((r'_1,\ldots,r'_{j+1}),\gamma'_{j+1})$ such that:

(a) If each $\mathbb{z}_i$ is a true statement, then $\mathbb{z}'$ is a true statement; and (b) If there is some $\mathbb{z}_i$ that is a false statement, then with high probability $\mathbb{z}'$ is a false statement. Notice that in order to merge multiple sumcheck transcripts it suffices to merge multiple sumcheck claims $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathtt{t}}$. Thus we will focus on merging such claims.

Before describing the reduction, we define a polynomial $I_{\mathbb{z}_1,\ldots,\mathbb{z}_{\mathtt{t}}} : \mathbb{F} \to \mathbb{F}^j$ that represents a curve through the $\mathtt{t}$ points described by the instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathtt{t}}$. That is, $I_{\mathbb{z}_1,\ldots,\mathbb{z}_{\mathtt{t}}}(i) = (r_{i,1}, \ldots, r_{i,j})$ for every $i \in [\mathtt{t}]$ (here we implicitly associate the set $[\mathtt{t}]$ with an arbitrary set $S \subseteq \mathbb{F}$ of size $\mathtt{t}$, known to all parties). By interpolation, the degree of $I_{\mathbb{z}_1,\ldots,\mathbb{z}_{\mathtt{t}}}$ is less than $\mathtt{t}$.

Given this definition, we describe the interactive reduction for the sumcheck protocol.

- Prover: Send the polynomial $g \in \mathbb{F}[X_1, X_2]$ defined as:

$$g(X_1, X_2) := \sum_{\alpha_{j+2},\ldots,\alpha_n \in \{0,1\}} p(I_{\mathbb{z}_1,\ldots,\mathbb{z}_{\mathtt{t}}}(X_1), X_2, \alpha_{j+2}, \ldots, \alpha_n) \ . \tag{1}$$

- Verifier: Receive a bivariate polynomial $\tilde{g} \in \mathbb{F}[X_1, X_2]$ of degree at most $j \cdot d \cdot (\mathtt{t} - 1)$ in X_1 and degree at most d in X_2.
  1. *Consistency:* Check that for every $i \in [\mathtt{t}]$ it holds that $\sum_{\alpha \in \{0,1\}} \tilde{g}(i, \alpha) = \gamma_{i,j}$. (Reject if not.)
  2. *Generate new instance:*
     - **(i)** Sample uniformly random field elements $\rho, r^* \leftarrow \mathbb{F}$ and send them to the prover.
     - **(ii)** Set $(r'_1, \ldots, r'_j) := I_{\mathbb{z}_1,\ldots,\mathbb{z}_{\mathtt{t}}}(\rho)$ and $\gamma_{j+1} := \tilde{g}(\rho, r^*)$ and output the new instance

$$\mathbb{z}' := \Big((r'_1, \ldots, r'_j, r^*), \gamma_{j+1}\Big) \ .$$

Analysis. It follows straightforwardly from the protocol that, if $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathtt{t}}$ are all true statements and the prover acts honestly, then $\mathbb{z}'$ is a true statement. We show that if any one of the statements $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathtt{t}}$ is false then with high probability so is $\mathbb{z}'$.

Let g be as defined in Equation (1) with respect to $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathtt{t}}$. Suppose that $\mathbb{z}_i$ is a false claim (i.e., $\sum_{\alpha_{j+1},\ldots,\alpha_n \in \{0,1\}} p(r_{i,1}, \ldots, r_{i,j}, \alpha_{j+1}, \ldots, \alpha_n) \neq \gamma_{i,j}$). Then, by definition,

$$\sum_{\alpha \in \{0,1\}} g(i, \alpha) = \sum_{\alpha_{j+1},\ldots,\alpha_n \in \{0,1\}} p(r_{i,1}, \ldots, r_{i,j}, \alpha_{j+1}, \ldots, \alpha_n) \neq \gamma_{i,j} \ .$$

During its consistency check, the verifier checks that $\sum_{\alpha \in \{0,1\}} \tilde{g}(i, \alpha) = \gamma_{i,j}$. Thus, in order for the verifier to not reject, a cheating prover must send $\tilde{g} \neq g$. By the Schwartz–Zippel lemma, since g and $\tilde{g}$ are low-degree polynomials (provided that the degree d and the number of instances being reduced $\mathtt{t}$ are small with respect to $|\mathbb{F}|$), the probability that the uniformly chosen ρ and r^* are such that $\tilde{g}(\rho, r^*) = g(\rho, r^*)$ is small. Whenever $\tilde{g}(\rho, r^*) \neq g(\rho, r^*)$ we have that

$$\sum_{\alpha_{j+2},\ldots,\alpha_n \in \{0,1\}} p(r'_1, \ldots, r'_j, r^*, \alpha_{j+2}, \ldots, \alpha_n) = g(\rho, r^*) \neq \tilde{g}(\rho, r^*) = \gamma_{j+1} \ ,$$

and so the resulting statement $\mathbb{z}' := ((r'_1, \ldots, r'_j, r^*), \gamma_{j+1})$ is false.

2.3.2 Defining interactive reducibility

We define interactive reducibility, which captures the capability of merging multiple transcripts/instances into a single transcript/instance while preserving correctness and soundness.

▶ **Definition 7.** *An ℓ-round public-coin protocol $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ where $\mathbf{V}_{\mathsf{IR}}$ runs in polynomial time is an* **interactive reduction** *for a relation R with k predicates and soundness error ε if there exists a sequence of predicates $f_0, f_1, \ldots, f_{\mathsf{k}}$ such that the following holds.*

- **Completeness:** *For every $(\mathbb{x}, \mathbb{w}) \in R$, $j \in [\mathsf{k}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$ such that $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 1$ for every $i \in [\mathsf{t}]$, it holds that:*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}), \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] = 1 \ .$$

- **Soundness:** *For every $\mathbb{x} \notin L(R)$, $j \in [\mathsf{k}]$, and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, if there exists $i \in [\mathsf{t}]$ where $f_{j-1}(\mathbb{x}, \mathbb{z}_i) = 0$ then for every (computationally unbounded) $\tilde{\mathbf{P}}_{\mathsf{IR}}$ it holds that:*

$$\Pr\left[\, f_j(\mathbb{x}, \mathbb{z}') = 1 \,\middle|\, \mathbb{z}' \leftarrow \langle \tilde{\mathbf{P}}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}) \rangle \,\right] \leq \varepsilon(\mathbb{x}, \mathsf{t}) \ .$$

- **Relation identity:** *$f_0(\mathbb{x}, \mathbb{z}) = 1$ if and only if $\mathbb{x} \in L(R)$.*
- **Triviality:** *$f_{\mathsf{k}}(\mathbb{x}, \mathbb{z})$ can be computed in time $\mathrm{poly}(|\mathbb{x}|, |\mathbb{z}|)$.*

We call $\mathbb{x}$ the base instance *and $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$* round instances.

An interactive reduction $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ has (polynomially) **bounded output length** if there exists $c \in \mathbb{N}$ such that, for every base instance $\mathbb{x}$, witness $\mathbb{w}$, and round instances $\mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}}$, the new instance $\mathbb{z}'$ output by $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ on inputs $(\mathbb{x}, \mathbb{w}, \mathbb{z}_1, \ldots, \mathbb{z}_{\mathsf{t}})$ has length at most $|\mathbb{x}|^c$.

2.3.3 IOPs from interactive reducibility

We show that any relation with an ℓ-round interactive reduction with k predicates (and bounded output length) has a $(\ell \cdot \mathsf{k})$-round public-coin IOP with query complexity $O(\mathsf{k})$. This is a variation of the protocol described in Section 2.2, adapted to work with interactive reducibility. For simplicity, in this overview, we present the protocol only for the case $\ell = 1$ (such as the sumcheck protocol).

To aid with notation, in the description of the protocol, we replace the set S_i (which in Section 2.2.1 contained the set of all transcripts generated up until the i-th iteration) with an array A_i where $A_i[j]$ contains all of the round instances generated in the i-th iteration that are associated with the j-th predicate of the interactive reduction. In iteration i, the interactive reduction will be run k times in parallel, where for every $j \in [\mathsf{k}]$ we run given the round instances stored in $A_{i-1}[j]$.

2.3.3.1 The protocol

Let $(\mathbf{P}_{\mathsf{IR}}, \mathbf{V}_{\mathsf{IR}})$ be a one-round interactive reduction for R with k predicates. The IOP prover $\mathbf{P}_{\mathsf{IOP}}$ receives as input an instance $\mathbb{x}$ and witness $\mathbb{w}$, and the IOP verifier $\mathbf{V}_{\mathsf{IOP}}$ receives as input the instance $\mathbb{x}$. They interact as follows.

1. For every $i \in \{0, \ldots, 2\mathsf{k}\}$, $\mathbf{P}_{\mathsf{IOP}}$ defines the $(\mathsf{k}+1)$-entry array A_i as follows

$$A_i[j] := \begin{cases} \{\perp\} & \text{if } j = 0 \\ \emptyset & \text{if } j \in \{1, \ldots, \mathsf{k}\} \end{cases} \ .$$

   The set $A_i[j]$ will store all instances corresponding to f_j collected by iteration i of the protocol.
2. For $i = 1, \ldots, 2\mathsf{k}$:
   a. $\mathbf{P}_{\mathsf{IOP}}$ sends A_{i-1} to $\mathbf{V}_{\mathsf{IOP}}$.
   b. $\mathbf{V}_{\mathsf{IOP}}$ sends a random $\rho_i \leftarrow \{0,1\}^{\mathsf{r}}$ (this corresponds to a message of $\mathbf{V}_{\mathsf{IR}}$).

c. $\mathbf{P}_{\mathsf{IOP}}$ sends $a_{i,j} := \mathbf{P}_{\mathsf{IR}}(\mathbb{x}, \mathbb{w}, A_{i-1}[j-1], \rho_i)$ for all $j \in [\mathsf{k}]$, and sets $A_i[j] := A_{i-1}[j] \cup \{\mathbb{z}_{i,j}\}$ where $\mathbb{z}_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1], \rho_i, a_{i,j})$ is the output of the interactive reduction verifier given base instance $\mathbb{x}$, round instances $A_{i-1}[j-1]$, verifier randomness ρ_i, and prover reply $a_{i,j}$.

3. $\mathbf{P}_{\mathsf{IOP}}$ sends $A_{2\mathsf{k}}$ and, for every $i \in \{0, \ldots, 2\mathsf{k}\}$, sends $B_i := A_i$. This concludes the interaction.

4. In the decision phase, $\mathbf{V}_{\mathsf{IOP}}$ is given oracle access to a transcript with the following structure:

$$(A_0, \rho_1, (a_{1,1}, \ldots, a_{1,\mathsf{k}}), A_1, \ldots, \rho_{2\mathsf{k}}, (a_{2\mathsf{k},1}, \ldots, a_{2\mathsf{k},\mathsf{k}}), A_{2\mathsf{k}}, (B_0, \ldots, B_{2\mathsf{k}})) \ .$$

$\mathbf{V}_{\mathsf{IOP}}$ performs the checks below.

a. *Subset consistency.* Read the arrays $B_0, B_1, \ldots, B_{2\mathsf{k}}$ in their entirety. For every $i \in [2\mathsf{k}]$ and $j \in \{0, \ldots, \mathsf{k}\}$ check that $B_{i-1}[j] \subseteq B_i[j]$.

b. *Transcript consistency.* Sample a random $i \in [2\ell]$. Read the arrays A_{i-1} and A_i sent by $\mathbf{P}_{\mathsf{IOP}}$ and the interaction ρ_i and $(a_{i,1}, \ldots, a_{i,\ell})$.

i. Check that $A_{i-1} = B_{i-1}$ and $A_i = B_i$.

ii. For every $j \in [\mathsf{k}]$, check that $A_i[j] = A_{i-1}[j] \cup \{\mathbb{z}'_{i,j}\}$ where

$$\mathbb{z}'_{i,j} := \mathbf{V}_{\mathsf{IR}}(\mathbb{x}, A_{i-1}[j-1], \rho_i, a_{i,j}) \ ,$$

is the output of the interactive reduction verifier given base instance $\mathbb{x}$, round instances $A_{i-1}[j-1]$, verifier randomness ρ_i, and prover reply $a_{i,j}$. (Reject if $\mathbf{V}_{\mathsf{IR}}$ rejects.)

c. *Final predicate holds.* Check that $f_{\mathsf{k}}(\mathbb{x}, \mathbb{z}) = 1$ for every $\mathbb{z} \in B_{2\mathsf{k}}[\mathsf{k}]$.

2.3.3.2 Analysis

The protocol has perfect completeness and soundness error $\max\{\frac{1}{2}, \binom{2\cdot\mathsf{k}}{\mathsf{k}} \cdot \mathsf{k} \cdot \epsilon\}$ where k is the number of predicates and ϵ is the soundness of the interactive reduction respectively. This can be shown in a similar manner to that described in Section 2.2.1. The main difference between the two protocols is in the analysis of the proof length. In the protocol of Section 2.2.1 the number of sent transcripts doubled in each round. In contrast, in the new protocol, one round instance is added for each predicate. In more detail, for every $i \in [2\mathsf{k}]$ and $j \in [\mathsf{k}]$, we have $|A_i[j]| = |A_{i-1}[j]| + 1$. Since $\mathsf{k} = \text{poly}(|\mathbb{x}|)$, the total number of round instances generated and sent is polynomial in $|\mathbb{x}|$. If the interactive reduction has bounded output length, then each of these round instances has polynomially-bounded length. We can therefore conclude that the overall proof length is $\text{poly}(|\mathbb{x}|)$.

2.3.4 Relations with interactive reducibility

Several relations of interest have interactive reductions.

2.3.4.1 General IPs

We show that any relation with a k-round interactive proof has an m-round interactive reduction with k/m predicates (for any m that divides k). To see this, consider round instances $\mathbb{z}$ that are sets of j-message partial transcripts of the IP. The interactive reduction advances each of the transcripts in the set $\mathbb{z}$ by m rounds, as in the IP. The predicates are defined with respect to the "state function" of the IP, which roughly denotes whether the prover has an accepting strategy with respect to this transcript or whether no strategy will

cause the verifier to accept with high probability (over the remaining interaction). See the full version of this paper for a formal definition of the state function of an IP through the concept of round-by-round soundness.

Notice that if this interactive reduction is used in the protocol of Section 2.3.3, this yields the protocol described in Section 2.2.1. This interactive reduction does not have bounded output length since the new round instance stores all of the previous transcripts and their continuations. Therefore the resulting IOP does not achieve $O(1)$ total query complexity.

2.3.4.2 Sumcheck protocol

Using the ideas described in Section 2.3.2, we show that any relation that can be reduced into k-variate sumcheck has a one-round interactive reduction with k predicates and *bounded output length*. As a result, any relation that can be reduced into k-variate sumcheck has a $O(\mathsf{k})$-round public-coin IOP with query complexity $O(1)$.

2.3.4.3 Shamir's protocol

Shamir's protocol [21] gives an IP for all of PSPACE, thereby showing the IP = PSPACE theorem. Extending the ideas developed in Section 2.3.2, we show a one-round interactive reduction with bounded output length and polynomially-many predicates for Shamir's protocol. This establishes that every language in PSPACE has a $\mathrm{poly}(|\mathbb{x}|)$-round public-coin IOP with query complexity $O(1)$.

2.3.4.4 Future directions

We leave the exploration of what other relations have interactive reductions to future work. Following the extensive use of polynomials in both the sumcheck protocol and Shamir's protocol, it seems likely that these techniques can be adapted to also work for low-depth circuits through the delegation protocol in [15].

References

1. Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, ITCS '17, pages 11:1–11:26, 2017.
2. Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference*, ITCS '18, pages 35:1–35:14, 2018.
3. Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '17, pages 25–36, 2017.
4. Gal Arnon, Alessandro Chiesa, and Eylon Yogev. A PCP theorem for interactive proofs. Cryptology ePrint Archive, Report 2021/915, 2022. To appear at EUROCRYPT 2022.
5. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.
6. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS '92.
7. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings of the 14th Theory of Cryptography Conference*, TCC '16-B, pages 31–60, 2016.

8 Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP = PSPACE. In *Proceedings of the 30th Annual Symposium on Discrete Algorithms*, SODA '19, pages 1–20, 2019.

9 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the 30th Annual Symposium on Discrete Algorithms*, SODA '19, pages 21–40, 2019.

10 Anne Condon, Joan Feigenbaum, Carsten Lund, and Peter W. Shor. Random debaters and the hardness of approximating stochastic functions. *SIAM Journal on Computing*, 26(2):369–400, 1997.

11 Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

12 Andrew Drucker. A PCP characterization of AM. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, ICALP '11, pages 581–592, 2011.

13 Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

14 Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1/2):1–53, 2002.

15 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, 2015.

16 Ishay Haviv, Oded Regev, and Amnon Ta-Shma. On the hardness of satisfiability with bounded occurrences in the polynomial-time hierarchy. *Theory of Computing*, 3(1):45–60, 2007.

17 Ker-I Ko and Chih-Long Lin. Non-approximability in the polynomial-time hierarchy. *Technical Report 94-2, Dept. of Computer Science, SUNY at Stony Brook*, 1994.

18 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

19 Christos H. Papadimitriou. Games against nature (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '83, pages 446–450, 1983.

20 Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages 49–62, 2016.

21 Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

Finding Errorless Pessiland in Error-Prone Heuristica

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan

Mikito Nanashima
Tokyo Institute of Technology, Japan

Abstract

Average-case complexity has two standard formulations, i.e., *errorless* complexity and *error-prone* complexity. In average-case complexity, a critical topic of research is to show the equivalence between these formulations, especially on the average-case complexity of NP.

In this study, we present a relativization barrier for such an equivalence. Specifically, we construct an oracle relative to which NP is easy on average in the error-prone setting (i.e., DistNP $\subseteq$ HeurP) but hard on average in the errorless setting even by $2^{o(n/\log n)}$-size circuits (i.e., DistNP $\nsubseteq$ AvgSIZE$[2^{o(n/\log n)}]$), which provides an answer to the open question posed by Impagliazzo (CCC 2011). Additionally, we show the following in the same relativized world:

Lower bound of meta-complexity $\mathsf{GapMINKT}^{\mathcal{O}} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}}[2^{o(n/\log n)}]$ and $\mathsf{GapMCSP}^{\mathcal{O}} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}}[2^{n^{\epsilon}}]$ for some $\epsilon > 0$.

Worst-case hardness of learning on uniform distributions P/poly is not weakly PAC learnable with membership queries on the uniform distribution by nonuniform $2^n/n^{\omega(1)}$-time algorithms.

Average-case hardness of distribution-free learning P/poly is not weakly PAC learnable on average by nonuniform $2^{o(n/\log n)}$-time algorithms.

Weak cryptographic primitives There exist a hitting set generator, an auxiliary-input one-way function, an auxiliary-input pseudorandom generator, and an auxiliary-input pseudorandom function against $\mathsf{SIZE}^{\mathcal{O}}[2^{o(n/\log n)}]$.

This provides considerable insights into Pessiland (i.e., the world in which no one-way function exists, and NP is hard on average), such as the relativized separation of the error-prone average-case hardness of NP and auxiliary-input cryptography. At the core of our oracle construction is a new notion of random restriction with masks.

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

Keywords and phrases average-case complexity, oracle separation, relativization barrier, meta-complexity, learning, auxiliary-input cryptography

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.25

Funding *Shuichi Hirahara*: JST, PRESTO Grant Number JPMJPR2024, Japan.
Mikito Nanashima: JST, ACT-X Grant Number JPMJAX190M, Japan.

Acknowledgements The authors would like to thank the anonymous reviewers for many helpful comments.

1 Introduction

Average-case complexity has been studied extensively in computational complexity theory. In the theory of average-case complexity, the computational cost of solving a distributional problem (L, D) well on average is investigated, where L is a language, and D is a polynomial-time samplable distribution on instances. Average-case complexity depends on the definition of "average-case easiness," and there are at least two natural ways to formulate this: *errorless*

and *error-prone*[1] average-case easiness. In both formulations, an efficient algorithm needs to output a correct answer with high probability over a choice of random instances sampled from distribution D. The difference is in the requirement when the algorithm cannot solve an instance. In the errorless setting, the algorithm is not allowed to output a wrong answer; instead, it is allowed to output a special symbol $\perp$, which represents the failure of the algorithm. In the error-prone setting, an algorithm is allowed to output a wrong answer, provided that the error probability of the algorithm is small.

The difference between the two notions originates from two different motivations of studying average-case complexity. On one hand, Levin [29] laid the foundation of the theory of average-case complexity of NP and introduced the notion of *average-case polynomial-time*, which is equivalent to errorless heuristic schemes [24, 7]. The motivation of Levin is to clarify which distributional NP problems are hard, as some NP-complete problems are indeed easy on average with respect to natural distributions. Levin proved the distributional NP-completeness of a problem called the tiling problem. Although Levin's theory is applicable to both of the average-case notions, it is more natural to consider the notion of errorless average-case easiness in this context: Practical heuristic algorithms, such as SAT solvers, can be considered as errorless heuristics. A SAT solver is usually guaranteed to output the correct answer if it halts, but the solver may "fail" on some instances, i.e., may require a long time to halt on some instances. Levin's theory demonstrates that some distributional NP problems are hard and are unlikely to be solved by such heuristic algorithms. On the other hand, the errorless notion is not (necessarily) appropriate for discussing the security of cryptographic primitives. The foundational work of Blum and Micali [6] and Yao [44] demonstrated that error-prone average-case hardness of some distributional NP problems is useful to build cryptographic primitives. Closing the gap between the errorless and error-prone average-case notions would unify the two motivations of studying average-case complexity. In his influential paper, Impagliazzo [24] explicitly raised this question as an important research direction. The question can be formally stated as follows.

▶ **Question 1.** *Is* DistNP $\subseteq$ HeurP *equivalent to* DistNP $\subseteq$ AvgP*?*

Here, AvgP (resp. HeurP) denotes the class of distributional problems solvable on average by a polynomial-time algorithm in the errorless (resp. error-prone) setting; see Section 3.1 for a formal definition. DistNP denotes the class of distributional NP problems, i.e., DistNP $= \{(L, D) : L \in \mathsf{NP}$ and D is a polynomial-time samplable distribution$\}$.

Giving an affirmative answer to Question 1 is necessary for basing the security of cryptography on the worst-case hardness of NP. An additional motivation was recently provided by Hirahara and Santhanam [22]: they identified a deep connection between the question of errorless versus error-prone average-case complexities and the question of constructing an instance checker for NP, which is another long-standing and important open question raised in the seminal work of Blum and Kannan [5].

Despite its importance, there does not seem to be an effective method for addressing this question, so it is natural to ask whether there is a technical barrier. This meta-approach is often considered in computational complexity theory and is useful for excluding hopeless proof techniques from consideration. For example, proof techniques that are captured by standard frameworks, such as relativization [4], natural proofs [38], and algebrization [1], are known to be incapable of resolving the P versus NP question. However, to the best of our knowledge,

[1] It is originally called the "heuristic" complexity, and the term "error-prone" is due to the follow-up work [22].

there is no barrier for the question of errorless versus error-prone average-case complexities. In fact, Impagliazzo [24, 25] raised the open question of presenting a relativization barrier to Question 1.

▶ **Question 2.** *Is there an oracle* $\mathcal{O}$ *such that* $\mathsf{DistNP}^{\mathcal{O}} \not\subseteq \mathsf{AvgP}^{\mathcal{O}}$ *and* $\mathsf{DistNP}^{\mathcal{O}} \subseteq \mathsf{HeurP}^{\mathcal{O}}$*?*

The main contribution of this study is to resolve this decade-old open question affirmatively. Before presenting the details of our results, we review the recent progress in complexity theory that demonstrates the notable power of the errorless average-case easiness of NP by *relativizing* proof techniques. Along the way, we provide additional questions related to errorless versus error-prone average-case complexities. We refer to the possible world in which $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (resp. $\mathsf{DistNP} \subseteq \mathsf{HeurP}$) but $\mathsf{P} \neq \mathsf{NP}$ as *errorless Heuristica* (resp. *error-prone Heuristica*). In any relativized errorless Heuristica, the following computational tasks regarding worst-case complexity are proved to be feasible.

Errorless Heuristica I: Approximating Complexity (Meta-Complexity)

Meta-complexity is a field that studies the computational complexity of determining computational complexity. One central meta-computational problem is MINKT; for an input $(x, t) \in \{0,1\}^n \times \mathbb{N}$, MINKT is the problem of determining the minimum description length of the program that prints x in t time, i.e., the t-time-bounded Kolmogorov complexity of x. Another well-studied problem is MCSP; for an input $x \in \{0,1\}^{2^n}$ (regarded as the truth table of a function), MCSP is the problem of determining the minimum size of the n-input circuit whose truth table corresponds to x, i.e., the circuit complexity of x.

Hirahara [16] revealed that the approximation versions of the aforementioned problems are efficiently solvable in the *worst case* based on the errorless average-case easiness. For every $\sigma\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, let $\mathsf{Gap}_\sigma\mathsf{MINKT}$ denote the problem of approximating the t-time-bounded Kolmogorov complexity of $x \in \{0,1\}^n$ within an additive error term $\sigma(\cdot, n)$. For every $\epsilon \in [0,1]$, let $\mathsf{Gap}_\epsilon\mathsf{MCSP}$ denote the problem of approximating the circuit complexity of $x \in \{0,1\}^{2^n}$ within a multiplicative approximation factor $2^{(1-\epsilon)n}$. The formal definitions of these problems are presented in Section 3.2, where they are defined as promise problems. Hirahara's theorem is stated as follows.

▶ **Theorem 1** ([16])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then there exist a function* $\sigma(s, n) = \sqrt{s} \cdot \mathsf{polylog}(n)$ *and a constant* $\epsilon > 0$ *such that* $\mathsf{Gap}_\sigma\mathsf{MINKT} \in \mathsf{pr\text{-}ZPP}$ *and* $\mathsf{Gap}_\epsilon\mathsf{MCSP} \in \mathsf{pr\text{-}BPP}$*. Furthermore, these results are relativized.*[2]

Errorless Heuristica II: PAC Learning

PAC (Probably Approximately Correct) learning is one of the well studied subjects in theoretical computer science, introduced by Valiant [40]. In the PAC learning model, a learner is required to learn *all* target functions f in the target class on *all* unknown example distributions D, i.e., the learner constructs a good approximator for f from passively collected data of the form $(x, f(x))$, where each x (called an example) is selected according to D. In other words, the performance of the learner is measured by the worst-case analysis on target functions and example distributions, and this task is not directly captured as a

[2] A subsequent result [18] improved the approximation errors using a potentially non-relativizing proof technique of [9].

distributional problem. Nevertheless, Hirahara and Nanashima [20] revealed that these worst-case requirements in PAC learning are performed based on only the average-case easiness of NP under a natural computational assumption on example distributions.

▶ **Theorem 2** ([20]). *If* DistNP $\subseteq$ AvgP*, then* P/poly *is PAC learnable in polynomial time on all unknown* P/poly*-samplable example distributions. Furthermore, this result is relativized.*

Errorless Heuristica III: No Auxiliary-Input Cryptography

The aforementioned results are sufficient to break the security of any efficiently computable *auxiliary-input* cryptographic primitive, as observed in [3, 21], which is yet another notable consequence of DistNP $\subseteq$ AvgP. An auxiliary-input primitive, introduced in [36, 37], is defined as a family of primitives and has the weak security condition that at least one primitive in the family is required to be secure depending on each adversary. In other words, an adversary for an auxiliary-input primitive needs to succeed in breaking *all* primitives in the family, and this task is not captured directly as a distributional NP problem. Nevertheless, we can efficiently break any auxiliary-input cryptographic primitive in errorless Heuristica.

▶ **Theorem 3.** *If* DistNP $\subseteq$ AvgP*, then there is no auxiliary-input one-way function. Furthermore, this result is relativized.*

The three theorems mentioned above demonstrate that several fascinating tasks concerning worst-case requirements can be performed in errorless Heuristica. By contrast, there is no result which shows the feasibility of a similar task in error-prone Heuristica. Thus, there are two possibilities: the errorless condition is essential in the aforementioned results, or they can be extended by similar (especially, relativizing) proof techniques. Determining which is correct is important to understand the capability and limitation of the technique for the worst-case to average-case reduction within NP developed by Hirahara [16]. Particularly, a significant line of work [26, 30, 2, 23, 31, 32] shows the characterization of a one-way function (OWF) based on the error-prone average-case hardness of several central problems in meta-complexity, including GapMINKT and GapMCSP. Therefore, if Hirahara's reduction can be extended to error-prone average-case analogues of these problems, then OWFs is characterized by the worst-case hardness of meta-computational problems. Despite many efforts, however, extending Hirahara's reduction is currently open. Proving Theorems 1, 2, and 3 in error-prone Heuristica is one natural and necessary approach for this research direction, where we consider the stronger assumption that DistNP $\subseteq$ HeurP (instead of the non-existence of OWFs) and attempt to solve easier problems such as breaking auxiliary-input cryptography.

▶ **Question 3.** *Do Theorems 1, 2, and 3 also hold in error-prone Heuristica, i.e., under the assumption that* DistNP $\subseteq$ HeurP*? Or, is there any barrier for such research directions?*

In this study, we address these questions and study the difference between the errorless average-case complexity and the error-prone average-case complexity from the perspective of relativization.

1.1 Our results

Our main contribution is the oracle construction for separating the error-prone average-case hardness and the errorless average-case hardness for distributional NP problems. Furthermore, the proposed oracle also separates the error-prone average-case hardness and (i) the hardness

of approximating complexity (i.e., the lower bound of meta-complexity), (ii) the hardness of PAC learning, and (iii) the existence of auxiliary-input cryptographic primitives. Therefore, the proposed oracle exhibits the relativization barrier for Question 3.

We remark several points before presenting the result. When we consider the adversary defined as (a family of) circuits for some cryptographic primitives (e.g., auxiliary-input primitives and hitting set generators), we regard a size function $s(n)$ of an adversary as a function in the length of a hidden seed instead of output of the primitive for simplicity. In addition, we regard a time-bound function of a learning algorithm as a function in the length of examples, i.e., the input size to the target function.

Now, we present the main theorem. The formal definition of each notion in the statement is presented in Section 3.

▶ **Theorem 4.** *For any constant $a > 0$, there exists an oracle $\mathcal{O}$ relative to which the following hold:*

- *(Error-prone average-case easiness of* NP*)* $\mathsf{DistNP}^{\mathcal{O}} \subseteq \mathsf{HeurP}^{\mathcal{O}}$.
- *(Errorless average-case hardness of* NP*)* $\mathsf{DistNP}^{\mathcal{O}} \nsubseteq \mathsf{AvgSIZE}^{\mathcal{O}}[2^{an/\log n}]$.
- *(Lower bound of meta-complexity)* $\mathsf{Gap}_{\sigma}\mathsf{MINKT}^{\mathcal{O}} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}}[2^{an/\log n}]$ *for any* $\sigma(s,n) = o(s) \cdot \mathsf{polylog}(n)$. *In addition, for each* $\epsilon \in [0,1]$*, there exists* $\delta \in (0,1)$ *such that* $\mathsf{Gap}_{\epsilon}\mathsf{MCSP}^{\mathcal{O}} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}}[2^{n^{\delta}}]$.
- *(Worst-case hardness of learning on uniform distributions)* $\mathsf{SIZE}^{\mathcal{O}}[n]$ *is not weakly PAC learnable with membership queries (MQ) on the uniform distribution by nonuniform $O(2^{an/\log n})$-time algorithms. Furthermore, there exists a polynomial $s(n)$ such that* $\mathsf{SIZE}^{\mathcal{O}}[s(n)]$ *is not weakly PAC learnable with MQ on the uniform distribution by nonuniform $2^n/n^{\omega(1)}$-time algorithms.*
- *(Average-case hardness of distribution-free learning) There exists a polynomial $s(n)$ such that* $\mathsf{SIZE}^{\mathcal{O}}[s(n)]$ *is not weakly PAC learnable on average by nonuniform $O(2^{an/\log n})$-time algorithms. Furthermore,* $\mathsf{SIZE}^{\mathcal{O}}[n]$ *is not weakly PAC learnable on average by nonuniform $O(2^{n^{\epsilon}})$-time algorithms for some constant $\epsilon > 0$.*
- *(Relaxed cryptographic primitives) There exist a hitting set generator (HSG), an auxiliary-input one-way function (AIOWF), an auxiliary-input pseudorandom generator (AIPRG), and an auxiliary-input pseudorandom function (AIPRF) against* $\mathsf{SIZE}^{\mathcal{O}}[2^{an/\log n}]$.

The lower bound in the oracle separation is considerably stronger than the polynomial lower bound and holds for the nonuniform computation model.

Wee [42] constructed an oracle relative to which $\mathsf{DistNP} \nsubseteq \mathsf{HeurP}$, and no AIOWF exists against $\mathsf{P/poly}$, which is the opposite separation of one of our results. Combined with Wee's result, our results show that auxiliary-input cryptography and the error-prone average-case hardness of NP are incomparable by any relativizing proof.

1.2 Related Work

The study of oracle separations is initiated by Baker, Gill, and, Solovay [4] to identify the barrier for resolving the P versus NP problem. The study of the average-case complexity is initiated by Levin [29], and later it was brushed up by Impagliazzo [24], where he introduced the notion of five worlds. In the same paper, Impagliazzo first addressed the question on the difference between the errorless complexity and the error-prone complexity. Each relativized world in Impagliazzo's five worlds is found in [4, 25, 42, 27, 8]. Specifically, Impagliazzo found a relativized heuristica in which $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ but $\mathsf{NP} \nsubseteq \mathsf{SIZE}[2^{n^{\epsilon}}]$ for some $\epsilon > 0$, and Wee found a relativized pessiland in which $\mathsf{DistNP} \nsubseteq \mathsf{HeurP}$, but neither AIOWF nor OWF exists. Watson [41] also constructed a relativized world in which there is no black-box

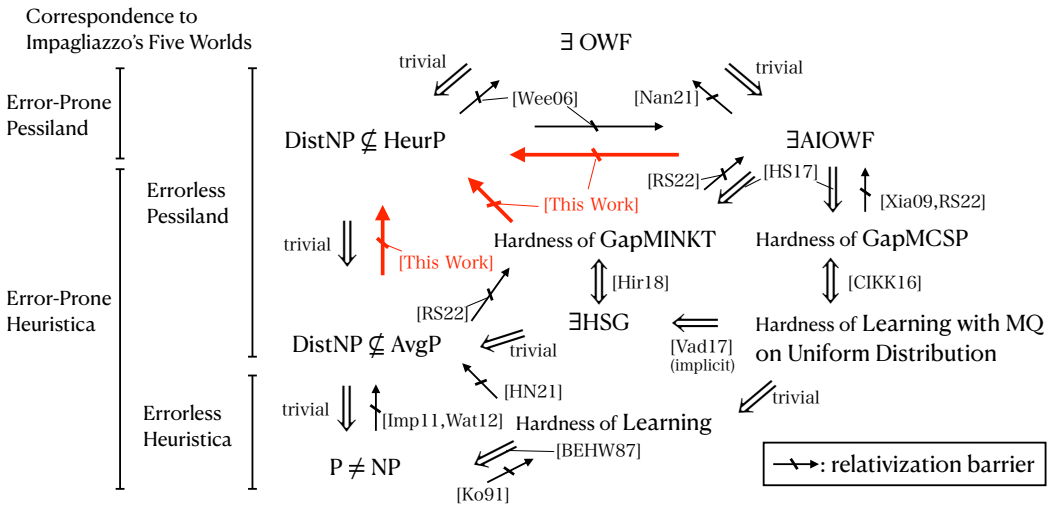


Figure 1 relativization barriers in heuristica and pessiland.

worst-case to average-case reduction for NP, but the reduction presented by Hirahara [16, 18] is non-black-box and overcomes the barrier against black-box reductions. Hirahara and Nanashima [20] improved the oracle construction proposed by Impagliazzo to the tight worst-case hardness of NP and also presented the relativized world in which DistNP $\subseteq$ AvgP, but PAC learning P/poly with MQ is sub-exponentially hard. Ko [28] showed the relativized world in which P $\neq$ NP, but a gap variant of the problem called MINLT is efficiently solvable, which is sufficient for PAC learning P/poly. Xiao [43] found the relativized world in which PAC learning P/poly with MQ is hard, but there is no AIOWF. Ren and Santhanam [39] presented various relativization barriers on the problems in meta-complexity, including the relativized world in which there is no efficient and robust reduction from distributional NP problems to the GapMINKT oracle. They also found the relativized world in which no AIOWF exists but GapMCSP and GapMINKT are hard (even in the error-prone average case). The oracle separation between AIOWF and OWF was discussed in [34]. The relationships among these oracle separation results is visualized in Figure 1.

Hirahara and Santhanam [22] also addressed the errorless complexity versus error-prone complexity problem, and they showed that the equivalence between a non-adaptive errorless to error-prone reduction for NP and an average-case instance checker for NP. They also discussed Question 1 for other classes of distributional problems such as DistPH and Dist(UP $\cap$ coUP) and showed that Dist(UP $\cap$ coUP) $\subseteq$ AvgP if and only if Dist(UP $\cap$ coUP) $\subseteq$ HeurP, i.e., they resolved Question 1 for the subclass UP $\cap$ coUP of NP.

2 Proof Techniques

We present ideas behind our oracle separation. The oracle construction is based on the one presented by Impagliazzo [25], in which the worst-case hardness and the errorless average-case easiness are separated for NP. First, we briefly review the idea and subsequently present its adjustment for the separation between the errorless average-case hardness and the error-prone average-case hardness for NP. For simplicity, we only consider the uniform distribution as the distribution over instances (instead of all sampleable distributions) and a lower bound for P/poly (instead of SIZE$[2^{an/\log n}]$) in this section.

Oracle Separation between NP $\not\subseteq$ P/poly and DistNP $\subseteq$ AvgP

The oracle construction by Impagliazzo [25] is based on the following observation: For a hidden random function $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$, the answers to most NP computations involving in f are determined by a random restriction of the truth-table of f. Therefore, by providing access to a restrictive NP oracle $\mathcal{A}$ that answers correctly only if the random restriction determines the answer (otherwise, $\mathcal{A}$ answers $\perp$), NP problems become easy on average. By contrast, we require all the information of f to perform all NP computations involving in f. Thus, NP problems remain hard in the worst-case sense in the presence of $\mathcal{A}$. We review how this idea can be implemented.

The oracle in [25] consists of two oracles $\mathcal{V}$ and $\mathcal{A}$ and a hidden internal random function[3] $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$, where $\mathcal{V}$ represents a verification oracle for the NP relation $R(x, f(x))$ which makes NP worst-case hard, and $\mathcal{A}$ represents a restrictive NP oracle which makes NP average-case easy while retaining the worst-case hardness. The NP oracle $\mathcal{A}$ is given a description of a nondeterministic oracle machine M, an input x, and a time bound T (of the form 1^{T^4} to prevent the circular call for $\mathcal{A}$), simulates $M^{\mathcal{V},\mathcal{A}}(x)$ in T time, and returns the answer, where we allow $\mathcal{A}$ to use only partial values of f on randomly selected positions. If the execution is determined only by the partial information, then $\mathcal{A}$ returns the result; otherwise, $\mathcal{A}$ returns $\perp$.

The average-case easiness of NP follows from the switching lemma for DNFs, where we regard each $f(y)_i$ as a binary variable for each input y and position i (assigned in the random selection of f) and $M^{\mathcal{V},\mathcal{A}}(x)$ (executed in T time) as a $m(n)\cdot T$-DNF formula[4]. If there is no query access to $\mathcal{A}$ by M, then the switching lemma implies that $M^{\mathcal{V}}(x)$ is determined only by the partial information of f for a large fraction of inputs x. In general cases, however, we need to take the recursive query access to $\mathcal{A}$ into account. To address this issue, we introduce a structure in f by multiple applications of random restrictions in the selection of f. Then, for a given time bound 1^{T^4}, we only apply from the first to $i_T := 2^{-1}\log\log T$-th random restrictions. If M queries $(M', x', 1^{T'^4})$ to $\mathcal{A}$ in T time, then $(T')^4 \le T$ holds. Because $i_{T'} = 2^{-1}\log\log T' \le i_T - 1$, the answer to the query to $\mathcal{A}$ is determined only by up to the $i_T - 1$-th random restriction. Thus, under an arbitrary condition on up to the (i_T-1)-th random restriction, all the answers from $\mathcal{A}$ (for executing $M^{\mathcal{V},\mathcal{A}}(x)$ in T time) are determined by the condition, and a certain DNF formula is determined regardless of query access to $\mathcal{A}$. Then, the average-case easiness follows from the switching lemma for the i_T-th random restriction (conditioned on up to the (i_T-1)-th random restriction). To apply the switching lemma for the average-case easiness, the parameter for the random restriction (i.e., the unset probability) is set to at most $n^{-\omega(1)}$ (we require a subexponentially small parameter for the subexponential lower bound for NP, as discussed in [20]).

By contrast, the worst-case hardness is shown by considering the $\mathsf{NP}^{\mathcal{V},\mathcal{A}}$ problem $L = \{\langle x, i\rangle : \exists y \text{ s.t. } \mathcal{V}(x,y) = 1 \text{ and } y_i = 1\}$ (in fact, $L \in \mathsf{UP}^{\mathcal{V},\mathcal{A}} \cap \mathsf{coUP}^{\mathcal{V},\mathcal{A}}$). Any polynomial-size circuit C can only access up to the $2^{-1}\log\log \mathsf{poly}(n) = O(\log\log n)$-th random restriction. Intuitively, if there still remain many unassigned values in the $O(\log\log n)$-th random restriction, then C should guess such values at random to find the witness $f(x)$ for L, which implies the worst-case hardness.

[3] This is a slightly modified analog of the original construction discussed in [20] for applying the standard switching lemma for DNFs in the proof instead of the switching lemma on matching variables.

[4] Specifically, the top-most $\vee$ is taken over a nondeterministic configuration path π for M and a choice of f, and each term corresponds to one choice of (π, f) such that $M^{\mathcal{V}}(x)$ accepts, where $\wedge$ in the term is applied to verify the consistency of the values of f at the (at most $m(n)\cdot T$) points $M^{\mathcal{V}}(x)$ queries.

The aforementioned oracle yields the *errorless* average-case easiness because $\mathcal{A}$ returns $\perp$ in the case in which the simulation of the given nondeterministic machine is not determined by the random restrictions. Therefore, a natural idea to separate the errorless and error-prone complexities is that we make $\mathcal{A}$ return a wrong answer in such cases. To implement this idea, the following concerns should be addressed. First, how should the answer from $\mathcal{A}$ be determined in such cases? Note that $\mathcal{A}$ cannot use the values of f assigned at higher levels in the structure to identify the wrong answer because it causes a circular problem, i.e., the DNF representing $M^{\mathcal{V},\mathcal{A}}(x)$ is not determined only by up to the $(i_T - 1)$-th random restriction anymore. Second, how should a distributional problem be determined for the errorless average-case hardness? Particularly, Hirahara and Santhanam [22] showed the equivalence between the errorless average-case easiness and the error-prone average-case easiness of $\mathsf{UP} \cap \mathsf{coUP}$ by relativizing proof techniques. Thus, we cannot hope to prove the errorless average-case hardness for the same $\mathsf{UP} \cap \mathsf{coUP}$ problem L under the error-prone average-case easiness of NP.

First Attempt for DistNP $\nsubseteq$ AvgP/poly and DistNP $\subseteq$ HeurP

The answer to the first question is relatively simple: we make $\mathcal{A}$ always answer 0. The intuition behind this is that an oracle machine given 1 as an answer from $\mathcal{A}$ (for some NP-type statement) can also obtain the witness for this assertion by the self-reducibility of NP; otherwise, the oracle machine can detect the error of $\mathcal{A}$ and output $\perp$. Thus, any error-prone algorithm can be translated into an errorless algorithm when $\mathcal{A}$ answers 1 as a wrong answer at some stage. By contrast, if $\mathcal{A}$ answers 0, i.e., declares "no witness," then there seems no efficient way to detect this error. Thus, we let $\mathcal{A}$ always answer 0, and this choice is indeed crucial in the proof.

By contrast, the answer to the second question is less obvious. Our approach is to construct a hitting set generator (HSG) instead of determining a distributional problem directly. A HSG (against $\mathsf{P/poly}$) is a (family of) efficiently computable function $G\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ which stretches the seed (i.e., $m(n) > n$) and hits any language recognized by a polynomial-size circuit. Specifically, if a polynomial-size circuit C accepts more than half of the strings in $\{0,1\}^{m(n)}$, then C also accepts $G(x)$ for some $x \in \{0,1\}^n$ (for infinitely many $n \in \mathbb{N}$). Constructing a HSG for the errorless average-case hardness is a natural approach because it immediately yields a natural distributional NP problem $(\mathrm{Im}G, \mathrm{Uniform})$ that is hard on average in the errorless setting, and Hirahara [17] demonstrated the equivalence between the errorless average-case hardness of PH and the existence of PH-computable HSGs.

A first attempt to construct a HSG is that we regard the random function $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ as a generator, where we let $m(n) > n$. Now, we replace the verification oracle $\mathcal{V}$ with $\mathcal{F}$ defined as $\mathcal{F}(x,i) = f(x)_i$ because the generator requires direct access to f for computing its values. Then, we define the candidate $G^{\mathcal{F},\mathcal{A}}\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ for a HSG as $G^{\mathcal{F},\mathcal{A}}(x) = \mathcal{F}(x,1) \circ \cdots \circ \mathcal{F}(x,m(n))\,(= f(x))$. However, this generator G is not a HSG, and G can be broken efficiently by using the partial information of f efficiently obtained from $\mathcal{A}$, informally as follows: For each random restriction, an expected fraction of unassigned values in f is $n^{-\omega(1)}$. Thus, for a given string $y \in \{0,1\}^{m(n)}$, we can easily detect the case of $y = G(x)$ for a large fraction of $x \in \{0,1\}^n$ by asking an NP-type query to $\mathcal{A}$ such as "Is there $x \in \{0,1\}^n$ such that $G(x)$ is *partially* consistent with y?" because the answer tends to be fixed to 1 only by the random restriction in $\mathcal{A}$ if such an x exists. After applying the random restrictions $\omega(1)$ times, the aforementioned strategy is sufficient for detecting all the cases of $y \in \mathrm{Im}G$. Thus, some different approach is required.

We remark that we can now regard executing a nondeterministic $M^{\mathcal{F},\mathcal{A}}(x)$ in T time as a T-DNF formula (instead of an $m(n) \cdot t$-DNF) because $\mathcal{F}$ accesses only one entry in f for each query.

Our Construction: Random Restriction with Masks

To construct a HSG, we introduce a new type of random restrictions, *random restriction with masks*, which is crucial to solve Question 2. A random restriction with masks to $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ with parameter $p \in [0,1]$ (i.e., the unset probability) is performed as follows: First, we select a random subset $S_1 \subseteq \{0,1\}^n$ of size $p \cdot 2^n$ and then apply a standard random restriction with unset probability p to a variable set $\{f(x)_i : x \in \{0,1\}^n \setminus S_1 \text{ and } i \in [m(n)]\}$, i.e., the random set S_1 performs as a "mask" that prevents restriction. This variant of random restriction is extended to multiple applications inductively as follows: Let S_i be the random subset (i.e., the mask) selected in the i-th random restriction with masks to f. Next, the $(i+1)$-th restriction (with parameter p) is performed by selecting a random subset $S_{i+1} \subseteq S_i$ of size $p \cdot |S_i|$ and applying random restriction to variables except for S_{i+1}.

We consider a modified oracle in which the oracle construction is the same as previously mentioned except that we apply random restrictions with masks instead of the standard random restrictions. For now, we select the unset probability $p(n) = n^{-\log n}$. This choice is sufficient for a HSG against $\mathsf{P/poly}$ and the statement that $\mathsf{DistNP} \nsubseteq \mathsf{AvgP/poly}$. Note that $p(n)$ should be selected more carefully according to the size complexity of the adversary in the formal argument (for the detail, see Section 4).

Specifically, we randomly select the aforementioned oracles $\mathcal{F}$ and $\mathcal{A}$ by selecting the internal random function $f\colon \{0,1\}^n \to \{0,1\}^{m(n)}$ with $\log n$ applications of random restrictions with masks for each $n \in \mathbb{N}$ (after applying random restrictions, we also select the remaining values of f at random). For each $n \in \mathbb{N}$, let $S_{n,\log n} \subseteq \{0,1\}^n$ be the random mask selected in the $\log n$-th restriction. Then, we have $|S_{n,\log n}| = p(n)^{\log n} \cdot 2^n = 2^{n-(\log n)^3}$. Thus, there exist exponentially many $z \in S_{n,\log n} \subseteq \{0,1\}^n$ (we call these hard indices) such that no value in $f(z)$ is assigned by the $\log n$-th restriction. Remember that for a query $(M, x, 1^{T^4})$, the oracle $\mathcal{A}$ applies only up to the $i_T := 2^{-1} \log\log T$-th random restriction. Since any polynomial-size adversary C can make a query only with $T = \mathsf{poly}(n)$, C can only access up to the $O(\log\log n)$-th restrictions. Therefore, any polynomial-size adversary cannot obtain any information about $f(z)$ from $\mathcal{A}$ for each hard index z, and oracle access to $\mathcal{F}(z,i) = f(z)_i$ is indistinguishable from access to a random function for such adversaries.

The aforementioned argument is sufficient for constructing a HSG. In fact, by defining the generator G as $G^{\mathcal{F},\mathcal{A}}(x) = \mathcal{F}(x,1) \circ \cdots \circ \mathcal{F}(x, m(n))$, we can show that G is a HSG against $\mathsf{P/poly}$ by a similar argument as in [20]. Furthermore, the random restriction method with masks has another advantage: even if we select exponentially large $m(n)$, it still provides hard indices z such that $\mathcal{A}$ does not reveal any information of $f(z)$ to polynomial-size adversaries. Specifically, by letting $m(n) = 2^n \cdot n$ (i.e., the length of the truth table of a mapping from n-bit to n-bit), we can prepare an auxiliary-input oracle $\mathcal{F}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that $\mathcal{F}(z,\cdot)$ is (computationally) indistinguishable from a random oracle for subexponentially many hard z's. Therefore, $F(\cdot,\cdot)$ is an AIOWF because it is known that a random oracle is also a OWF with probability 1 over the choice of the random oracle (cf. [27, 11]). Furthermore, by the technique presented in [45], we can construct an AIPRG based on the auxiliary-input analog of a random oracle with less security loss than the general methods to convert OWFs into PRGs (e.g., [15]). In the formal proof of Theorem 4, we first construct such an AIPRG and subsequently show the related hardness notion (e.g., HSGs and the hardness of learning)

to prevent security loss. Note that the aforementioned argument does not yield standard cryptographic primitives such as OWFs because the set of hard indices are selected at random, and there is no efficient sampling algorithm that selects a hard index with high probability.

A random restriction with masks assigns fewer variables than a standard random restriction. Therefore, the remaining problem is whether the error-prone average-case easiness is preserved in the modified oracle construction. This issue can be addressed by the choice of the answer (i.e., 0) from $\mathcal{A}$ when the simulation is not determined by random restrictions. The proof is outlined as follows (for the formal proof, see Section 5).

For convenience, we regard that a random restriction with masks is performed as follows: (i) a standard random restriction is applied to remaining variables at the stage, (ii) the random subset $S_{i+1} \subseteq S_i$ is selected in the same manner, and (iii) the values in $f(z)$ are returned to unassigned for each $z \in S_{i+1}$. Let us call the first step (resp. the second and third steps) a *restriction* (resp. *reverse*) step. The random restriction in the restriction step is merely a standard one. Thus, by the standard switching lemma, we can show that the value of a T-DNF ϕ (representing the execution of a nondeterministic machine in T time) is determined with high probability at this stage. Therefore, it is sufficient to show that the answer from $\mathcal{A}$ rarely changes in the reverse step.

For simplicity, we use the notation $*$ to refer to the cases in which the DNF ϕ is not fixed by the random restriction. Then, there are $3 \times 3 = 9$ possibilities about the change in the state on the restricted ϕ, i.e., from $\{0, 1, *\}$ (in the restriction step) to $\{0, 1, *\}$ (in the reverse step). Obviously, we do not need to consider the following 3 cases: $\{0\} \to \{0\}$, $\{1\} \to \{1\}$, and $\{*\} \to \{*\}$. Since we cancel some assignments in the reverse step, the following 4 cases do not occur: $\{*\} \to \{0, 1\}$, $\{0\} \to \{1\}$, and $\{1\} \to \{0\}$. Furthermore, because $\mathcal{A}$ answers 0 in the case of $*$, we do not need to consider the case of $\{0\} \to \{*\}$. Therefore, the remaining case is only $\{1\} \to \{*\}$.

We show that the case of $\{1\} \to \{*\}$ rarely occurs as follows. Since the T-DNF formula ϕ is satisfied in the restriction step, there must exist a satisfied term τ of size T. If ϕ becomes unfixed in the reverse step, then τ is also unfixed. This event occurs only if there exists $z \in \{0, 1\}^n$ such that some variable $f(z)_i$ is contained in τ (for some i), and z is selected on the choice of the random subset in the reverse step. Since τ covers at most T indices z in literals, this probability is at most $T \cdot p(n) = T \cdot n^{-\log n}$. Particularly, for solving a NP problem by $\mathcal{A}$, we only need to simulate a nondeterministic machine in $\mathsf{poly}(n)$ times, so we can let $T = \mathsf{poly}(n)$. Therefore, the error probability that the answer from $\mathcal{A}$ is changed in the reverse step is negligible.

Limitations of Our Technique and Future Direction

We remark that the aforementioned argument above heavily relies on the characteristics of DNFs (i.e., nondeterministic machines). Currently, it is unclear whether the proposed argument can be extended to a general case of constant-depth circuits, even for depth-3 $\wedge$-$\vee$-$\wedge$-circuits (which corresponds to Π_2^p). By contrast, the oracle separation between the worst-case hardness and the errorless average-case easiness for NP in [25] is naturally extended for PH, as explicitly discussed in [20] by considering the switching lemma for constant-depth circuits. Therefore, we pose the following open question for the further research on the difference between the errorless and error-prone average-case complexity.

▶ **Question 4.** *Is there any oracle $\mathcal{O}$ relative to which* $\mathsf{DistNP}^{\mathcal{O}} \not\subseteq \mathsf{AvgP}^{\mathcal{O}}/\mathsf{poly}$ *and* $\mathsf{DistPH}^{\mathcal{O}} \subseteq \mathsf{HeurP}^{\mathcal{O}}$*? Or, is there a relativizing proof which shows that* $\mathsf{DistPH} \subseteq \mathsf{HeurP} \implies \mathsf{DistNP} \subseteq \mathsf{AvgP}/\mathsf{poly}$*?*

The fact that we failed to extend our results to PH might suggest the feasibility of proving $\mathsf{DistPH} \subseteq \mathsf{HeurP} \Longrightarrow \mathsf{DistNP} \subseteq \mathsf{AvgP/poly}$. Furthermore, we failed to improve our lower bound $2^{o(n/\log n)}$ on the time complexity of errorless average-case algorithms to $2^{o(n)}$. In this light, we conjecture that the worst-case-to-average-case connection of Hirahara [19], which shows that $\mathsf{DistNP} \subseteq \mathsf{AvgP} \Longrightarrow \mathsf{UP} \subseteq \mathsf{DTIME}(2^{O(n/\log n)})$, can be extended to the error-prone average-case complexity by using a relativizing proof.

▶ **Conjecture 5.** *For every oracle $\mathcal{O}$, if $\mathsf{DistNP}^{\mathcal{O}} \subseteq \mathsf{HeurP}^{\mathcal{O}}$, then $\mathsf{UP}^{\mathcal{O}} \subseteq \mathsf{BPTIME}^{\mathcal{O}}[2^{O(n/\log n)}]$.*

3 Preliminaries

For each $n \in \mathbb{N}$, let $[n] = \{1, \dots, n\}$. For each $x \in \{0,1\}^n$ and $i \in [n]$, we let x_i denote the i-th bit of x and $x_{\leq i}$ denote $x_1 \circ \cdots \circ x_i$. For a distribution D, we write $x \leftarrow D$ to refer to a random sampling x according to D. For a finite set S, we also use the notation $x \leftarrow_u S$ to denote the uniform sampling from S. For each $n \in \mathbb{N}$, we let U_n denote the uniform distribution over $\{0,1\}^n$ or a random variable selected uniformly at random from $\{0,1\}^n$ in context. We use the notation negl to represent a certain negligible function, i.e., for any polynomial $p(n)$, $\mathsf{negl}(n) < 1/p(n)$ for sufficiently large $n \in \mathbb{N}$. For a randomized algorithm A using $r(n)$ random bits on an n-bit input, we use $A(x;s)$ to refer to the execution of $A(x)$ with a random tape s for $x \in \{0,1\}^n$ and $s \in \{0,1\}^{r(n)}$.

For any oracles $\mathcal{O}_0$ and $\mathcal{O}_1$, we let $\mathcal{O}_0 + \mathcal{O}_1$ denote the combination, i.e., for any $b \in \{0,1\}$ and any $x \in \{0,1\}^*$, $(\mathcal{O}_0 + \mathcal{O}_1)(b \circ x) = \mathcal{O}_b(x)$.

In this paper, we assume the basic knowledge of probability theory, including the union bound, Markov's inequality, Hoeffding's inequality, and the Borel–Cantelli lemma.

For each $p \in [0,1]$ and set S of variables taking binary values, we define a p-random restriction ρ to S as a partial assignment $\rho \colon S \to \{0,1,*\}$ (where $*$ represents "unassigned") randomly selected as follows: for each $x \in S$,

$$\rho(x) = \begin{cases} * & \text{with probability } p \\ 0 & \text{with probability } (1-p)/2 \\ 1 & \text{with probability } (1-p)/2. \end{cases}$$

For every restriction ρ to S and function f defined on S, we let $f|_\rho$ denote the restricted function obtained by applying a partial assignment to f according to ρ.

3.1 Average-Case Complexity

We present the notions in average-case complexity theory. Further backgrounds can be found in a survey [7].

We say that a family $D = \{D_n\}_{n\in\mathbb{N}}$ of distributions, where each D_n is a distribution on $\{0,1\}^n$, is (polynomial-time) samplable if there exists a randomized sampling algorithm S such that the distribution of $S(1^n)$ is identical to D_n for each $n \in \mathbb{N}$. We consider a family of distributions as a single distribution on instances. We define a distributional problem as a pair of a language $L \subseteq \{0,1\}^*$ and a distribution $D = \{D_n\}_{n\in\mathbb{N}}$ on instances. For a standard complexity class $\mathcal{C}$ (e.g., NP), we define its average-case extension $\mathsf{Dist}\mathcal{C}$ as $\mathsf{Dist}\mathcal{C} = \{(L, D) : L \in \mathcal{C}, D \text{ is samplable}\}$.

We present the *errorless* average-case easiness. We say that a distributional problem (L, D) has an *errorless* heuristic algorithm A with failure probability $\epsilon \colon \mathbb{N} \to (0,1)$ if (1) A outputs $L(x)$ $(:= \mathbb{1}\{x \in L\})$ or $\perp$ (which represents "failure") for every $x \in \mathrm{supp}(D)$, and (2)

the failure probability that $A(x)$ outputs $\perp$ over the choice of $x \leftarrow D$ is bounded above by $\epsilon(n)$ for each $n \in \mathbb{N}$. Note that an errorless heuristic algorithm never outputs an incorrect value $\neg L(x)$ for any $x \in \mathrm{supp}(D)$. Then, for every $\epsilon : \mathbb{N} \to (0,1)$, we define a class $\mathsf{Avg}_\epsilon\mathsf{P}$ as a class of distributional problems that have a polynomial-time errorless heuristic algorithm with failure probability $\epsilon(n)$. Furthermore, we say that a distributional problem (L, D) has an *errorless* heuristic scheme A if A is given an instance $x \in \mathrm{supp}(D)$ and $\epsilon \in (0,1)$ as input and satisfies the condition of an errorless heuristic algorithm with failure probability ϵ. We define a class AvgP as a class of distributional problems that have a polynomial-time errorless heuristic scheme. It is not hard to verify that $\mathsf{AvgP} \subseteq \mathsf{Avg}_{1/p(n)}\mathsf{P}$ for any polynomial $p(n)$.

Next, we present the *error-prone* average-case easiness. We say that a distributional problem (L, D) has an *error-prone* heuristic algorithm A with failure probability $\epsilon\colon \mathbb{N} \to (0,1)$ if the failure probability that $A(x) \neq L(x)$ over the choice of $x \leftarrow D$ is bounded above by $\epsilon(n)$ for each $n \in \mathbb{N}$. Note that an error-prone heuristic algorithm may output an incorrect value $\neg L(x)$, but the error probability is bounded above by $\epsilon(n)$. Then, for every $\epsilon : \mathbb{N} \to (0,1)$, we define a class $\mathsf{Heur}_\epsilon\mathsf{P}$ as a class of distributional problems that have a polynomial-time error-prone heuristic algorithm with failure probability $\epsilon(n)$. We also define an *error-prone* heuristic scheme and the class HeurP in the same manner as the errorless case.

We also define classes $\mathsf{AvgP/poly}$, $\mathsf{HeurP/poly}$, $\mathsf{AvgSIZE}[s(n)]$, and $\mathsf{HeurSIZE}[s(n)]$ for each size parameter $s(n)$ in the same manner as above.

3.2 Meta-Complexity

Next, we define problems $\mathsf{GapMINKT}$ and $\mathsf{GapMCSP}$ formally. In this study, we fix a universal Turing machine U arbitrarily to specify the Kolmogorov complexity.

▶ **Definition 6** (Kolmogorov complexity, $\mathsf{GapMINKT}$). *For each $t \in \mathbb{N}$ and $x \in \{0,1\}^*$, we define the t-time-bounded Kolmogorov complexity $\mathsf{K}^t(x)$ of x as*

$$\mathsf{K}^t(x) = \min_{p \in \{0,1\}^*} \{|p| : U(p) \text{ outputs } x \text{ in } t \text{ time}\}.$$

We also define $\mathsf{K}(x)$ by $\mathsf{K}(x) = \lim_{t\to\infty} \mathsf{K}^t(x)$.

For a function $\sigma\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\mathsf{Gap}_\sigma\mathsf{MINKT}$ is a promise problem (Π_Y, Π_N) defined as $\Pi_Y = \left\{(x, 1^s, 1^t) : \mathsf{K}^t(x) \leq s\right\}$ and $\Pi_N = \{(x, 1^s, 1^t) : \mathsf{K}(x) > s + \sigma(s, |x|)\}$[5].

▶ **Definition 7** (Circuit complexity, $\mathsf{GapMCSP}$). *For each $n \in \mathbb{N}$ and $x \in \{0,1\}^{2^n}$, we define the circuit complexity $\mathsf{cc}(x)$ of x as the minimum size of an n-input circuit whose truth table corresponds to x.*

For a constant $\epsilon \in [0,1]$, $\mathsf{Gap}_\epsilon\mathsf{MCSP}$ is a promise problem (Π_Y, Π_N) defined as $\Pi_Y = \{(x, 1^s) : n \in \mathbb{N}, x \in \{0,1\}^{2^n}, \mathsf{cc}(x) \leq s\}$ and $\Pi_N = \{(x, 1^s) : n \in \mathbb{N}, x \in \{0,1\}^{2^n}, \mathsf{cc}(x) > 2^{(1-\epsilon)n} \cdot s\}$.

3.3 Learning

We define a concept class as a subset of $\{f\colon \{0,1\}^n \to \{0,1\} : n \in \mathbb{N}\}$. For any concept class $\mathcal{C}$ and $n \in \mathbb{N}$, we let $\mathcal{C}_n$ represent $\mathcal{C} \cap \{f\colon \{0,1\}^n \to \{0,1\}\}$. Then, the (weak) PAC learning model with membership queries (MQ) is defined as follows. We refer to a family $D = \{D_n\}_{n\in\mathbb{N}}$, where each D_n is a distribution on $\{0,1\}^n$, as an example distribution in the learning context.

[5] Note that this formulation of $\mathsf{GapMINKT}$ has a relaxed (i.e., easier) requirement than one discussed in [16, 18], in the sense that we do not consider the time-bound in "no" cases. In other words, we only need to distinguish efficiently generated strings from strings no short program can generate even in time-unbounded settings.

▶ **Definition 8** (PAC learning [40]). *Let $\mathcal{C}$ be a concept class, D be an example distribution, and $t\colon \mathbb{N} \to \mathbb{N}$ be a time-bound function. We say that a (possibly nonuniform) randomized oracle machine L, referred to as a weak learner, (weakly) learns $\mathcal{C}$ with MQ on D in time $t(n)$ if L satisfies the following conditions for some polynomial $p(n)$:*

1. *L is given $n \in \mathbb{N}$ as the input and oracle access to $\mathsf{EX}_{f,D}$ (called an example oracle) and MQ_f (called a membership query oracle), which are determined by a target function $f \in \mathcal{C}_n$ and the example distribution D.*
2. *For each access (with no input), $\mathsf{EX}_{f,D}()$ returns an example of the form $(x, f(x))$, where x is selected identically and independently according to D_n. Furthermore, for each access with input $x \in \{0,1\}^n$, $\mathsf{MQ}_f(x)$ returns $f(x)$.*
3. *For each $n \in \mathbb{N}$ and target function $f\colon \{0,1\}^n \to \{0,1\}$, L outputs a circuit $h\colon \{0,1\}^n \to \{0,1\}$ that is $(\frac{1}{2} - \frac{1}{p(n)})$-close to f under D with probability at least $2/3$, i.e., L satisfies the following condition:*

$$\Pr_{L,\mathsf{EX}}\left[L^{\mathsf{EX}_{f,D},\mathsf{MQ}_f}(n) \text{ outputs } h \text{ such that } \Pr_{x\leftarrow D}[h(x) \neq f(x)] \leq \frac{1}{2} - \frac{1}{p(n)}\right] \geq \frac{2}{3}.$$

4. *$L^{\mathsf{EX}_{f,D},\mathsf{MQ}_f}(n)$ halts in time $t(n)$ for each $n \in \mathbb{N}$.*

We say that a concept class $\mathcal{C}$ is weakly PAC learnable with MQ on D in $t(n)$ time if there exists a $t(n)$-time weak learner for $\mathcal{C}$.

When the example distribution is uniform, a randomized learner can simulate EX based on MQ and its randomness with no loss of time complexity. Thus, we ignore EX in learning on the uniform distribution without loss of generality.

We also define an average-case analog of PAC learning, in which a target function is randomly selected according to some fixed distribution F (called a target distribution). By contrast, we consider the distribution-free setting on example distributions, formally, as follows:

▶ **Definition 9** (learning on average). *Let $\mathcal{C}$ be a concept class, $t\colon \mathbb{N} \to \mathbb{N}$ be a time-bound function, and $F = \{F_n\}_{n\in\mathbb{N}}$ be a target distribution, where each F_n is a distribution on $\mathcal{C}_n$. We say that a (possibly nonuniform) randomized oracle machine L, referred to as a weak learner, (weakly) learns $\mathcal{C}$ on average with respect to F in time $t(n)$ if L satisfies the following condition for some polynomial $p(n)$: For any $n \in \mathbb{N}$ and example distribution D_n on $\{0,1\}^n$, $L^{\mathsf{EX}_{f,D_n}}(n)$ halts in time $t(n)$ (for each $f \in \mathrm{supp}(F_n)$) and*

$$\Pr_{f\leftarrow F_n}\left[\Pr_{L,\mathsf{EX}_{f,D}}\left[L^{\mathsf{EX}_{f,D}}(n) \text{ outputs a circuit } h \text{ such that } \Pr_{x\leftarrow D}[h(x) \neq f(x)] \leq \frac{1}{2} - \frac{1}{p(n)}\right] \geq \frac{2}{3}\right] \geq \frac{1}{p(n)}.$$

We say that a concept class $\mathcal{C}$ is not weakly PAC learnable on average in $t(n)$ time if there exists a polynomial-time samplable target distribution F such that there is no $t(n)$-time weak learner that satisfies the condition above with respect to F.

3.4 Cryptography

We introduce cryptographic primitives. Let $\mathcal{C}$ be a complexity class of adversaries (e.g., $\mathsf{P/poly}$). We regard the complexity parameter (e.g., time and size) on $\mathcal{C}$ as a function in the size of a hidden seed for primitives.

▶ **Definition 10** (auxiliary-input one-way function). *Let $n, m\colon \mathbb{N} \to \mathbb{N}$ be polynomials. We say that $f = \{f_z\colon \{0,1\}^{n(|z|)} \to \{0,1\}^{m(|z|)}\}_{z\in\{0,1\}^*}$ is an auxiliary-input one-way function (AIOWF) against $\mathcal{C}$ if each $f_z(x)$ is polynomial-time computable from (z,x), and for any adversary A in $\mathcal{C}$, there exists an infinite subset $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$,*

$$\Pr\left[f_z(A(z, f_z(U_{n(|z|)}))) = f_z(U_{n(|z|)})\right] < \mathsf{negl}(|z|).$$

▶ **Definition 11** (auxiliary-input pseudorandom generator). *Let $n, m\colon \mathbb{N} \to \mathbb{N}$ be polynomials. We say that $G = \{G_z\colon \{0,1\}^{n(|z|)} \to \{0,1\}^{m(|z|)}\}_{z\in\{0,1\}^*}$ is an auxiliary-input pseudorandom generator (AIPRG) against $\mathcal{C}$ if each $G_z(x)$ is polynomial-time computable from (z,x), $n(\ell) < m(\ell)$ holds for any $\ell \in \mathbb{N}$, and for any adversary A in $\mathcal{C}$, there exists an infinite subset $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$,*

$$\left|\Pr\left[A(z, G_z(U_{n(|z|)})) = 1\right] - \Pr\left[A(z, U_{m(|z|)}) = 1\right]\right| < \mathsf{negl}(|z|).$$

▶ **Definition 12** (auxiliary-input pseudorandom function). *We say that $F = \{F_z\colon \{0,1\}^{|z|} \times \{0,1\}^{|z|} \to \{0,1\}\}_{z\in\{0,1\}^*}$ is an auxiliary-input pseudorandom function (AIPRF) against $\mathcal{C}$ if each F_z is polynomial-time computable from z and its input, and for any adversary $A^?$ in (an oracle machine analog of) $\mathcal{C}$, there exists an infinite subset $Z_A \subseteq \{0,1\}^*$ such that for every $z \in Z_A$,*

$$\left|\Pr_{A,u\sim\{0,1\}^{|z|}}\left[A^{F_z(u,\cdot)}(z) = 1\right] - \Pr_{A,\phi_{|z|}}\left[A^{\phi_{|z|}(\cdot)}(z) = 1\right]\right| < \mathsf{negl}(|z|),$$

where $\phi_{|z|}\colon \{0,1\}^{|z|} \to \{0,1\}$ denotes a truly random function.

When the auxiliary-input z is obvious in context, we write $n(|z|)$ and $m(|z|)$ as n and m, respectively.

▶ **Definition 13** (hitting set generator). *Let $\ell, m\colon \mathbb{N} \to \mathbb{N}$ be polynomials. We say that $G = \{G_n\}_{n\in\mathbb{N}}$, where $G_n\colon \{0,1\}^{\ell(n)} \to \{0,1\}^{m(n)}$ is a hitting set generator (HSG) against $\mathcal{C}$ if each G_n is polynomial-time computable, $\ell(n) < m(n)$ holds for each $n \in \mathbb{N}$, and G hits any language recognized by adversaries in $\mathcal{C}$ in the following sense: For any adversary A in $\mathcal{C}$, let $L_A = \{L_{A,n}\}_{n\in\mathbb{N}}$ be a language recognized by A, where $L_{A,n} \subseteq \{0,1\}^n$ for each $n \in \mathbb{N}$. Then, for infinitely many $n \in \mathbb{N}$, the following holds:*

$$|L_{A,m(n)}| > 2^{m(n)-1} \Longrightarrow L_{A,m(n)} \cap \mathrm{Im} G_n \neq \emptyset.$$

4 Oracle Construction

In this section, we formally present the proposed oracle construction. Let $a > 0$ be a parameter.

▶ **Construction.** *$\mathcal{O}_a = \mathcal{F} + \mathcal{A}$, where each oracle is randomly selected according to the following process:*

1. *Let $t(n) = 2^{an/\log n}$ be the upper bound on the time of nonuniform adversaries and $c = 7a$.*
2. *Define functions p and $i_{\max}$ as $p(n) = t(n)^{-6}$ and $i_{\max}(n) = \frac{1}{c}\log\log t(n)$. Here, p is a parameter of random restriction, and $i_{\max}$ is the number of applications of random restrictions.*
3. *For each $n \in \mathbb{N}$, define a set $V_{n,0}$ of variables taking binary values as follows:*

$$V_{n,0} = \{F_{z,x,\ell} : z, x \in \{0,1\}^n, \ell \in [n]\}.$$

4. *For each* $n \in \mathbb{N}$, *let* $S_{n,0} = \{0,1\}^n$.
5. *For each* $n \in \mathbb{N}$ *and* $i \in [i_{\max}(n)-1]$, *we inductively (on* i*) select a* $p(n)$*-random restriction* $\rho^*_{n,i}$ *to* $V_{n,i-1}$ *and a random subset* $S_{n,i} \subseteq S_{n,i-1}$ *of size* $p(n) \cdot |S_{n,i-1}|$. *Then, we define a restriction* $\rho_{n,i}$ *to* $V_{n,i-1}$ *and a subset* $V_{n,i} \subseteq V_{n,i-1}$ *as follows:*

$$\rho_{n,i}(z,x,\ell) = \begin{cases} * & \text{if } z \in S_{n,i} \\ \rho^*_{n,i}(z,x,\ell) & \text{otherwise} \end{cases}$$

$$V_{n,i} = \rho^{-1}_{n,i}(*) \left(= {\rho^*_{n,i}}^{-1}(*) \cup \{F_{z,x,\ell} : z \in S_{n,i}\}\right).$$

   We also define $\rho_{n,i_{\max}(n)}$ *as a full assignment to* $V_{n,i_{\max}(n)-1}$ *selected uniformly at random. Let* $\rho_{n,i} \equiv \rho_{n,i_{\max}(n)}$ *for each* $i \geq i_{\max}(n)+1$. *We use the notation* $\rho_{n,\leq i}$ *to represent the composite restriction* $\rho_{n,1} \cdots \rho_{n,i}$ *to* $V_{n,0}$ *for each* n *and* i.
6. *Define* $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$, *where* $\mathcal{F}_n \colon \{0,1\}^n \times \{0,1\}^n \times [n] \to \{0,1\}$, *as* $\mathcal{F}_n(z,x,\ell) = \rho_{n,\leq i_{\max}(n)}(z,x,\ell)$.
7. *Define* $\mathcal{A}$ *as follows: On input* $(M,x,1^{T^{2^c}})$, *where* M *is a nondeterministic oracle machine,* $x \in \{0,1\}^*$, *and* $T \in \mathbb{N}$, *the oracle* $\mathcal{A}(M,x,1^{T^{2^c}})$ *returns* 0 *or* 1 *according to the following procedure:*
   1: *Let* $i_T := \frac{1}{c}\log\log T$.
   2: *Construct a* T*-DNF* ϕ *on variables in* $V_{n,0}$ *representing the execution of* $M^{\mathcal{F}+\mathcal{A}}(x)$ *in* T *steps, where the top-most OR corresponds to the nondeterminism on a possible choice of* $\mathcal{F}$ *(say,* $\mathcal{F}'$*) and an accepting path of* $M^{\mathcal{F}'+\mathcal{A}}(x)$, *and each term performs verification whether* M*'s at most* T *queries (say,* $(z_1,x_1,\ell_1),\ldots,(z_q,x_q,\ell_q)$ *for some* $q \leq T$*) are consistent with the actual choices of* $\mathcal{F}$, *i.e., for each* $i \in [q]$, *the term contains* F_{z_i,x_i,ℓ_i} *if* $\mathcal{F}'(z_i,x_i,\ell_i) = 1$*; otherwise,* $\neg F_{z_i,x_i,\ell_i}$ *as a literal.*
   3: *If* $\phi|_{\rho_{1,\leq i_T},\ldots,\rho_{T,\leq i_T}} \equiv b$ *for some* $b \in \{0,1\}$, *then* ***return*** b, *otherwise,* ***return*** 0.

   We can verify that $\mathcal{A}$ is well-defined (i.e., not circular on recursive calls for $\mathcal{A}$) as follows.

▶ **Proposition 14.** *For each input, the value of* $\mathcal{A}(M,x,1^{T^{2^c}})$ *is determined only by* $\rho_{n,j}$ *(equivalently,* $\rho^*_{n,j}$ *and* $S_{n,j}$*) for* $n \leq T$ *and* $j \leq i_T$ *(remember that* $i_T = \frac{1}{c}\log\log T$*).*

Proof. We show the proposition by induction on T. Remember that, on input $(M,x,1^{T^{2^c}})$, the oracle $\mathcal{A}$ first makes a T-DNF ϕ based on M independently of the values of $\mathcal{F}$.

Suppose that M makes some query $(M',x',1^{T'^{2^c}})$ to $\mathcal{A}$ for constructing ϕ. Since the length of such a query is at most T, we have $T'^{2^c} \leq T$ and

$$i_{T'} = \frac{1}{c}\log\log T' \leq \frac{1}{c}\log\log T^{\frac{1}{2^c}} = \frac{1}{c}\log\log T - 1 = i_T - 1.$$

By the induction hypothesis, the answer of $\mathcal{A}(M',x',1^{T'^{2^c}})$ is determined by only $\rho_{n,j}$ for $n \leq T'$ and $j \leq i_T - 1$, and so is ϕ. Then, $\mathcal{A}$ determines the answer by restricting ϕ by $\rho_{n,j}$ for $n \leq T$ and $j \leq i_T$. Therefore, $\mathcal{A}(M,x,1^{T^{2^c}})$ is determined only by $\rho_{n,j}$ for $n \leq T$ and $j \leq i_T$. ◀

When the parameter a is clear from the context, we omit the subscript a from $\mathcal{O}_a$.

5 Error-Prone Average-Case Easiness of NP

In this section, we show the error-prone average-case easiness of NP.

▶ **Theorem 15.** *With probability 1 over the choice of* $\mathcal{O}_a$, $\mathsf{DistNP}^{\mathcal{O}_a} \subseteq \mathsf{HeurP}^{\mathcal{O}_a}$ *holds.*

First, we introduce several notations. For each choice of $\mathcal{O}$, we define the oracle $\mathcal{A}^*$ in the same manner as $\mathcal{A}$ except we apply $\rho_{1,\le i_T-1}\rho^*_{1,i_T},\ldots,\rho_{T,\le i_T-1}\rho^*_{T,i_T}$ to ϕ instead of $\rho_{1,\le i_T},\ldots,\rho_{T,\le i_T}$. Note that $\mathcal{A}^*$ executes a given nondeterministic machine M with access to $\mathcal{A}$ (rather than $\mathcal{A}^*$) to construct the corresponding DNF ϕ. We can verify that $\mathcal{A}^*$ is well-defined (i.e., not circular) in the same manner as Proposition 14.

Now, we show that $\mathcal{A}$ and $\mathcal{A}^*$ do not differ considerably.

▶ **Lemma 16.** *For each input $(M,x,1^{T^{2^c}})$ to $\mathcal{A}$, we have that*

$$\Pr_{\mathcal{O}}\left[\mathcal{A}(M,x,1^{T^{2^c}})\neq\mathcal{A}^*(M,x,1^{T^{2^c}})\right]=O(T^{-4}).$$

Proof. Let $i:=i_T=(1/c)\log\log T$. For all $n\le T$ and $j\le i-1$, we fix $\rho^*_{n,j}$, $S_{n,j}$, and $\rho^*_{n,i}$ arbitrarily; let C_T denote this condition. Notice that the DNF formula ϕ_{C_T} constructed by $\mathcal{A}$ and $\mathcal{A}^*$ is determined only by C_T because all answers to recursive calls for $\mathcal{A}$ are determined by C_T as in Proposition 14. Let $\phi'_{C_T}=\phi_{C_T}|_{\rho_{1,\le i-1},\ldots,\rho_{T,\le i-1}}$. Then, the value of $\mathcal{A}(M,x,1^{T^{2^c}})$ (resp. $\mathcal{A}^*(M,x,1^{T^{2^c}})$) is determined by $\phi'_{C_T}|_{\rho_{1,i},\ldots,\rho_{T,i}}$ (resp. $\phi'_{C_T}|_{\rho^*_{1,i},\ldots,\rho^*_{T,i}}$).

For any DNF formula ϕ and a restriction ρ, there are the following three cases: (i) $\phi|_\rho\equiv 0$, (ii) $\phi|_\rho\equiv 1$, or (iii) $\phi|_\rho$ does not become a constant (we write this case as $\phi|_\rho\equiv *$). Following this case analysis, there exist $3^2=9$ cases on $(\phi'_{C_T}|_{\rho^*_{1,i},\ldots,\rho^*_{T,i}},\phi'_{C_T}|_{\rho_{1,i},\ldots,\rho_{T,i}})$. However, since each $\rho_{n,i}$ is a subrestriction of $\rho^*_{n,i}$ (i.e., $\rho_{n,i}$ assigns values only to variables that are also assigned by $\rho^*_{n,i}$), the following 4 cases do not occur: $(0,1),(1,0),(*,0)$, and $(*,1)$. Further, in the cases of $(0,0),(1,1),(*,*)$, and $(*,0)$, the answers by $\mathcal{A}^*$ and $\mathcal{A}$ do not differ because $\mathcal{A}^*$ and $\mathcal{A}$ return 0 in the case of $*$. Thus, we only need to consider the case of $(1,*)$.

By the aforementioned argument, the probability in the lemma is expressed as follows:

$$\Pr_{\mathcal{O}}\left[\mathcal{A}(M,x,1^{T^{2^c}})\neq\mathcal{A}^*(M,x,1^{T^{2^c}})\right]=\mathrm{Exp}_{C_T}\left[\Pr_{S_{1,i},\ldots,S_{T,i}}\left[\phi'_{C_T}|_{\rho_{1,i},\ldots,\rho_{T,i}}\equiv *\middle| C_T,\phi'_{C_T}|_{\rho^*_{1,i},\ldots,\rho^*_{T,i}}\equiv 1\right]\right].$$

To bound the probability in the right-hand side for each condition, we consider the case in which $\phi'_{C_T}|_{\rho^*_{1,i},\ldots,\rho^*_{T,i}}\equiv 1$. Then, we can select a term τ in ϕ'_{C_T} such that $\tau|_{\rho^*_{1,i},\ldots,\rho^*_{T,i}}\equiv 1$. For each $n\le T$, define $Z_n\subseteq\{0,1\}^n$ as

$$Z_n=\{z\in\{0,1\}^n:\exists(x,\ell)\in\{0,1\}^n\times[n]\text{ s.t. a variable }F_{z,x,\ell}\text{ is contained in }\tau\}.$$

Since ϕ'_{C_T} is a T-DNF, $|Z_n|\le T$ for each $n\le T$. Furthermore, for any $n\in\mathbb{N}$ such that $i_{\max}(n)\le i-1$, we have that $Z_n=\emptyset$ because all such variables must be assigned by $\rho_{n,\le i-1}$. If $\phi'_{C_T}|_{\rho_{1,i},\ldots,\rho_{T,i}}\equiv *$, then $\tau|_{\rho_{1,i},\ldots,\rho_{T,i}}\equiv *$ must hold. This event occurs only if $\bigcup_{n\le T}(S_{n,i}\cap Z_n)\neq\emptyset$ holds. Since each $S_{n,i}$ is selected from $S_{n,i-1}$ uniformly at random, this probability is bounded above by

$$\begin{aligned}
\Pr\left[\bigcup_{n\le T}(S_{n,i}\cap Z_n)\neq\emptyset\right]&\le\sum_{n\le T}\Pr\left[S_{n,i}\cap Z_n\neq\emptyset\right]\\
&\le\sum_{n\le T:i_{\max}(n)\ge i}|Z_n|\cdot|S_{n,i}|/|S_{n,i-1}|\\
&\le O(T)\cdot\sum_{n:t^{-1}(T)\le n\le T}p(n)\\
&\le O(T^2\cdot t(t^{-1}(T))^{-6})\\
&=O(T^{-4}).
\end{aligned}$$

Thus, we conclude that

$$\Pr_{\mathcal{O}}\left[\mathcal{A}(M,x,1^{T^{2^c}}) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})\right] = \mathrm{Exp}_{C_T}\left[\Pr_{S_{1,i},\dots,S_{T,i}}\left[\phi'_{C_T}|_{\rho_{1,i},\dots,\rho_{T,i}} \equiv * \middle| C_T, \phi'_{C_T}|_{\rho^*_{1,i},\dots,\rho^*_{T,i}} \equiv 1\right]\right]$$
$$\leq O(T^{-4}). \quad \blacktriangleleft$$

Furthermore, we can show the average-case easiness of NP under the oracle access to $\mathcal{A}^*$ (instead of $\mathcal{A}$). This part is similar to in [25, 20]. Thus, we defer the proof to Appendix A.

▶ **Lemma 17** ([25, 20]). *Let M be a $t_M(n)$-time nondeterministic oracle machine and S be a randomized polynomial-time oracle sampling machine. We assume that $S(1^n)$ takes at most $t_S(n)$ time to generate an instance of length n. Then, the following event occurs with probability 1 over the choice of $\mathcal{O}$: for any $n \in \mathbb{N}$,*

$$\Pr_{x \leftarrow S^{\mathcal{O}}(1^n)}\left[M^{\mathcal{O}}(x) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})\right] \leq O(n^{-4}),$$

where $T = \max\{n^{2^c}, t_M(n)^{2^c}, t_S(n)^{2^c}\}$.

Now, we derive the average-case easiness of NP from Lemmas 16 and 17 .

Proof of Theorem 15. We consider an arbitrary distributional NP problem (L,D) and assume that L is specified by a $t_M(n)$-time nondeterministic oracle machine M, and D is specified by a randomized $t_S(n)$-time oracle sampling machine S. Let $T = \max\{n^{2^c}, t_M(n)^{2^c}, t_S(n)^{2^c}\}$ as in Lemma 17.

We construct an error-prone heuristic algorithm B for (L,D) as follows: On input $x \in \{0,1\}^n$, B queries $(M,x,1^{T^{2^c}})$ to $\mathcal{A}$ and returns the same answer. In the following, we will verify that the error probability of B (over the choice of $\mathcal{O}$ and $S^{\mathcal{O}}(1^n)$) is bounded above by $O(n^{-4})$ for each input size n. Then, by applying Markov's inequality and the Borel–Cantelli lemma, the error probability of B is bounded above by n^{-2} for all sufficiently large n with probability 1 over the choice of $\mathcal{O}$. Since the number of tuples (M,S) is countable, we can conclude that all distributional NP problems have an error-prone heuristic algorithm with error probability at most n^{-2} with probability 1 over the choice of $\mathcal{O}$. Based on the argument in [24, Proposition 3], this is sufficient for the statement that $\mathsf{DistNP}^{\mathcal{O}} \subseteq \mathsf{HeurP}^{\mathcal{O}}$.

Therefore, it is sufficient to show that, for any $n \in \mathbb{N}$,

$$\Pr_{\mathcal{O},x \leftarrow S^{\mathcal{O}}(1^n)}\left[M^{\mathcal{O}}(x) \neq \mathcal{A}(M,x,1^{T^{2^c}})\right] \leq O(n^{-4}).$$

Obviously, this event occurs only if (i)$\mathcal{A}(M,x,1^{T^{2^c}}) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})$ or (ii) $M^{\mathcal{O}}(x) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})$ occur. By Lemmas 16 and 17 and the union bound, we have

$$\Pr_{\mathcal{O},x \leftarrow S^{\mathcal{O}}(1^n)}\left[M^{\mathcal{O}}(x) \neq \mathcal{A}(M,x,1^{T^{2^c}})\right]$$
$$\leq \Pr_{\mathcal{O},x \leftarrow S^{\mathcal{O}}(1^n)}\left[\mathcal{A}(M,x,1^{T^{2^c}}) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})\right] + \Pr_{\mathcal{O},x \leftarrow S^{\mathcal{O}}(1^n)}\left[M^{\mathcal{O}}(x) \neq \mathcal{A}^*(M,x,1^{T^{2^c}})\right]$$
$$= O(T^{-4}) + O(n^{-4}) = O(n^{-4}) + O(n^{-4}) = O(n^{-4}). \quad \blacktriangleleft$$

6 Errorless Average-Case Hardness of NP

In this section, we show the hardness part of our oracle separation. First, we show the existence of AIPRG relative to $\mathcal{O}_a$. Then we show the other hardness results, including the errorless average-case hardness for NP, as corollaries.

We use the following theorem, which shows the existence of PRGs based on a random oracle.

▶ **Theorem 18** ([45]). *For each $n \in \mathbb{N}$, let $\mathcal{R}_n \colon \{0,1\}^n \to \{0,1\}^n$ be a random function oracle, i.e., $\mathcal{R}_n$ is selected uniformly at random from $\{f \colon \{0,1\}^n \to \{0,1\}^n\}$.*

There exist a polynomial-time deterministic oracle machine $G^?$ and constants $c \geq 1$ and $b, \epsilon > 0$ satisfying the following: For any $n \in \mathbb{N}$ and $x \in \{0,1\}^{cn}$, $G^{\mathcal{R}_n}(x)$ generates a binary string of length $4cn$, and all oracle circuits $C^?$ of size 2^{bn} satisfy that

$$\left| \Pr_{U_{cn}} \left[C^{\mathcal{R}_n}(G^{\mathcal{R}_n}(U_{cn})) = 1 \right] - \Pr_{U_{4cn}} \left[C^{\mathcal{R}_n}(U_{4cn}) = 1 \right] \right| \leq 2^{-\epsilon n},$$

with probability at least $1 - 2^n$ over the choice of $\mathcal{R}_n$.

Furthermore, the result above is relativized, i.e., the above holds in the presence of an arbitrary oracle $\mathcal{O}$ independent of the choice of R_n.

Now, we show the existence of AIPRG relative to $\mathcal{O}_a$.

▶ **Theorem 19.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, there exists an AIPRG $G^{\mathcal{O}_a} = \{G_z^{\mathcal{O}_a}\}_{z \in \{0,1\}^*}$ against $\mathsf{SIZE}^{\mathcal{O}_a}[2^{\epsilon an / \log n}]$ for some absolute constant $\epsilon > 0$, where $G_z^{\mathcal{O}_a} \colon \{0,1\}^{|z|} \to \{0,1\}^{3|z|}$ for each $z \in \{0,1\}^*$.*

Proof. Let $G^?$ and c be the oracle machine and the constant in Theorem 18, respectively. Then, we define an AIPRG $G' = \{G'_z\}_{z \in \{0,1\}^*}$ as $G'^{\mathcal{O}}_z(x) = G^{\mathcal{F}_{z'}}(x')_{\leq 3|z|}$, where $x \in \{0,1\}^{|z|}$, $z' = z_{\leq \lfloor |z|/c \rfloor}$, $x' = x_{\leq c|z'|}$, and $\mathcal{F}_{z'} \colon \{0,1\}^{|z'|} \to \{0,1\}^{|z'|}$ is defined as $\mathcal{F}_{z'}(y) = \mathcal{F}(z', y, 1) \circ \cdots \circ \mathcal{F}(z', y, |z'|)$. The validity of the truncation is verified as that $|x'| = c|z'| \leq c \cdot |z|/c = |z| = |x|$ and $|G^{\mathcal{F}_{z'}}(x')| = 4|x'| = 4c|z'| \geq 4c(|z|/c - 1) \geq 4|z| - 4c \geq 3|z|$ for any z with $|z| \geq 4c$.

Let $\epsilon = 1/2c$ and $s(n) = 2^{\epsilon an / \log n}$. We show that G' above is an AIPRG against $\mathsf{SIZE}^{\mathcal{O}}[s(n)]$. Suppose there exists a family $C = \{C_n\}_{n \in \mathbb{N}}$ of oracle circuits of size $s(n)$ that breaks G', i.e., for any sufficiently large $n \in \mathbb{N}$ and any $z \in \{0,1\}^n$,

$$\left| \Pr_{U_n} \left[C_n^{\mathcal{O}}(z, G'^{\mathcal{O}}_z(U_n)) = 1 \right] - \Pr_{U_{3n}} \left[C_n^{\mathcal{O}}(z, U_{3n}) = 1 \right] \right| > \frac{1}{\mathsf{poly}(n)}.$$

Fix $n \in \mathbb{N}$ arbitrarily, and let $n' = \lfloor n/c \rfloor$. We consider an arbitrary choice of $\mathcal{O}$ except for the values of $\rho_{n', i_{\max}(n')}$ (we write this condition as R for convenience). Fix $z \in \{0,1\}^n$ such that $z' = z_{\leq n'} \in S_{n', i_{\max}(n')-1}$ arbitrary (where $S_{\cdot,\cdot}$ is the random set in the oracle construction). We refer to such z as a hard index.

Since the size of C_n is at most $s(n) = 2^{\frac{an}{2c \log n}} \leq 2^{\frac{a(\lfloor n/c \rfloor)}{\log(\lfloor n/c \rfloor)}} = t(n')$ for sufficiently large n (where t is the time-bound function in the oracle construction), the answers to queries by C_n of the form $\mathcal{A}(M, y, 1^{T^{2^c}})$ do not depend on the values of $\mathcal{F}_{z'}$ and are determined by condition R because they are determined only by the restrictions $\rho_{\cdot,j}$ for $j \leq c^{-1} \log \log T \leq c^{-1} \log \log s(n)^{1/2^c} \leq c \log \log t(n') - 1 < i_{\max}(n')$. Now, we consider an arbitrary choice of $\rho_{n', i_{\max}(n')}$ except for the values of $\mathcal{F}_{z'}$ and denote this condition by R'. It is not hard to verify that $\mathcal{F}_{z'}$ is selected uniformly at random even under the conditions R and R'. Therefore, under the conditions R and R', we can identify the query access to $\mathcal{O}$ by C with the query access to another oracle $\mathcal{O}'$ (determined only by R and R') and a random function oracle $\mathcal{F}_{z'}$ that are selected independently of $\mathcal{O}'$.

For any $n \in \mathbb{N}$, let E_n be an event (over the choice of $\mathcal{F}_{z'}$) that there exists a circuit C' of size $2^{bn'}$, where b represents the constant in Theorem 18, such that

$$\begin{aligned} &\left| \Pr_{U_n} \left[C'^{\mathcal{O}}(G'^{\mathcal{F}_{z'}}_z(U_n)) = 1 \right] - \Pr_{U_{3n}} \left[C'^{\mathcal{O}}(U_{3n}) = 1 \right] \right| \\ &\qquad = \left| \Pr_{U_n} \left[C'^{\mathcal{O}', \mathcal{F}_{z'}}(G^{\mathcal{F}_{z'}}(U_n)) = 1 \right] - \Pr_{U_{3n}} \left[C'^{\mathcal{O}', \mathcal{F}_{z'}}(U_{3n}) = 1 \right] \right| > \frac{1}{\mathsf{poly}(n)}. \end{aligned}$$

Then, by Theorem 18 (relative to $\mathcal{O}'$), we have $\Pr_{\mathcal{O}}[E_n|R,R'] \leq 2^{-\Omega(n)}$. By the Borel–Cantelli lemma, E_n occurs only for finitely many n's with probability 1 over the choice of $\mathcal{O}$ conditioned on R, R' (i.e., the choice of $\mathcal{F}_{z'}$). By taking the expectation over R, R', we can show that, with probability 1 over the choice of $\mathcal{O}$, there is no family C' of $2^{bn'}$-size circuits satisfying that for any sufficiently large $n \in \mathbb{N}$, there exists a hard index $z \in \{0,1\}^n$ such that

$$\left| \Pr_{U_n}\left[C'^{\mathcal{O}}(G'^{\mathcal{F}_{z'}}_z(U_n)) = 1\right] - \Pr_{U_{3n}}\left[C'^{\mathcal{O}}(U_{3n}) = 1\right] \right| > \frac{1}{\mathsf{poly}(n)}.$$

However, the circuit C in the assumption violates this statement by embedding a hard index z as auxiliary-input because the size is at most $n + s(n) = O(2^{(a/2c)\cdot n/\log n}) = o(2^{bn'})$. Therefore, we conclude that, with probability 1 over the choice of $\mathcal{O}$, there is no such a circuit C of size $s(n)$, and G' is an AIPRG against $\mathsf{SIZE}[s(n)]$. ◀

Now, we present the consequences of the existence of AIPRG. First, it is the well-established that any PRG is also OWF (cf. [12, Proposition 3.3.8]). This result is trivially extended to the case of auxiliary-input primitives, and the following holds.

▶ **Corollary 20.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, there exists an AIOWF against $\mathsf{SIZE}^{\mathcal{O}_a}[2^{\epsilon an/\log n}]$ relative to $\mathcal{O}_a$ for some absolute constant $\epsilon > 0$.*

Next, AIPRG implies HSG by regarding the auxiliary-input as a part of the hidden input to HSG. It is not hard to verify the security (refer to [34, Lemma 18] for the formal proof).

▶ **Corollary 21.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, there exists an HSG $G^{\mathcal{O}_a} = \{G_n^{\mathcal{O}_a}\}_{n\in\mathbb{N}}$ against $\mathsf{SIZE}^{\mathcal{O}_a}[2^{\epsilon an/\log n}]$ for some absolute constant $\epsilon > 0$, where $G_n^{\mathcal{O}_a} \colon \{0,1\}^{2n} \to \{0,1\}^{3n}$ for each $n \in \mathbb{N}$.*

Furthermore, the existence of HSGs implies the errorless average-case hardness of NP, as observed in [21] in the context of natural proofs and the average-case complexity of the minimum circuit size problem.

▶ **Corollary 22.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, $\mathsf{DistNP}^{\mathcal{O}_a} \nsubseteq \mathsf{AvgSIZE}^{\mathcal{O}_a}[2^{\frac{\epsilon an}{\log n}}]$ for some absolute constant $\epsilon > 0$.*

Proof. Let $G^{\mathcal{O}}$ and ϵ be the HSG and the constant in Corollary 21, respectively. Then, we define the language $L^{\mathcal{O}}$ as $L^{\mathcal{O}} := \mathrm{Im} G^{\mathcal{O}}$. Obviously, $L^{\mathcal{O}} \in \mathsf{NP}^{\mathcal{O}}$ and $(L^{\mathcal{O}}, \{U_n\}_{n\in\mathbb{N}}) \in \mathsf{DistNP}^{\mathcal{O}}$. Thus, it is sufficient to show that $(L^{\mathcal{O}}, \{U_n\}_{n\in\mathbb{N}}) \notin \mathsf{Avg}_{1/4}\mathsf{SIZE}^{\mathcal{O}}[2^{\frac{\epsilon an}{4\log n}}]$.

For contradiction, we assume that $(L^{\mathcal{O}}, \{U_n\}_{n\in\mathbb{N}}) \in \mathsf{Avg}_{1/4}\mathsf{SIZE}^{\mathcal{O}}[2^{\frac{\epsilon an}{4\log n}}]$. Then, there exists a family $C = \{C_n\}_{n\in\mathbb{N}}$ of $O(2^{\frac{\epsilon an}{4\log n}})$-size oracle circuits for $(L^{\mathcal{O}}, \{U_n\}_{n\in\mathbb{N}})$, i.e., for any sufficiently large $n \in \mathbb{N}$,

$$\Pr_{y \leftarrow_u \{0,1\}^{3n}}\left[C^{\mathcal{O}}_{3n}(y) = \bot\right] \leq 1/4$$

and for each $y \in \{0,1\}^{3n}$,

$$C^{\mathcal{O}}_{3n}(y) \in \{\mathbb{1}(y \in L^{\mathcal{O}}), \bot\}.$$

Note that the size of C_{3n} is at most $O(2^{\frac{\epsilon a\cdot 3n}{4\log 3n}})$.

Next, we define an adversary C' for $G^{\mathcal{O}}$ as follows: for a given $y \in \{0,1\}^{3n}$ (i.e., the length of seed is $2n$), $C'^{\mathcal{O}}_{2n}$ simulates $C^{\mathcal{O}}_{3n}(y)$, and if C_{3n} returns 1 or $\perp$, then C'_{2n} outputs 0; otherwise (i.e., if C_{3n} returns 0), C'_{2n} outputs 1. Then, based on the aforementioned inequalities, it is not hard to verify that for any sufficiently large $n \in \mathbb{N}$,

$$\Pr_{y \leftarrow_u \{0,1\}^{3n}}\left[C'^{\mathcal{O}}_{2n}(y) = 0\right] \leq \frac{1}{4} + \frac{|\{G^{\mathcal{O}}(x) : x \in \{0,1\}^{2n}\}|}{|\{0,1\}^{3n}|} \leq \frac{1}{4} + \frac{2^{2n}}{2^{3n}} < \frac{1}{2},$$

and for any $x \in \{0,1\}^{2n}$, we have $C^{\mathcal{O}}_{3n}(G^{\mathcal{O}}(x)) \in \{1, \perp\}$ and $C'^{\mathcal{O}}_{2n}(G^{\mathcal{O}}(x)) = 0$.

Therefore, C' succeeds in avoiding $\mathrm{Im}G^{\mathcal{O}}$, and the size is bounded above by $O(2^{\frac{\epsilon a n}{\log n}}) = O(2^{\frac{\epsilon a(2n)}{\log(2n)}})$. This contradicts Corollary 21. Thus, we conclude that $(L^{\mathcal{O}}, \{U_n\}_{n\in\mathbb{N}}) \notin \mathsf{Avg}_{1/4}\mathsf{SIZE}^{\mathcal{O}}[2^{\frac{\epsilon a n}{4\log n}}]$. ◀

Furthermore, based on the GGM construction in [13], we can translate AIPRGs into AIPRFs. In the security proof, the seed length is preserved, and an adversary of size $s(n)$ for the PRF is translated into an adversary of size $s(n) \cdot \mathsf{poly}(n)$ for the original PRG, where poly is a polynomial depending on the computational cost of the PRG. This observation implies the following:

▶ **Corollary 23.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, there exists an AIPRF $f^{\mathcal{O}_a} = \{f^{\mathcal{O}_a}_z\}_{z\in\{0,1\}^*}$ against $\mathsf{SIZE}^{\mathcal{O}_a}[2^{\epsilon an/\log n}]$ for some absolute constant $\epsilon > 0$, where $f^{\mathcal{O}_a}_z : \{0,1\}^{|z|} \times \{0,1\}^{|z|} \to \{0,1\}$ for each $z \in \{0,1\}^*$.*

Nanashima [33] observed that the existence of AIPRF implies the average-case hardness of distribution-free learning, where the complexity of the concept class depends on the complexity of computing AIPRF. Thus, we obtain the following, where we apply the standard transformation from nonuniform Turing machines to circuit families.

▶ **Corollary 24.** *There exist a polynomial $s(n)$ and a constant $\epsilon > 0$ such that for any $a > 0$, $\mathsf{SIZE}[s(n)]$ is not PAC learnable on average by nonuniform $O(2^{\epsilon an/\log n})$-time algorithms with probability 1 over the choice of $\mathcal{O}_a$.*

Without loss of generality, we can let $s(n) = n^b$ in above for some $b > 0$. By the simple padding argument, where we stretch an n-bit example into an $s(n)$-bit example, the size complexity of the target function becomes $O(n)$ (for the input length $s(n)$) in above. Since $2^{s(n)^{1/(b+1)}} = o(2^{\epsilon an/\log n})$, we have the following:

▶ **Corollary 25.** *There exists $\epsilon > 0$ such that for any $a > 0$, $\mathsf{SIZE}[n]$ is not PAC learnable on average by nonuniform $O(2^{n^\epsilon})$-time algorithms with probability 1 over the choice of $\mathcal{O}_a$.*

The existence of AIPRF also implies the (worst-case) hardness of PAC learning $\mathsf{SIZE}[s(n)]$ on the uniform distribution, as observed in [3], where $s(n)$ is a polynomial depending on the complexity of computing the AIPRF. In our case, we can directly construct a hard-to-learn class and show the hardness of learning $\mathsf{SIZE}[n]$.

▶ **Theorem 26.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, $\mathsf{SIZE}^{\mathcal{O}_a}[n]$ is not weakly PAC learnable with MQ on the uniform distribution by nonuniform $O(2^{\epsilon an/\log n})$-time algorithms relative to $\mathcal{O}^a$ for some absolute constant $\epsilon > 0$.*

The proof of Theorem 26 is an analog of the proof in [20]. For completeness, we present the formal proof in Appendix B.

Furthermore, Oliveira and Santhanam [35] showed the speedup phenomena in PAC learning with MQ on the uniform distribution. One of their results is stated below.

▶ **Theorem 27** (speedup lemma [35]). *For any polynomial $s(n)$ and constant $\epsilon > 0$, there exists a polynomial $s'(n)$ such that if $\mathsf{SIZE}[s(n)]$ is not weakly PAC learnable with MQ on uniform distribution by nonuniform $O(2^{n^\epsilon})$-time algorithms, then $\mathsf{SIZE}[s'(n)]$ is not weakly PAC learnable with MQ on uniform distribution by nonuniform $2^n/n^{\omega(1)}$-time algorithms. Furthermore, this result is relativized.*

Theorems 26 and 27 immediately imply the following.

▶ **Corollary 28.** *There exists a polynomial $s(n)$ such that for any $a > 0$, $\mathsf{SIZE}[s(n)]$ is not PAC learnable with MQ on the uniform distribution by nonuniform $2^n/n^{\omega(1)}$-time algorithms with probability 1 over the choice of $\mathcal{O}_a$.*

Finally, we mention the hardness of approximation problems in meta-complexity. The hardness of $\mathsf{GapMINKT}$ follows from the existence of HSG in the same manner as Corollary 22.

▶ **Corollary 29.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, $\mathsf{Gap}_\sigma\mathsf{MINKT}^{\mathcal{O}_a} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}_a}[2^{\epsilon an/\log n}]$ for any $\sigma(s,n) = o(s) \cdot \mathsf{polylog}(n)$, where $\epsilon > 0$ is an absolute constant.*

Proof sketch. Suppose that $\mathsf{Gap}_\sigma\mathsf{MINKT}^{\mathcal{O}} \in \mathsf{pr\text{-}SIZE}^{\mathcal{O}}[2^{(\epsilon/4)an/\log n}]$ for some $\sigma(s,n) = o(s) \cdot \mathsf{polylog}(n)$, where $\epsilon > 0$ is the constant in Corollary 21. Then, there exists an $O(2^{(\epsilon/4)an/\log n})$-size oracle circuit C for $\mathsf{Gap}_\sigma\mathsf{MINKT}^{\mathcal{O}}$. Based on C, we can construct an adversary for an arbitrary HSG $G^{\mathcal{O}}\colon \{0,1\}^{2n} \to \{0,1\}^{3n}$ because, for each $n \in \mathbb{N}$ and $x \in \{0,1\}^{2n}$, it holds that $\mathsf{K}^{t,\mathcal{O}}(G^{\mathcal{O}}(x)) \leq 2n + O(1)$ for a proper choice of $t = \mathsf{poly}(n)$ and $\Pr_{y \leftarrow_u \{0,1\}^{3n}}[\mathsf{K}^{\mathcal{O}}(y) \geq 3n - 2\,(> 2n + O(1) + \sigma(2n + O(1), n))] \geq 3/4$. It is not hard to verify that the size of the adversary based on C is at most $O(2^{(\epsilon/4)3an/\log 3n})$. Thus, this contradicts Corollary 21. ◀

Furthermore, Carmosino, Impagliazzo, Kabanets, and Kolokolova [10] constructed a PAC learning algorithm for $\mathsf{P/poly}$ with MQ on the uniform distribution based on an algorithm for $\mathsf{GapMCSP}$ (originally, the existence of natural proofs). As the contraposition, we obtain the following from Corollary 28.

▶ **Corollary 30.** *Let $a > 0$ be an arbitrary constant. With probability 1 over the choice of $\mathcal{O}_a$, for each $\epsilon > 0$, there exists $\delta > 0$ such that $\mathsf{Gap}_\epsilon\mathsf{MCSP}^{\mathcal{O}_a} \notin \mathsf{pr\text{-}SIZE}^{\mathcal{O}_a}[2^{n^\delta}]$.*

Theorem 4 follows from Theorems 15, 19, and 26 and Corollaries 20– 30 by selecting an appropriately large parameter in the oracle construction according to $a > 0$ in the statement of Theorem 4.

References

1 S Aaronson and A Wigderson. Algebrization: A New Barrier in Complexity Theory. *ACM Trans. Comput. Theory*, 1(1), February 2009.

2 E. Allender, M. Cheraghchi, D. Myrisiotis, H. Tirumala, and I. Volkovich. One-Way Functions and a Conditional Variant of MKTP. In *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

3 B. Applebaum, B. Barak, and D. Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'08, pages 211–220, 2008.

4 T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} =? \mathcal{NP}$ Question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

5 Manuel Blum and Sampath Kannan. Designing Programs that Check Their Work. *J. ACM*, 42(1):269–291, 1995. doi:10.1145/200836.200880.

6 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.

7 A. Bogdanov and L. Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

8 G. Brassard. Relativized Cryptography. *IEEE Transactions on Information Theory*, 29(6):877–894, 1983.

9 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.

10 M. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the 31st Conference on Computational Complexity*, CCC'16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

11 R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 305–313, 2000.

12 O. Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

13 O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, August 1986.

14 J. Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.

15 J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A Pseudorandom Generator from Any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, March 1999.

16 S. Hirahara. Non-Black-Box Worst-Case to Average-Case Reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258, 2018.

17 S Hirahara. Characterizing Average-Case Complexity of PH by Worst-Case Meta-Complexity. In *IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 50–60, 2020.

18 S. Hirahara. Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions. In *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *LIPIcs*, pages 20:1–20:47, Dagstuhl, Germany, 2020.

19 S. Hirahara. Average-Case Hardness of NP from Exponential Worst-Case Hardness Assumptions. In *53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*, 2021.

20 S. Hirahara and M. Nanashima. On Worst-Case Learning in Relativized Heuristica. In *62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, 2021.

21 S. Hirahara and R. Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

22 S. Hirahara and R. Santhanam. Errorless versus Error-prone Average-case Complexity. In *The 13th Innovations in Theoretical Computer Science (ITCS 2022)*, 2022.

23 R. Ilango, H. Ren, and R. Santhanam. Hardness on any Samplable Distribution Suffices: New Characterizations of One-Way Functions by Meta-Complexity. *Electron. Colloquium Comput. Complex.*, page 82, 2021.

24 R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of IEEE Tenth Annual Conference on Structure in Complexity Theory*, pages 134–147, 1995.

25 R. Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 104–114, 2011.

26 R. Impagliazzo and L. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, FOCS'90, pages 812–821, 1990.

27 R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 44–61, New York, NY, USA, 1989. ACM.

28 K. Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM Journal on Computing*, 20(5):962–986, 1991.

29 L. A Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, February 1986.

30 Y. Liu and R. Pass. On One-way Functions and Kolmogorov Complexity. In *IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 1243–1254, 2020. doi:10.1109/FOCS46700.2020.00118.

31 Y. Liu and R. Pass. A Note on One-way Functions and Sparse Languages. *Electron. Colloquium Comput. Complex.*, page 92, 2021.

32 Y. Liu and R. Pass. On the Possibility of Basing Cryptography on EXP $\neq$ BPP. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 11–40. Springer, 2021.

33 M. Nanashima. A Theory of Heuristic Learnability. In *Proceedings of the 34th Conference on Learning Theory, COLT'21*. PMLR, August 2021.

34 M. Nanashima. On Basing Auxiliary-Input Cryptography on NP-Hardness via Nonadaptive Black-Box Reductions. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *LIPIcs*, pages 29:1–29:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

35 I. Oliveira and R. Santhanam. Conspiracies between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Proceedings of the 32nd Computational Complexity Conference*, CCC'17, Dagstuhl, DEU, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

36 R. Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 133–138, 1991.

37 R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Proceedings of the 2nd Israel Symposium on Theory and Computing Systems*, ISTCS'93, pages 3–17, June 1993.

38 A. A. Razborov and S. Rudich. Natural Proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

39 H. Ren and R. Santhanam. A Relativization Perspective on Meta-Complexity. In *The 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, 2022.

40 L. Valiant. A Theory of the Learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

41 Thomas Watson. Relativized Worlds without Worst-Case to Average-Case Reductions for NP. *ACM Trans. Comput. Theory*, 4(3), September 2012.

42 H Wee. Finding Pessiland. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 429–442, Berlin, Heidelberg, 2006. Springer-Verlag.

43 D. Xiao. On basing ZK $\neq$ BPP on the hardness of PAC learning. In *Proceedings of the 24th Conference on Computational Complexity*, CCC'09, pages 304–315, 2009.

44 A. Yao. Theory and Application of Trapdoor Functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, FOCS'82, pages 80–91, November 1982.

45 M. Zimand. Efficient Privatization of Random Bits. In *IN WORKSHOP ON RANDOMIZED ALGORITHMS*, 1997.

A Proof of Lemma 17

Fix $n \in \mathbb{N}$ arbitrarily. Remember that $T = \max\{n^{2^c}, t_M(n)^{2^c}, t_S(n)^{2^c}\}$. Let $i_T = c^{-1}\log\log T$.

We first show that the instance $x \in \{0,1\}^n$ generated by $S^{\mathcal{O}}(1^n)$ is determined by only $\rho_{n',j}$ for $n' \le T$ and $j \le i_T - 1$ with probability at least $1 - O(n^{-5})$. Then, we show that $\mathcal{A}^*(M, x, 1^{T^{2^c}})$ returns $M^{\mathcal{O}}(x)$ with probability at least $1 - O(n^{-5})$ under the condition that x is determined by only $\rho_{n',j}$ for $n' \le T$ and $j \le i_T - 1$. If we assume these, then by the union bound, we have the lemma as

$$\Pr_{\mathcal{O}, x \leftarrow S^{\mathcal{O}}(1^n)}\left[M^{\mathcal{O}}(x) \neq \mathcal{A}^*(M, x, 1^{T^{2^c}})\right] \le O(n^{-5}) + O(n^{-5}) = O(n^{-5})\ (= O(n^{-4})).$$

Now, we show the first claim. Since $t_S(n) \le T^{1/2^c}$, the answers of $\mathcal{A}$ to queries made by $S^{\mathcal{O}}(1^n)$ are determined by only $\rho_{n',j}$ for $n' \le T^{1/2^c}$ and $j \le i_T - 2$. Under an arbitrary condition on restrictions $\rho_{n',j}$ for $n' \le T^{1/2^c}$ and $j \le i_{t_S(n)^{1/2^c}} \le i_T - 2$, the output $S^{\mathcal{O}}(1^n)$ is determined by only $\rho_{n',j}$ for $n' \le T$ and $j \le i_T - 1$ unless S queries $(z, x, \ell) \in \cup_{n' \le T^{1/2^c}} {\rho^*_{n',i_T-1}}^{-1}(*)$ to $\mathcal{F}$. Note that, if $n' \in \mathbb{N}$ satisfies $n' < t^{-1}(T^{1/2^c})$, then we have $i_{\max}(n') < c^{-1}\log\log(T^{1/2^c}) = c^{-1}\log\log T - 1 = i_T - 1$. Thus, ${\rho^*_{n',i_T-1}}^{-1}(*) = \emptyset$. Otherwise, each element in ${\rho^*_{n',i_T-1}}^{-1}(*)$ is selected from V_{n',i_T-2} independently with probability $p(n')$. Since $S^{\mathcal{O}}(1^n)$ accesses to $\mathcal{A}$ at most $T^{1/2^c}$ times, such a conditional probability is bounded above by

$$\begin{aligned} T^{1/2^c} \max_{t^{-1}(T^{1/2^c}) \le n' \le T^{1/2^c}} p(n') &= T^{1/2^c} p(t^{-1}(T^{1/2^c})) \\ &= T^{1/2^c} t(t^{-1}(T^{1/2^c}))^{-6} \\ &= (T^{1/2^c})^{-5} \\ &\le n^{-5}, \end{aligned}$$

where the inequality holds because $T \ge n^{2^c}$.

Next, we show the second claim. Under the condition that the given instance $x \in \{0,1\}^n$ is determined by only $\rho_{n',j}$ for $n' \le T$ and $j \le i_T - 1$, the T-DNF formula ϕ constructed in $\mathcal{A}^*(M, x, 1^{T^{2^c}})$ is determined only by $\rho_{n',j}$ for $n' \le T$ and $j \le i_T - 1$. Then, applying the restriction $\rho_{n',j}$ for $n' \le T$ and $j \le i_T$ under this condition is regarded as a $p(n')$-random restriction to V_{n',i_T-1} for each $n' \le T$, where we can ignore small n' such that $n' < t^{-1}(T)$ because $i_{\max}(n') < c^{-1}\log\log T = i_T$ for such n'. By applying the switching lemma (particularly, the baby switching lemma) for T-DNF [14], the probability that ϕ does not become a constant is at most

$$\begin{aligned} O\left(T \max_{t^{-1}(T) \le n' \le T} p(n')\right) &= O\left(T \cdot t(t^{-1}(T))^{-6}\right) \\ &= O(T^{-5}) \\ &= O(n^{-5}), \end{aligned}$$

where the last equation holds because $T \ge n^{2^c} \ge n$. Remember that $\mathcal{A}^*$ always returns the correct answer whenever ϕ becomes constant. Therefore, the second claim holds.

B Proof of Theorem 26

We first introduce the following lemma.

▶ **Lemma 31** ([11]). *Let $S, T \subseteq \{0,1\}^*$ be finite subsets of the same size N, and let $b\colon S \to T$ be a bijection. Let $A^?$ be a deterministic oracle machine that makes at most q queries to b. If $A^f(y) = f^{-1}(y)$ for all $y \in T$, then b has the representation of length at most $2\log\binom{N}{a} + \log((N-a)!)$ when A is given, where $a = N/(q+1)$.*

Now, we present the formal proof of Theorem 26.

Proof of Theorem 26. For every choice of $\mathcal{O}_a$, we define a concept class $\mathcal{C}^{\mathcal{O}_a}$ as

$$\mathcal{C}^{\mathcal{O}_a} = \{\mathcal{F}_{|z|}(z,\cdot,1) : z \in \{0,1\}^*\}.$$

Then, we show that $\mathcal{C}^{\mathcal{O}_a}$ is not weakly PAC learnable with MQ on the uniform distribution by nonuniform $t_L(n) = O(2^{(a/2)n/\log n})$-time algorithms with probability 1 over the choice of $\mathcal{O}_a$. Since $\mathcal{C}^{\mathcal{O}_a} \subseteq \mathsf{SIZE}^{\mathcal{O}_a}[n]$, the theorem also holds.

Let $\epsilon(n) = n^{-\log n}$. We fix $n \in \mathbb{N}$ arbitrarily. We consider an arbitrary nonuniform randomized oracle machine (i.e., a learner) L. For each $z \in \{0,1\}^n$, we define I_z as an event (over the choice of $\mathcal{O}$) that L succeeds in learning $f_z(x) \equiv F_n(z,x,1) \in \mathcal{C}_n^{\mathcal{O}}$ in $t_L(n)$ time with advantage $\epsilon(n)$, i.e.,

$$I_z = \left(\Pr_L\left[L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n) \to h^{\mathcal{O}} \text{ s.t. } \Pr_x[h^{\mathcal{O}}(x) = f_z(x)] \geq 1/2 + \epsilon(n) \text{ in } t_L(n) \text{ time}\right] \geq 2/3\right).$$

We will show that $\Pr_{\mathcal{O}}[\wedge_{z\in\{0,1\}^n} I_z] \leq 2^{-2^{\Omega(n)}}$. For now, we assume this and show the hardness of learning $\mathcal{C}^{\mathcal{O}}$. Since any nonuniform $t_L(n)$-time oracle machine has a binary representation of length at most $O(t_L(n))$ (for each $n \in \mathbb{N}$), the event E_n that there exists a nonuniform $t_L(n)$-time oracle machine succeeds in learning $\mathcal{C}_n^{\mathcal{O}}$ is at most $2^{O(t_L(n))} \cdot 2^{-2^{\Omega(n)}} = \mathsf{negl}(n)$ by the union bound. By the Borel–Cantelli lemma, these events E_n occur only for finitely many $n \in \mathbb{N}$ with probability 1 over the choice of $\mathcal{O}$. In such cases, there is no nonuniform $t_L(n)$-time algorithm that succeeds in weak learning for $\mathcal{C}^{\mathcal{O}}$.

Now, we show that $\Pr_{\mathcal{O}}[\wedge_{z\in\{0,1\}^n} I_z] \leq 2^{-2^{\Omega(n)}}$. For any $z, x \in \{0,1\}^n$, we use a notation $L^{\mathcal{O}}(n)(x)$ to refer to the following procedure: We execute $L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n)$ and if L outputs some hypothesis $h^?$ in $t_L(n)$ time, then we also execute $h^{\mathcal{O}}(x)$. For any $z \in \{0,1\}^n$, we define an event J_z as the event (over the choice of $\mathcal{O}$) that L or its hypothesis directly access a target function f_z by $\mathcal{F}$, i.e.,

$$J_z = \left(\Pr_{L,x\sim\{0,1\}^n}\left[\mathcal{F}(z,x',\ell) \text{ is queried for some } (x',\ell) \in \{0,1\}^n \times [n] \text{ during } L^{\mathcal{O}}(n)(x)\right] \geq \epsilon(n)^4\right).$$

We say that $z \in \{0,1\}^n$ is a hard index (relative to $\mathcal{O}$) if $z \in S_{n,i_{\max}(n)-1}$. Then, we have that

$$\begin{aligned}
\Pr_{\mathcal{O}}\left[\bigwedge_{z\in\{0,1\}^n} I_z\right] &\leq \Pr_{\mathcal{O}}\left[\bigwedge_{z\in\{0,1\}^n:\text{hard}} I_z\right] \\
&\leq \Pr_{\mathcal{O}}\left[\bigwedge_{z\in\{0,1\}^n:\text{hard}} (I_z \vee J_z)\right] \\
&\leq \Pr_{\mathcal{O}}\left[\bigwedge_{z:\text{hard}} J_z\right] + \Pr_{\mathcal{O}}\left[\exists z : \text{hard s.t. } I_z \wedge \neg J_z\right].
\end{aligned}$$

In the following, we show that each term is bounded above by $2^{-2^{\Omega(n)}}$, which implies the theorem.

▷ Claim 32. $\Pr_{\mathcal{O}}\left[\bigwedge_{z:\mathrm{hard}} J_z\right] \le 2^{-2^{\Omega(n)}}$.

Proof. Let $N = |S_{n,i_{\max}(n)-1}|$. For any choice of $S_{n,i_{\max}(n)-1}$, we can divide a random selection of $\rho_{n,i_{\max}(n)}$ into the following two steps without loss of generality: (i) select N random functions $x_1, \ldots, x_N \in \{0,1\}^{n2^n}$ uniformly at random (where we regard each x_j as a truth table of a mapping from n bits to n bits), and (ii) select a random bijection $b\colon S_{n,i_{\max}(n)-1} \to \{x_1, \ldots, x_N\}$ to assign each value of $F(z,\cdot,\cdot)$ as $F(z,\cdot,\cdot) \equiv b(z)$ for each $z \in S_{n,i_{\max}(n)-1}$.

We consider an arbitrary choice of $\mathcal{O}$ except for the aforementioned bijection b and use the notation C to refer to such a partial choice of $\mathcal{O}$. We regard C as a condition on the choice of $\mathcal{O}$. We say that the partial choice C is bad if there are two distinct indices $j_1, j_2 \in [N]$ such that $x_{i_1} = x_{i_2}$. Since $x_1, \ldots, x_N$ is uniformly and independently selected from 2^{n2^n} elements, by the union bound, we obtain that

$$\Pr_{\mathcal{O}}[C \text{ is bad}] \le N^2 \cdot 2^{-n2^n} \le 2^{2n} \cdot 2^{-n2^n} = 2^{-\Omega(2^n)}.$$

Thus, we have that

$$\begin{aligned}
\Pr_{\mathcal{O}}\left[\bigwedge_{z:\mathrm{hard}} J_z\right] &= \mathrm{Exp}_C\left[\Pr_{\mathcal{O}}\left[\bigwedge_{z:\mathrm{hard}} J_z \middle| C\right]\right] \\
&\le \mathrm{Exp}_C\left[\Pr_{\mathcal{O}}\left[\bigwedge_{z:\mathrm{hard}} J_z \middle| C\right] \middle| C \text{ is not bad}\right] + \Pr_{O}[C \text{ is bad}] \\
&\le \mathrm{Exp}_C\left[\Pr_{b}\left[\bigwedge_{z:\mathrm{hard}} J_z \middle| C\right] \middle| C \text{ is not bad}\right] + 2^{-\Omega(2^n)}.
\end{aligned}$$

Therefore, it is sufficient to show that for every not bad condition C,

$$\Pr_{b}\left[\bigwedge_{z:\mathrm{hard}} J_z \middle| C\right] = 2^{-2^{\Omega(n)}}.$$

To show the aforementioned bound, we assume that $\bigwedge_{z:\mathrm{hard}} J_z$ holds under a not bad condition C (notice that C determines all hard indices), i.e., for any hard index $z \in \{0,1\}^n$,

$$\Pr_{L,x\sim\{0,1\}^n}\left[\mathcal{F}(z,\cdot,\cdot) \text{ is queried during } L^{\mathcal{O}}(n)(x)\right] \ge \epsilon(n)^4$$

By the standard probabilistic argument, we can reduce the upper bound from $1-\epsilon(n)^4$ to 2^{-2n} on the probability that $\mathcal{F}(z,\cdot,\cdot)$ is not queried by L or its hypothesis by repeating $L^{\mathcal{O}}(n)(x)$ $2n/\epsilon(n)^4$ times. Then, by the union bound for all hard indices, there exists a random seed $r \in \{0,1\}^{t_L(n)\cdot 2n/\epsilon(n)^4}$ such that for any hard index z, $\mathcal{F}(z,\cdot,\cdot)$ is queried during at least one execution of $L^{\mathcal{O}}(n)(x)$ by using the randomness r. We remark that all queries to $\mathcal{O}$ by L or its hypothesis are determined by C except for $\mathcal{F}(z,\cdot,\cdot)$ for each hard index z. This is because L and its hypothesis are executed in time $O(t_L(n)) = O(2^{(a/2)n/\log n}) \le 2^{an/\log n} = t(n)$ (for sufficiently large n), i.e., all answers from $\mathcal{A}$ depend on only $\rho_{\cdot,j}$ for $j \le c^{-1}\log\log t(n)^{1/2^c} = c^{-1}\log\log t(n) - 1 < i_{\max}(n)$.

Therefore, by executing L with the randomness r and tracing queries to $\mathcal{F}$, we can obtain a deterministic inverter for b of the query complexity at most $t_L(n)\cdot 2n/\epsilon(n)^4$, where the inverter simulates membership queries by using its input and its own query access to b. Particularly, the inverter accesses b only for answering the queries of the form $\mathcal{F}(z',\cdot,\cdot)$

for some $z' \in S_{n,i_{\max}(n)-1}$. However, by Lemma 31, such a bijection b is represented by $2\log\binom{N}{a} + \log((N-a)!)$ bits, where $a = N/(t_L(n)\cdot 2n\epsilon(n)^{-4}+1)$, when L and r are given. Thus, we obtain that

$$a \geq \frac{2^n \cdot p(n)^{i_{\max}(n)}}{O(t_L(n)n\epsilon(n)^{-4})} \geq \frac{2^n \cdot 2^{-\frac{6an}{\log n}\cdot\frac{1}{c}\log n}}{O(2^{(a/2)n/\log n} n^{4\log n+1})} \geq \frac{2^n \cdot 2^{-\frac{6}{7}n}}{2^{O(n/\log n)}} \geq 2^{\Omega(n)},$$

and

$$\begin{aligned}
\Pr_b\left[\bigwedge_{z:\text{hard}} J_z \middle| C\right] &\leq \frac{\binom{N}{a}^2 \cdot (N-a)! \cdot 2^{O(t_L(n)\cdot 2n\epsilon(n)^{-4})}}{N!} \\
&\leq \binom{N}{a} \cdot \frac{1}{a!} \cdot 2^{O(2^{(a/2)n/\log n})} \\
&\leq \left(\frac{Ne}{a}\right)^a \cdot \frac{1}{\sqrt{2\pi a}} \left(\frac{e}{a}\right)^a \cdot 2^{2^{O(n/\log n)}} \\
&\leq \left(e(t_L(n)\cdot 2n\epsilon(n)^{-4}+1)\right)^a \cdot \left(\frac{e}{a}\right)^a \cdot 2^{2^{O(n/\log n)}} \\
&\leq \left(\frac{2^{O(n/\log n)}}{a}\right)^a \cdot 2^{2^{O(n/\log n)}} \\
&\leq 2^{-a} \cdot 2^{2^{O(n/\log n)}} = 2^{-2^{\Omega(n)}}.
\end{aligned}$$

◁

▷ Claim 33. $\Pr_{\mathcal{O}}[\exists z : \text{hard s.t. } I_z \wedge \neg J_z] \leq 2^{-2^{\Omega(n)}}$.

Proof. We consider an arbitrary choice of $\mathcal{O}$ except for values of $\rho_{n,i_{\max}(n)}$ (we write this condition as C). Note that hard indices are determined by the condition C, and for any hard index $z \in \{0,1\}^n$, f_z is a random function even under the condition C.

Suppose that z is a hard index, and $\neg I_z \wedge J_z$ occurs. By Markov's inequality, we derive the following from $\neg I_z$:

$$\Pr_L\left[\Pr_x\left[\mathcal{F}(z,\cdot,\cdot) \text{ is queried during } L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n)(x)\right] \leq 4\epsilon(n)^3\right] \geq 1-\epsilon(n)/4.$$

Since J_z holds, we also have

$$\Pr_L\left[L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n) \to h^{\mathcal{O}} \text{ s.t. } \Pr_x[h^{\mathcal{O}}(x) = f_z(x)] \geq 1/2+\epsilon(n) \text{ in } t_L(n) \text{ time }\right] \geq \frac{2}{3}.$$

From the aforementioned two inequalities, there exists a random string r for L such that

- $L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n;r)$ outputs some hypothesis $h^{\mathcal{O}}$ in $t_L(n)$ time without querying $(z,\cdot,\cdot)$ to $\mathcal{F}$;
- $\Pr_x[h^{\mathcal{O}}(x) \text{ queries } (z,\cdot,\cdot) \text{ to } \mathcal{F}] \leq 4\epsilon(n)^3$; and
- $\Pr_x[h^{\mathcal{O}}(x) = f_z(x)] \geq 1/2+\epsilon(n)$.

Since L and h are only executed in $O(t_L(n)) \leq t(n)$ time (for sufficiently large n), any query $(M, x', 1^{T^{2^c}})$ to $\mathcal{A}$ by L and h satisfies that $T^{2^c} \leq t(n)$ and $i_T = c^{-1}\log\log T = c^{-1}\log\log t(n) - 1 < i_{\max}(n)$. Therefore, if L and h do not query $(z,\cdot,\cdot)$ to $\mathcal{F}$, then the answers from $\mathcal{O}$ do not depend on $\rho_{n,i_{\max}(n)}$, i.e., they are determined only by the condition C.

In this case, we show that a truth table of f_z has a short description under the condition C. This implies the upper bound on the probability of this case because a random function does not have such a short description with extremely high probability.

The short description of f_z is obtained as follows. Let $B_z \subseteq \{0,1\}^n$ be the subset consisting of x such that $h^{\mathcal{O}}(x)$ queries $\mathcal{F}(z,\cdot,\cdot)$. By the second property of the above, $|B_z| \leq 2^n \cdot 4\epsilon(n)^3$ holds. We execute $L^{\mathcal{O},\mathsf{MQ}_{f_z}}(n;r)$ to obtain $h^{\mathcal{O}}$, and we write down all

answers from the membership query oracle MQ_{f_z} in Q, i.e., Q is a binary string of length at most $t_L(n)$. By the first property on r, the answers from $\mathcal{O}$ are determined by the condition C. Next, we execute the outputted hypothesis $h^{\mathcal{O}}(x)$ on each input $x \in \{0,1\}^n \setminus B_z$. From these predictions and auxiliary information $f_z(B_z) = \{(x, f_z(x)) : x \in B_z\}$, we obtain a function $\tilde{f} : \{0,1\}^n \to \{0,1\}$ defined as

$$\tilde{f}(x) = \begin{cases} h^{\mathcal{O}}(x) & \text{if } x \notin B_z \\ f_z(x) & \text{if } x \in B_z. \end{cases}$$

Then, by the third property, $\tilde{f}$ is $(1/2 - \epsilon(n))$-close to f_z. We define $e \in \{0,1\}^{2^n}$ as $e_{x+1} = f_z(x) \oplus \tilde{f}(x)$, where we identify $x \in \{0,1\}^n$ with an integer in $[0, 2^n - 1]$. Then, the Hamming weight of e is at most $2^n \cdot (1/2 - \epsilon(n))$, and e is represented by a binary string $\tilde{e}$ of length at most $(1 - \Omega(\epsilon(n)^2)) \cdot 2^n$ by lexicographic indexing among binary strings of the same weight. Obviously, f_z is reconstructed from $\tilde{f}$ and $\tilde{e}$. Therefore, based on the aforementioned constructions, f_z is represented only by $L, r, Q, f_z(B_z)$, and $\tilde{e}$ on the condition C. The total number of such representations is at most

$$\begin{aligned} |L| + t_L(n) + t_L(n) + (n+1) \cdot |B_z| + (1 - \Omega(\epsilon(n)^2)) \cdot 2^n & \\ \leq O(t_L(n)) + \left(1 + 4(n+1)\epsilon(n)^3 - \Omega(\epsilon(n)^2)\right) 2^n & \\ \leq O(t_L(n)) + \left(1 - \Omega(\epsilon(n)^2)\right) \cdot 2^n. & \end{aligned}$$

Therefore, we have that for any condition C and any hard index $z \in \{0,1\}^n$,

$$\begin{aligned} \Pr_{\mathcal{O}}\left[I_z \wedge \neg J_z | C\right] &\leq \frac{2^{O(t_L(n)) + \left(1 - \Omega(\epsilon(n)^2)\right) \cdot 2^n}}{2^{2^n}} \\ &\leq 2^{2^{O(n/\log n)} - \Omega(n^{-2\log n}) \cdot 2^n} \\ &\leq 2^{-2^{\Omega(n)}}. \end{aligned}$$

Thus, we conclude that

$$\begin{aligned} \Pr_{\mathcal{O}}\left[\bigvee_{z:\text{hard}} I_z \wedge \neg J_z\right] &= \mathrm{Exp}_C\left[\Pr_{\mathcal{O}}\left[\bigvee_{z:\text{hard}} I_z \wedge \neg J_z \middle| C\right]\right] \\ &\leq \mathrm{Exp}_C\left[\sum_{z \in \{0,1\}^n} \Pr_{\mathcal{O}}\left[z \text{ is hard and } I_z \wedge \neg J_z | C\right]\right] \\ &\leq 2^n \cdot 2^{-2^{\Omega(n)}} = 2^{-2^{\Omega(n)}}. \end{aligned}$$

◁

◀

Symmetry of Information from Meta-Complexity

Shuichi Hirahara ✉
National Institute of Informatics, Tokyo, Japan

Abstract

Symmetry of information for time-bounded Kolmogorov complexity is a hypothetical inequality that relates time-bounded Kolmogorov complexity and its conditional analogue. In 1992, Longpré and Watanabe showed that symmetry of information holds if NP is easy in the worst case, which has been the state of the art over the last three decades. In this paper, we significantly improve this result by showing that symmetry of information holds under the weaker assumption that NP is easy on average. In fact, our proof techniques are applicable to any resource-bounded Kolmogorov complexity and enable proving symmetry of information from an efficient algorithm that computes resource-bounded Kolmogorov complexity.

We demonstrate the significance of our proof techniques by presenting two applications. First, using that symmetry of information does not hold for Levin's Kt-complexity, we prove that randomized Kt-complexity cannot be computed in time $2^{o(n)}$ on inputs of length n, which improves the previous quasi-polynomial lower bound of Oliveira (ICALP 2019). Our proof implements Kolmogorov's insightful approach to the P versus NP problem in the case of randomized Kt-complexity. Second, we consider the question of excluding Heuristica, i.e., a world in which NP is easy on average but $\mathsf{NP} \neq \mathsf{P}$, from Impagliazzo's five worlds: Using symmetry of information, we prove that Heuristica is excluded if the problem of approximating time-bounded conditional Kolmogorov complexity $\mathrm{K}^t(x \mid y)$ up to some additive error is NP-hard for $t \gg |y|$. We complement this result by proving NP-hardness of approximating *sublinear*-time-bounded conditional Kolmogorov complexity up to a multiplicative factor of $|x|^{1/(\log\log|x|)^{O(1)}}$ for $t \ll |y|$. Our NP-hardness proof presents a new connection between sublinear-time-bounded conditional Kolmogorov complexity and a secret sharing scheme.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases resource-bounded Kolmogorov complexity, average-case complexity, pseudorandomness, hardness of approximation, unconditional lower bound

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.26

Funding JST, PRESTO Grant Number JPMJPR2024, Japan.

Acknowledgements I thank Mikito Nanashima for helpful discussion, Rahul Santhanam for raising the question of finding an equivalent notion of the weak universal heuristic scheme, which inspired Lemma 7.5, and anonymous reviewers for strongly inspiring me to include Appendix A.1.

1 Introduction

One of the basic facts in information theory is symmetry of mutual information: For two random variables X and Y, the mutual information $I(X : Y)$ is symmetric: $I(X : Y) = I(Y : X)$. In terms of Shannon entropy H(-), this equality is equivalent to

$$\mathrm{H}(Y) - \mathrm{H}(Y \mid X) = \mathrm{H}(X) - \mathrm{H}(X \mid Y).$$

An alternative way of measuring the amount of information is to use *Kolmogorov complexity*. The Kolmogorov complexity $\mathrm{K}(x)$ of a finite string $x \in \{0,1\}^*$ is defined to be the length of a shortest program that prints x. While Shannon entropy can measure the average amount of information in a *collection* of strings, the notion of Kolmogorov complexity enables

quantifying the amount of information in an *individual* string, thereby giving a finer notion than Shannon entropy. Kolmogorov complexity has had fundamental impacts on theoretical computer science [55].

Kolmogorov and Levin [74] established a fundamental property of Kolmogorov complexity: *Symmetry of information for Kolmogorov complexity* states that

$$\mathrm{K}(y) - \mathrm{K}(y \mid x) = \mathrm{K}(x) - \mathrm{K}(x \mid y) \pm O(\log(|x| + |y|)) \tag{1}$$

for any strings x and $y \in \{0,1\}^*$. Here, $\mathrm{K}(x \mid y)$ denotes the conditional Kolmogorov complexity of x given y, i.e., the length of a shortest program that prints x given y as input.

One disadvantage of Kolmogorov complexity, which was already noted in the seminal paper of Kolmogorov [51], is that $\mathrm{K}(x)$ does not take into account the complexity of generating the string x. Kolmogorov suggested considering t-time-bounded Kolmogorov complexity, denoted by $\mathrm{K}^t(x)$, which is the shortest size of a program that prints x in time t. According to Levin [52], as early as 1967 Kolmogorov suggested time-bounded versions of symmetry of information as an interesting avenue of research. We consider the following time-bounded version of symmetry of information.[1]

▶ **Definition 1.1.** Symmetry of information for time-bounded Kolmogorov complexity (SoI) *refers to the following hypothesis: There exists a polynomial p such that for any strings $x \in \{0,1\}^*$ and $y \in \{0,1\}^*$, for every $t \geq |x| + |y|$,*

$$\mathrm{K}^{p(t)}(x \mid y) + \mathrm{K}^{p(t)}(y) - \log p(t) \leq \mathrm{K}^t(x,y). \tag{SoI}$$

Throughout this paper, we refer to this hypothesis as SoI.

Note that $\mathrm{K}^t(x,y) \leq \mathrm{K}^{t/4}(y \mid x) + \mathrm{K}^{t/4}(x) + O(1)$ unconditionally holds for all large t because strings x and y can be computed by running a program of length $\mathrm{K}^{t/4}(x)$ that outputs x in time $t/4$ and a program of length $\mathrm{K}^{t/4}(y \mid x)$ that takes x as input and outputs y in time $t/4$.[2] Consequently, SoI implies that

$$\mathrm{K}^{p(t)}(x \mid y) + \mathrm{K}^{p(t)}(y) \leq \mathrm{K}^{t/4}(y \mid x) + \mathrm{K}^{t/4}(x) + O(\log t),$$

which is a natural time-bounded analogue of Equation (1).

The original Kolmogorov–Levin proof of Equation (1) is based on an exhaustive search of all strings of a given length. By applying the original proof to a space-bounded setting, symmetry of information for *space-bounded* Kolmogorov complexity was proved by Longpré and Mocas [57]. Interestingly, as is repeatedly stated by Levin (e.g., in [53, 19, 54]), even before the notion of NP-completeness was formalized, Kolmogorov envisioned that investigating symmetry of information for time-bounded Kolmogorov complexity would be a good approach toward $\mathsf{P} \neq \mathsf{NP}$. To quote [53],

> "this information symmetry theorem may be a good test case to prove that for some tasks exhaustive search cannot be avoided (in today's terms, $\mathsf{P} \neq \mathsf{NP}$)."

[1] Previous works [57, 58] studied a slightly different version called *polynomial*-time-bounded symmetry of information, in which the time bound t of SoI is fixed to an arbitrary polynomial. SoI is stronger than polynomial-time-bounded symmetry of information, which makes Theorem 1.2 stronger. Moreover, it is not hard to observe that the results of [57, 58] hold for SoI. We also mention that our version of SoI is useful to show worst-case-to-average-case connections; see Section 7.

[2] For our definition of time-bounded Kolmogorov complexity (Definition 3.1), there is a larger overhead of the simulation of the programs. For simplicity, we ignore this minor issue in this introduction.

In 1992, Longpré and Watanabe [58][3] showed that $\mathsf{P} = \mathsf{NP}$ implies SoI, and that SoI implies the non-existence of one-way functions. Their results can be seen as formalizing Kolmogorov's insightful approach to $\mathsf{P} \neq \mathsf{NP}$: Finding a *single* sequence of pairs (x, y) of strings that violates SoI implies $\mathsf{P} \neq \mathsf{NP}$! Despite the rich literature on Kolmogorov complexity [55], providing an exact characterization of SoI remains elusive. In fact, it has been a long-standing open question to improve the results of [58] over the last three decades.

In this work, we improve the result of [58] by showing that SoI holds under the weaker assumption that NP is easy on average.

▶ **Theorem 1.2.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then SoI holds.*

Here, the statement $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ means that for every language L in NP and for every polynomial-time samplable distribution $\mathcal{D}$,[4] there exists an errorless heuristic scheme that solves L on inputs sampled from $\mathcal{D}$; we refer the reader to the survey of Bogdanov and Trevisan [16] for background on average-case complexity theory.

We find Theorem 1.2 surprising: The assumption of Theorem 1.2 is that NP is easy on *most* instances, whereas SoI is an inequality for *every* pair of strings x and y; thus, it is natural to expect that SoI is related to the worst-case complexity of some problem. Indeed, we show that SoI is closely related to the worst-case complexity of GapMINKT, which is the problem of approximating time-bounded Kolmogorov complexity. We also show that, under the plausible assumption that E requires exponential-size circuits, SoI is sandwiched between the existence of an *errorless* heuristic scheme (AvgP) for computing $\mathrm{K}^t(\text{-})$ and the existence of an *error-prone* heuristic scheme (HeurP) for computing $\mathrm{K}^t(\text{-})$, thereby narrowing SoI.

▶ **Theorem 1.3** (informal; see Theorem 8.2 for the formal version)**.** *Assume that* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$*. In the following list, we have 1* $\iff$ *2* $\implies$ *3* $\implies$ *4 and 3* $\implies$ *5.*

1. $\mathrm{GapMINKT} \in \mathsf{P}$.
2. *For every polynomial-time samplable distribution* $\mathcal{D}$ *and for all sufficiently large polynomials* t*, there exists an* errorless *heuristic scheme that computes* $\mathrm{K}^{t(n)}(x)$ *for a random input* $x \sim \mathcal{D}_n$*.*
3. *SoI holds.*
4. *For every polynomial-time samplable distribution* $\mathcal{D}$ *and for all sufficiently large polynomials* t*, there exists an* error-prone *heuristic scheme that computes* $\mathrm{K}^{t(n)}(x)$ *for a random input* $x \sim \mathcal{D}_n$*.*
5. $\mathrm{Gap}_\tau\mathrm{MINKT} \in \mathsf{DTIME}(2^{O(n/\log n)})$ *for some function* $\tau(n,t) = 2^{O(n/\log n)}$*.*

The difference between errorless and error-prone heuristics is as follows: Error-prone heuristics compute a problem on most instances but are allowed to output a wrong answer. Errorless heuristics also compute a problem on most instances; in addition, they must output either a correct answer or a special failure symbol "$\perp$". Hirahara and Santhanam [38] showed the equivalence between errorless and error-prone average-case complexities of any problem that admits an instance checker in the sense of Blum and Kannan [15]. In particular, Theorem 1.3 implies that SoI is exactly characterized by the worst-case complexity of GapMINKT if time-bounded Kolmogorov complexity admits an instance checker (and E requires exponential-size circuits). Whether an instance checker can be constructed or not is an interesting open question.

[3] The conference version of [58] was presented in ISAAC 1992.

[4] In fact, even if we restrict the distribution $\mathcal{D}$ to $\{\mathcal{U}, \mathcal{T}\}$, the statement remains the same. Here, $\mathcal{U}$ denotes the uniform distribution and $\mathcal{T}$ denotes the tally distribution supported only on $\{1\}^*$. This is a consequence of the theorems of Impagliazzo and Levin [45], Buhrman et al. [18]; see [35].

Previously, under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, [35] showed *weak symmetry of information*, which states that $|x| + \mathrm{K}^{\mathsf{poly}(t)}(y) - O(\log t) \leq \mathrm{K}^t(x,y)$ holds with high probability over $x \sim \{0,1\}^n$ uniformly chosen at random. Our proof is largely inspired by the proof of weak symmetry of information. In fact, it is "not hard" to prove Theorem 1.2 itself using the proof techniques developed in [35][5]. Independently of this work, Goldberg and Kabanets [28] proved Theorem 1.2 using the same proof techniques and used it to provide an alternative proof of main results of [35].[6]

Our main contribution is to develop more general proof techniques that are applicable to any resource-bounded Kolmogorov complexity and to demonstrate their significance: In general, for any (randomized) resource-bounded Kolmogorov complexity $\mathrm{K}\mu(\text{-})$, we show that symmetry of information for $\mathrm{K}\mu(\text{-})$ follows from an efficient algorithm that computes $\mathrm{K}\mu(\text{-})$. Since symmetry of information is a fundamental property of Kolmogorov complexity, we expect that our new proof techniques will have a variety of applications in future. In the following two subsections, we present two specific applications that concern two central questions of computational complexity theory: (1) $\mathsf{P} \neq \mathsf{NP}$ and (2) $\mathsf{P} \neq \mathsf{NP} \Longrightarrow \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

1.1 Kolmogorov's Approach to the P versus NP Problem

Is Kolmogorov's approach to the P versus NP problem an illusion made in the premature era of theoretical computer science? We present strong evidence that this is not the case: Using our new proof techniques of proving symmetry of information, we demonstrate that Kolmogorov's insightful approach is indeed useful to prove lower bounds! Specifically, we consider a randomized version of Levin's Kt-complexity. The randomized Kt-complexity of a string x, denoted by $\mathrm{rKt}(x)$, is defined as the minimum of $|M| + \log t$ over all the randomized programs M that print x with probability at least $\frac{2}{3}$ in time t [62]. Let GapMrKtP be the promise problem of approximating $\mathrm{rKt}(x)$ on input x up to an additive error of $O(\log |x|)$. Using an exhaustive search, it is easy to observe that GapMrKtP $\in$ pr-$\mathsf{BPTIME}(2^{O(n)})$. We prove that this exhaustive search cannot be avoided.

▶ **Theorem 1.4.** GapMrKtP $\notin$ i.o.$\mathsf{BPTIME}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$.

The proof of Theorem 1.4 is given by implementing Kolmogorov's approach in the case of rKt: Ronneburger [66] proved that symmetry of information for Levin's Kt-complexity does not hold. Using this result, we observe that symmetry of information for rKt is also false (conditionally). Then, using our proof techniques, we prove symmetry of information for rKt from an efficient hypothetical algorithm that computes GapMrKtP, which leads to a contradiction.

Previously, Oliveira [62] proved GapMrKtP $\notin$ pr-$\mathsf{BPTIME}(2^{\log^{O(1)} n})$ using a different proof technique. Theorem 1.4 improves this result in the following two respects:[7]

1. The lower bound is improved from a quasi-polynomial $2^{\log^{O(1)} n}$ to a strongly exponential $2^{\epsilon n}$, which is optimal up to a constant exponent.
2. Our lower bound works against algorithms that are correct infinitely often (i.o.).

[5] if the feasibility of proving Theorem 1.2 is given as advice; see Appendix A.1 for more comments.
[6] Independently, we also find a similar alternative proof; see Section 7.
[7] A caveat is that our lower bound is applicable to BPP, i.e., randomized algorithms that satisfy the promise of BPP-type algorithms over all the inputs, whereas the lower bound of [62] is also applicable to pr-BPP. In this respect, these two lower bounds are incomparable. We leave an open problem of improving our lower bound to i.o.pr-$\mathsf{BPTIME}(2^{\epsilon n})$. However, we note that $\mathsf{BPP} = \mathsf{P} =$ pr-BPP under the plausible assumption that E requires exponential-size circuits [46], in which case there is no difference between the two lower bounds.

The fact that Theorem 1.4 improves the previous state-of-the-art result of [62] indicates the significance of the new approach of proving lower bounds via symmetry of information. [34] developed yet another proof technique of showing lower bounds for resource-bounded Kolmogorov complexity and showed that $\mathrm{K}^{t(|x|)}(x)$ cannot be computed in polynomial time if $t(n) = n^{\omega(1)}$. Interestingly, this previous proof technique does not seem to be applicable to the setting of Theorem 1.4,[8] which suggests the "novelty" of Kolmogorov's approach.

Why was Kolmogorov's approach not implemented before? Previously, the only proof technique of proving symmetry of information was due to Kolmogorov and Levin [74]; for example, the aforementioned work of Longpré and Watanabe [58] showed symmetry of information from a hypothetical efficient algorithm that computes some problem in PH, by applying the original Kolmogorov–Levin proof to the resource-bounded case. It was not known if symmetry of information can be proved from an efficient algorithm that computes resource-bounded Kolmogorov complexity. Our general proof techniques of showing symmetry of information from efficient algorithms make it possible to implement Kolmogorov's insightful approach, for the first time. Our results demonstrate the significance of Kolmogorov's approach and hint that it may indeed lead to the resolution of the P versus NP problem in future.

1.2 Toward Excluding Heuristica

One of the central open questions in computational complexity theory is to show the equivalence between the worst-case and average-case complexities of NP, e.g., $\mathsf{P} = \mathsf{NP} \iff \mathsf{DistNP} \subseteq \mathsf{AvgP}$. Equivalently, this question is well known as whether Heuristica can be excluded from Impagliazzo's five possible worlds [43]. Heuristica is a hypothetical world in which $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. There are at least three types of results explaining the difficulty of excluding Heuristica: Standard proof techniques for relating worst-case and average-case complexities, such as black-box nonadaptive reductions [23, 17], hardness amplification procedures [73, 72], and relativizing proof techniques [44, 36], are known to be incapable of excluding Heuristica. To make progress on this central problem, we need to develop new types of proof techniques that are not subject to any of these technical barriers.

Recently, several new proof techniques of analyzing average-case complexity have been developed based on *meta-complexity*. Meta-complexity refers to the complexity of a problem that asks for complexity. One representative example of meta-computational problems is MINKT [50], which asks to compute $\mathrm{K}^t(x)$ given $(x, 1^t)$ as input. [31] developed a non-black-box reduction technique that goes beyond the limits of black-box reductions presented by Feigenbaum and Fortnow [23], Bogdanov and Trevisan [17], and showed $\mathrm{GapMINKT} \in \mathsf{P}$ if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Informally, GapMINKT is the problem of approximating the time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ up to an additive error of $O(\log t)$ on input $(x, 1^t)$.[9] As a consequence, Heuristica can be excluded if GapMINKT is NP-hard. Subsequently, [35] developed proof techniques that simultaneously overcome the limits of black-box reductions [23, 17] and the impossibility of hardness amplification procedures [73, 72], and proved (among other results) that $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ follows from the worst-case assumption that $\mathsf{UP} \not\subseteq \mathsf{DTIME}(2^{O(n/\log n)})$. The next goal along this research line is to develop a proof

[8] By combining the infinitely-often pseudo-deterministic subexponential-time algorithm of Oliveira and Santhanam [63] with [34], it is possible to prove $\mathrm{GapMrKtP} \notin \text{pr-}\mathsf{BPTIME}(2^{\log^{O(1)} n})$. However, this ends up with essentially the same proof as [62].

[9] More precisely, GapMINKT is the problem of computing a value v such that $\mathrm{K}^{p(t)}(x) - \log p(t) \leq v \leq \mathrm{K}^t(x)$ on input $(x, 1^t)$ such that $|x| \geq t$, where p is some polynomial; see Fact 3.4.

technique that overcomes the limits of relativizing barriers, for which quantitatively tight results are known: Building on the work of Impagliazzo [44], Hirahara and Nanashima [36] showed that a relativizing proof technique is incapable of improving the time complexity $2^{O(n/\log n)}$ achieved in [35] to $2^{o(n/\log n)}$. Chen et al. [20] *bypassed* this relativization barrier by showing that $\mathsf{NTIME}[n]$ is hard on average for quasi-linear-time algorithms if $\mathsf{UP} \not\subseteq \mathsf{DTIME}\left(2^{\sqrt{n \log n}}\right)$. To make further progress, we would need to develop a proof technique that *overcomes*[10] the relativization barrier.

In search of non-relativizing proof techniques, we review another line of research on meta-complexity. A long-standing open question on meta-complexity, which dates back to as early as the 1960s [69, 8], is to prove NP-hardness of meta-computational problems, such as MINKT. Recently, there has been substantial progress on this question in a line of research [7, 37, 40, 42, 41, 56, 4]. A salient feature of the proof techniques developed in this line of research is that these are apparently non-relativizing. Specifically, Ko [50] constructed an oracle under which $\mathsf{NP} \neq \mathsf{P}$ but GapMINKT is easy, thereby showing that a relativizing proof technique is incapable of showing NP-hardness of GapMINKT. In contrast, building on the work of Ilango [40], Allender et al. [4], Liu and Pass [56] showed NP-hardness of the problem of computing sublinear-time-bounded conditional Kolmogorov complexity, which we denote by MINcKT ("c" stands for "conditional").

Toward excluding Heuristica, we aim at combining the two lines of research reviewed above. One interpretation of SoI is an approximate equality between time-bounded conditional Kolmogorov complexity and its unconditional version. Specifically, SoI implies that

$$\begin{aligned} \mathrm{K}^{p(t)}(x \mid y) \leq \mathrm{K}^{t}(x,y) - \mathrm{K}^{p(t)}(y) + O(\log t) &\leq \mathrm{K}^{t/4}(x \mid y) + \mathrm{K}^{t/4}(y) - \mathrm{K}^{p(t)}(y) + O(\log t). \\ &= \mathrm{K}^{t/4}(x \mid y) + \mathrm{cd}^{t/4,p(t)}(y) + O(\log t). \end{aligned} \tag{2}$$

Here, $\mathrm{cd}^{s,t}(y) := \mathrm{K}^{s}(y) - \mathrm{K}^{t}(y)$ is called the (s,t)-*time-bounded computational depth* of y [9, 35] and is known to be small for most strings y [9, 11] (see Lemma 8.4 for a formal statement). Equation (2) means that the conditional Kolmogorov complexity $\mathrm{K}^{t'}(x \mid y)$ can be approximated up to an additive error of $\mathrm{cd}^{t/4,p(t)}(y) + O(\log t)$ for some $t' \in [t/4, p(t)]$ if time-bounded Kolmogorov complexity can be efficiently computed.[11] In other words, SoI enables reducing the problem of approximating *conditional* Kolmogorov complexity to the problem of approximating *unconditional* Kolmogorov complexity; i.e., GapMINcKT is reducible to GapMINKT. As a consequence of Theorem 1.2, we generalize the approach of [31] for excluding Heuristica as follows:

▶ **Theorem 1.5.** *If* $\mathrm{Gap}_{\tau}\mathrm{MINcKT}$ *is* NP*-hard under randomized reductions for every polynomial* τ*, then Heuristica does not exist; that is,* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *if and only if* $\mathsf{P} = \mathsf{NP}$*. Here, the problem* $\mathrm{Gap}_{\tau}\mathrm{MINcKT}$ *asks to compute an integer* $k \in \mathbb{N}$ *such that*

$$\mathrm{K}^{\tau(|x|,|y|,t)}(x \mid y) - \log \tau(|x|,|y|,t) \leq k \leq \mathrm{K}^{t}(x \mid y) + \mathrm{cd}^{t,\tau(|x|,|y|,t)}(y)$$

on input $(x, y, 1^t)$.

[10] We say that a proof technique A *overcomes* a barrier B if A proves a statement S such that proof techniques subject to B are incapable of proving S. The results of [20] do not necessarily overcome the relativization barriers of [44, 36] because the conclusion that $\mathsf{NTIME}[n]$ is hard on average is weaker than $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

[11] It is unlikely that the additive error term $\mathrm{cd}^{t/4,p(t)}(y)$ can be improved by using relativizing proof techniques; see Remark 6.4.

More generally, our proof techniques provide a general approach of reducing conditional resource-bounded Kolmogorov complexity $\mathrm{K}\mu(\text{-} \mid \text{-})$ to its unconditional analogue $\mathrm{K}\mu(\text{-})$ for every resource-bounded Kolmogorov complexity $\mathrm{K}\mu$: Using a hypothetical efficient algorithm that computes $\mathrm{K}\mu(\text{-})$, we prove symmetry of information for $\mathrm{K}\mu$. Then, we use symmetry of information to reduce conditional complexity $\mathrm{K}\mu(\text{-} \mid \text{-})$ to $\mathrm{K}\mu(\text{-})$.

Next, we investigate whether $\mathrm{Gap}_\tau\mathrm{MINcKT}$ can be proved to be NP-hard. Building on the proof techniques developed in [40, 4, 56], we prove NP-hardness of $\mathrm{Gap}_\tau\mathrm{MINcKT}$ if the approximation parameter $\tau(|x|,|y|,t)$ is sublinear in $|y|$, e.g., $\tau(|x|,|y|,t) \leq |x|^{O(1)} \cdot |y|^{0.99} \cdot t^{O(1)}$.

▶ **Theorem 1.6** (informal; see Theorem 6.6 for the formal version). *Let $c > 1$ be an arbitrary constant. Let $\tau\colon \mathbb{N}^3 \to \mathbb{N}$ be a function such that $\tau(n,m,t) \leq n^c \cdot m^{1-1/c} \cdot t^c$ for all large n, m, and $t \in \mathbb{N}$. Then,* $\mathrm{Gap}_\tau\mathrm{MINcKT}$ *is* NP*-hard under randomized polynomial-time reductions. Moreover, it is* NP*-hard to approximate $\mathrm{K}^t(x \mid y)$ to within a factor of $|x|^{1/(\log\log|x|)^{O(1)}}$.*

The contributions of Theorem 1.6 are two-fold.

1. We improve an inapproximability factor. Most previous hardness proofs in the literature [7, 37, 40, 42, 41, 56, 4] reduce the set cover problem to meta-computational problems. In particular, the inapproximability factor obtained in [4, 56] is at most $O(\log|x|)$. Our inapproximability factor $|x|^{1/(\log\log|x|)^{O(1)}}$ is a nearly exponential improvement and is close to the trivial upper bound of $|x|$. In addition, we prove the NP-hardness of approximating $\mathrm{K}^t(x \mid y)$ even when there is an additive error term $\mathrm{cd}^{t,\tau(|x|,|y|,t)}(y)$.
   Our strong inapproximability is not only a quantitative improvement but also a *qualitative* improvement. In fact, the two previous results [4, 56] prove NP-hardness of two different versions of conditional Kolmogorov complexity, McKTP and MINcKT. Here, McKTP is the problem of computing the KT-complexity of x given y, i.e., the minimum of $|M| + t$ over all programs M that compute x given y as input in time t. Because of the previous weak inapproximability, these two results are proved by different proofs. Our inapproximability is strong enough to prove NP-hardness of McKTP and MINcKT simultaneously.
2. We present a new connection between a secret sharing scheme and sublinear-time-bounded conditional Kolmogorov complexity. This new connection abstracts essential proof ideas implicitly developed in the line of research and enables us to provide a streamlined proof.

Perspective

How should we interpret our results? Optimistically, our results indicate that excluding Heuristica might be reachable by combining current proof techniques. We are not aware of any technical barrier that suggests Theorems 1.5 and 1.6 cannot inherently be matched. It is currently plausible that all the technical barriers to excluding Heuristica that we are aware of ([23, 17, 73, 72, 44, 36]) can be overcame in this way. Pessimistically, our results can be understood as indicating the difference between sublinear-time-bounded Kolmogorov complexity and time-bounded Kolmogorov complexity. However, we emphasize that the fact that Theorems 1.2 and 1.5 do not work for sublinear-time bounded Kolmogorov complexity is not an inherent limitation of our proof techniques. Our proof techniques make it possible to prove symmetry of information even for some version of sublinear-time bounded Kolmogorov complexity *with error* [25] by using a pseudorandom generator construction from average-case hardness. Whether there is a technical barrier that explains the inherent difficulty of the current approach remains to be explored.[12]

[12] A candidate is the limits of oracle-independent reductions of [39].

1.3 Related Work

Symmetry of Information

Lee and Romashchenko [52] unconditionally showed that for every polynomial p, for some polynomial q, for any strings x and y of length n,

$$\mathrm{KAMD}^{q(n)}(x \mid y) + \mathrm{KAMD}^{q(n)}(y) - O(\log^3 n) \leq \mathrm{K}^{p(n)}(x, y).$$

Here, KAMD denotes an Arthur–Merlin variant of distinguishing Kolmogorov complexity. Consequently, symmetry of information holds up to an additive error of $O(\log^3 n)$ under the assumption that $\mathrm{K}^{q'(n)}(x \mid y) \leq \mathrm{KAMD}^{q(n)}(x \mid y) + O(\log n)$ holds for any strings x and y. They also showed that this assumption is in fact equivalent to $\mathsf{P} = \mathsf{NP}$ and hence gave no improvement over the conditional result of [58].

Symmetry of Information and Hardness

Symmetry of information is used to show hardness results for Kolmogorov complexity [34]. Similarly, an inequality analogous to SoI is used to show NP-hardness of a multi-output variant of MCSP [42]. The *Minimum Circuit Size Problem* (MCSP [47]) is the meta-computational problem that asks to compute the size of a minimum circuit that computes a given function $f\colon \{0,1\}^n \to \{0,1\}$ represented as the truth table of length 2^n. Ilango [40] showed that a conditional variant of MCSP is NP-hard under randomized reductions. Subsequently, Ilango et al. [42] proved NP-hardness of a multi-output variant of MCSP. Their proof is based on an inequality reminiscent of SoI, which enables translating the hardness result for the conditional variant of MCSP [40] to an unconditional variant.

The Shortest Vector Problem

The complexity landscape about GapMINcKT is reminiscent of the shortest vector problem GapSVP. If "approximation factors" are small, then GapMINcKT and GapSVP are NP-hard [30]. If "approximation factors" are large, then these problems admit worst-case-to-average-case connections. However, there are fundamental differences: On one hand, GapSVP is known to be in $\mathsf{NP} \cap \mathsf{coNP}$ for a large approximation factor [29, 1] and thus is unlikely to be NP-hard. Moreover, this phenomenon is inherent: The limits of black-box reductions presented by Bogdanov and Trevisan [17] show that any problem reducible to DistNP by nonadaptive black-box reductions must be in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. The worst-case-to-average-case connections presented for GapSVP are given by black-box reductions [2, 60] and thus subject to this barrier. On the other hand, the worst-case-to-average-case connection for GapMINcKT (Theorem 1.5) is not subject to the barrier of [17] since it is proved by a *non-black-box* reduction, i.e., a reduction that exploits the efficiency of a hypothetical heuristic algorithm for DistNP. There is no evidence against NP-hardness of $\mathrm{Gap}_\tau\mathrm{MINcKT}$ for any large polynomial τ; on the contrary, the NP-hardness of $\mathrm{Gap}_\tau\mathrm{MINcKT}$ for sublinear-time-bounded functions τ can be seen as supporting evidence.

2 Proof Techniques

In this section, we outline our proof techniques.

2.1 Symmetry of Information in Heuristica

We first sketch a proof of Theorem 1.2, which shows SoI in Heuristica. In doing so, we present a general technique of proving symmetry of information from any efficient algorithm that computes resource-bounded Kolmogorov complexity and any pseudorandom generator construction. We need two lemmas from previous work, which we review below.

Our proof of Theorem 1.2 is based on the meta-complexity of time-bounded Kolmogorov complexity: We crucially use the fact that time-bounded Kolmogorov complexity can be approximated in Heuristica.

▶ **Lemma 2.1** ([31, 33]). *If* DistNP $\subseteq$ AvgP*, then* GapMINKT $\in$ P.

This result makes it possible to *approximate* $\mathrm{K}^t(x)$. For the purpose of exposition, we assume below that there exists an algorithm which, on input (x,t), computes $\mathrm{K}^t(x)$ *exactly* in time $\mathsf{poly}(|x|,t)$. It is worth emphasizing that our proof deviates from the original Kolmogorov–Levin proof in this respect: Our proof is "meta-computational" in that we use GapMINKT $\in$ P, whereas the Kolmogorov–Levin proof cannot be "meta-computational" since the resource-unbounded Kolmogorov complexity K(-) cannot be computed in finite time steps.

The second lemma is the reconstruction property of a *k-wise direct product generator* [34], which turned out to be a fundamental tool for analyzing Kolmogorov complexity [33, 65, 32, 35]. A k-wise direct product generator $\mathrm{DP}_k\colon \{0,1\}^n \times \{0,1\}^{nk} \to \{0,1\}^{nk+k}$ is defined as follows:

$$\mathrm{DP}_k(x;z) := (z^1,\dots,z^k,\langle z^1,x\rangle,\dots,\langle z^k,x\rangle)$$

for every $x \in \{0,1\}^n$ and every $z=(z^1,\dots,z^k) \in (\{0,1\}^n)^k$, where $\langle x,y\rangle$ denotes the inner product of x and $y \in \{0,1\}^n$ over GF(2), i.e., $\langle x,y\rangle := (\sum_{i=1}^n x_i y_i) \mod 2$. The k-wise direct product generator $\mathrm{DP}_k(x;\text{-})$ is a pseudorandom generator secure against an algorithm D if $\mathrm{K}(x \mid D) > k + O(\log |x|)$. More formally, we have the following property:

▶ **Lemma 2.2** (Deterministic Reconstruction for DP_k; see [35]). *Assume that* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$*. Then, there exists a polynomial* p *such that, for every* $n \in \mathbb{N}$*,* $x \in \{0,1\}^n$*, parameters* $k, \epsilon^{-1}, s \in \mathbb{N}$*, and for every randomized circuit* D *of size* s *such that*

$$\left|\Pr_{z,r}[D(\mathrm{DP}_k(x;z);r)=1] - \Pr_{w,r}[D(w;r)=1]\right| \geq \epsilon,$$

where $z \sim \{0,1\}^{nk}$*,* $w \sim \{0,1\}^{nk+k}$*, and* $r \sim \{0,1\}^s$*, it holds that*

$$\mathrm{K}^{p(ns/\epsilon)}(x \mid D) \leq k + \log p(ns/\epsilon).$$

The assumption of Lemma 2.2 is satisfied in Heuristica, as Buhrman et al. [18] showed that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$ if DistNP $\subseteq$ AvgP. We mention in passing that the proof of Lemma 2.1 also uses Lemma 2.2.

Now, we sketch the proof of SoI from an efficient algorithm that computes K^t(-) and the pseudorandom generator construction DP(-;-). Fix strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ and an integer $t \geq n+m$. We claim

$$\mathrm{K}^{q(t)}(x \mid y) + \mathrm{K}^{q(t)}(y) - \log q(t) \leq \mathrm{K}^t(x,y)$$

for some universal polynomial q. The proof is given by analyzing the following three values for $z \sim \{0,1\}^{nk}$, $w \sim \{0,1\}^{nk+k}$, $z' \sim \{0,1\}^{m\ell}$, and $w' \sim \{0,1\}^{m\ell+\ell}$:

$$\begin{array}{lccc} \mathrm{K}^{t'}(& \mathrm{DP}_k(x;z), & \mathrm{DP}_\ell(y;z') &), \\ \mathrm{K}^{t'}(& w, & \mathrm{DP}_\ell(y;z') &), \\ \mathrm{K}^{t'}(& w, & w' &), \end{array}$$

where t', k, and ℓ are parameters chosen later.

First, by a standard counting argument, we have

$$\mathrm{K}^{t'}(w,w') \approx |w| + |w'| \tag{3}$$

with high probability over the random choice of w and w'.

Next, by the contrapositive of Lemma 2.2, $\mathrm{DP}_\ell(y;\text{-})$ is a pseudorandom generator secure against uniform algorithms if $\ell \ll \mathrm{K}^{p(t)}(y)$, where p is some polynomial; thus, we obtain

$$\mathrm{K}^{t'}(w, \mathrm{DP}_\ell(y;z')) \approx \mathrm{K}^{t'}(w,w'). \tag{4}$$

In more detail, consider a randomized circuit D that takes w' as input as well as random bits w and checks whether $\mathrm{K}^{t'}(w,w') \geq \theta$ for a threshold θ. By Lemma 2.2, if D can distinguishes $\mathrm{DP}_\ell(y;z')$ and w', we would get $\mathrm{K}^{p(t)}(y) \leq \ell + \log p(t)$. We choose $\ell := \mathrm{K}^{p(t)}(y) - \log p(t) - 1$ so that this inequality does not hold. Equation (4) follows from the fact that D cannot distinguish the pseudorandom distribution $\mathrm{DP}_\ell(y;z')$ from the uniform distribution w'.

By combining Equations (3) and (4), we conclude that

$$\mathrm{K}^{t'}(w, \mathrm{DP}_\ell(y;z')) \gtrsim |w| + |w'| = |z| + k + |z'| + \ell \approx |z| + k + |z'| + \mathrm{K}^{p(t)}(y) \tag{5}$$

with high probability over the random choice of w and z'.

On the other hand, observe that for some $t' := \mathsf{poly}(t)$,

$$\mathrm{K}^{t'}(\mathrm{DP}_k(x;z), \mathrm{DP}_\ell(y;z')) \leq \mathrm{K}^t(x,y) + |z| + |z'| + O(\log n) \tag{6}$$

holds because the strings $\mathrm{DP}_k(x;z)$ and $\mathrm{DP}_\ell(y;z')$ can be computed from k, ℓ, z, z', and a program of size $\mathrm{K}^t(x,y)$ that outputs (x,y) in time t.

Comparing Equations (5) and (6), for a sufficiently large $k := \mathrm{K}^t(x,y) - \mathrm{K}^{p(t)}(y) + O(\log t)$, there exists a threshold θ' such that $\mathrm{K}^{t'}(w, \mathrm{DP}_\ell(y;z')) \geq \theta'$ with high probability over the random choice of w and z', whereas $\mathrm{K}^{t'}(\mathrm{DP}_k(x;z), \mathrm{DP}_\ell(y;z')) < \theta'$ for every z and z'. Let D_y be a randomized circuit that takes an input w and random bits z' and checks whether $\mathrm{K}^{t'}(w, \mathrm{DP}_\ell(y;z')) < \theta'$. Then, we obtain

$$1 = \Pr_{z,z'}[D_y(\mathrm{DP}_k(x;z);z') = 1] \gg \Pr_{w,z'}[D_y(w;z') = 1] \approx 0$$

Using Lemma 2.2, we obtain

$$\mathrm{K}^{\mathsf{poly}(t)}(x \mid D_y) \leq k + O(\log t) = \mathrm{K}^t(x,y) - \mathrm{K}^{p(t)}(y) + O(\log t).$$

Since the circuit D_y can be efficiently constructed from y, we have $\mathrm{K}^t(D_y \mid y) \leq O(\log t)$. It follows that for some large polynomial q,

$$\mathrm{K}^{q(t)}(x \mid y) \leq \mathrm{K}^t(x \mid D_y) + \mathrm{K}^t(D_y \mid y) + O(1) \leq \mathrm{K}^t(x,y) - \mathrm{K}^{p(t)}(y) + O(\log t)$$

as desired. This completes the proof of Theorem 1.2, except that the circuits D and D_y must carefully be defined using an algorithm for GapMINKT. A complete proof can be found in Section 4. We also present a simple proof of Theorem 1.2 based on weak symmetry of information of [35] in Appendix A and clarify that the techniques of proving Theorem 1.2 were already developed in [35].

2.2 Lower Bounds for rKt via Kolmogorov's Approach

Observe the generality of the proof technique described in Section 2.1: The only two ingredients of the proof of symmetry of information are (1) a hypothetical algorithm that computes resource-bounded Kolmogorov complexity and (2) a pseudorandom generator construction. The same proof strategy can be applied to any resource-bounded Kolmogorov complexity; the tightness of the resulting SoI is determined by the advice complexity of a pseudorandom generator construction.

We review the notion of pseudorandom generator construction and its advice complexity. A *pseudorandom generator construction* $G\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is an efficiently computable function that has a (randomized) *reconstruction procedure* R with the following property: For every $x \in \{0,1\}^n$, if an oracle D distinguishes the output distribution $G(x;\text{-})$ from the uniform distribution, then there exists an advice string $\alpha \in \{0,1\}^a$ such that the randomized algorithm R outputs x given α as advice with high probability. The length a of the advice string is referred to as the *advice complexity* of G. Trevisan and Vadhan [71] showed that the advice complexity a is always at least $m - d - 3$. In the case of the k-wise direct product generator $\mathrm{DP}_k(x;\text{-})$, the advice complexity is $k + O(\log n)$, which is close to the lower bound of $m - d - 3 = k - 3$. The fact that the additive term of SoI is $O(\log t)$ comes from the nearly optimal advice complexity of $\mathrm{DP}_k(x;\text{-})$. The disadvantage of DP_k, however, is that the seed length $d = nk$ is too large, which prevents us from obtaining the tight lower bound of Theorem 1.4.

There are several different constructions of pseudorandom generators in the literature (e.g., [70, 64, 68, 34]). For example, the pseudorandom generator construction G of Raz et al. [64] satisfies the following properties: The advice complexity of G is at most $m + O(\log^3 n)$. In particular, if some t-time algorithm distinguishes the output distribution $G(x;\text{-})$ from the uniform distribution, then $\mathrm{rKt}(x) \le m + O(\log^3 n) + O(\log t)$. Moreover, the seed length d is $O(\log^3 n)$, which is small enough for our application.

Now, we sketch the proof of Theorem 1.4. For simplicity, we claim that the problem MrKtP of computing rKt *exactly* cannot be solved in time $2^{o(n)}$. Toward a contradiction, assume $\mathrm{MrKtP} \in \mathsf{BPTIME}(2^{o(n)})$. Applying the general proof technique of Section 2.1 to (1) the algorithm that computes MrKtP and (2) the pseudorandom generator construction G, we obtain symmetry of information for rKt:

$$\mathrm{rKt}(x \mid y) + \mathrm{rKt}(y) \le \mathrm{rKt}(x,y) + O(\log^3 n) + o(n)$$

for any strings x and y of length n (Lemma 5.6). Next, we construct a pair (x,y) that violates this symmetry of information. Ronneburger [66] constructed such a pair in the case of Kt-complexity. We apply the same proof idea to the case of rKt-complexity. The pair (x,y) is constructed as follows: Exhaustively search a string $y \in \{0,1\}^n$ such that $\mathrm{rKt}(y) \ge n$ in time $2^{n+o(n)}$. Next, exhaustively search a string $x \in \{0,1\}^n$ such that $\mathrm{rKt}(x,y) \ge 2n - o(n)$ in time $2^{n+o(n)}$. The existence of such a string x is guaranteed by symmetry of information because $\mathrm{rKt}(x \mid y) + \mathrm{rKt}(y) \ge 2n$ holds for a string x such that $\mathrm{rKt}(x \mid y) \ge n$. Overall, the pair (x,y) is constructed in time

$$2^{n+o(n)} + 2^{n+o(n)} = 2^{n+o(n)},$$

However, by the definition of rKt, we obtain $\mathrm{rKt}(x,y) \le O(\log n) + \log 2^{n+o(n)} \le n + o(n)$, which contradicts the lower bound of $\mathrm{rKt}(x,y)$. We present the details in Section 5.

2.3 Secret Sharing Schemes and Sublinear-Time-Bounded Conditional Kolmogorov Complexity

We now present a new connection between a secret sharing scheme and sublinear-time-bounded conditional Kolmogorov complexity.

To do so, however, we first need to clarify the definition of $\mathrm{K}^t(x \mid y)$ for sublinear-time bounds t (i.e., $t \ll |x|$ and $t \ll |y|$). For the standard computational model of Turing machines, it takes at least $|y|$ time steps just to read the entire input y. To define sublinear-time-bounded Kolmogorov complexity in a meaningful way, we assume that algorithms are given random access to each bit of y; in other words, the input y is given as the oracle that, on query $i \in \{1, \ldots, |y|, |y|+1\}$, answers the i-th bit of y if $i \leq |y|$ and a special stop symbol "$\perp$" if $i = |y|+1$. Similarly, it takes at least $|x|$ steps for a Turing machine to output all the bits of x; instead, we assume that an algorithm is given an index $i \in \{1, \ldots, |x|, |x|+1\}$ as input and is asked to compute the i-th bit of x if $i \leq |x|$ and "$\perp$" if $i = |x|+1$. This notion of sublinear-time-bounded Kolmogorov complexity is standard in the literature (e.g., [3, 32]). See Section 3 for the formal definition.

Next, we review the notion of secret sharing scheme. A *secret sharing scheme*, introduced by Blakley [14], Shamir [67], is an algorithm that shares a secret among n parties so that any authorized set of parties can reconstruct the secret but no unauthorized set of parties can obtain any information about the secret. Specifically, let $\mathcal{A} \subseteq 2^{[n]}$ be an *access structure*, i.e., a monotone collection of subsets of $[n]$. We say that a pair $(\mathrm{Share}, \mathrm{Rec})$ of algorithms is a secret sharing scheme for $\mathcal{A}$ if it satisfies the following two properties:

Correctness: For a secret $x \in \{0,1\}^*$ and an authorized set $T \in \mathcal{A}$ of parties,

$$\mathrm{Rec}(\mathrm{Share}(x)_T) = x,$$

where $\mathrm{Share}(x)_T$ denotes the set of shares that are shared to some party $i \in T$.

Perfect Privacy: For an unauthorized set $T \notin \mathcal{A}$ of parties, X and $\mathrm{Share}(X)_T$ are statistically independent for every random variable X.

We refer the reader to the survey of Beimel [12] for more background on secret sharing schemes.

We present a generic reduction that takes an arbitrary secret sharing scheme and reduces the problem of computing the minimum size of an authorized set to sublinear-time-bounded conditional Kolmogorov complexity.

▷ Claim 2.3 (informal; see Theorem 6.14 for the formal version). Let $\mathcal{A} = \{\mathcal{A}_\varphi\}_{\varphi \in \{0,1\}^*}$ be a family of access structures for which there exists an efficient secret sharing scheme. Let $w(\mathcal{A}_\varphi) := \min_{T \in \mathcal{A}_\varphi} |T|$. Then, there is a randomized polynomial-time reduction that takes φ as input and outputs (x, y, t, λ) such that

$$w(\mathcal{A}_\varphi) \cdot \lambda \approx \mathrm{K}^t(x \mid y),$$

where $t \ll |y|$.

Benaloh and Leichter [13] showed that for any monotone formula φ, there exists an efficient secret sharing scheme for the access structure $\mathcal{A}_\varphi$ represented by φ.[13] Applying their secret sharing scheme to Claim 2.3, we obtain a reduction from the Minimum Monotone Satisfying

[13] More generally, Karchmer and Wigderson [49] showed that the access structure represented by any monotone span program admits an efficient secret sharing scheme.

Assignment (MMSA) problem to GapMINcKT. MMSA is known to be hard to approximate within a factor of $|\varphi|^{1/(\log\log|\varphi|)^{O(1)}}$ even for depth-3 monotone formulas[14] [22, 21], which will complete the proof of Theorem 1.6.

We now sketch a proof for Claim 2.3. Here is an outline of the reduction, which generalizes the reduction presented by [4, 56]: We share a random string $x \sim \{0,1\}^\ell$ among n parties using a secret sharing scheme $(\mathrm{Share}_\varphi, \mathrm{Rec}_\varphi)$ for $\mathcal{A}_\varphi$. Let $(s_1, \ldots, s_n) := \mathrm{Share}_\varphi(x)$, where s_i is the share of the i-th party. Randomly choose $k_1, \ldots, k_n \sim \{0,1\}^\lambda$, which are called "keys". We define an oracle $y' \colon \{0,1\}^\lambda \to \{0,1\}^m$ such that $y'(k_i) := s_i$ and $y'(k) := 0^m$ if $k \notin \{k_i \mid i \in [n]\}$. Let $y \in \{0,1\}^{m \cdot 2^\lambda}$ denote the truth table of y', i.e., the concatenation of the values of $y(k)$ for all the inputs $k \in \{0,1\}^\lambda$ in lexicographical order. We claim that for some time bound t,

$$\mathrm{K}^t(x \mid y) \lesssim w(\mathcal{A}_\varphi) \cdot \lambda + |\varphi| \tag{7}$$

and that

$$\mathrm{K}^t(x \mid y) \gtrsim w(\mathcal{A}_\varphi) \cdot \lambda. \tag{8}$$

It is easy to see Equation (7): Let $T \in \mathcal{A}_\varphi$ be an authorized set of parties. Consider an oracle program that, given oracle access to y, takes the set $\{k_i \in \{0,1\}^\lambda \mid i \in T\}$ as input, queries k_i to the oracle y and obtains the i-th share $s_i = y'(k_i)$ for all $i \in T$, and then reconstructs the secret $x = \mathrm{Rec}_\varphi(\{s_i \mid i \in T\})$. The size of this program is at most approximately $|T| \cdot \lambda + |\varphi|$. To see (8), assume, toward a contradiction, that there exists an oracle program M of size $\ll w(\mathcal{A}_\varphi) \cdot \lambda$ such that M prints x in time t given oracle access to y. Since $t \ll |y|$ and $k_1, \ldots, k_n$ are chosen randomly, it can be shown that the program M of size $|M|$ can read at most approximately $|M|/\lambda$ shares from y. The intuition behind this is that the i-th share s_i cannot be accessed without knowing the i-th key k_i, which must be hard-wired in the program M. Let T be the set of indices $i \in [n]$ such that the i-th share s_i is read by M. Then, we have $|T| \lesssim |M|/\lambda \ll w(\mathcal{A}_\varphi)$, which implies that $T \notin \mathcal{A}_\varphi$. However, by the perfect privacy of the secret sharing scheme, it is not possible to reconstruct x from $\{s_i \mid i \in T\}$, which is a contradiction.

There are two issues in the proof sketch above. First, it can be the case that $w(\mathcal{A}_\varphi) \cdot \lambda \ll |\varphi|$, in which case Equation (7) is too loose. Second, we need to show that an additive error term $\mathrm{cd}^{t,\tau(|x|,|y|,t)}(y)$ is small compared to $w(\mathcal{A}_\varphi) \cdot \lambda$. Fortunately, there is a simple trick that simultaneously fixes these two issues: Share D independent secrets $x^1, \ldots, x^D \sim \{0,1\}^\ell$ among n parties, construct oracles $y^1, \ldots, y^D$ for each secret, and reduce φ to $(x, y, t, D\lambda)$ for $x := (x^1, \ldots, x^D)$ and $y := (y^1, \ldots, y^D)$. Then we get

$$w(\mathcal{A}_\varphi) \cdot \lambda \lesssim \frac{1}{D} \cdot \mathrm{K}^t(x \mid y) \lesssim w(\mathcal{A}_\varphi) \cdot \lambda + \frac{|\varphi|}{D},$$

whose last term is negligible for a large enough D. This fixes the first issue. Moreover, it can be shown that the "amortized" computational depth $\frac{1}{D} \cdot \mathrm{cd}^t(y)$ goes to 0 as D increases, which fixes the second issue.

Finally, we explain the relationship between our reduction and the previous reductions in [40, 4, 56]. Previously, the set cover problem was reduced to MINcKT and a conditional variant of MCSP by implicitly using a secret sharing scheme that can share a random string and satisfies the imperfect privacy. Specifically, let $S_1, \cdots, S_n \subseteq [m]$ be an instance

[14] It is worth mentioning that MMSA for depth-2 monotone formulas corresponds to the set cover problem.

of the set cover problem. Let $\mathcal{A}$ be the collection of the sets $T \subseteq [n]$ that form a cover; i.e., $\bigcup_{i \in T} S_i = [m]$. Then, in [40, 4, 56], the following secret sharing scheme for $\mathcal{A}$ was implicitly used: For a secret $x = (x^1, \dots, x^m) \sim (\{0,1\}^\ell)^m$, the i-th share s_i is defined to be $\{x^j \mid j \in S_i\}$ for each $i \in [n]$. This secret sharing scheme satisfies correctness, i.e., the property that x can be reconstructed from $s_T := \{s_i \mid i \in T\}$ for any authorized set $T \in \mathcal{A}$; it also satisfies imperfect privacy in the sense that, for any unauthorized set $T \notin \mathcal{A}$, there exists $j \in [m] \setminus \bigcup_{i \in T} S_i$ such that no information about x^j is leaked from s_T. Our generalized reduction abstracts the essential ideas developed in [40, 4, 56] and would be useful for making further progress on the final goal of proving NP-hardness of GapMINKT and excluding Heuristica.

Organization

In Section 4, we present a formal proof of SoI in Heuristica. In Section 5, we prove the lower bound for rKt. In Section 6, we examine the complexity of conditional Kolmogorov complexity. In Section 7, we present an application of SoI to average-case complexity. In Section 8, we present several statements that follow from SoI.

3 Preliminaries

Notation

$[n]$ denotes $\{1, \dots, n\}$. For a function $p\colon \mathbb{N} \to \mathbb{N}$ and $i \in \mathbb{N}$, the function $p^{(i)}\colon \mathbb{N} \to \mathbb{N}$ is recursively defined as follows: $p^{(0)}(n) := n$ and $p^{(i+1)}(n) := p^{(i)}(p(n))$ for every $n \in \mathbb{N}$.

Kolmogorov Complexity

For a string $x \in \{0,1\}^*$ and an index $i \in \mathbb{N}$, let x_i denote the i-th bit of x if $i \leq |x|$ and $\perp$ if $i > |x|$. We fix an efficient universal Turing machine U. Time-bounded Kolmogorov complexity is formally defined as follows.

▶ **Definition 3.1** (Time-bounded Kolmogorov complexity). *For strings $x, y \in \{0,1\}^*$, a time bound $t \in \mathbb{N} \cup \{\infty\}$, and an oracle A, the A-oracle t-time-bounded Kolmogorov complexity of x given y is defined as*

$$\mathrm{K}^{t,A}(x \mid y) := \min\left\{|d| \;\middle|\; U^{A,y,d} \text{ outputs } x_i \text{ on input } i \text{ in time } t \text{ for every } i \in [1, \dots, |x|+1]\right\}.$$

Here, $U^{A,y,d}$ indicates that the universal Turing machine is given oracle access to A and each bit of y and $d \in \{0,1\}^*$. We omit the superscript A if $A = \emptyset$, the superscript t if $t = \infty$, and "$\mid y$" if y is the empty string.

A simple counting argument implies the following basic fact of Kolmogorov-randomness.

▶ **Fact 3.2.** *For any integer $s \geq 1$ and any string $y \in \{0,1\}^*$, the number of strings $x \in \{0,1\}^*$ such that $\mathrm{K}(x \mid y) < s$ is less than 2^s.*

Proof. The number of programs of length less than s is at most $\sum_{i=0}^{s-1} 2^i < 2^s$. ◀

▶ **Definition 3.3** ([50, 33]). *For a polynomial $\tau\colon \mathbb{N} \to \mathbb{N}$, let*

$$\Pi_{\mathrm{YES}} := \left\{(x, 1^t, 1^s) \;\middle|\; \mathrm{K}^t(x) \leq s\right\},$$
$$\Pi_{\mathrm{NO}} := \left\{(x, 1^t, 1^s) \;\middle|\; \mathrm{K}^{\tau(|x|,t)}(x) > s + \log \tau(|x|, t)\right\}.$$

We define $\mathrm{Gap}_\tau\mathrm{MINKT}$ *to be the promise problem* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$. *We say that* $\mathrm{GapMINKT} \in \mathsf{P}$ *if there exists some polynomial* τ *such that* $\mathrm{Gap}_\tau\mathrm{MINKT} \in \mathsf{P}$.

The problem GapMINKT can be equivalently formulated as follows.

▶ **Fact 3.4** ([35]). *The following are equivalent.*

1. $\mathrm{GapMINKT} \in \mathsf{P}$.
2. *There exist a polynomial-time algorithm* $\widetilde{\mathrm{K}}$ *and a polynomial* p *such that*

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) \leq \mathrm{K}^t(x)$$

for every string $x \in \{0,1\}^*$ *and every integer* $t \geq |x|$.

We recall the notion of time-bounded computational depth.

▶ **Definition 3.5** (Time-Bounded Computational Depth [9, 35]). *For time bounds* $s \in \mathbb{N}$ *and* $t \in \mathbb{N} \cup \{\infty\}$ *and a string* $x \in \{0,1\}^*$, *the* (s,t)-time-bounded computational depth *of* x *is defined as*

$$\mathrm{cd}^{s,t}(x) := \mathrm{K}^s(x) - \mathrm{K}^t(x).$$

We omit the superscript t *if* $t = \infty$.

4 Symmetry of Information from Average-Case Easiness of NP

In this section, we present a formal proof of Theorem 1.2. In fact, under a plausible assumption that E requires exponential-size circuits to compute, we prove that SoI follows from an approximation algorithm for time-bounded Kolmogorov complexity.

▶ **Theorem 4.1.** *If* $\mathrm{GapMINKT} \in \mathsf{P}$ *and* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$, *then SoI holds.*

Theorem 4.1 immediately implies Theorem 1.2.

Proof of Theorem 1.2. It is known that the two hypotheses of Theorem 4.1 are implied by the assumption that NP is easy on average: Buhrman et al. [18] showed that $\mathsf{E} \not\subseteq \bigcap_{\epsilon>0} \mathsf{i.o.SIZE}(2^{\epsilon n})$ if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$; We also have $\mathrm{GapMINKT} \in \mathsf{P}$ if $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ by Lemma 2.1. ◀

We now present the formal proof of Theorem 4.1.

Proof of Theorem 4.1. Let $\widetilde{\mathrm{K}}$ be the polynomial-time algorithm of Fact 3.4 which satisfies the property that there exists a polynomial p such that for every $x \in \{0,1\}^*$ and every $t \geq |x|$,

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x; 1^t) \leq \mathrm{K}^t(x) \tag{9}$$

Fix strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ and an integer $t \geq n + m$. The proof of SoI is given by analyzing the following three values for $z \sim \{0,1\}^{nk}$, $w \sim \{0,1\}^{nk+k}$, $z' \sim \{0,1\}^{m\ell}$, and $w' \sim \{0,1\}^{m\ell+\ell}$:

$$\begin{aligned}
&\widetilde{\mathrm{K}}(\ \mathrm{DP}_k(x;z),\ \mathrm{DP}_\ell(y;z')\ ;1^{t'}),\\
&\widetilde{\mathrm{K}}(\ w,\ \mathrm{DP}_\ell(y;z')\ ;1^{t'}),\\
&\widetilde{\mathrm{K}}(\ w,\ w'\ ;1^{t'}),
\end{aligned}$$

where $t' = t^{O(1)}$, $k \approx \mathrm{K}^t(x,y) - \ell$, and $\ell \approx \mathrm{K}^{\mathsf{poly}(t)}(y)$ are parameters chosen later.

First, observe that Fact 3.2 implies that $\mathrm{K}(w, w') \geq |w| + |w'| - 2$ with probability at least $\frac{3}{4}$. Let $\theta := |w| + |w'| - 2 - \log p(t')$; using Equation (9), we obtain

$$\Pr_{w,w'}\left[\widetilde{\mathrm{K}}(w, w'; 1^{t'}) \geq \theta\right] \geq \frac{3}{4}. \tag{10}$$

Next, we set the parameter ℓ to $\mathrm{K}^{p'(t)}(y) - \log p'(t) - 1$, where p' is some large polynomial. Consider a randomized circuit D that takes w' as input as well as random bits w and outputs 1 if and only if $\widetilde{\mathrm{K}}(w, w'; 1^{t'}) \geq \theta$. By the contrapositive of Lemma 2.2, $\mathrm{DP}_\ell(y; \text{-})$ is a pseudorandom generator secure against D; i.e., D cannot distinguish $\mathrm{DP}_\ell(y; z')$ and w' in the sense that

$$\left|\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y; z'); 1^{t'}) \geq \theta\right] - \Pr_{w,w'}\left[\widetilde{\mathrm{K}}(w, w'; 1^{t'}) \geq \theta\right]\right| < \frac{1}{4},$$

which, together with Equation (10), implies that

$$\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y; z'); 1^{t'}) \geq \theta\right] \geq \frac{1}{2}.$$

Finally, we compare $\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y; z'); 1^{t'})$ with $\widetilde{\mathrm{K}}(\mathrm{DP}_k(x; z), \mathrm{DP}_\ell(y; z'); 1^{t'})$. On one hand, since $|w| = |z| + k$ and $|w'| = |z'| + \ell$, we have

$$\Pr_{w,z'}\left[\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y; z'); 1^{t'}) \geq |z| + |z'| + k + \ell - 2 - \log p(t')\right] \geq \frac{1}{2}. \tag{11}$$

On the other hand, observe that for some $t' := \mathsf{poly}(t)$,

$$\widetilde{\mathrm{K}}(\mathrm{DP}_k(x; z), \mathrm{DP}_\ell(y; z'); 1^{t'}) \leq \mathrm{K}^{t'}(\mathrm{DP}_k(x; z), \mathrm{DP}_\ell(y; z')) \leq \mathrm{K}^t(x, y) + |z| + |z'| + O(\log n)$$

holds because the strings $\mathrm{DP}_k(x; z)$ and $\mathrm{DP}_\ell(y; z')$ can be computed from k, ℓ, z, z', and a program of size $\mathrm{K}^t(x, y)$ that outputs (x, y) in time t. We now set $k := \mathrm{K}^t(x, y) - \ell + O(\log t)$ so that

$$\Pr_{z,z'}\left[\widetilde{\mathrm{K}}(\mathrm{DP}_k(x; z), \mathrm{DP}_\ell(y; z'); 1^{t'}) < |z| + |z'| + k + \ell - 2 - \log p(t')\right] = 1. \tag{12}$$

Let D_y be a randomized circuit that takes an input w and random bits z' and outputs 1 if and only if $\widetilde{\mathrm{K}}(w, \mathrm{DP}_\ell(y; z'); 1^{t'}) < |z| + |z'| + k + \ell - 2 - \log p(t')$. It follows from Equations (11) and (12) that

$$\Pr_{z,z'}\left[D_y(\mathrm{DP}_k(x; z); z') = 1\right] - \Pr_{w,z'}\left[D_y(w; z') = 1\right] \geq 1 - \frac{1}{2} = \frac{1}{2}.$$

Using Lemma 2.2, we obtain

$$\mathrm{K}^{\mathsf{poly}(t)}(x \mid D_y) \leq k + O(\log t) = \mathrm{K}^t(x, y) - \mathrm{K}^{p'(t)}(y) + O(\log t).$$

It follows that for some large polynomial q,

$$\begin{aligned}\mathrm{K}^{q(t)}(x \mid y) \leq \mathrm{K}^t(x \mid D_y) + \mathrm{K}^t(D_y \mid y) + O(1) &\leq \mathrm{K}^t(x, y) - \mathrm{K}^{p'(t)}(y) + O(\log t) \\ &\leq \mathrm{K}^t(x, y) - \mathrm{K}^{q(t)}(y) + \log q(t)\end{aligned}$$

as desired. ◀

5 A Lower Bound for Randomized Kt Complexity

Here, we implement Kolmogorov's insightful approach to the P versus NP problem for randomized Kt complexity. We recall the formal definition of rKt, which was introduced by Oliveira [62].

▶ **Definition 5.1** ([62]). *For every $x \in \{0,1\}^*$ and every $\lambda \in [0,1]$, the randomized time-bounded Kolmogorov complexity* $\mathrm{rKt}_\lambda(x)$ *of x is defined as*

$$\mathrm{rKt}_\lambda(x \mid y) := \min\left\{ |d| + \lceil \log t \rceil \;\middle|\; \Pr_{r \sim \{0,1\}^t}[\, U(d,y,r) \text{ outputs } x \text{ in time } t \,] \geq \lambda \right\}.$$

We omit the subscript "λ" if $\lambda = \frac{2}{3}$ and "$\mid y$" if y is the empty string. The language MrKtP *is defined as*

$$\mathrm{MrKtP} := \{(x, 1^s) \mid \mathrm{rKt}(x) \leq s\}.$$

The definition of rKt_λ is robust with respect to a choice of λ.

▶ **Lemma 5.2** (Lu and Oliveira [59]). *For all $x \in \{0,1\}^*$ and $\lambda \in (0,1]$,*

$$\mathrm{rKt}(x) \leq \mathrm{rKt}_\lambda(x) + O(\log(1/\lambda)).$$

For simplicity, we first prove the following weaker lower bound than Theorem 1.4.

▶ **Theorem 5.3.** *There exists a constant $\epsilon > 0$ such that* $\mathrm{MrKtP} \notin \mathsf{BPTIME}(2^{\epsilon n})$.

In what follows, let $\epsilon > 0$ be a sufficiently small positive constant. The O notation below does not depend on ϵ. We say that a *pseudo-deterministic algorithm M computes y on input x* [27] if

$$\Pr_M[M(x) = y] \geq \frac{2}{3},$$

where the probability is taken over an internal coin flip sequence of the randomized algorithm M. We use the following pseudorandom generator construction.

▶ **Lemma 5.4** (see the proof of [32, Theorem 4.7]). *For all sufficiently large $n, m \in \mathbb{N}$ such that $m \leq 2n$, there exists a triple $(G, A, R^{(\cdot)})$ such that*

$$G\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m,$$
$$A\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m,$$
$$R^{(\cdot)}\colon \{0,1\}^m \times \{0,1\}^d \times \{0,1\}^r \to (\{0,1\}^n)^L,$$

and for every $x \in \{0,1\}^n$ and any $D\colon \{0,1\}^m \to \{0,1\}$ such that

$$\Pr_{\substack{z \sim \{0,1\}^d \\ w' \sim \{0,1\}^{O(m)}}}[D(G(x;z);w') = 1] - \Pr_{\substack{w \sim \{0,1\}^m \\ w' \sim \{0,1\}^{O(m)}}}[D(w;w') = 1] \geq \frac{1}{m},$$

it holds that

$$\Pr_{\substack{w \sim \{0,1\}^r \\ z \sim \{0,1\}^d}}\left[x \in R^D(A(x,z),z,w)\right] \geq \frac{1}{2m^2}.$$

Here, we have $d = O(\log^3 n)$, $r = O(m)$, and $L = \mathsf{poly}(m)$. Moreover, G and A can be computed in time $\mathsf{poly}(n)$ *and R^D can be computed in time* $\mathsf{poly}(n)$ *with oracle access to D.*

Proof Sketch. The construction is given by the improvement of Trevisan's extractor [70] given by Raz et al. [64]. Specifically, we encode x using a list-decodable error-correcting code and regard the encoding of x as the truth table of a hard function f. Then, the pseudorandom generator construction G is defined as the Nisan–Wigderson generator [61] instantiated with the function f and the weak design of [64]. ([32, Theorem 4.7] is stated as a black-box *hitting set generator* construction; however, it is easy to observe that the construction actually provides a *pseudorandom generator* construction, which is a stronger object.) ◀

▶ **Corollary 5.5.** *Under the same hypothesis of Lemma 5.4, it holds that*

$$\mathrm{rKt}^D(x) \leq m + O(\log^3 n).$$

Proof. By an averaging argument, there exists a string $z \in \{0,1\}^d$ such that

$$\Pr_w \left[x \in R^D(A(x,z),z,w)\right] \geq \frac{1}{2m^2}.$$

Let $\alpha := A(x,z) \in \{0,1\}^m$. The string x can be described by the following pseudo-deterministic algorithm M: M takes the advice string $\alpha \in \{0,1\}^m$ and $z \in \{0,1\}^d$ as input, picks $w \sim \{0,1\}^r$ randomly, and outputs a random element of $R^D(\alpha,z,w)$. Since the list size is at most $L = \mathsf{poly}(m)$, the probability that the output of M is equal to x is at least $\lambda := \frac{1}{2m^2} \cdot \frac{1}{L}$. Therefore, we obtain $\mathrm{rKt}_\lambda(x) \leq m + d + O(\log n)$. The result follows from Lemma 5.2. ◀

▶ **Lemma 5.6.** *If* $\mathrm{MrKtP} \in \mathsf{BPTIME}(2^{\epsilon n})$*, then for all sufficiently large* $n \in \mathbb{N}$ *and any strings* $x \in \{0,1\}^n$ *and* $y \in \{0,1\}^n$*, it holds that*

$$\mathrm{rKt}(x \mid y) + \mathrm{rKt}(y) \leq \mathrm{rKt}(x,y) + O(\epsilon n).$$

Proof. We may assume without loss of generality that $\mathrm{rKt}(y) \gg \log^3 n$. Let m and m' be parameters chosen later. Let $G\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ and $G'\colon \{0,1\}^n \times \{0,1\}^{d'} \to \{0,1\}^{m'}$ be the pseudorandom generator construction of Lemma 5.4 with parameters m and m', respectively.

Fix $z \in \{0,1\}^d$ and $z' \in \{0,1\}^{d'}$. Since the strings $(G(x;z), G'(y;z'))$ can be described by z, z', and a program that computes (x,y), we obtain

$$\mathrm{rKt}(G(x;z), G'(y;z')) \leq \mathrm{rKt}(x,y) + |z| + |z'| + O(\log n) = \mathrm{rKt}(x,y) + O(\log^3 n). \tag{13}$$

Pick $w \sim \{0,1\}^m$ and $z' \sim \{0,1\}^{d'}$ randomly. We claim that $\mathrm{rKt}(w, G'(y;z')) \geq |w| + |w'| - O(\log n)$ with high probability. Toward a contradiction, assume that

$$\Pr_{w,z'} \left[\mathrm{rKt}(w, G'(y;z')) < |w| + |w'| - O(\log n)\right] \geq \frac{1}{2}.$$

By Fact 3.2, we have

$$\Pr_{w,w'} \left[\mathrm{rKt}(w,w') \geq |w| + |w'| - O(\log n)\right] \geq 1 - o(1).$$

We define an oracle D so that $D(w';w) := 1$ if and only if $\mathrm{rKt}(w,w') < |w| + |w'| - O(\log n)$. Then, the two inequalities above show that D distinguishes the output distribution of $G'(y;\text{-})$ from the uniform distribution. By Corollary 5.5, we obtain

$$\mathrm{rKt}^D(y) \leq m' + O(\log^3 n).$$

Using the assumption that MrKtP $\in \mathsf{BPTIME}(2^{\epsilon n})$, the oracle D can be computed by a randomized algorithm running in time $2^{O(\epsilon m')}$. It follows that

$$\mathrm{rKt}(y) \leq m' + O(\epsilon m') = (1 + O(\epsilon)) \cdot m'.$$

We choose m' so that this inequality does not hold; that is,

$$m' := (1 - O(\epsilon)) \cdot \mathrm{rKt}(y).$$

Then, we obtain a contradiction and conclude that

$$\Pr_{w,z'}[\mathrm{rKt}(w, G'(y; z')) \geq m + m' - O(\log n)] \geq \frac{1}{2}. \tag{14}$$

Define D_y to be an oracle such that $D_y(w; z') := 1$ if and only if $\mathrm{rKt}(w, G'(y; z')) \leq \mathrm{rKt}(x, y) + O(\log^3 n)$. By Equations (14) and (13), D_y distinguishes the output distribution of $G(x; \text{-})$ from the uniform distribution if

$$m + m' - O(\log n) \geq \mathrm{rKt}(x, y) + O(\log^3 n).$$

We define $m := \mathrm{rKt}(x, y) - m' + O(\log^3 n)$ so that this inequality holds. By Corollary 5.5, we obtain

$$\mathrm{rKt}^{D_y}(x) \leq m + O(\log^3 n).$$

Since the oracle D_y can be computed in time $2^{O(\epsilon m)}$ given y as hard-wired input, we conclude that

$$\begin{aligned}
\mathrm{rKt}(x \mid y) &\leq m + O(\epsilon m). \\
&\leq \mathrm{rKt}(x, y) - m' + O(\epsilon n). \\
&\leq \mathrm{rKt}(x, y) - (1 - O(\epsilon)) \cdot \mathrm{rKt}(y) + O(\epsilon n). \\
&\leq \mathrm{rKt}(x, y) - \mathrm{rKt}(y) + O(\epsilon n).
\end{aligned}$$

◀

Proof of Theorem 5.3. The outline of the proof is as follows: First, we find a string $y \in \{0,1\}^n$ such that $\mathrm{rKt}(y) \geq n$ by an exhaustive search over all strings $y \in \{0,1\}^n$. Next, we find a string $x \in \{0,1\}^n$ such that $\mathrm{rKt}(x, y) \geq 2n - O(\epsilon n)$ by an exhaustive search over all strings $x \in \{0,1\}^n$. The existence of such a string x is guaranteed by Lemma 5.6. Finally, we observe that the pair (x, y) can be pseudo-deterministically computed in time $2^{n+O(\epsilon n)}$, which contradicts the lower bound on $\mathrm{rKt}(x, y)$. Details follow.

Using the assumption that MrKtP $\in \mathsf{BPTIME}(2^{\epsilon n})$, let M be the randomized algorithm that computes $\mathrm{rKt}(y)$ on input y of length n in time $2^{O(\epsilon n)}$ with probability at least $1 - 2^{-2n}$. Fix $n \in \mathbb{N}$. Let y_n be the lexicographically first string $y \in \{0,1\}^n$ such that $\mathrm{rKt}(y) \geq n$. Note that the existence of such a string y_n is guaranteed by Fact 3.2.

We claim that there exists a pseudo-deterministic algorithm M_1 that computes y_n on input $n \in \mathbb{N}$ in time $2^{n+O(\epsilon n)}$. For all strings $y \in \{0,1\}^n$ in lexicographical order, the algorithm M_1 tests whether $\mathrm{rKt}(y) \geq n$ using M, and outputs the first string y that passes this test. Since the error probability of M is at most 2^{-2n}, by a union bound, the algorithm M_1 computes y_n pseudo-deterministically with probability $1 - 2^{-n}$. The running time of M_1 is at most $2^{n+O(\epsilon n)}$.

Next, let x_n be the lexicographically first string $x \in \{0,1\}^n$ such that

$$\mathrm{rKt}(x, y_n) \geq 2n - c \cdot \epsilon n, \tag{15}$$

where c is a large constant to be chosen independently of ϵ. We prove that x_n is well defined, i.e., there exists a string $x \in \{0,1\}^n$ such that $\mathrm{rKt}(x, y_n) \geq 2n - c \cdot \epsilon n$. By Fact 3.2, there exists a string $x \in \{0,1\}^n$ such that $\mathrm{rKt}(x \mid y) \geq n$. By Lemma 5.6,

$$\mathrm{rKt}(x, y) + O(\epsilon n) \geq \mathrm{rKt}(x \mid y) + \mathrm{rKt}(y) \geq 2n.$$

Choosing a large constant c, we obtain $\mathrm{rKt}(x, y) \geq 2n - c \cdot \epsilon n$, as desired.

We claim that there exists a pseudo-deterministic algorithm M_2 that computes x_n given y_n as input in time $2^{n+O(\epsilon n)}$. For all strings $x \in \{0,1\}^n$ in lexicographical order, the algorithm M_2 tests whether $\mathrm{rKt}(x, y_n) \geq 2n - c \cdot \epsilon n$ using M, and outputs the first string x that passes this test. Using a union bound, the error probability of M_2 is bounded by 2^{-n}. The running time of M_2 is at most $2^{n+O(\epsilon n)}$.

Finally, by combining the algorithms M_1 and M_2, we obtain an algorithm that computes (x_n, y_n) pseudo-deterministically in time $2^{n+O(\epsilon n)}$. By the definition of rKt, this implies that

$$\mathrm{rKt}(x_n, y_n) \leq n + O(\epsilon n).$$

We also have $\mathrm{rKt}(x_n, y_n) \geq 2n - O(\epsilon n)$ by Equation (15). Therefore, we obtain $2n - O(\epsilon n) \leq n + O(\epsilon n)$, which is a contradiction for a sufficiently small constant $\epsilon > 0$. ◀

We observe that MrKtP is in the exponential-time variant of PP.

▶ **Proposition 5.7.** MrKtP $\in$ PEXP.

Proof. Since PP is closed under truth-table reductions [26], it suffices to show that MrKtP is reducible to PP via an exponential-time truth-table reduction. Let $(x, 1^s)$ be an input such that $x \in \{0,1\}^n$ and $s \in \mathbb{N}$. For each string $d \in \{0,1\}^*$ of length at most s, we ask the PP oracle whether $U(d, r)$ outputs x in time t for at least a $\frac{2}{3}$-fraction of $r \in \{0,1\}^t$, where $t := 2^{s-|d|}$. We accept the input $(x, 1^s)$ if and only if a positive answer is returned from the oracle. ◀

We expect that Theorem 5.3 can be extended to a lower bound against PP-type algorithms. However, the original motivation of Oliveira [62] is to study a gap version of MrKtP, for which the upper bound of pr-BPE can be proved. Specifically, let GapMrKtP denote the promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ such that $\Pi_{\text{YES}} = \{(x, s) \mid \mathrm{rKt}(x) \leq s\}$ and $\Pi_{\text{NO}} = \{(x, s) \mid \mathrm{rKt}(x) > s + c \log |x|\}$, where c is a sufficiently large constant. One can observe GapMrKtP $\in$ pr-BPE [62]. Theorem 5.3 can be extended to this promise problem:

▶ **Theorem 5.8** (Theorem 1.4, restated)**.** GapMrKtP $\notin$ i.o.BPTIME$(2^{\epsilon n})$ *for some constant* $\epsilon > 0$.

Here, abusing a notation, for a class $\mathfrak{C}$ of languages, we say that a promise problem $\Pi \in \mathfrak{C}$ if there exists a language $L \in \mathfrak{C}$ such that $\Pi_{\text{YES}} \subseteq L \subseteq \{0,1\}^* \setminus \Pi_{\text{NO}}$ for $(\Pi_{\text{YES}}, \Pi_{\text{NO}}) = \Pi$. The lower bound of Theorem 5.8 does work against a bounded probabilistic algorithm that satisfies the promise of BPP-type algorithms over *all* the inputs. In contrast, the upper bound pr-BPE = pr-BPTIME$(2^{O(n)})$ holds only for an algorithm that may not satisfy the promise of BPP-type algorithms for *some* inputs. This leaves a qualitative gap between the lower bound and the upper bound on GapMrKtP; closing this gap is an interesting open question. We note, however, that there is no gap under the plausible assumption that E $\not\subseteq$ i.o.SIZE$(2^{\epsilon n})$ for some constant $\epsilon > 0$, in which case pr-BPP = P = BPP [46].

Proof of Theorem 5.8. The proof is essentially the same with Theorem 5.3. We only explain how to modify the proof of Theorem 5.3.

Toward a contradiction, we assume the existence of a randomized algorithm M that computes some language L *infinitely often*, where L is a language consistent with GapMrKtP. For all large $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$ and $s \in [n]$, we assume that the binary encoding of (x,s) is exactly equal to $m(n)$, where $m(n) = \Theta(n)$ is some function. This ensures that for infinitely many $n \in \mathbb{N}$, the algorithm M computes L correctly on input (x,s) for every $x \in \{0,1\}^n$ and every $s \in [n]$.

First, we explain how to extend the lower bound of Theorem 5.3 to the infinitely-often version. Let n be the length of $x_n \in \{0,1\}^n$ and $y_n \in \{0,1\}^n$ constructed in the proof of Theorem 5.3. By inspection, in the proof of Theorem 5.3, we use the algorithm M on inputs of length at most n', where $n' = O(n)$. Any input x of length less than n' can be padded to an input of length n' by mapping x to $x' := x1^{n'-|x|}$. This ensures that the length of any input to M that we use in the proof is exactly equal to n'. Moreover, the correctness of the proof remains unchanged because $\mathrm{rKt}(x) = \mathrm{rKt}(x') \pm O(\log n)$.

Second, we explain how to deal with the promise problem. In the proof of Theorem 5.3, we exhaustively search the lexicographically first string $y_n \in \{0,1\}^n$ such that $\mathrm{rKt}(y_n) \geq n$. Instead, we search the lexicographically first string $y_n \in \{0,1\}^n$ such that $(y_n, n - 2c \cdot \log n) \notin L$. Since $L \subseteq \{0,1\}^* \setminus \Pi_{\mathrm{No}}$, this ensures that $\mathrm{rKt}(y_n) \geq n - c\log n$. Similarly, we search the lexicographically first string $x_n \in \{0,1\}^n$ such that $((x_n, y_n), 2n - O(\epsilon n)) \notin L$. This ensures that $\mathrm{rKt}(x_n, y_n) \geq 2n - O(\epsilon n)$. Since the language L can be decided by the randomized algorithm M, we can compute the pair (x_n, y_n) pseudo-deterministically in time $2^{n+O(\epsilon n)}$. This is a contradiction. ◀

6 The Complexity of Conditional Kolmogorov Complexity

In this section, we examine the complexity of the problem GapMINcKT of approximating time-bounded conditional Kolmogorov complexity.

▶ **Definition 6.1.** *For a polynomial $\tau\colon \mathbb{N}^3 \to \mathbb{N}$, we define the promise problem* $\mathrm{Gap}_\tau\mathrm{MINcKT}$ *to be* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *such that*

$$\Pi_{\mathrm{YES}} := \left\{ (x, y, 1^t, 1^s) \;\middle|\; \mathrm{K}^t(x \mid y) \leq s - \mathrm{cd}^{t,\tau(|x|,|y|,t)}(y) \right\},$$
$$\Pi_{\mathrm{NO}} := \left\{ (x, y, 1^t, 1^s) \;\middle|\; \mathrm{K}^{\tau(|x|,|y|,t)}(x \mid y) > s + \log \tau(|x|,|y|,t) \right\}.$$

6.1 Approximating Conditional Kolmogorov Complexity in Heuristica

We observe that SoI enables reducing GapMINcKT to GapMINKT.

▶ **Proposition 6.2.** *Assume that SoI holds. If* $\mathrm{GapMINKT} \in \mathsf{P}$*, then there exists a polynomial $\tau\colon \mathbb{N}^3 \to \mathbb{N}$ such that* $\mathrm{Gap}_\tau\mathrm{MINcKT} \in \mathsf{P}$.

Proof. Let $\widetilde{\mathrm{K}}$ be the polynomial-time algorithm of Fact 3.4 such that for some polynomial p,

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) \leq \mathrm{K}^t(x)$$

for every string $x \in \{0,1\}^*$ and every integer $t \geq |x|$. Applying this inequality to xy and y, we obtain

$$\mathrm{K}^{p(t)}(xy) - \log p(t) \leq \widetilde{\mathrm{K}}(xy, 1^t) \leq \mathrm{K}^t(xy),$$
$$\mathrm{K}^{p(t)}(y) - \log p(t) \leq \widetilde{\mathrm{K}}(y, 1^t) \leq \mathrm{K}^t(y)$$

for every $t \geq |x| + |y|$. Using these inequalities, we next analyze $\widetilde{\mathrm{K}}(xy, 1^{p(t)}) - \widetilde{\mathrm{K}}(y, 1^{p^{(3)}(t)})$. On one hand, we have

$$\begin{aligned}\widetilde{\mathrm{K}}(xy, 1^{p(t)}) - \widetilde{\mathrm{K}}(y, 1^{p^{(3)}(t)}) &\leq \mathrm{K}^{p(t)}(xy) - \mathrm{K}^{p^{(4)}(t)}(y) + \log p^{(4)}(t) \\ &\leq \mathrm{K}^{t}(x \mid y) + \mathrm{K}^{t}(y) + O(1) - \mathrm{K}^{p^{(4)}(t)}(y) + \log p^{(4)}(t) \\ &= \mathrm{K}^{t}(x \mid y) + \mathrm{cd}^{t,p^{(4)}(t)}(y) + \log p^{(4)}(t) + O(1).\end{aligned}$$

On the other hand, we have

$$\begin{aligned}\widetilde{\mathrm{K}}(xy, 1^{p(t)}) - \widetilde{\mathrm{K}}(y, 1^{p^{(3)}(t)}) &\geq \mathrm{K}^{p^{(2)}(t)}(xy) - \log p^{(2)}(t) - \mathrm{K}^{p^{(3)}(t)}(y) \\ &\geq \mathrm{K}^{p^{(3)}(t)}(x \mid y) - 2\log p^{(3)}(t),\end{aligned}$$

where the last inequality follows from SoI.

We now set $\tau(n, m, t) := p^{(4)}(t+n+m)$ so that $p^{(4)}(t) \leq \tau(|x|, |y|, t)$ for any $x, y \in \{0,1\}^*$ and $t \in \mathbb{N}$. Define an algorithm B so that $B(x, y, 1^t) := \widetilde{\mathrm{K}}(xy, 1^{p(t')}) - \widetilde{\mathrm{K}}(y, 1^{p^{(3)}(t')}) - \frac{1}{2} \cdot \log \tau(n, m, t)$, where $t' := \max\{|x| + |y|, t\}$. Then, we have

$$\mathrm{K}^{\tau(n,m,t)}(x \mid y) - \log \tau(n, m, t) \leq B(x, y, 1^t) \leq \mathrm{K}^{t}(x \mid y) + \mathrm{cd}^{t,\tau(n,m,t)}(y)$$

for $n := |x|$ and $m := |y|$.

In order to show that $\mathrm{Gap}_\tau\mathrm{MINcKT} \in \mathsf{P}$, consider an algorithm M such that M accepts an input $(x, y, 1^t, 1^s)$ if and only if $B(x, y, 1^t) \leq s$. It is easy to see that M accepts every YES instance and rejects every NO instance of $\mathrm{Gap}_\tau\mathrm{MINcKT}$. ◀

▶ **Corollary 6.3.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$*, then* $\mathrm{Gap}_\tau\mathrm{MINcKT} \in \mathsf{P}$ *for some polynomial* τ*.*

Proof. This immediately follows from Proposition 6.2, Theorem 1.2, , and Lemma 2.1 ◀

As a consequence, Heuristica can be excluded if GapMINcKT is NP-hard under randomized reductions.

Proof of Theorem 1.5. Assume that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. By Corollary 6.3, there exists a polynomial τ such that $\mathrm{Gap}_\tau\mathrm{MINcKT} \in \mathsf{P}$. By the NP-hardness assumption, we obtain $\mathsf{NP} \subseteq \mathsf{BPP} = \mathsf{P}$, where the last equality follows from the theorem of Buhrman et al. [18]. ◀

▶ Remark 6.4 (On the optimality of the additive error). Generalizing Corollary 6.3, it holds that $\mathsf{Dist}\Sigma_2^{\mathsf{p}} \subseteq \mathsf{AvgP}$ implies that $\mathrm{GapMINcKT}^{\mathsf{NP}} \in \mathsf{P}$, where $\mathrm{GapMINcKT}^{\mathsf{NP}}$ is an NP-oracle version of GapMINcKT which asks to approximate $\mathrm{K}^{t,\mathrm{SAT}}(x \mid y)$ within an additive error $\mathrm{cd}^{t,p(t)}(y) + O(\log t)$. Below, we informally argue that a relativizing proof technique is unlikely to improve the error term $\mathrm{cd}^{t,p(t)}(y)$ of $\mathrm{GapMINcKT}^{\mathsf{NP}}$: In [34], NP-hardness of $\mathrm{MINcKT}^{\mathsf{NP}}$ was proved. The NP-hardness reduction in fact shows that for every $L \in \mathsf{NP}$, an instance y for L can be reduced to the problem of approximating $\mathrm{K}^{t,\mathsf{NP}}(x \mid y)$ up to an additive error Δ in time $2^{O(\Delta + \log|y|)}$. In particular, applying this reduction to $\mathrm{GapMINcKT}^{\mathsf{NP}}$, L can be solved in time $2^{O(\mathrm{cd}^{t,p(t)}(y) + \log|y| + \log t)}$ on input $(y, 1^t)$ if $\mathrm{GapMINcKT}^{\mathsf{NP}} \in \mathsf{P}$. This induces a universal heuristic scheme for L, which yields an algorithm that solves L in time $2^{O(n/\log n)}$ (see Section 7). Now, if $\mathrm{K}^{t,\mathrm{SAT}}(x \mid y)$ could be approximated with an additive error $o(\mathrm{cd}^{t,p(t)}(y))$ under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{p}} \subseteq \mathsf{AvgP}$, then we would obtain an improved algorithm that solves every language $L \in \mathsf{NP}$ in time $2^{o(n/\log n)}$, which contradicts the relativization barrier of [36].

6.2 NP-Hardness of Sublinear-Time-Bounded Conditional Kolmogorov Complexity

We now show that $\mathrm{Gap}_{\tau}\mathrm{MINcKT}$ is NP-hard under randomized reductions if $\tau(|x|,|y|,t)$ is sublinear in the length of y. In fact, we prove that it is NP-hard to approximate $\mathrm{K}^t(x \mid y)$ to within a factor of $|x|^{1/(\log\log|x|)^{O(1)}}$. We define a version of GapMINcKT that incorporates a multiplicative factor as follows.

▶ **Definition 6.5.** *For a polynomial $\tau\colon \mathbb{N}^3 \to \mathbb{N}$ and a function $\sigma\colon \mathbb{N} \to \mathbb{N}$, we define the promise problem* $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ *to be* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *such that*

$$\Pi_{\mathrm{YES}} := \left\{(x, y, 1^t, 1^s) \;\middle|\; \mathrm{K}^t(x \mid y) \le s - \mathrm{cd}^{t,\tau(|x|,|y|,t)}(y)\right\},$$
$$\Pi_{\mathrm{NO}} := \left\{(x, y, 1^t, 1^s) \;\middle|\; \mathrm{K}^{\tau(|x|,|y|,t)}(x \mid y) > \sigma(|x|)\cdot s + \log\tau(|x|,|y|,t)\right\}.$$

Note that $\mathrm{Gap}_{\tau,1}\mathrm{MINcKT} = \mathrm{Gap}_{\tau}\mathrm{MINcKT}$ is trivially reducible to $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ for every $\sigma \ge 1$. We show that $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ is NP-hard for $\sigma(|x|) = |x|^{1/(\log\log|x|)^{O(1)}}$.

▶ **Theorem 6.6.** *Let $c > 1$ be an arbitrary constant and $\tau\colon \mathbb{N}^3 \to \mathbb{N}$ be a function such that $\tau(n,m,t) \le n^c \cdot m^{1-1/c} \cdot t^c$ for all large n, m, and $t \in \mathbb{N}$. Then,* $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ *is* NP*-hard under one-query randomized polynomial-time reductions for some function $\sigma\colon \mathbb{N} \to \mathbb{N}$ such that $\sigma(n) = n^{1/(\log\log n)^{O(1)}}$.*

6.2.1 Secret Sharing Scheme

We review the notion of secret sharing scheme below.

▶ **Definition 6.7** (Access Structure)**.** *An* access structure *$\mathcal{A} \subseteq 2^{[n]}$ is a "monotone" collection of subsets of $[n]$; that is, for every $T \supseteq S \in \mathcal{A}$, we have $T \in \mathcal{A}$. The minimum weight of $\mathcal{A}$ is defined to be $w(\mathcal{A}) := \min\{|T| \mid T \in \mathcal{A}\}$.*

We will prove that there is a generic reduction from the problem of estimating the weight of access structures $\mathcal{A}$ to GapMINcKT if there exists an efficient secret sharing scheme for $\mathcal{A}$.

▶ **Definition 6.8** (Secret Sharing [12])**.** *A* secret sharing scheme *for $\mathcal{A}$ is a pair* $(\mathrm{Share}, \mathrm{Rec})$ *of a randomized algorithm* Share *and a deterministic algorithm* Rec *with the following properties for every $\ell \in \mathbb{N}$:*

1. Correctness: *For every $T \in \mathcal{A}$ and for every string $x \in \{0,1\}^\ell$, any output of* $\mathrm{Share}(x)$ *is a sequence $(y_1, \ldots, y_n)$ of n strings that satisfies*

$$\mathrm{Rec}(y_T) = x,$$

   where $y_T := \{(i, y_i) \mid i \in T\}$.
2. Privacy: *For every $T \notin \mathcal{A}$ and for every random variable X on $\{0,1\}^\ell$, the random variables X and $\mathrm{Share}(X)_T$ are statistically independent.*

We observe that the privacy condition can be stated in terms of Kolmogorov complexity.[15]

[15] We mention in passing that Kolmogorov complexity-theoretic versions of privacy conditions are studied in, e.g., [10, 48].

▶ **Lemma 6.9.** *Let* (Share, Rec) *be a secret sharing scheme for an access structure* $\mathcal{A}$ *over* $[n]$*. Then, for every* ℓ *and* $k \in \mathbb{N}$*, it holds that*

$$\Pr\left[\min_{T \notin \mathcal{A}} \mathrm{K}(X \mid \mathrm{Share}(X)_T) \geq \ell - n - k\right] \geq 1 - 2^{-k},$$

where X *is the uniform distribution over* $\{0,1\}^\ell$ *and the probability is taken over* X *as well as the internal randomness of* Share*.*

Proof. Fix an arbitrary subset $T \notin \mathcal{A}$ of $[n]$. Let $y_T \in \mathrm{supp}(\mathrm{Share}(X)_T)$ be an outcome of $\mathrm{Share}(X)_T$. By Fact 3.2, we obtain

$$\Pr\left[\mathrm{K}(X \mid y_T) < \ell - n - k\right] \leq 2^{-n-k}.$$

By the privacy of the secret sharing scheme, the random variables X and $\mathrm{Share}(X)_T$ are statistically independent. By averaging the inequality above over all $y_T \in \mathrm{supp}(\mathrm{Share}(X)_T)$, we obtain

$$\Pr\left[\mathrm{K}(X \mid \mathrm{Share}(X)_T) < \ell - n - k\right] \leq 2^{-n-k}.$$

Finally, we take the union bound over all subsets $T \notin \mathcal{A}$ and conclude that

$$\Pr\left[\exists T \notin \mathcal{A},\ \mathrm{K}(X \mid \mathrm{Share}(X)_T) < \ell - n - k\right] \leq 2^{-k}. \qquad \blacktriangleleft$$

The "efficiency" of a secret sharing scheme is defined as follows.

▶ **Definition 6.10.** *A family* $\mathcal{A} = \{\mathcal{A}_\varphi\}_{\varphi \in \{0,1\}^*}$ *of access structures is said to* admit efficient secret sharing schemes *if there exists a pair* (Share, Rec) *of a randomized polynomial-time algorithm* Share *and a deterministic polynomial-time algorithm* Rec *such that for every* $\varphi \in \{0,1\}^*$*, the pair* $(\mathrm{Share}(\varphi, \text{-}), \mathrm{Rec}(\varphi, \text{-}))$ *is a secret sharing scheme for the access structure* $\mathcal{A}_\varphi$*.*

Benaloh and Leichter [13] showed that access structures represented by monotone formulas admit efficient secret sharing schemes.

▶ **Lemma 6.11** ([13])**.** *Let* $\mathcal{A} := \{\mathcal{A}_\varphi\}_{\varphi \in \{0,1\}^*}$ *be the family of access structures* $\mathcal{A}_\varphi := \{T \subseteq [n] \mid \varphi(\chi_T) = 1\}$*, where* φ *is a monotone formula on* n *variables and* $\chi_T \in \{0,1\}^n$ *denotes the characteristic vector of* $T \subseteq [n]$*. Then,* $\mathcal{A}$ *admits efficient secret sharing schemes.*

6.2.2 Minimum Monotone Satisfying Assignment

In order to prove Theorem 6.6, we reduce the Minimum Monotone Satisfying Assignment (MMSA) problem to GapMINcKT.

▶ **Definition 6.12** (Minimum Monotone Satisfying Assignment; MMSA)**.** *For a monotone formula* φ *on* n *variables, the* weight *of an assignment* $\alpha \in \{0,1\}^n$ *is defined to be* $\sum_{i=1}^n \alpha_i$*. Let* $\mathrm{MMSA}(\varphi)$ *denote the minimum weight of* $\alpha \in \{0,1\}^n$ *such that* $\varphi(\alpha) = 1$*.*

Observe that $\mathrm{MMSA}(\varphi) = w(\mathcal{A}_\varphi)$ for the family $\mathcal{A}$ of access structures of Lemma 6.11. It is known that MMSA is NP-hard to approximate:

▶ **Lemma 6.13** ([22, 21])**.** *For some function* $g(n) = n^{1/(\log\log n)^{O(1)}}$*, it is* NP*-hard to solve the promise problem* $\mathrm{Gap}_g\mathrm{MMSA} = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *defined as follows:*

$$\Pi_{\mathrm{YES}} := \{(\varphi, s) \mid \mathrm{MMSA}(\varphi) \leq s\},$$
$$\Pi_{\mathrm{NO}} := \{(\varphi, s) \mid \mathrm{MMSA}(\varphi) > s \cdot g(|\varphi|)\},$$

where $|\varphi|$ *denotes the length of the binary string that represents* φ*.*

Proof Sketch. Dinur and Safra [22] presented a polynomial-time reduction from any constraint satisfaction problem (CSP) to $\mathrm{Gap}_g\mathrm{MMSA}$ with the following property: Let ψ be an instance of the CSP. Let (φ, s) be the output of the reduction. Let D be the arity of ψ. If ψ is satisfiable, then $\mathrm{MMSA}(\varphi) \le s$. If there is no assignment satisfying a $1/\left(2(2Dg)^D\right)$ fraction of the constraints of ψ, then $\mathrm{MMSA}(\varphi) > s \cdot g$, where g is an arbitrary parameter. Dinur et al. [21] showed that it is NP-hard to decide whether a given CSP with n variables and of arity $D = (\log\log n)^{O(1)}$ is satisfiable or every assignment satisfies at most a $1/\mathsf{poly}(n)$ fraction of the constraints. Combining these two hardness results, we conclude that $\mathrm{Gap}_g\mathrm{MMSA}$ is NP-hard, where $g \ge \mathsf{poly}(n)^{1/D} \cdot D^{-D} \ge n^{1/(\log\log n)^{O(1)}}$. ◀

6.2.3 A Generic Connection from Secret Sharing Schemes

We now formally state a generic reduction that reduces the problem of estimating the weight of access structures to GapMINcKT.

▶ **Theorem 6.14.** *Let $\mathcal{A} = \{\mathcal{A}_\varphi\}_{\varphi\in\{0,1\}^*}$ be a family of access structures that admits efficient secret sharing schemes. Let $\tau\colon \mathbb{N}^3 \to \mathbb{N}$ be a function such that $t \le \tau(n,m,t) \le n^c \cdot m^{1-1/c} \cdot t^c$ for some constant $c > 1$. Then, there exists a randomized polynomial-time algorithm R that takes $\varphi \in \{0,1\}^*$ as input and outputs $x, y \in \{0,1\}^*$ and $t, \rho \in \mathbb{N}$ such that*

$$\frac{\rho}{8c} \cdot w(\mathcal{A}_\varphi) + \log t' \le \mathrm{K}^{t'}(x \mid y) \le \mathrm{K}^t(x \mid y) \le \rho \cdot w(\mathcal{A}_\varphi) - \mathrm{cd}^t(y)$$

holds with probability at least $1 - o(1)$ over the internal randomness of R, where $t' := \tau(|x|, |y|, t)$.

It is easy to observe that this reduction implies NP-hardness of GapMINcKT.

Proof of Theorem 6.6. Let $\mathcal{A} = \{\mathcal{A}_\varphi\}_\varphi$ be the family of access structures for monotone formulas given in Lemma 6.11. Let R be the reduction of Theorem 6.14. Consider a reduction R' that reduces an instance (φ, s) of $\mathrm{Gap}_g\mathrm{MMSA}$ to an instance $(x, y, 1^t, 1^{s'})$ of $\mathrm{Gap}_\tau\mathrm{MINcKT}$ such that $(x, y, t, \rho) := R(\varphi)$ and $s' := s \cdot \rho$.

Let $(\Pi_{\textsc{Yes}}, \Pi_{\textsc{No}}) := \mathrm{Gap}_g\mathrm{MMSA}$. We claim below that the reduction R' reduces $\mathrm{Gap}_g\mathrm{MMSA}$ to $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ for some σ, which implies NP-hardness of $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ by Lemma 6.13. If $(\varphi, s) \in \Pi_{\textsc{Yes}}$, then $w(\mathcal{A}_\varphi) = \mathrm{MMSA}(\varphi) \le s$. By the properties of R, with high probability, it holds that $\mathrm{K}^t(x \mid y) \le \rho \cdot w(\mathcal{A}_\varphi) - \mathrm{cd}^t(y) \le s' - \mathrm{cd}^{t,t'}(y)$, which implies that $(x, y, 1^t, 1^{s'})$ is a YES instance of $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$. If $(\varphi, s) \in \Pi_{\textsc{No}}$, then $w(\mathcal{A}_\varphi) = \mathrm{MMSA}(\varphi) > g(|\varphi|) \cdot s$. Also by the properties of R, with high probability, it holds that $\mathrm{K}^{t'}(x \mid y) \ge \frac{\rho}{8c} \cdot w(\mathcal{A}_\varphi) + \log t' > \frac{g(|\varphi|)}{8c} \cdot s' + \log t'$, which implies that $(x, y, 1^t, 1^{s'})$ is a NO instance of $\mathrm{Gap}_{\tau,\sigma}\mathrm{MINcKT}$ for some $\sigma(|x|) := \frac{g(|\varphi|)}{8c} \ge |\varphi|^{1/(\log\log|\varphi|)^{O(1)}} \ge |x|^{1/(\log\log|x|)^{O(1)}}$, where the last inequality follows from the fact that $|\varphi| \ge |x|^{\Omega(1)}$. ◀

It remains to prove Theorem 6.14. To show that the additive error $\mathrm{cd}^t(y)$ is relatively small, we use the fact that for D independent random samples $y^1, \ldots, y^D$ from a distribution $\mathcal{D}$ samplable by polynomial-size circuits, the amortized time-bounded Kolmogorov complexity of $y^1, \ldots, y^D$ approaches the entropy of $\mathcal{D}$ asymptotically.

▶ **Lemma 6.15** ([6, 5])**.** *Let $\mathcal{D} = \{\mathcal{D}_x\}_{x\in\{0,1\}^*}$ be a family of distributions sampled by circuits of size $\mathsf{poly}(|x|)$. Then, there exist a polynomial p and a constant $\delta > 0$ such that for every $x \in \{0,1\}^*$ and for every $D \ge p(|x|)$,*

$$\frac{1}{D} \cdot \mathrm{K}^t(y^1, \ldots, y^D) \leq \mathrm{H}(\mathcal{D}_x) + \frac{1}{2} \cdot D^{-\delta},$$
$$\frac{1}{D} \cdot \mathrm{K}(y^1, \ldots, y^D) \geq \mathrm{H}(\mathcal{D}_x) - \frac{1}{2} \cdot D^{-\delta}$$

holds with probability at least $1 - 2^{-|x|}$ *over a random choice of* D *independent samples* $y^1, \ldots, y^D$ *from* $\mathcal{D}_x$*, where* $t := p(|x|)$.

This lemma immediately implies that the computational depth of independent samples from $\mathcal{D}_x$ is small.

▶ **Corollary 6.16.** *Under the same assumptions as Lemma 6.15, with probability at least* $1 - 2^{-|x|}$ *over a random choice of* D *independent samples* $y^1, \ldots, y^D$ *from* $\mathcal{D}_x$*, it holds that*

$$\mathrm{cd}^t(y^1, \ldots, y^D) \leq D^{1-\delta}.$$

Proof. It follows from Lemma 6.15 that

$$\mathrm{cd}^t(y^1, \ldots, y^D) = \mathrm{K}^t(y^1, \ldots, y^D) - \mathrm{K}(y^1, \ldots, y^D) \leq D^{1-\delta}. \qquad \blacktriangleleft$$

We are now ready to prove Theorem 6.14.

Proof of Theorem 6.14. Fix an input $\varphi \in \{0,1\}^*$. Let n be the number of parties in the access structure $\mathcal{A}_\varphi$. Let $\lambda = O(\log |\varphi|)$, t, and ℓ be parameters chosen later.

We first define a family $\mathcal{D} = \{\mathcal{D}_\varphi\}_{\varphi \in \{0,1\}^*}$ of distributions by using the following sampling procedure: Choose $x \sim \{0,1\}^\ell$ uniformly at random. Let $(s_1, \ldots, s_n) := \mathrm{Share}(\varphi, x)$ and let m be the length of each share; i.e., $m := |s_i|$. Pick $k_1, \ldots, k_n \sim \{0,1\}^\lambda$ randomly. We define a string $y \in \{0,1\}^{2^\lambda \cdot m}$, which we identify with a function $y \colon \{0,1\}^\lambda \to \{0,1\}^m$: For every $q \in \{0,1\}^\lambda$, let $y(q) := s_i$ if $q = k_i$ for some $i \in [n]$ and let $y(q) := 0^m$ otherwise. Note that y is not well defined if $(k_i)_{i \in [n]}$ is not pairwise distinct; however, we will show that y is well defined with high probability. The output of the sampling procedure is defined as (x, y, k, s), where $k := (k_1, \ldots, k_n)$ and $s := (s_1, \ldots, s_n)$.

The algorithm R operates as follows: For a given input $\varphi \in \{0,1\}^*$, pick D independent samples $(x^1, y^1, k^1, s^1), \ldots, (x^D, y^D, k^D, s^D)$ from $\mathcal{D}_\varphi$. Let $x := (x^1, \ldots, x^D)$, $y := (y^1, \ldots, y^D)$, and $\rho := 2\lambda D$. The output of R is defined to be (x, y, t, ρ).

Below, we prove the correctness of R using a sequence of claims. Let $k := (k_i^d \mid i \in [n], d \in [D])$ and let $s := (s_i^d \mid i \in [n], d \in [D])$. We first observe that k is Kolmogorov-random with high probability:

▷ Claim 6.17. With probability at least $1 - o(1)$, it holds that

$$\mathrm{K}(k \mid s) \geq nD\lambda - \log n. \tag{16}$$

Proof. Since $k = (k_i^d)_{i \in [n], d \in [D]}$ is uniformly distributed over $(\{0,1\}^\lambda)^{nD}$ and independent of s, the claim follows from Fact 3.2. ◁

▷ Claim 6.18. If Equation (16) holds, then $k = (k_i^d)_{i \in [n], d \in [D]}$ is pairwise distinct and hence y is well defined.

Proof. If there exists $(i, d) \neq (i', d')$ such that $k_i^d = k_{i'}^{d'}$, then k can be described by a program of size $nD\lambda - \lambda + O(\log nD)$. Choosing a large enough $\lambda = O(\log |\varphi|)$, this implies that $\mathrm{K}(k) \leq nD\lambda - \lambda + O(\log nD) < nD\lambda - \log n$, which contradicts Equation (16). ◁

Below, we assume that Equation (16) holds. We prove an upper bound of $\mathrm{K}^t(x \mid y)$.

▷ Claim 6.19. $\mathrm{K}^t(x \mid y) \leq \rho \cdot w(\mathcal{A}_\varphi) - \mathrm{cd}^t(y)$ holds with probability at least $1 - o(1)$.

Proof. Let $T \in \mathcal{A}_\varphi$ be a minimum authorized set of parties such that $|T| = w(\mathcal{A}_\varphi)$. We present an oracle program M^y that takes an index d and outputs $x^d \in \{0,1\}^\ell$ as follows: M^y takes $d \in [D]$ as input and T, $\left\{k_i^d \;\middle|\; i \in T, d \in [D]\right\}$ and φ as hard-wired input, computes $\left\{s_i^d \;\middle|\; i \in T\right\} = \left\{y^d(k_i) \;\middle|\; i \in T\right\}$ by making queries to the oracle y, and outputs $\mathrm{Rec}(\varphi, s_T^d)$, which is equal to x^d by the correctness of a secret sharing scheme and the assumption that $T \in \mathcal{A}_\varphi$. The size of the oracle machine M is at most

$$O(|T| \cdot \log n) + |T| \cdot D \cdot \lambda + |\varphi| + O(\log D) \leq 2\lambda D \cdot w(\mathcal{A}_\varphi) - D^{1-\delta},$$

where $\delta > 0$ is the constant from Corollary 6.16 and this inequality follows by choosing sufficiently large $D \geq \mathsf{poly}(|\varphi|)$. Applying Corollary 6.16 to the distribution of y, we obtain $\mathrm{cd}^t(y) \leq D^{1-\delta}$ with probability at least $1 - o(1)$; thus, $|M| \leq \rho \cdot w(\mathcal{A}_\varphi) - \mathrm{cd}^t(y)$. The running time of M is at most $\mathsf{poly}(|\varphi|)$, which is independent of λ. We choose t so that this running time is at most t. ◁

The remainder of the proof is devoted to proving the lower bound.

▷ Claim 6.20. Let $\theta := \frac{\rho}{8c} \cdot w(\mathcal{A}_\varphi) + \log t'$. Then, $\mathrm{K}^{t'}(x \mid y) \geq \theta$ with probability at least $1 - o(1)$.

We first clarify the condition of random variables under which the claim holds. Let $s_{[n]}^{[D]\setminus\{d\}} := (s_i^{d'} \mid i \in [n], d' \in [D] \setminus \{d\})$.

▷ Claim 6.21. With probability at least $1 - o(1)$, it holds that for every unauthorized set $T \notin \mathcal{A}_\varphi$ of parties and for every $d \in [D]$,

$$\mathrm{K}\left(x^d \;\middle|\; s_T^d, s_{[n]}^{[D]\setminus\{d\}}, k\right) \geq \ell - n - 2\log D. \tag{17}$$

Proof. Fix $d \in [D]$. Since the random variable x^d is independent of $s_{[n]}^{[D]\setminus\{d\}}$ and k, by Lemma 6.9,

$$\min_{T \notin \mathcal{A}_\varphi} \mathrm{K}\left(x^d \;\middle|\; s_T^d, s_{[n]}^{[D]\setminus\{d\}}, k\right) \geq \ell - n - 2\log D$$

holds with probability at least $1 - \frac{1}{D^2}$. The claim follows by taking a union bound over all $d \in [D]$. ◁

In what follows, we assume Equations (16) and (17) and prove $\mathrm{K}^{t'}(x \mid y) \geq \theta$, which will complete the proof of Claim 6.20. Assume, by way of contradiction, that $\mathrm{K}^{t'}(x \mid y) < \theta$. Let $M \in \{0,1\}^*$ be the description of an oracle program such that $|M| \leq \theta + O(\log \ell)$ and $M^y(d)$ outputs $x^d \in \{0,1\}^\ell$ in time $t'' := t'\ell$ for every $d \in [D]$; the machine M can be constructed from a program that witnesses $\mathrm{K}^{t'}(x \mid y) < \theta$. For each $d \in [D]$, let $T(M,d)$ be the set of indices $i \in [n]$ such that some bit of $y^d(k_i^d) \in \{0,1\}^m$ is queried during the computation of $M^y(d)$.

▷ Claim 6.22. Let M be an oracle program that runs in time t'' and let $\alpha := \sum_{d=1}^{D} |T(M,d)|$. Then,

$$\mathrm{K}(k \mid s) \leq |M| + (nD - \alpha) \cdot \lambda + \alpha \cdot (\log t'' + \log nD) + O(\log nD).$$

Proof. By the definition of $T(M,d)$, for every $i \in T(M,d)$, there exists a time step $t_{i,d} \in [t'']$ such that M^y makes a query k_i^d to the oracle y on input $d \in [D]$. To describe k given s, consider the following program M': M' takes

$$\{(i,d,t_{i,d}) \in [n] \times [D] \times [t''] \mid i \in T(M,d)\},$$
$$\left\{k_i^d \in \{0,1\}^\lambda \;\middle|\; (i,d) \in [n] \times [D], i \notin T(M,d)\right\},$$

and M as input and simulates M^y on input d for every $d \in [D]$. At the time step $t_{i,d}$ of the simulation of $M^y(d)$ for some $i \in T(M,d)$, M' reads the query k_i^d that $M^y(d)$ makes to the oracle y^d and answers the query with s_i^d. M' continues the simulation until the time step t'', at which point M' knows k_i^d for every $i \in T(M,d)$. Finally, M' outputs k. The input of M' can be encoded as a string of length $\alpha \cdot (\log t'' + \log nD) + (nD - \alpha) \cdot \lambda + |M| + O(\log nD)$. ◁

It follows from Claim 6.22 and Equation (16) that

$$nD\lambda - \log n \leq |M| + (nD - \alpha) \cdot \lambda + \alpha \cdot (\log t'' + \log nD) + O(\log nD),$$

which can be simplified to

$$(\lambda - \log t'' - \log nD) \cdot \alpha \leq |M| + O(\log nD). \tag{18}$$

Since $t''/\ell = t' = \tau(|x|,|y|,t) \leq |x|^c \cdot |y|^{1-1/c} \cdot t^c \leq 2^{(1-1/c)\cdot\lambda} \cdot (D \cdot |\varphi| \cdot \ell \cdot m)^{O(c)}$, we may choose $\lambda = O(\log |\varphi|)$ large enough so that $t'' \leq 2^{(1-1/2c)\cdot\lambda - \log nD}$. Then, by Equation (18), we obtain

$$\alpha \leq \frac{2c}{\lambda} \cdot (|M| + O(\log nD)) < \frac{4c}{\lambda} \cdot (\theta - \log t') = \frac{\rho}{2\lambda} \cdot w(\mathcal{A}_\varphi) = D \cdot w(\mathcal{A}_\varphi),$$

where the second inequality follows from $|M| + O(\log nD) \leq \theta + O(\log nD\ell t') < 2\theta$.[16] It follows that there exists $d \in [D]$ such that $|T(M,d)| \leq \alpha/D < w(\mathcal{A}_\varphi)$, which implies that

$$T(M,d) \notin \mathcal{A}_\varphi.$$

By Equation (17), we obtain

$$\mathrm{K}\left(x^d \;\middle|\; s^d_{T(M,d)}, s^{[D]\setminus\{d\}}_{[n]}, k\right) \geq \ell - n - 2\log D.$$

However, this contradicts the following claim.

▷ Claim 6.23. Let M be an oracle program such that $M^y(d)$ outputs x^d for every $d \in [D]$. Then, for every $d \in [D]$, it holds that

$$\mathrm{K}\left(x^d \;\middle|\; s^d_{T(M,d)}, s^{[D]\setminus\{d\}}_{[n]}, k\right) \leq |M| + O(\log D).$$

Proof. Since $M^y(d)$ outputs x^d without making any query in $\left\{k_i^d \;\middle|\; i \in [n] \setminus T(M,d)\right\}$, we can define another oracle $z = (z^1, \ldots, z^D)$ such that $z^{d'} := y^{d'}$ for every $d' \in [D] \setminus \{d\}$ and $z^d(k_i^d) := s_i^d$ for every $i \in T(M,d)$ and $z^d(k) := 0^m$ otherwise, so that there is no difference between y and z on inputs queried by $M^y(d)$; therefore, we obtain $x^d = M^y(d) = M^z(d)$. The claim follows by observing that the oracle z can be constructed from M, $d \in [D]$, $s^d_{T(M,d)}$, $s^{[D]\setminus\{d\}}_{[n]}$, and k. ◁

[16] We may assume without loss of generality that $w(\mathcal{A}_\varphi) \geq 1$; then, we have $\theta \geq \Omega(\rho) \geq \Omega(D)$, which can be assumed to be larger than logarithmic terms.

We conclude that $\ell - n - 2\log D \leq |M| + O(\log D) \leq O(\theta) \leq O(n\lambda D)$, which is a contradiction by letting $\ell \gg O(n\lambda D)$. This completes the proof of Claim 6.20. ◀

▶ Remark 6.24. It is not hard to observe that our NP-hardness hardness reduction also proves the NP-hardness of McKTP [4] because the upper bound of Claim 6.19 also holds for $\mathrm{KT}(x \mid y)$.

7 A Simple Proof for Worst-Case to Average-Case Connections

In this section, using SoI, we present an alternative proof of the connection from the worst-case complexity of the polynomial hierarchy PH to the average-case complexity of PH.

▶ **Theorem 7.1** ([35]). $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ *implies that* $\mathsf{PH} \subseteq \mathsf{DTIME}(2^{O(n/\log n)})$.

One important component of the proof of Theorem 7.1 is the notion of universal heuristic scheme.

▶ **Definition 7.2** (Universal Heuristic Scheme [35]). *A* (strong) universal heuristic scheme *for a language L is a pair (S, C) of polynomial-time algorithms such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$,*
1. *if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $C(x, 1^t, 1^k) = 1$, and*
2. *if $C(x, 1^t, 1^k) = 1$, then $S(x, 1^t, 1^{2^k}) = L(x)$.*

S and C are referred to as a solver *and a* checker*, respectively.*

A strong universal heuristic scheme enables the construction of an efficient algorithm:

▶ **Lemma 7.3** ([35]). *For every language L, if there exists a strong universal heuristic scheme for L, then $L \in \mathsf{DTIME}(2^{O(n/\log n)})$.*

Our goal is to construct a strong universal heuristic scheme for every language in PH. To this end, we introduce a weaker version of a universal heuristic scheme:

▶ **Definition 7.4.** *A* weak universal heuristic scheme *for a language L is a polynomial-time algorithm S such that, for some polynomial p, for any $n \in \mathbb{N}$, any $t \geq p(n)$, and any $x \in \{0,1\}^n$, if $\mathrm{cd}^{t,p(t)}(x) \leq k$, then $S(x, 1^t, 1^{2^k}) = L(x)$.*

We observe that weak and strong universal heuristic schemes are in fact equivalent in Heuristica.

▶ **Lemma 7.5.** *If* $\mathrm{GapMINKT} \in \mathsf{P}$*, the following are equivalent for every language L.*
1. *There exists a strong universal heuristic scheme for L.*
2. *There exists a weak universal heuristic scheme for L.*

Under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$, [35] showed the equivalence between the existence of a strong universal heuristic scheme for L and $\{L\} \times \mathrm{PSAMP} \subseteq \mathsf{Avg_PP}$, where $\mathsf{Avg_PP}$ denotes the class of distributional problems solvable by algorithms whose running time is bounded above by some polynomial-time-computable average-case polynomial-time bound. Lemma 7.5 adds a new equivalent statement to this.

Proof of Lemma 7.5.

strong $\to$ weak. For a strong universal heuristic scheme (S, C), the solver S satisfies the definition of a weak universal heuristic scheme.

weak $\to$ strong. The idea of constructing a checker is to estimate the time-bounded computational depth of an input by using an algorithm for GapMINKT.

Let S be a weak universal heuristic scheme for L and let p be the polynomial in Definition 7.4. Let $\widetilde{\mathrm{K}}$ be the polynomial-time algorithm of Fact 3.4 such that for every $x \in \{0,1\}^*$ and every $t \geq |x|$,[17]

$$\mathrm{K}^{p(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) \leq \mathrm{K}^t(x).$$

Observe that

$$\mathrm{cd}^{p(t),p^{(2)}(t)}(x) - \log p(t) \leq \widetilde{\mathrm{K}}(x, 1^t) - \widetilde{\mathrm{K}}(x, 1^{p^{(2)}(t)}) \leq \mathrm{cd}^{t,p^{(3)}(t)}(x) + \log p^{(3)}(t). \tag{19}$$

We define a checker C as follows: $C(x, 1^t, 1^k) = 1$ if and only if $\widetilde{\mathrm{K}}(x, 1^t) - \widetilde{\mathrm{K}}(x, 1^{p^{(2)}(t)}) \leq k + \log p^{(3)}(t)$. We define a solver S' so that $S'(x, 1^t, 1^{2^k}) := S'(x, 1^{p(t)}, 1^{2^{k'}})$, where $k' := k + \log p(t) + \log p^{(3)}(t)$.

Below, we claim that (S', C) is a strong universal heuristic scheme by showing that it satisfies the two properties of Definition 7.2.

1. If $\mathrm{cd}^{t,p^{(3)}(t)}(x) \leq k$, then by the upper bound of Equation (19), we have $C(x, 1^t, 1^k) = 1$.
2. If $C(x, 1^t, 1^k) = 1$, then by the definition of C and by the lower bound of Equation (19), we obtain $\mathrm{cd}^{p(t),p^{(2)}(t)}(x) \leq k + \log p(t) + \log p^{(3)}(t) = k'$. It follows from the property of the weak heuristic scheme S that $S'(x, 1^t, 1^{2^k}) = S(x, 1^{p(t)}, 1^{2^{k'}}) = L(x)$. ◀

The following lemma shows that any string that can be efficiently compressed with some PH oracle can also be compressed without the oracle if $\mathsf{DistPH} \subseteq \mathsf{AvgP}$.

▶ **Lemma 7.6** ([33]). *Let A be an oracle. Assume that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}$. Then, there exists a polynomial p such that, for every $x \in \{0,1\}^*$ and every $t \geq |x|$,*

$$\mathrm{K}^{p(t)}(x) \leq \mathrm{K}^{t,A}(x) + \log p(t).$$

We now use SoI to construct a weak universal heuristic scheme for every language in PH.

▶ **Lemma 7.7.** *Let $k \in \mathbb{N}$. If $\mathsf{Dist}\Sigma^{\mathsf{p}}_{k+1} \subseteq \mathsf{AvgP}$, then for every language $L \in \Sigma^{\mathsf{p}}_k$, there exists a weak universal heuristic scheme for L.*

Proof. We prove this by induction on $k \in \mathbb{N}$. The base case ($k = 0$) is trivial because every language $L \in \Sigma^{\mathsf{p}}_0 = \mathsf{P}$ admits a weak universal heuristic scheme. Let $k \geq 1$. Let V be a language in Π^{p}_{k-1} such that $x \in L$ if and only if $V(x, y) = 1$ for some $y \in \{0,1\}^{\mathsf{poly}(|x|)}$. For every $x \in L$, let y_x be the lexicographically first string y such that $V(x, y) = 1$. The following claim is the key to the construction of a weak universal heuristic scheme.

▷ Claim 7.8. There exists a polynomial q such that for every $x \in L$ and every $t \geq |x|$,

$$\mathrm{K}^{q(t)}(y_x \mid x) \leq \mathrm{cd}^{t,q(t)}(x) + \log q(t).$$

Proof. By a standard search-to-decision reduction, y_x can be computed from x in polynomial time with oracle access to some oracle A in Σ^{p}_k; thus, it follows that

$$\mathrm{K}^{p^{(2)}(t)}(y_x, x) \leq \mathrm{K}^{p(t),A}(y_x, x) + \log p^{(2)}(t) \leq \mathrm{K}^t(x) + \log p^{(2)}(t) + O(1),$$

where the first inequality follows from Lemma 7.6.

[17] We may assume without loss of generality that the polynomial p in Fact 3.4 is the same polynomial with Definition 7.4.

We claim that $\mathrm{K}^{q(t)}(y_x \mid x) \leq \mathrm{cd}^{t,q(t)}(x) + \log q(t)$ for some polynomial q. Note that SoI holds because of Theorem 1.2. Using SoI, we obtain

$$\begin{aligned}\mathrm{K}^{p^{(3)}(t)}(y_x \mid x) &\leq \mathrm{K}^{p^{(2)}(t)}(y_x, x) - \mathrm{K}^{p^{(3)}(t)}(x) + \log p^{(3)}(t) \\ &\leq \mathrm{K}^{t}(x) - \mathrm{K}^{p^{(3)}(t)}(x) + O(\log p^{(3)}(t)) \\ &= \mathrm{cd}^{t,p^{(3)}(t)}(x) + O(\log p^{(3)}(t)),\end{aligned}$$

where the first inequality follows from SoI. The claim follows by letting $q(t) := p^{(3)}(t)^{O(1)}$. ◁

By the induction hypothesis, there exists a weak universal heuristic scheme S for V. Let p be the polynomial in Definition 7.4.

We now present a weak universal heuristic scheme S' for L: The algorithm S' takes $(x, 1^t, 1^{2^k})$ as input and computes the set Y of strings $y \in \{0,1\}^*$ such that there exists a program of length at most $k + \log q(t)$ that takes x as input and outputs y in time $q(t)$. Equivalently, we define

$$Y := \left\{ y \in \{0,1\}^* \;\middle|\; \mathrm{K}^{q(t)}(y \mid x) \leq k + \log q(t). \right\}$$

Note that $|Y| \leq 2^{k+\log q(t)+1}$ and Y can be computed in time $\mathsf{poly}(|x|, t, 2^k)$. The algorithm S' outputs 1 if and only if there exists a string $y \in Y$ such that $S((x,y), 1^{t'}, 1^{2^{k'}}) = 1$, where $t' = t^{O(1)}$ and $k' = O(k + \log t)$ are parameters chosen later. Clearly, S' is a polynomial-time algorithm.

We claim the correctness of S'. Let q' be a polynomial chosen later. Assume that $\mathrm{cd}^{t,q'(t)}(x) \leq k$. We claim that for some parameter $t' = q(t)^{O(1)}$ and for every $y \in Y$, the $(t', p(t'))$-time-bounded computational depth of (x,y) is at most k', which will imply that the output of the weak universal heuristic scheme S is correct on input (x,y). For every $y \in Y$, we have

$$\begin{aligned}\mathrm{cd}^{t',p(t')}(x,y) &\leq \mathrm{K}^{q(t)}(x) + \mathrm{K}^{q(t)}(y \mid x) - \mathrm{K}^{p(t')}(x,y) + O(1) \\ &\leq \mathrm{cd}^{q(t),2p(t')}(x) + k + \log q(t) + O(1) \\ &\leq 2k + \log q(t) + O(1) =: k',\end{aligned}$$

where the first inequality follows from the definition of time-bounded computational depth, the second inequality follows from the fact that $\mathrm{K}^{2p(t')}(x) \leq \mathrm{K}^{p(t')}(x,y) + O(1)$ and $y \in Y$, and the third inequality follows from the assumption that $\mathrm{cd}^{q(t),2p(t')}(x) \leq \mathrm{cd}^{t,q'(t)}(x) \leq k$, where we define $q'(t) := 2p(t')$. By the correctness of the weak universal heuristic scheme S, we obtain $S((x,y), 1^{t'}, 1^{2^{k'}}) = V(x,y)$. If $x \in L$, Claim 7.8 implies that $y_x \in Y$; thus, we have $S((x,y_x), 1^{t'}, 1^{2^{k'}}) = V(x,y_x) = 1$, which implies that S' outputs 1. If $x \notin L$, then $V(x,y) = 0$ for every string y; thus, we obtain $S((x,y), 1^{t'}, 1^{2^{k'}}) = V(x,y) = 0$, which implies that S' outputs 0. ◀

Proof of Theorem 7.1. By Lemma 7.7, every language $L \in \mathsf{PH}$ admits a weak universal heuristic scheme. By Lemmas 2.1 and 7.5, the weak universal heuristic scheme can be converted into a strong universal heuristic scheme. Finally, using Lemma 7.3, we obtain $L \in \mathsf{DTIME}(2^{O(n/\log n)})$. ◀

8 What Is Implied by Symmetry of Information?

Toward giving an exact characterization of SoI, we investigate what SoI implies. We show that SoI is sandwiched between the existence of *errorless* heuristic schemes (denoted by AvgP) for MINKT and the existence of *error-prone* heuristic schemes (denoted by HeurP) for MINKT under the plausible assumption that E requires exponential-sized circuits. We start with the definition of error-prone and errorless heuristic schemes.

▶ **Definition 8.1.** *For a function $t\colon \mathbb{N} \to \mathbb{N}$ and a family $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions, an algorithm A is said to be an* error-prone heuristic scheme *for $\mathrm{K}^t(\text{-})$ with respect to $\mathcal{D}$ if for every $n \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,*

1. *$A(x; n, \delta)$ halts in time $\mathsf{poly}(n/\delta)$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$.*
2. *$\Pr_{x\sim\mathcal{D}_n}\left[A(x; n, \delta) \neq \mathrm{K}^{t(n)}(x)\right] \leq \delta$.*

If A satisfies the additional condition that

resume *$A(x; n, \delta) \in \left\{\mathrm{K}^{t(n)}(x), \perp\right\}$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$,*

then A is said to be an errorless heuristic scheme *for $\mathrm{K}^t(\text{-})$ with respect to $\mathcal{D}$. We write $(\mathrm{K}^t(\text{-}), \mathcal{D}) \in \mathsf{AvgP}$ and $(\mathrm{K}^t(\text{-}), \mathcal{D}) \in \mathsf{HeurP}$ if there exists an errorless heuristic scheme and an error-prone heuristic scheme for $\mathrm{K}^t(\text{-})$ with respect to $\mathcal{D}$, respectively.*

Definition 8.1 is different from the standard definition given in [16] in the following two respects.

1. The classes AvgP and HeurP are usually defined as the class of *decision* problems; here, we require that heuristic algorithms output an integer $\mathrm{K}^{t(n)}(x) \in \mathbb{N}$ on input $x \in \mathrm{supp}(\mathcal{D}_n)$.
2. The output $\mathrm{K}^{t(n)}(x)$ of the distributional problem $(\mathrm{K}^t(\text{-}), \mathcal{D})$ depends on the size parameter n.[18]

We mention that it is possible to state Theorem 8.2 below using only the standard definitions, though the statement becomes somewhat awkward; see Footnote 19. We now state the main result of this section.

▶ **Theorem 8.2.** *Assume that $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ for some constant $\epsilon > 0$. In the following list, we have 1 $\iff$ 2 $\implies$ 3 $\implies$ 4 and 3 $\implies$ 5.*

1. $\mathrm{GapMINKT} \in \mathsf{P}$.
2. *For every $\mathcal{D} \in \mathrm{PSAMP}$, there exists a polynomial t_0 such that $(\mathrm{K}^t(\text{-}), \mathcal{D}) \in \mathsf{AvgP}$ for every polynomial $t \geq t_0$.*[19]
3. *SoI holds.*
4. *For every $\mathcal{D} \in \mathrm{PSAMP}$, there exists a polynomial t_0 such that $(\mathrm{K}^t(\text{-}), \mathcal{D}) \in \mathsf{HeurP}$ for every polynomial $t \geq t_0$.*
5. *$\mathrm{Gap}_\tau\mathrm{MINKT} \in \mathsf{DTIME}(2^{O(n/\log n)})$ for some function $\tau(n, t) = 2^{O(n/\log n)}$.*

Here, PSAMP denotes the class of families $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ of distributions such that there exists a randomized polynomial-time algorithm S such that the distribution induced by $S(1^n)$ is identical to $\mathcal{D}_n$.

Longpré and Watanabe [58] showed that SoI implies that $\mathrm{K}^t(\text{-})$ admits an error-prone heuristic scheme with respect to the uniform distribution. Theorem 8.2 extends their result to an arbitrary polynomial-time samplable distribution. To prove this, we need the following result from [58].

---

[18] The output does not depend on the size parameter n in the special case that $n = |x|$ for every $x \in \mathrm{supp}(\mathcal{D}_n)$.

[19] This statement can be equivalently stated as the statement that $(\mathrm{MINKT}, \mathcal{D}^t) \in \mathsf{AvgP}$, where MINKT is the language defined as $\left\{(x, 1^t, 1^s) \;\middle|\; \mathrm{K}^t(x) \leq s\right\}$ and $\mathcal{D}^t$ denotes the family $\{\mathcal{D}^t_n\}_{n\in\mathbb{N}}$ of distributions such that $\mathcal{D}^t_n$ is the distribution that picks $(x, 1^s) \sim \mathcal{D}_n$ and outputs a sample $(x, 1^{t(n)}, 1^s)$.

▶ **Lemma 8.3** ([58]). *If SoI holds, then there exist a polynomial p and a polynomial-time algorithm M such that for every $x \in \{0,1\}^*$ and every $t \geq p(|x|)$ such that $\mathrm{cd}^{t,p(t)}(x) \leq k$, on input $(x, 1^t, 1^{2^k})$, M outputs the lexicographically first program of length $\mathrm{K}^t(x)$ that outputs x in time t.*

Interestingly, this was proved even before the notion of computational depth was introduced by Antunes et al. [9]. Lemma 8.3 is reminiscent of a weak universal heuristic scheme for the search version of MINKT; however, it does not satisfy the definition of a weak universal heuristic scheme in that the parameter t cannot be chosen independently of the instance of MINKT. For completeness, we present the proof of Lemma 8.3.

Proof. Let $x \in \{0,1\}^*$, $t \in \mathbb{N}$, and $k \in \mathbb{N}$ be given inputs. Let $d_{x,t} \in \{0,1\}^*$ be the lexicographically first program of length $\mathrm{K}^t(x)$ that outputs x in time t. By SoI, for some polynomial p, we have

$$\begin{aligned}\mathrm{K}^{p(2t)}(d_{x,t} \mid x) &\leq \mathrm{K}^{2t}(d_{x,t}, x) - \mathrm{K}^{p(2t)}(x) + \log p(2t) \\ &\leq |d_{x,t}| + O(1) - \mathrm{K}^{p(2t)}(x) + \log p(2t) = \mathrm{cd}^{t,p(2t)}(x) + \log p(2t) + O(1),\end{aligned}$$

where the last inequality holds because $(d_{x,t}, x)$ can be computed from the description $d_{x,t}$ of the program. Consequently, there exists a polynomial q such that

$$\mathrm{K}^{q(t)}(d_{x,t} \mid x) \leq \mathrm{cd}^{t,q(t)}(x) + \log q(t). \tag{20}$$

We now describe the algorithm M: Given an input $(x, 1^t, 1^{2^k})$ such that $\mathrm{cd}^{t,q(t)}(x) \leq k$, the algorithm M computes the set Y of strings $d \in \{0,1\}^*$ such that $\mathrm{K}^{q(t)}(d \mid x) \leq k + \log q(t)$ and the program described by d outputs x in time t. The set Y can be computed in time $\mathsf{poly}(|x|, t, 2^k)$ by enumerating all the programs of length at most $k + \log q(t)$ that take x as input. Let $s := \min\{|d| \mid d \in Y\}$. The algorithm M outputs the lexicographically first string $d \in Y \cap \{0,1\}^s$.

To see the correctness of M, fix an input $(x, 1^t, 1^{2^k})$ such that $\mathrm{cd}^{t,q(t)}(x) \leq k$. We claim that M outputs $d_{x,t}$. Observe that $s \geq \mathrm{K}^t(x)$ by the definition of $\mathrm{K}^t(x)$. Moreover, it follows from Equation (20) that $d_{x,t} \in Y$; hence, we obtain $s \leq |d_{x,t}| = \mathrm{K}^t(x)$. Since $s = \mathrm{K}^t(x)$, the lexicographically first string $d \in Y \cap \{0,1\}^s$ is equal to $d_{x,t}$. ◀

Under a plausible derandomization hypothesis, Antunes and Fortnow [11] showed that if a string x is drawn from a polynomial-time samplable distribution, then the computational depth of x is small with high probability. The same conclusion holds under SoI.

▶ **Lemma 8.4** (see [35, Theorem 9.6 and Corollary 9.8]). *If SoI holds, then for every $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}} \in \mathrm{PSAMP}$, there exists a polynomial t such that for every $n \in \mathbb{N}$,*

$$\Pr_{x\sim\mathcal{D}_n}\left[\mathrm{cd}^{t(n)}(x) > k\right] \leq 2^{-k+\log t(n)}.$$

▶ **Theorem 8.5.** *If SoI holds, then for every $\mathcal{D} \in \mathrm{PSAMP}$, there exist a polynomial t and an error-prone heuristic scheme for $\mathrm{K}^t(\text{-})$ with respect to $\mathcal{D}$.*

Proof. Let M be the polynomial-time algorithm and p be the polynomial of Lemma 8.3. Let t_0 be the polynomial of Lemma 8.4.

We describe an error-prone heuristic scheme A: The algorithm A takes $(x; n, \delta)$ as input, where x is sampled from $\mathcal{D}_n$ and $\delta^{-1} \in \mathbb{N}$ is an error parameter. Let $k := \log(1/\delta) + \log t_0(n)$. The algorithm A outputs the length of the program computed by $M(x, 1^{t(n)}, 1^{2^k})$. Clearly, the algorithm A runs in time $\mathsf{poly}(n/\delta)$.

We claim the correctness of A, i.e., for every $n \in \mathbb{N}$ and every $\delta^{-1} \in \mathbb{N}$,

$$\Pr_{x \sim \mathcal{D}_n}\left[A(x; n, \delta) \neq \mathrm{K}^{t(n)}(x)\right] \leq \delta.$$

By the property of M, the algorithm M fails to output a program of size $\mathrm{K}^{t(n)}(x)$ on input $(x, 1^{t(n)}, 1^{2^k})$ only if $\mathrm{cd}^{t(n),p(t(n))}(x) > k$, which implies that $\mathrm{cd}^{t_0(n)}(x) \geq \mathrm{cd}^{t(n)}(x) > k$. By Lemma 8.4, this happens with probability at most $2^{-k+\log t_0(n)} \leq \delta$. We conclude that the probability that $A(x; n, \delta) \neq \mathrm{K}^{t(n)}(x)$ is at most δ, as desired. ◀

Next, we present an errorless heuristic version of Theorem 8.5.

▶ **Theorem 8.6.** *If SoI holds and* GapMINKT $\in \mathsf{P}$*, then for every* $\mathcal{D} \in$ PSamp*, there exist a polynomial* t *and an errorless heuristic scheme for* $\mathrm{K}^t(\text{-})$ *with respect to* $\mathcal{D}$*.*

Proof. The proof is similar to that of Theorem 8.5, except that we use the algorithm for GapMINKT to translate the error-prone algorithm of Theorem 8.5 to an errorless algorithm in a way similar to Lemma 7.5.

Let p be the larger of the polynomial of Lemma 8.3 and the polynomial of Fact 3.4. Let C be the checker defined in the proof of Lemma 7.5, which satisfies the following properties: For every $x \in \{0,1\}^*$ and every $t \geq |x|$ and every $t \in \mathbb{N}$,

1. if $\mathrm{cd}^{t,p^{(3)}(t)}(x) \leq k$, then $C(x, 1^t, 1^k) = 1$, and
2. if $C(x, 1^t, 1^k) = 1$, then $\mathrm{cd}^{p(t),p^{(2)}(t)}(x) \leq k + \log p(t) + \log p^{(3)}(t)$.

We now describe an errorless heuristic scheme A: The algorithm A takes $(x; n, \delta)$ as input. Let $t := t(n)$ and $k := \log(1/\delta) + \log t$. The algorithm A outputs the special failure symbol $\perp$ if $C(x, 1^t, 1^k) = 0$. Otherwise, A outputs the length of the program computed by $M(x, 1^{p(t)}, 1^{2^{k'}})$, where we define $k' := k + \log p(t) + \log p^{(3)}(t)$. Clearly, the algorithm A runs in time $\mathsf{poly}(n/\delta)$.

To prove the correctness of A, we first show that A is errorless. For every $n \in \mathbb{N}$ and every $x \in \mathrm{supp}(\mathcal{D}_n)$, we claim

$$A(x; n, \delta) \in \{\mathrm{K}^t(x), \perp\}$$

by considering the following two cases: (1) If the checker C outputs 0, then by definition, A outputs $\perp$. (2) Otherwise, we have $C(x, 1^t, 1^k) = 1$, which implies that $\mathrm{cd}^{p(t),p^{(2)}(t)}(x) \leq k'$. In the latter case, by the properties of M, the output of A is equal to $\mathrm{K}^t(x)$.

Next, we show that the failure probability of A is at most δ. Observe that A fails only if $C(x, t, 1^k) = 0$, which implies that $\mathrm{cd}^t(x) \geq \mathrm{cd}^{t,p^{(3)}(t)}(x) > k$. This happens with probability at most $2^{-k+\log t} \leq \delta$ by Lemma 8.4. We conclude that

$$\Pr_{x \sim \mathcal{D}_n}[A(x; n, \delta) = \perp] \leq \delta.$$ ◀

Finally, we construct a slightly sub-exponential-time algorithm for GapMINKT from SoI.

▶ **Theorem 8.7.** *If SoI holds, then for every constant* $\delta > 0$*, there exists an algorithm* M *that, on input* (x, t) *such that* $|x| \leq t \leq 2^{|x|^{1-\delta}}$*, outputs a program of length* $\mathrm{K}^t(x)$ *that outputs* x *in time* t'*, where* $t' = 2^{O(|x|/\log|x|)}$*. The algorithm* M *runs in time* $2^{O(|x|/\log|x|)}$ *on input* (x, t)*.*

In particular, the length of the program d computed by M on input (x, t) satisfies that

$$\mathrm{K}^{t'}(x) \leq |d| \leq \mathrm{K}^t(x). \tag{21}$$

Proof of Theorem 8.7. Let M be the algorithm and p be the polynomial from Lemma 8.3. Fix a string $x \in \{0,1\}^*$ of length n and an integer $t \in \mathbb{N}$. Following [35], we consider the following telescoping sum:

$$\sum_{i=1}^{I} \mathrm{cd}^{p^{(i-1)}(t),p^{(i)}(t)}(x) = \mathrm{cd}^{t,p^{(I)}(t)}(x) \leq n + O(1) \leq 2n,$$

where I is a parameter chosen later. By taking the minimum term on the left-hand side, there exists an index $i \in [I]$ such that

$$\mathrm{cd}^{p^{(i-1)}(t),p^{(i)}(t)}(x) \leq 2n/I. \tag{22}$$

We define an algorithm M' as follows. Given (x,t) as input, let $t_0 := \max\{t, p(n)\}$ and $t_i := p^{(i)}(t_0)$ for every $i \in [I]$. Let d_i be the program computed by $M(x, 1^{t_i}, 1^{2^{2n/I}})$ for every $i \in [I]$. The algorithm M' outputs the shortest program d_i that outputs x in time t_i.

The running time of M' is at most

$$\mathsf{poly}(t_I, 2^{2n/I}) \leq t^{c^I} \cdot 2^{O(n/I)} \leq 2^{O(c^I \cdot n^{1-\delta} + n/I)},$$

where c is some universal constant. By letting $I := \epsilon \log n$ for a sufficiently small constant $\epsilon > 0$, the running time can be bounded by $2^{O(n/\log n)}$.

We prove the correctness of M'. Let i^* be the index that satisfies Equation (22). Then, by the correctness of M, the program d_{i^*} prints x in time t_{i^*} and $|d_{i^*}| = \mathrm{K}^{t_{i^*}}(x)$. This means that the output of M' is well defined. Let d_i be the program computed by M'. By the definition of M', we have $|d_i| \leq |d_{i^*}| = \mathrm{K}^{t_{i^*}}(x) \leq \mathrm{K}^t(x)$. Moreover, the program d_i prints x in time $t_i \leq t_I$. ◀

Proof of Theorem 8.2. The implication from Item 2 to 1 is proved in [33]. Theorems 4.1 and 8.6 prove the implication from Item 1 to 2. Theorem 4.1 proves the implication from Item 1 to 3. Theorem 8.5 proves the implication from Item 3 to 4.

The implication from Item 3 to 5 easily follows from Theorem 8.7: We describe a $2^{O(n/\log n)}$-time algorithm that solves $\mathrm{Gap}_\tau\mathrm{MINKT}$. Given an instance $(x, 1^t, 1^s)$ of $\mathrm{Gap}_\tau\mathrm{MINKT}$, if $t \leq |x|^2$, then we use the search algorithm M of Theorem 8.7 and output 1 if and only if the length of the program computed by $M(x,t)$ is at most s. The search algorithm M runs in time $2^{O(|x|/\log|x|)}$. If $t > |x|^2$, then we use a trivial exhaustive search to find a shortest program that prints x in time t; this exhaustive search runs in time $2^{O(|x|)} \cdot t^{O(1)}$. In both cases, the algorithm runs in time $2^{O(n/\log n)}$, where n denotes the length $\Theta(|x| + t)$ of the instance $(x, 1^t, 1^s)$. The correctness of the algorithm follows from Equation (21). ◀

References

1. Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *J. ACM*, 52(5):749–765, 2005. doi:10.1145/1089023.1089025.
2. Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 99–108, 1996. doi:10.1145/237814.237838.
3. 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.
4. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-Way Functions and a Conditional Variant of MKTP. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 7:1–7:19, 2021. doi:10.4230/LIPIcs.FSTTCS.2021.7.

5 Eric Allender, John Gouwar, Shuichi Hirahara, and Caleb Robelle. Cryptographic Hardness Under Projections for Time-Bounded Kolmogorov Complexity. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, pages 54:1–54:17, 2021. doi:10.4230/LIPIcs.ISAAC.2021.54.

6 Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum Circuit Size, Graph Isomorphism, and Related Problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018. doi:10.1137/17M1157970.

7 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.

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. doi:10.1016/j.jcss.2010.06.004.

9 Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Computational depth: Concept and applications. *Theor. Comput. Sci.*, 354(3):391–404, 2006. doi:10.1016/j.tcs.2005.11.033.

10 Luis Antunes, Sophie Laplante, Alexandre Pinto, and Liliana C. M. Salvador. Cryptographic Security of Individual Instances. In *Proceedings of Information Theoretic Security (ICITS)*, pages 195–210, 2007. doi:10.1007/978-3-642-10230-1_17.

11 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-Case Running Times for Average-Case Algorithms. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 298–303, 2009. doi:10.1109/CCC.2009.12.

12 Amos Beimel. Secret-Sharing Schemes: A Survey. In *The Third International Workshop on Coding and Cryptology (IWCC)*, pages 11–46, 2011. doi:10.1007/978-3-642-20901-7_2.

13 Josh Cohen Benaloh and Jerry Leichter. Generalized Secret Sharing and Monotone Functions. In *Proceedings of the International Cryptology Conference (CRYPTO)*, pages 27–35, 1988. doi:10.1007/0-387-34799-2_3.

14 George Robert Blakley. Safeguarding cryptographic keys. In *International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979. doi:10.1109/MARK.1979.8817296.

15 Manuel Blum and Sampath Kannan. Designing Programs that Check Their Work. *J. ACM*, 42(1):269–291, 1995. doi:10.1145/200836.200880.

16 Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.

17 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.

18 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.

19 Harry Buhrman and Elvira Mayordomo. An Excursion to the Kolmogorov Random Strings. *J. Comput. Syst. Sci.*, 54(3):393–399, 1997. doi:10.1006/jcss.1997.1484.

20 Lijie Chen, Shuichi Hirahara, and Neekon Vafa. Average-case Hardness of NP and PH from Worst-case Fine-grained Assumptions. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 67:1–67:17, 2022.

21 Irit Dinur, Prahladh Harsha, and Guy Kindler. Polynomially Low Error PCPs with polyloglog n Queries via Modular Composition. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 267–276, 2015. doi:10.1145/2746539.2746630.

22 Irit Dinur and Shmuel Safra. On the hardness of approximating label-cover. *Inf. Process. Lett.*, 89(5):247–254, 2004. doi:10.1016/j.ipl.2003.11.007.

23 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.

24 Lance Fortnow and Martin Kummer. On Resource-Bounded Instance Complexity. *Theor. Comput. Sci.*, 161(1&2):123–140, 1996. doi:10.1016/0304-3975(95)00097-6.

25 Lance Fortnow, Troy Lee, and Nikolai K. Vereshchagin. Kolmogorov Complexity with Error. In *STACS 2006, 23rd Annual Symposium on Theoretical Aspects of Computer Science, Marseille, France, February 23-25, 2006, Proceedings*, pages 137–148, 2006. doi:10.1007/11672142_10.

26 Lance Fortnow and Nick Reingold. PP is Closed Under Truth-Table Reductions. *Inf. Comput.*, 124(1):1–6, 1996. doi:10.1006/inco.1996.0001.

27 Eran Gat and Shafi Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *Electron. Colloquium Comput. Complex.*, page 136, 2011. URL: https://eccc.weizmann.ac.il/report/2011/136.

28 Halley Goldberg and Valentine Kabanets. A Simpler Proof of the Worst-Case to Average-Case Reduction for Polynomial Hierarchy via Symmetry of Information. *Electronic Colloquium on Computational Complexity (ECCC)*, 007, 2022.

29 Oded Goldreich and Shafi Goldwasser. On the Limits of Nonapproximability of Lattice Problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000. doi:10.1006/jcss.1999.1686.

30 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. doi:10.4086/toc.2012.v008a023.

31 Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

32 Shuichi Hirahara. Characterizing Average-Case Complexity of PH by Worst-Case Meta-Complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 50–60, 2020.

33 Shuichi Hirahara. Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 20:1–20:47, 2020. doi:10.4230/LIPIcs.CCC.2020.20.

34 Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1038–1051, 2020. doi:10.1145/3357713.3384251.

35 Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 292–302, 2021. doi:10.1145/3406325.3451065.

36 Shuichi Hirahara and Mikito Nanashima. On Worst-Case Learning in Relativized Heuristica. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2021.

37 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018. doi:10.4230/LIPIcs.CCC.2018.5.

38 Shuichi Hirahara and Rahul Santhanam. Errorless versus Error-prone Average-case Complexity. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 38:1–38:23, 2022.

39 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

40 Rahul Ilango. Approaching MCSP from Above and Below: Hardness for a Conditional Variant and $\mathsf{AC}^0[p]$. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 34:1–34:26, 2020. doi:10.4230/LIPIcs.ITCS.2020.34.

41 Rahul Ilango. Constant Depth Formula and Partial Function Versions of MCSP are Hard. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2020.

42 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-Hardness of Circuit Minimization for Multi-Output Functions. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 22:1–22:36, 2020. doi:10.4230/LIPIcs.CCC.2020.22.

43 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.

44 Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011. doi:10.1109/CCC.2011.34.

45 Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990. doi:10.1109/FSCS.1990.89604.

46 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997. doi:10.1145/258533.258590.

47 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.

48 Tarik Kaced. Almost-perfect secret sharing. In *Proceedings of the International Symposium on Information Theory (ISIT)*, pages 1603–1607, 2011. doi:10.1109/ISIT.2011.6033816.

49 Mauricio Karchmer and Avi Wigderson. On Span Programs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 102–111, 1993. doi:10.1109/SCT.1993.336536.

50 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.

51 Andrei N Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

52 Troy Lee and Andrei E. Romashchenko. Resource bounded symmetry of information revisited. *Theor. Comput. Sci.*, 345(2-3):386–405, 2005. doi:10.1016/j.tcs.2005.07.017.

53 Leonid A. Levin. The Tale of One-Way Functions. *Probl. Inf. Transm.*, 39(1):92–103, 2003. doi:10.1023/A:1023634616182.

54 Leonid A. Levin. How do we succeed in tasks like proving Fermat's Theorem or predicting the Higgs boson?, 2021. An invited talk given at STOC 2021. The transcript is available at https://www.cs.bu.edu/fac/lnd/expo/stoc21/txt.pdf.

55 Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Texts in Computer Science. Springer, 2008. doi:10.1007/978-0-387-49820-1.

56 Yanyi Liu and Rafael Pass. On One-way Functions from NP-Complete Problems. *Electron. Colloquium Comput. Complex.*, 28:59, 2021. URL: https://eccc.weizmann.ac.il/report/2021/059.

57 Luc Longpré and Sarah Mocas. Symmetry of Information and One-Way Functions. *Inf. Process. Lett.*, 46(2):95–100, 1993. doi:10.1016/0020-0190(93)90204-M.

58 Luc Longpré and Osamu Watanabe. On Symmetry of Information and Polynomial Time Invertibility. *Inf. Comput.*, 121(1):14–22, 1995. doi:10.1006/inco.1995.1120.

59 Zhenjian Lu and Igor Carboni Oliveira. An Efficient Coding Theorem via Probabilistic Representations and Its Applications. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 94:1–94:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.94.

60 Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007. doi:10.1137/S0097539705447360.

61 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.

62 Igor Carboni Oliveira. Randomness and Intractability in Kolmogorov Complexity. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 32:1–32:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.32.

63 Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 665–677, 2017. doi:10.1145/3055399.3055500.

64 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.

65 Hanlin Ren and Rahul Santhanam. Hardness of KT Characterizes Parallel Cryptography. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 35:1–35:58, 2021. doi:10.4230/LIPIcs.CCC.2021.35.

66 Detlef Ronneburger. *Kolmogorov Complexity and Derandomization*. PhD thesis, Rutgers UniversityDept. of Computer Science Laboratory for Computer Sci. Research Hill Center, NJ, USA, 2004.

67 Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

68 Amnon Ta-Shma, Christopher Umans, and David Zuckerman. Lossless Condensers, Unbalanced Expanders, And Extractors. *Combinatorica*, 27(2):213–240, 2007. doi:10.1007/s00493-007-0053-2.

69 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.

70 Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001. doi:10.1145/502090.502099.

71 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.

72 Emanuele Viola. On Constructing Parallel Pseudorandom Generators from One-Way Functions. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 183–197, 2005. doi:10.1109/CCC.2005.16.

73 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.

74 AK Zvonkin and LA Levin. The complexity of finite objects and the algorithmic concepts of randomness and information. *UMN (Russian Math. Surveys)*, 25(6):83–124, 1970.

A Symmetry of Information from Weak Symmetry of Information

In this appendix, we present a simple proof of SoI based on weak symmetry of information. Weak symmetry of information is formally stated as follows:

▶ **Lemma A.1** (Weak Symmetry of Information [35]). *If* $\mathrm{GapMINKT} \in \mathsf{P}$ *and* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$*, then there exists a polynomial* p *such that, for any* $n, m \in \mathbb{N}$*, any* $t \geq n + m$*, any* $\epsilon > 0$*, and any* $x \in \{0,1\}^n$*,*

$$\Pr_{w \sim \{0,1\}^m}\left[\mathrm{K}^t(x,w) \geq \mathrm{K}^{p(t/\epsilon)}(x) + m - \log p(t/\epsilon)\right] \geq 1 - \epsilon.$$

Alternative Proof of Theorem 4.1. Let M be an algorithm for $\mathrm{Gap}_\tau\mathrm{MINKT}$, where τ is some polynomial. Fix $x, y \in \{0,1\}^*$, and $t \in \mathbb{N}$ such that $t \geq |x| + |y|$. Let $n := |x|$. We prove SoI by analyzing the behavior of

$$M(\mathrm{DP}_k(x;z) \cdot y, 1^{t'}, 1^s)$$

over a random choice of $z \sim \{0,1\}^{nk}$, where k, s and t' are parameters chosen later. To this end, we first compare the time-bounded Kolmogorov complexity of $\mathrm{DP}_k(x;z) \cdot y$ with that of $w \cdot y$ for random choices of $z \sim \{0,1\}^{nk}$ and $w \sim \{0,1\}^{nk+k}$: On one hand, since $\mathrm{DP}_k(x;z) \cdot y$ can be described by z, $k \leq O(t)$, and a program that outputs (x,y), we have

$$\mathrm{K}^{t'}(\mathrm{DP}_k(x;z) \cdot y) \leq \mathrm{K}^t(x,y) + |z| + \log t', \tag{23}$$

where $t' = \mathsf{poly}(t)$ is some polynomial. On the other hand, by Lemma A.1, there exists a polynomial p_0 such that

$$\mathrm{K}^{t'}(w \cdot y) \geq \mathrm{K}^{p_0(t')}(y) + |w| - \log p_0(t') \tag{24}$$

with probability at least $\frac{1}{2}$ over a random choice of $w \sim \{0,1\}^{nk+k}$.

Comparing Equations (23) and (24), when k is sufficiently large, the output distribution of $\mathrm{DP}_k(x;\text{-})$ can be distinguished from the uniform distribution by using the algorithm M. In more detail, let $s := \mathrm{K}^t(x,y) + |z| + \log t'$, which is the right-hand side of Equation (23). Let $k := \mathrm{K}^t(x,y) - \mathrm{K}^{p_0(t')}(y) + \log p_0(t') + \log t' + \log \tau(n',t') + 1$ so that the right-hand side of Equation (24) is greater than $s + \log \tau(n',t')$, where $n' := |w| + |y|$. Then, Equation (23) implies that $\mathrm{DP}_k(x;z) \cdot y$ is a YES instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$; thus, we obtain

$$\Pr_z\left[M(\mathrm{DP}_k(x;z) \cdot y, 1^{t'}, 1^s) = 1\right] = 1.$$

Equation (24) implies that $w \cdot y$ is a NO instance of $\mathrm{Gap}_\tau\mathrm{MINKT}$ with probability at least $\frac{1}{2}$; thus, we obtain

$$\Pr_w\left[M(w \cdot y, 1^{t'}, 1^s) = 1\right] \leq \frac{1}{2}.$$

Define a circuit D_y so that $D_y(w) := M(w \cdot y, 1^{t'}, 1^s)$ for every input $w \in \{0,1\}^{nk+k}$; then, it follows that

$$\Pr_z\left[D_y(\mathrm{DP}_k(x;z)) = 1\right] - \Pr_w\left[D_y(w) = 1\right] \geq \frac{1}{2},$$

which, by Lemma 2.2, implies that

$$\mathrm{K}^{p_1(t)}(x \mid D_y) \leq k + \log p_1(t)$$

for some polynomial p_1. Finally, observe that the circuit D_y can be described by using $y \in \{0,1\}^*$, n, k, t', and $s \in \mathbb{N}$ as well as an $O(1)$-size program for M; therefore, we obtain

$$\mathrm{K}^{p_1(t)}(x \mid y) \leq \mathrm{K}^t(x \mid D_y) + O(\log t) \leq k + O(\log t) \leq \mathrm{K}^t(x,y) - \mathrm{K}^{p_0(t')}(y) + O(\log t).$$

By choosing a large enough polynomial p, it follows that

$$\mathrm{K}^{p(t)}(x \mid y) + \mathrm{K}^{p(t)}(y) \leq \mathrm{K}^t(x,y) + \log p(t). \qquad \blacktriangleleft$$

A.1 Why Was Symmetry of Information Not Proved Before?

The alternative proof of Theorem 1.2 is reminiscent of the lemma of [35] that constructs a universal heuristic scheme from an algorithm for $\mathrm{Gap}(\mathrm{K}^{\mathsf{PH}}$ vs $\mathrm{K})$. In retrospect, the proof techniques for Theorem 1.2 were already developed in [35]. It is natural to ask why Theorem 1.2 was not proved in [35]. The reason is that previous results suggested the infeasibility of proving Theorem 1.2, as we explain below.

Weak symmetry of information is implicitly proved in [33] under the strong assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. Specifically, [33] showed that if $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, then

$$\mathrm{K}^{\mathsf{poly}(t)}(x) \leq \mathrm{K}^{t,\mathsf{PH}}(x) + O(\log t)$$

for every $x \in \{0,1\}^*$ and every $t \geq |x|$.[20] That is, the PH-oracle time-bounded Kolmogorov complexity of x is approximately equal to $\mathrm{K}^t(x)$. By using the Kolmogorov–Levin proof of symmetry of information (as in [58]), it can be shown that

$$\mathrm{K}^{\mathsf{poly}(t),\mathsf{PH}}(y \mid x) + \mathrm{K}^{\mathsf{poly}(t),\mathsf{PH}}(x) \leq \mathrm{K}^t(x,y) + O(\log t).$$

Now, for a random $y \sim \{0,1\}^m$, we have $\mathrm{K}^{\mathsf{poly}(t),\mathsf{PH}}(y \mid x) \approx |y|$ by Fact 3.2. Therefore, we obtain weak symmetry of information:

$$|y| + \mathrm{K}^{\mathsf{poly}(t)}(x) \leq \mathrm{K}^t(x,y) + O(\log t)$$

holds with high probability over a random choice of $y \sim \{0,1\}^m$.

One may be tempted to try to prove symmetry of information by extending the result of [33] to

$$\mathrm{K}^{\mathsf{poly}(t)}(x \mid y) \leq \mathrm{K}^{t,\mathsf{PH}}(x \mid y) + O(\log t) \tag{25}$$

under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$. However, this statement is in fact equivalent to the equivalence between the average-case easiness of PH and the worst-case easiness of PH (i.e., $\mathsf{P} = \mathsf{PH} \iff \mathsf{DistPH} \subseteq \mathsf{AvgP}$) [52, 24], which is a long-standing open question and cannot be proved by relativizing proof techniques [44, 36]. These results suggest the difficulty of proving symmetry of information because it relates conditional Kolmogorov complexity and unconditional Kolmogorov complexity, which previously seemed to imply Equation (25).

[35] proved $\mathsf{DistPH} \subseteq \mathsf{AvgP} \implies \mathsf{PH} \subseteq \mathsf{DTIME}(2^{O(n/\log n)})$, which is the first non-trivial worst-case-to-average-case connection of PH. [35] also proved weak symmetry of information under the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ than [33]. What was overlooked in [35] is that symmetry of information does not necessarily imply Equation (25). In fact, under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, SoI implies a weaker statement that

$$\mathrm{K}^{\mathsf{poly}(t)}(x \mid y) \leq \mathrm{K}^{t,\mathsf{PH}}(x \mid y) + \mathrm{cd}^{t,\mathsf{poly}(t)}(y) + O(\log t). \tag{26}$$

This statement looks quite similar to Equation (25), especially because the computational depth of y is small for most strings y (Lemma 8.4). Although Equation (26) is not sufficient to obtain $\mathsf{P} = \mathsf{PH}$, it does suffice to prove $\mathsf{PH} \subseteq \mathsf{DTIME}(2^{O(n/\log n)})$, which provides the alternative proof of the main results of [35] presented in Section 7. The fact that SoI provides the worst-case-to-average-case connections indicates the importance of SoI in average-case complexity theory.

[20] $\mathrm{K}^{t,\mathsf{PH}}(x)$ is an informal notation that represents the A-oracle t-time-bounded Kolmogorov complexity of x for an oracle $A \in \mathsf{PH}$. The statement is true for every oracle $A \in \mathsf{PH}$.

Pseudorandomness of Expander Random Walks for Symmetric Functions and Permutation Branching Programs

Louis Golowich
Harvard University, Cambridge, MA, USA

Salil Vadhan
Harvard University, Cambridge, MA, USA

Abstract

We study the pseudorandomness of random walks on expander graphs against tests computed by symmetric functions and permutation branching programs. These questions are motivated by applications of expander walks in the coding theory and derandomization literatures. A line of prior work has shown that random walks on expanders with second largest eigenvalue λ fool symmetric functions up to a $O(\lambda)$ error in total variation distance, but only for the case where the vertices are labeled with symbols from a binary alphabet, and with a suboptimal dependence on the bias of the labeling. We generalize these results to labelings with an arbitrary alphabet, and for the case of binary labelings we achieve an optimal dependence on the labeling bias. We extend our analysis to unify it with and strengthen the expander-walk Chernoff bound. We then show that expander walks fool permutation branching programs up to a $O(\lambda)$ error in ℓ_2-distance, and we prove that much stronger bounds hold for programs with a certain structure. We also prove lower bounds to show that our results are tight. To prove our results for symmetric functions, we analyze the Fourier coefficients of the relevant distributions using linear-algebraic techniques. Our analysis for permutation branching programs is likewise linear-algebraic in nature, but also makes use of the recently introduced singular-value approximation notion for matrices (Ahmadinejad et al. 2021).

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Expander graph, Random walk, Pseudorandomness

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.27

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2022/024/` [9]

Funding *Louis Golowich*: Supported by Harvard College Herchel Smith Fellowship.
Salil Vadhan: Supported by NSF grant CCF-1763299 and a Simons Investigator Award.

Acknowledgements S.V. thanks Dean Doron, Raghu Meka, Omer Reingold, and Avishay Tal, conversations with whom inspired some of this research. We also thank the anonymous CCC reviewers for helpful comments that have improved the presentation.

1 Introduction

Random walks on expander graphs have numerous applications in computer science due to their pseudorandom properties (see e.g. [13] for a survey). Typically, an expander random walk is used to provide a randomness-efficient means for generating a sequence of vertices $v_0, \ldots, v_{t-1}$. In a given application, this expander walk will be used to "fool" certain desired test functions f, in the sense that the distribution of $f(v_0, \ldots, v_{t-1})$ is approximately the same whether the vertices $v_0, \ldots, v_{t-1}$ are sampled from a random walk on an expander, or independently and uniformly at random (which is equivalent to using a random walk on a complete graph with self loops). In this paper, we prove tight bounds on the extent to which expander graph random walks fool certain functions f of interest, namely, symmetric

functions as well as functions computable by permutation branching programs. These results improve on a recent line of work [11, 6, 5]. Our results also yield further implications, including a strengthening of the expander-walk Chernoff bound [7, 12].

An expander graph is a graph that is sparse but well connected. In this paper we consider regular λ-spectral expanders, which are constant-degree graphs for which all nontrivial eigenvalues of the random walk matrix have absolute value at most λ. Intuitively, the spectrum of an expander graph approximates that of the complete graph, so an expander provides a sparsification of the complete graph. Random walks on expander graphs therefore provide a derandomized approximation for random walks on complete graphs. A major aim of this paper is to obtain tight bounds on the error in this approximation.

Many explicit constructions of λ-spectral expanders are known for arbitrarily small $\lambda > 0$ (e.g. [18, 17, 19, 4]). Random walks on such expanders have many applications, such as in randomness-efficient error reduction, error-correcting codes, and small-space derandomization (see the surveys [13, 21, 10]). Randomness-efficient error reduction uses the ability of expander random walks to fool threshold functions, while Ta-Shma's recent breakthrough construction of ϵ-balanced codes [20] uses their ability to fool the parity function. Meanwhile, work on small-space derandomization starting from [15] uses the ability of expander walks to fool branching programs. In this paper, we prove new bounds on the extent to which expander walks fool symmetric functions (which include the threshold and parity functions), as well as (permutation) branching programs.

Specifically, we strengthen and generalize a result of Cohen et al. [5], which shows that a random walk on a sequence of λ-spectral expanders fools symmetric functions up to a $O(\lambda)$ error in total variation distance. Our result extends the result of Cohen et al. [5] to labelings of the vertices by symbols from an arbitrary alphabet and, in the binary case, achieves the optimal dependence on the bias of the labeling; the Cohen et al. [5] result only applies to binary labelings, and has a suboptimal dependence on the labeling bias. We also unite this total variation bound with a tail bound, which yields a strengthening of the expander-walk Chernoff bound. We furthermore show that expander random walks fool width-w permutation branching programs up to a $O(\lambda)$ error in ℓ_2-distance and a $O(\sqrt{w} \cdot \lambda)$ error in total variation distance, which extends a result of [2, 14] to walks of length > 2, and also strengthens the $O(w^4 \cdot \sqrt{\lambda})$ total variation bound of Cohen et al. [6]. For programs possessing a certain structure, we prove much stronger bounds. We also present several lower bounds that show our upper bounds to be tight.

The organization of the remainder of this extended abstract is as follows. Section 2 describes the main problem we consider, and introduces notation. Section 3 describes our contributions. We present proof outlines of our results for symmetric functions and for permutation branching programs in Section 4 and Section 5 respectively. For complete proofs of all results, the reader is referred to the full version of this paper [9].

2 Problem overview

For a sequence $\mathcal{G} = (G_1, \ldots, G_{t-1})$ of graphs on a shared vertex set V, let $\mathrm{RW}^t_{\mathcal{G}}$ denote the random variable taking values in V^t that is given by taking a length-t random walk on V, where the ith step is taken in the graph G_i. If all $G_i = G$ then we write $\mathrm{RW}^t_{\mathcal{G}} = \mathrm{RW}^t_G$.

For some fixed integer $d \geq 2$, we are given a labeling $\mathrm{val} : V \to [d] = \{0, \ldots, d-1\}$, which we extend to act on sequences componentwise, that is, $\mathrm{val}(v_0, \ldots, v_{t-1}) = (\mathrm{val}(v_0), \ldots, \mathrm{val}(v_{t-1}))$. We let the tuple $p = (p_0, \ldots, p_{d-1}) \in [0,1]^d$ specify the weights of the labels, so that p_b equals the fraction of vertices with label $b \in [d]$.

In this paper, we study the distribution of $\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}})$ for a sequence $\mathcal{G}$ of λ-spectral expanders. In particular, letting J denote the complete graph with self-loops, we will compare the distributions of $f(\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}))$ and $f(\mathrm{val}(\mathrm{RW}^t_J))$ for certain test functions f on $[d]^t$. Specifically, we study functions f that are either symmetric or computable by a permutation branching program.

Let $\Sigma : [d]^{[t]} \to [t+1]^{[d]}$ be the histogram function, so that $(\Sigma a)_b = |\{i \in [t] : a_i = b\}|$ denotes the number of copies of b in the sequence a. All symmetric functions factor through Σ, so to study symmetric functions we restrict attention to Σ.

3 Contributions

This section describes our main results. The reader is referred to the full version [9] for theorem statements containing explicit constants.

3.1 Symmetric functions

A major objective of this paper is to study the extent to which expander walks fool symmetric functions. In our notation, for a sequence $\mathcal{G}$ of λ-spectral expanders, we would like to bound the distance between the distributions of $\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}})$ and $\Sigma\,\mathrm{val}(\mathrm{RW}^t_J)$ as a function of λ, regardless of the choice of $\mathcal{G}$. Rather than directly comparing these distributions, in the following theorem we bound the change in $\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}})$ when one of the graphs G_u in the sequence $\mathcal{G}$ is changed. We then apply a hybrid argument by changing the graphs in $\mathcal{G}$ to J one at a time.

Thus the consideration of arbitrary expander sequences $\mathcal{G}$ is inherent in our proof. Yet as a side benefit, we are able to show fine-grained bounds on the distance between $\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}})$ and $\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}'})$ when $\mathcal{G}$ and $\mathcal{G}'$ only differ at a few steps. Such bounds are used in a follow-up work [8] to prove a new Berry-Esseen theorem for expander walks.

The following theorem considers the case of $d = 2$ possible labels; we will subsequently show a similar result for $d > 2$. In a slight abuse of notation below, we let G both denote a graph and its random walk matrix. We use $\|\cdot\|$ to denote the spectral norm of a matrix.

▶ **Theorem 1.** *Fix positive integers $u < t$. Let $\mathcal{G} = (G_i)_{1\le i\le t-1}$ and $\mathcal{G}' = (G'_i)_{1\le i\le t-1}$ be sequences of regular $1/100$-spectral expanders on a shared vertex set V such that $G_i = G'_i$ for all $i \neq u$. Fix a labeling $\mathrm{val} : V \to [2]$ that assigns each label $b \in [2]$ to p_b-fraction of the vertices. Then for every $c \ge 0$,*

$$\sum_{j\in[t+1]:|j-p_1 t|\ge c} \left|\Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}'}) = (t-j,j)] - \Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}) = (t-j,j)]\right| = O\left(\frac{\|G'_u - G_u\| \cdot e^{-c^2/8t}}{t}\right).$$

Theorem 1 bounds the change in the distribution of $\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}})$ when the graph at a single step in $\mathcal{G}$ is changed. A key point is that the bound decays linearly in t. That is, the longer the walk, the less effect changing one of the graphs has. By changing all the graphs to the complete graph with self loops J one step at a time, we obtain the following corollary.

▶ **Corollary 2.** *For all positive integers t and all $0 \le \lambda \le 1/100$, let $\mathcal{G} = (G_i)_{1\le i\le t-1}$ be a sequence of regular λ-spectral expanders on a shared vertex set V with labeling $\mathrm{val} : V \to [2]$. Then for every $c \ge 0$,*

$$\sum_{j\in[t+1]:|j-p_1 t|\ge c} \left|\Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}) = (t-j,j)] - \Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{J}) = (t-j,j)]\right| = O(\lambda \cdot e^{-c^2/8t}).$$

The bounds in Theorem 1 and Corollary 2 both provide unified bounds for two different notions of distance, namely total variation distance and tail bounds. Specifically, when $c = 0$ then the results above bound total variation distance, while as c grows large they provide tail bounds, as $p_1 t$ is the expected value of $(\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t))_1$.

Both the total variation and tail bounds above are novel, to the best of our knowledge. Our tails bounds can be viewed as strengthening the expander-walk Chernoff bound [7, 12], and indeed our proof of Theorem 1 draws on similar techniques as used in Healy's [12] proof of the expander-walk Chernoff bound. Recall that for a sequence $\mathcal{G}$ of λ-spectral expanders with λ bounded away from 1, the expander-walk Chernoff bound states that

$$\sum_{j \in [t+1] : |j - p_1 t| \geq c} \Pr[\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t) = (t-j, j)] = O(e^{-\Omega(c^2/t)}),$$

that is, the tails of $\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)$ decay approximately as quickly as the tails of the binomial distribution as $c^2/t \to \infty$. Corollary 2 shows the stronger statement that as $\lambda \to 0$, the tails of $\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)$ converge to the tails of the binomial distribution $\Sigma \operatorname{val}(\mathrm{RW}_J^t)$, even when $c^2/t = O(1)$.

The $c = 0$ case of Corollary 2 shows a $O(\lambda)$ bound on the total variation distance between $\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)$ and $\Sigma \operatorname{val}(\mathrm{RW}_J^t)$. Equivalently, this result shows that every symmetric function $f : \{0,1\}^t \to \{0,1\}$ satisfies $|\mathbb{E}[f(\operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t))] - \mathbb{E}[f(\operatorname{val}(\mathrm{RW}_J^t))]| = O(\lambda)$, that is, random walks on λ-spectral expanders $O(\lambda)$-fool symmetric functions. This bound improves upon a line of prior work [11, 6, 5]. Guruswami and Kumar [11] initiated this line of work by showing a $O(\lambda)$ bound on the total variation distance between $\Sigma \operatorname{val}(\mathrm{RW}_G^t)$ and $\Sigma \operatorname{val}(\mathrm{RW}_J^t)$ for the special case where G is the 2-vertex sticky random walk. Cohen et al. [6] then showed a $O(\lambda(\log(1/\lambda))^{3/2})$ bound on this total variation distance for arbitrary expanders G with a balanced labeling, that is, when $p_0 = p_1 = 1/2$. A follow-up paper of Cohen et al. [5] generalized to arbitrary p, and improved the total variation distance bound to $O(\lambda/\sqrt{\min(p)})$, where $\min(p) = \min\{p_0, p_1\}$. In contrast, the $c = 0$ case of Corollary 2 strengthens this bound to $O(\lambda)$ regardless of p. Our results also allow for sequences $\mathcal{G}$ of λ-spectral expanders with different graphs at different steps, whereas the prior work [6, 5] assumed that the graph was the same at each step.

Theorem 1, Corollary 2, and all of the prior work [11, 6, 5] assumes a binary labeling $\operatorname{val} : V \to \{0,1\}$ on the expander graph's vertices. Jalan and Moshkovitz [16] asked whether these results generalize to labelings $\operatorname{val} : V \to [d]$ for $d > 2$. We provide an affirmative answer to this question in the following results, which generalizing the total variation distance bounds in Theorem 1 and Corollary 2 to arbitrary $d \geq 2$. Below, we let $\min(p) = \min_{b \in [d]} p_b$.

▶ **Theorem 3.** *For every integer $d \geq 2$ and every distribution $p \in [0,1]^d$ over the labels $[d]$, there exists a constant $\lambda_0 = \lambda_0(d, p) > 0$ such that the following holds. For all positive integers $u < t$, let $\mathcal{G} = (G_i)_{1 \leq i \leq t-1}$ and $\mathcal{G}' = (G'_i)_{1 \leq i \leq t-1}$ be sequences of λ_0-spectral expanders on a shared vertex set V, such that for all $i \neq u$ we have $G_i = G'_i$. Let $\operatorname{val} : V \to [d]$ be any labeling that assigns each label $b \in [d]$ to p_b-fraction of the vertices. Then*

$$d_{TV}\left(\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}'}^t), \Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)\right) = O\left(\left(\frac{d}{\min(p)}\right)^{O(d)} \cdot \frac{\|G'_u - G_u\|}{t}\right).$$

▶ **Corollary 4.** *For all integers $t \geq 1$ and $d \geq 2$, let $\mathcal{G} = (G_i)_{1 \leq i \leq t-1}$ be a sequence of λ-spectral expanders on a shared vertex set V with labeling $\mathrm{val} : V \to [d]$ that assigns each label $b \in [d]$ to p_b-fraction of the vertices. Then*

$$d_{TV}\left(\Sigma \,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}), \Sigma \,\mathrm{val}(\mathrm{RW}^t_J)\right) = O\left(\left(\frac{d}{\min(p)}\right)^{O(d)} \cdot \lambda\right).$$

In the results above, it is helpful to think of d and p as fixed, so that Corollary 4 gives a $O(\lambda)$ bound on total variation distance. When $d = 2$, Theorem 1 and Corollary 2 with $c = 0$ show that the factor of $(d/\min(p))^{O(d)}$ in the bounds above can be removed. We suspect that this $(d/\min(p))^{O(d)}$ dependence for $d > 2$ is not tight, and we leave the determination of the optimal dependence on d and p as an open question.

To show that the $O(\lambda)$ upper bounds on total variation distance described above are tight, we present the following lower bound.

▶ **Theorem 5.** *For every $0 < \lambda < 1$ and $p = (p_0, p_1)$, there exists a sufficiently large $t_0 = t_0(p, \lambda) \in \mathbb{N}$ and a λ-spectral expander $G = G_{\lambda,p}$ with vertex labeling $\mathrm{val} : V \to [2]$ that has label weights given by p, such that for every $t \geq t_0$,*

$$d_{TV}(\Sigma \,\mathrm{val}(\mathrm{RW}^t_G), \Sigma \,\mathrm{val}(\mathrm{RW}^t_J)) = \Omega(\lambda).$$

Theorem 5 generalizes a similar result of Guruswami and Kumar [11] for the special case of $p_0 = p_1 = 1/2$, and indeed our proof method is similar to theirs. Cohen et al. [5] showed a similar $\Omega(\lambda)$ lower bound for all t but only when $p_0 = p_1 = 1/2$. Their result is incomparable to ours, as Theorem 5 considers all p but only sufficiently large t.

The graph $G_{\lambda,p}$ in Proposition 8 is the λ-sticky, p-biased random walk, which generalizes the sticky walk studied by Guruswami and Kumar [11] for the case where $p = (1/2, 1/2)$. From a given vertex $v \in V$, with probability $1 - \lambda$ the sticky walk chooses the next vertex $v' \in V$ uniformly at random, and with probability λ it instead chooses a random v' that has the same label $\mathrm{val}(v') = \mathrm{val}(v)$. This sticky walk is in some sense a canonical λ-spectral expander, and arises in all of our lower bounds in this paper.

The main idea to prove Theorem 5 is that by the Markov chain CLT, as $t \to \infty$ then $((\Sigma \,\mathrm{val}(\mathrm{RW}^t_{G_{\lambda,p}}))_1 - p_1 t)/\sqrt{p_0 p_1 t}$ converges in distribution (that is, in Kolmogorov distance) to a normal distribution with variance $(1 + \lambda)/(1 - \lambda)$. In contrast, the CLT implies that the normalized binomial distribution $((\Sigma \,\mathrm{val}(\mathrm{RW}^t_J))_1 - p_1 t)/\sqrt{p_0 p_1 t}$ converges to a normal distribution with variance 1. Theorem 5 then follows because the distance between these two normals is $\Omega(\lambda)$. All the details are provided the full version [9].

3.2 Permutation branching programs

This section describes our main results on the extent to which expander walks fool permutation branching programs.

To begin, we recall the formal definition of a permutation branching program $\mathcal{B}$, which sequentially reads in inputs a_i and updates its internal state according to a permutation $B_i(a_i)$.

▶ **Definition 6.** *A **permutation branching program** $\mathcal{B}$ of length t, width w, and degree d is a collection of functions $B_i : [d] \times [w] \to [w]$ for $i \in [t]$ such that for $b \in [d]$, each restriction $B_i(b) = B_i|_{\{b\} \times [w]} : [w] \to [w]$ is a permutation. The program is said to **compute** the function $B : [d]^t \to [w]$ defined by*[1]

$$B(a) = (B_{t-1}(a_{t-1}) \circ \cdots \circ B_0(a_0))(0).$$

[1] Without loss of generality the initial state is assumed to be $0 \in [w]$.

We first present a bound that makes no assumptions on the structure of the program.

▶ **Theorem 7.** *For integers $t \geq 1$, $w \geq 2$, and $d \geq 2$, let G be a λ-spectral expander with $\lambda < .1$, and assign some vertex labeling* $\mathrm{val} : V \to [d]$*. Let $B : [d]^t \to [w]$ be computed by a permutation branching program $\mathcal{B}$ of length t, width w, and degree d. Then*

$$d_{\ell_2}(B(\mathrm{val}(\mathrm{RW}_G^t)), B(\mathrm{val}(\mathrm{RW}_J^t))) = O(\lambda).$$

Note that the bound in Theorem 7 has no dependence on the width w of the branching program, but only bounds ℓ_2 rather than total variation distance. Applying the Cauchy-Schwartz inequality to this ℓ_2-bound gives the total variation bound

$$d_{\mathrm{TV}}(B(\mathrm{val}(\mathrm{RW}_G^t)), B(\mathrm{val}(\mathrm{RW}_J^t))) = O(\sqrt{w} \cdot \lambda). \tag{1}$$

This bound improves upon the work of Cohen et al. [6], who showed a $O(w^4 \cdot \sqrt{\lambda})$ bound on $d_{\mathrm{TV}}(B(\mathrm{val}(\mathrm{RW}_G^t)), B(\mathrm{val}(\mathrm{RW}_J^t)))$ for the special case where $d = 2$ and $p_0 = p_1 = 1/2$.

Theorem 7 is closely related to the analysis of the Impagliazzo-Nisan-Wigderson [15] pseudorandom generator studied by Hoza et al. [14], which also uses expander walks to fool permutation branching programs. Both Theorem 7 and the results of Hoza et al. [14] are also proven using similar matrix approximation notions. However, Hoza et al. [14] consider many length-2 expander walks, whereas we consider a single longer walk.

The following lower bound shows that Theorem 7 is tight.

▶ **Proposition 8.** *For every $0 \leq \lambda \leq 1$ and every $p = (p_0, p_1)$, there exists a λ-spectral expander $G = G_{\lambda,p}$ with vertex labeling* $\mathrm{val} : V \to [2]$ *that assigns each label $b \in [2]$ to p_b-fraction of the vertices, such that the following hold:*

1. *There exists a permutation branching program $\mathcal{B}$ of length $t = 2$, width $w = 2$, and degree $d = 2$ such that*

$$\left|\Pr[B(\mathrm{val}(\mathrm{RW}_G^t)) = 0] - \Pr[B(\mathrm{val}(\mathrm{RW}_J^t)) = 0]\right| = 2p_0p_1\lambda.$$

2. *There exists a permutation branching program $\mathcal{B}$ of length $t = \lfloor 1/\min\{p_0, p_1\}\rfloor + 1$, width $w = t + 1$, and degree $d = 2$ such that*

$$\left|\Pr[B(\mathrm{val}(\mathrm{RW}_G^t)) = 0] - \Pr[B(\mathrm{val}(\mathrm{RW}_J^t)) = 0]\right| \geq \frac{\lambda}{2e^2}.$$

For these lower bounds, a smaller program length and width corresponds to a stronger result, as the length and width can be increased arbitrarily with padding. Proposition 8 implies a $\Omega(\lambda)$ lower bound for both the ℓ_2 and total variation distance between $B(\mathrm{val}(\mathrm{RW}_G^t))$ and $B(\mathrm{val}(\mathrm{RW}_J^t))$. This ℓ_2 lower bound meets the upper bound in Theorem 7. However, whereas the $\Omega(\lambda)$ total variation lower bound has no dependence on the program width w, the $O(\sqrt{w} \cdot \lambda)$ upper bound in (1) decays with w. It is an open question to resolve this gap.

The graph $G_{\lambda,p}$ in Proposition 8 is same the λ-sticky, p-biased random walk used to show Theorem 5, as described in Section 3.1. More details can be found in the full version [9].

Although Proposition 8 shows that Theorem 7 is tight in general, much stronger bounds hold for certain permutation branching programs.

▶ **Theorem 9.** *For integers $t \geq 1$, $w \geq 2$, and $d \geq 2$, let $\mathcal{G}$ be a sequence of λ-spectral expanders on a shared vertex set V with labeling* $\mathrm{val} : V \to [d]$*. Let $B^t : [d]^t \to [w]$ denote the sum modulo w, that is $B^t(a) = \sum_{i\in[t]} a_i \pmod w$. Then there exists a constant $c = c(d, w, p, \lambda) < 1$ such that*

$$d_{TV}(B^t(\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t)), B^t(\mathrm{val}(\mathrm{RW}_J^t))) \leq \sqrt{w} \cdot c^t.$$

That is, expander walks fool the small modular functions B^t, which are naturally computed by permutation branching programs, up to an exponentially small error. This result can be viewed as a generalization of the previously known fact that expander walks fool the parity function up to an exponentially small error, as can be recovered by letting $w = 2$ and $d = 2$ in Theorem 9. This fact that expander walks are good parity samplers played a pivotal role in Ta-Shma's breakthrough construction of almost optimal ϵ-balanced codes [20].

For arbitrary $w \geq 2$, Guruswami and Kumar [11] showed that the total variation distance between $B^t(\mathrm{val}(\mathrm{RW}^t_G))$ and $B^t(\mathrm{val}(\mathrm{RW}^t_J))$ is exponentially small in t when G is the 2-vertex sticky random walk. Theorem 9 generalizes this exponential decay bound to arbitrary expander walks.

Theorem 9 presents a particular class of permutation branching programs $\mathcal{B}$ for which $B(\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}))$ approaches a uniform distribution exponentially quickly. In the full version [9], we provide a more general class of such permutation branching programs $\mathcal{B}$, and deduce Theorem 9 as a special case. For illustrative purposes to avoid more cumbersome notation, we have omitted the more general case here.

4 Proof overview for symmetric functions

In this section, we outline the proof of Theorem 1, which contains many of the key technical insights in our paper. In particular, the proof of Theorem 3 follows the same general argument, so for the exposition in this section we focus on Theorem 1. All of the proof details can be found in the full version [9].

As in Theorem 1, for some $u < t$ let $\mathcal{G} = (G_i)_{1 \leq i \leq t-1}$ and $\mathcal{G}' = (G'_i)_{1 \leq i \leq t-1}$ be sequences of 1/100-spectral expanders that agree at all positions $i \neq u$, and again fix a vertex labeling $\mathrm{val} : V \to [2]$. Define $g \in [-1,1]^{[t+1]} \subseteq [-1,1]^{\mathbb{Z}}$ to be the difference between the probability mass functions of $(\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}'}))_1$ and $(\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}))_1$, that is,

$$g_j = \Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}'}) = (t-j, j)] - \Pr[\Sigma\,\mathrm{val}(\mathrm{RW}^t_{\mathcal{G}}) = (t-j, j)].$$

In this notation, the $c = 0$ case of Theorem 1 states that g has ℓ_1-norm $\|g\|_1 = O(\|G'_u - G_u\|/t)$, which is bounded by $O(\lambda/t)$ if G'_u and G_u are λ-spectral expanders.

We first show that the ℓ_2-norm of g satisfies

$$\|g\| = O\left(\frac{\|G'_u - G_u\|}{t} \cdot \frac{1}{(p_0 p_1 t)^{1/4}}\right). \tag{2}$$

The proof of this bound is sketched below in Section 4.1. We will then explain in Section 4.2 how to go from this ℓ_2-bound to the desired ℓ_1-bound. We compare our techniques to those of prior work in Section 4.3, and in particular we draw connections with Healy's proof of the expander-walk Chernoff bound [12].

4.1 Bounding the ℓ_2-distance $\|g\|$

In this section, we sketch the proof of the ℓ_2-bound (2). Because the Fourier transform preserves ℓ_2-norms, we will bound the ℓ_2-norm $\|\hat{g}\| = \|g\|$ of the Fourier transform $\hat{g}$ of g. Recall that here the Fourier transform is given by $\hat{g}(\theta) = \sum_{j \in \mathbb{Z}} e^{-i\theta j} g_j$, and has ℓ_2-norm $\|\hat{g}\| = \sqrt{\int_{\theta=-\pi}^{\pi} |\hat{g}(\theta)|^2 d\theta/2\pi}$.

To motivate this shift to the Fourier basis, recall that the Fourier transform interchanges convolution and multiplication, so that addition of independent random variables translates to multiplication of the Fourier transforms of their probability density functions (i.e. multiplication of their *characteristic functions*). Such products can be easier to analyze than convolutions, so the Fourier transform is a natural tool for analyzing sums of independent random variables, as is exemplified in proofs of the central limit theorem. Theorem 1 and Corollary 2 intuitively show that the expander walk distribution $(\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t))_1$ is close to the sum of independent variables, so it is also natural to analyze this distribution with the Fourier transform.

Whereas we apply the Fourier transform over the group $\mathbb{Z}$ to the random variable $\Sigma \operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)$ (which is distributed over $\mathbb{Z}$), the prior work of Cohen et al. [6] and Cohen et al. [5] applied the Fourier transform over the group $(\mathbb{Z}/2)^t$ to the random variable $\operatorname{val}(\mathrm{RW}_{\mathcal{G}}^t)$ (which is distributed over $\{0,1\}^t \cong (\mathbb{Z}/2)^t$). As described above, our approach seems well suited for symmetric functions, and it generalizes naturally to give Theorem 3 and Corollary 4 for alphabet sizes $d > 2$. In contrast, Cohen et al. [6] only consider $d = 2$, but they are able to apply their techniques to other classes of functions such as bounded-depth circuits, which we do not consider. More comparisons to prior techniques are provided in Section 4.3.

To begin, we express $\hat{g}(\theta)$ linear-algebraically. Specifically, let $\vec{1} = (1/\sqrt{|V|}, \dots, 1/\sqrt{|V|})$ denote the uniform unit vector, and define the diagonal matrix $P_\theta = \operatorname{diag}(x_\theta) \in \mathbb{C}^{V \times V}$, where $x_\theta \in \mathbb{C}^V$ is the vector with $(x_\theta)_v = e^{-i\theta(\operatorname{val}(v) - p_1)}$. Then it can be verified that

$$e^{i\theta p_1 t} \cdot \hat{g}(\theta) = \vec{1}^\top \left(\prod_{i=u+1}^{t} G_i P_\theta \right) (G'_u - G_u) \left(\prod_{i=0}^{u-1} P_\theta G_i \right) \vec{1},$$

where the products above multiply from right-to-left, and we take $G_0 = G_t = J$. This equality can be seen by expanding the right hand side above as a sum over all length-t walks $v_0, \dots, v_{t-1}$ on V. Therefore because $G'_u - G_u$ annihilates $\vec{1}$ from both sides, we have

$$|\hat{g}(\theta)| \le \left\| \left(\vec{1}^\top \left(\prod_{i=u+1}^{t} G_i P_\theta \right) \right)^\perp \right\| \cdot \|G'_u - G_u\| \cdot \left\| \left(\left(\prod_{i=0}^{u-1} P_\theta G_i \right) \vec{1} \right)^\perp \right\|, \tag{3}$$

where the notation $x^\perp$ denotes the projection of a vector x onto the orthogonal complement of $\vec{1}$. We will also use $x^\parallel$ to denote the projection of x onto $\vec{1}$.

We bound the rightmost factor above by induction on u. Splitting off a factor of $P_\theta G_{u-1}$ gives

$$\begin{aligned}
&\left\| \left(\left(\prod_{i=0}^{u-1} P_\theta G_i \right) \vec{1} \right)^\perp \right\| \\
&\quad \le \|(P_\theta \vec{1})^\perp\| \cdot \left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^\parallel \right\| + \|P_\theta\| \cdot \lambda(G_{u-1}) \cdot \left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^\perp \right\| \\
&\quad \le \|(P_\theta \vec{1})^\perp\| \cdot \left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^\parallel \right\| + \frac{1}{100} \cdot \left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^\perp \right\|,
\end{aligned} \tag{4}$$

where the last inequality follows because $\|P_\theta\| = 1$ and G_{u-1} is a 1/100-spectral expander. Thus if we can bound the first term on the right hand side of (4) by some $B(u)$ that decays less rapidly than 100^{-u} (i.e. $B(u) = \Theta(\beta^{-u})$ for $\beta < 100$), we can inductively bound the left

hand side by $B(u) + B(u-1)/100 + B(u-2)/100^2 + \cdots = O(B(u))$. Specifically, we will show this bound for $B(u) = \Theta(\sqrt{p_0 p_1} \cdot \theta \cdot e^{-\Omega(p_0 p_1 (u-1)\theta^2)})$. Intuitively, it suffices to bound what happens to the component parallel to $\vec{1}$, because the component orthogonal to $\vec{1}$ is shrunk by a factor of 100 with each application of $P_\theta G_i$.

Letting $F = J + (1/10)(I - J)$ be the matrix that preserves $\vec{1}$ and scales its orthogonal complement by $1/10$, then because by assumption all $i \neq u$ have $\lambda(G_i) \leq 1/100$, it follows that $\|F^{-1} G_i F^{-1}\| \leq 1$. Thus

$$\left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^{\|} \right\| = \left\| \vec{1}^\top \left(\prod_{i=0}^{u-2} F P_\theta F \cdot F^{-1} G_i F^{-1} \right) \vec{1} \right\| \leq \|F P_\theta F\|^{u-1}.$$

Next via some technical calculations, we show that for all $-\pi < \theta \leq \pi$,

$$\|(P_\theta \vec{1})^\perp\| = \frac{\|x_\theta^\perp\|}{\sqrt{|V|}} = \Theta(\sqrt{p_0 p_1} \cdot \theta). \tag{5}$$

For intuition, observe that if $p_0 p_1$ or θ equals 0, then all entries of x_θ are the same, so $x_\theta^\perp = 0$. Using (5), we also deduce that

$$\|F P_\theta F\| \leq 1 - \Omega((\|(P_\theta \vec{1})^\perp\|^2) = e^{-\Omega(p_0 p_1 \theta^2)}.$$

Here for intuition, as F is a 1/10-spectral expander, we should expect $\|F P_\theta F\|$ to be close to $\|J P_\theta J\| = \|(P_\theta \vec{1})^{\|}\| = \sqrt{1 - \|(P_\theta \vec{1})^\perp\|^2} = 1 - \Omega(\|(P_\theta \vec{1})^\perp\|^2)$. Thus (4) becomes

$$\left\| \left(\left(\prod_{i=0}^{u-1} P_\theta G_i \right) \vec{1} \right)^{\perp} \right\| \leq O\left(\sqrt{p_0 p_1} \cdot \theta \cdot e^{-\Omega(p_0 p_1 (u-1)\theta^2)} \right) + \frac{1}{100} \cdot \left\| \left(\left(\prod_{i=0}^{u-2} P_\theta G_i \right) \vec{1} \right)^{\perp} \right\|.$$

Recursively applying this inequality to bound the last term on its right hand side then gives

$$\left\| \left(\left(\prod_{i=0}^{u-1} P_\theta G_i \right) \vec{1} \right)^{\perp} \right\| = O\left(\sqrt{p_0 p_1} \cdot \theta \cdot e^{-\Omega(p_0 p_1 u \cdot \theta^2)} \right).$$

We now apply the above bound on $\|((\prod_{i=0}^{u-1} P_\theta G_i)\vec{1})^\perp\|$, along with an analogous bound on $\|(\vec{1}^\top (\prod_{i=u+1}^{t} G_i P_\theta))^\perp\|$, in (3) to give

$$|\hat{g}(\theta)| = O\left(p_0 p_1 \cdot \theta^2 \cdot e^{-\Omega(p_0 p_1 t \cdot \theta^2)} \cdot \|G_u' - G_u\| \right).$$

We then obtain the desired ℓ_2-bound (2) by squaring and integrating this bound with the substitution $q = c\sqrt{p_0 p_1 t} \cdot \theta$ for a sufficiently small constant $c > 0$:

$$\begin{aligned} \|g\| = \|\hat{g}\| &= O\left(p_0 p_1 \cdot \|G_u' - G_u\| \cdot \sqrt{\int_{-\pi}^{\pi} \theta^4 e^{-\Omega(p_0 p_1 t \cdot \theta^2)} \frac{d\theta}{2\pi}} \right) \\ &= O\left(\frac{\|G_u' - G_u\|}{t \cdot (p_0 p_1 t)^{1/4}} \cdot \sqrt{\int_{-\infty}^{\infty} q^4 e^{-q^2} dq} \right) \\ &= O\left(\frac{\|G_u' - G_u\|}{t \cdot (p_0 p_1 t)^{1/4}} \right). \end{aligned}$$

4.2 Going from an ℓ_2 to ℓ_1 bound

In this section, we show how to extend the techniques for bounding $\|g\|$ described above to bound $\|g\|_1$, and more generally to prove Theorem 1.

First observe that by the expander-walk Chernoff bound, $\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t)$ and $\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}'}^t)$ are mostly supported in an interval of length $\ell \approx O(\sqrt{t})$ about their mean. Applying the Cauchy-Schwartz inequality to (2) on this interval (which costs a factor of $\sqrt{\ell} \approx O(t^{1/4})$ to convert from ℓ_2 to ℓ_1), and the expander-walk Chernoff bound on the tails lying outside of the interval, yields a total variation bound of

$$\|g\|_1 = O\left(\frac{\|G'_u - G_u\|}{t} \cdot \left(\frac{\log(\|G'_u - G_u\|/t)}{p_0 p_1}\right)^{1/4}\right).$$

However, the above ℓ_1-bound does not help us prove Theorem 1 when c^2/t is large. Furthermore, even to prove the $c = 0$ case Theorem 1, we need to remove the factor $(\log(\|G'_u - G_u\|/t)/p_0p_1)^{1/4}$ from the bound above.

To obtain these improvements, we first generalize (2) to bound the ℓ_2-norm of the vector $g^{(sr)} = (e^{sr(j-p_1t)}g_j)_{j\in[t+1]}$ for $s = \pm 1$ and various values of $r \geq 0$. The proof of this bound on $\|g^{(sr)}\|$ for general r simply generalizes the argument presented in Section 4.1. The special case $r = 0$ recovers $g = g^{(0)}$, while when $r > 0$ then the sum of the elements of $g^{(sr)}$ equals the difference between the moment generating functions $\mathbb{E}[e^{sr((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t))_1 - p_1t)}]$ and $\mathbb{E}[e^{sr((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}'}^t))_1 - p_1t)}]$ that are used in the proofs of Chernoff bounds.

We then partition $[t+1] \subseteq \mathbb{Z}$ into intervals of length approximately $\sqrt{p_0p_1t}$, and we bound the ℓ_1-norm of g restricted to each interval by applying the Cauchy-Schwartz inequality with our ℓ_2-bound on $g^{(sr)}$ for appropriately chosen s, r. Summing these bounds over all intervals lying at least some distance c from p_1t yields Theorem 1.

Intuitively, as $\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t)$ and $\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}'}^t)$ have standard deviation $\Theta(\sqrt{p_0p_1t})$, we would expect these distributions to be somewhat evenly distributed across an interval of length $\sqrt{p_0p_1t}$. This is the regime where Cauchy-Schwartz is tight. Appropriately choosing s, r allows us to "isolate" a given length-$\sqrt{p_0p_1t}$ interval, by ensuring that the components of $g^{(sr)}$ in that interval dominate components outside that interval.

4.3 Comparison with techniques in prior work

Our techniques described above to prove Theorem 1 are closely related to Healy's [12] proof of the expander-walk Chernoff bound. In some sense, Healy's proof [12] makes up "half" of our proof: Healy's proof bounds the moment-generating function $\mathbb{E}[e^{sr((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t))_1 - p_1t)}]$, but does not bound the characteristic function $\mathbb{E}[e^{-i\theta((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t))_1 - p_1t)}]$ as described in Section 4.1 (as the Fourier coefficient $\hat{g}(\theta)$ by definition equals the difference between the characteristic functions of $(\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t))_1$ and $(\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}'}^t))_1$). Intuitively, our proof combines the moment generating and characteristic function bounds, as in order to bound $\|g^{(sr)}\|$, we bound the difference $e^{i\theta p_1 t} \cdot \hat{g}^{(sr)}(\theta)$ between the generating functions $\mathbb{E}[e^{(sr-i\theta)((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t))_1 - p_1t)}]$ and $\mathbb{E}[e^{(sr-i\theta)((\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}'}^t))_1 - p_1t)}]$.

Although Cohen et al. [6] and Cohen et al. [5] also studied the extent to which expander walks fool symmetric functions, their proofs are less similar to ours. Most notably, both of these papers use Fourier analysis over the group $(\mathbb{Z}/2\mathbb{Z})^t$ by viewing $\mathrm{val}(\mathrm{RW}_G^t)$ as a distribution on $(\mathbb{Z}/2\mathbb{Z})^t$. In contrast, we use Fourier analysis over $\mathbb{Z}^{d-1}$ by viewing $\Sigma\,\mathrm{val}(\mathrm{RW}_{\mathcal{G}}^t)$ as a distribution on $\mathbb{Z}^d$ (or $\mathbb{Z}^{d-1}$, if we drop the first component). This explains why our results generalize more naturally to the case $d > 2$, which is not considered in [6, 5]. We

could also do our analysis using discrete Fourier analysis over $(\mathbb{Z}/m)^{d-1}$ instead, for any $m \geq t$, but then the modulus m is superfluous (as it is cleaner to avoid modular reduction) and only makes the notation more cumbersome.

5 Proof overview for permutation branching programs

In this section, we outline the proof of Theorem 7, which uses singular-value approximations as described below. We do not outline the proof of our other results for permutation branching programs, specifically Theorem 9, and instead refer the reader to the full version [9], as this latter result uses techniques somwhat similar to those described above in Section 4.

5.1 Proof outline

We now describe the proof of Theorem 7. As in the theorem statement, for arbitrary integers $t \geq 1$, $w \geq 2$, and $d \geq 2$, let $\mathcal{B}$ be a permutation branching program of length t, width w, and degree d that computes some function $B : [d]^t \to [w]$. Let G be a λ-spectral expander with $\lambda < .1$, and assign some vertex labeling $\mathrm{val} : G \to [d]$. We again let $g \in [-1,1]^{[w]}$ denote the difference between the distributions $B(\mathrm{val}(\mathrm{RW}_G^t))$ and $B(\mathrm{val}(\mathrm{RW}_J^t))$ of interest, that is,

$$g_j = \Pr[B(\mathrm{val}(\mathrm{RW}_G^t)) = j] - \Pr[B(\mathrm{val}(\mathrm{RW}_J^t)) = j].$$

In this notation, Theorem 7 states that $\|g\| = O(\lambda)$.

As in the proof of Theorem 1, we begin by expressing g linear-algebraically. Let $\tilde{P}$ be the operator on the vector space $\mathbb{R}^V \otimes \mathbb{R}^t \otimes \mathbb{R}^w$ given by

$$\tilde{P} = \sum_{v \in V, i \in [t]} \delta_v \delta_v^\top \otimes \delta_{i+1} \delta_i^\top \otimes B_i(\mathrm{val}(v)),$$

where $i+1$ is taken $(\mathrm{mod}\ t)$ above, and by abuse of notation $B_i(\mathrm{val}(v)) \in \mathbb{R}^{w \times w}$ refers to the permutation matrix associated to the permutation $B_i(\mathrm{val}(v)) : [w] \to [w]$. Also for $W = G$ or J, let $\tilde{W} = W \otimes I \otimes I$. Then for every $j \in [w]$,

$$g = (\vec{1} \otimes \delta_0 \otimes I)^\top ((\tilde{G}\tilde{P})^t - (\tilde{J}\tilde{P})^t)(\vec{1} \otimes \delta_0 \otimes \delta_0). \tag{6}$$

This equality can again be seen by expanding the right hand side above as a sum over all length-t walks on V.

We will bound the right hand side using singular-value approximations [1, 3]. A matrix $W' \in \mathbb{C}^{N \times N}$ is a singular-value ϵ-approximation of another matrix $W \in \mathbb{C}^{N \times N}$, written $W' \overset{\mathrm{sv}}{\approx}_\epsilon W$, if for all $x, y \in \mathbb{C}^N$,

$$|x^*(W' - W)y| \leq \frac{\epsilon}{2}(\|x\|^2 + \|y\|^2 - \|x^*W\|^2 - \|Wy\|^2),$$

where x^* denotes the conjugate transpose of x. The following properties were shown by Ahmadinejad et al. [1, 3]:

1. $\tilde{G} \overset{\mathrm{sv}}{\approx}_\lambda \tilde{J}$.

Assume that $W' \overset{\mathrm{sv}}{\approx}_\epsilon W$. Then:

2. For every matrix X with spectral norm $\|X\| \leq 1$, then $W'X \overset{\mathrm{sv}}{\approx}_\epsilon WX$.
3. If $\epsilon < .1$, then $(W')^t \overset{\mathrm{sv}}{\approx}_{\epsilon+5\epsilon^2} W^t$. (Importantly, the bound $\epsilon + O(\epsilon^2)$ does not grow with t.)
4. $\|W' - W\| \leq \epsilon$.

We now bound the right hand side of (6) using singular value approximations. Because by definition $\|\tilde{P}\| = 1$, property 1 and property 2 above imply that $\tilde{G}\tilde{P} \overset{\text{sv}}{\approx}_{\lambda} \tilde{J}\tilde{P}$. Then property 3 implies that $(\tilde{G}\tilde{P})^t \overset{\text{sv}}{\approx}_{\lambda+5\lambda^2} (\tilde{J}\tilde{P})^t$, and property 4 then gives that $\|(\tilde{G}\tilde{P})^t - (\tilde{J}\tilde{P})^t\| \leq \lambda + 5\lambda^2$, so $\|g\| \leq \lambda + 5\lambda^2 = O(\lambda)$.

5.2 Comparison with techniques in prior work

The proof of Theorem 7 described above is closely related to the analysis of the Impagliazzo-Nisan-Wigderson (INW) [15] pseudorandom generator in Hoza et al. [14]. Hoza et al. [14] use unit-circle approximations [2] to show that length-2 walks on λ-spectral expanders fool permutation branching programs up to a $O(\lambda)$ ℓ_2-error; the INW generator they study recursively applies many such length-2 walks. We generalize this $O(\lambda)$ bound to walks of arbitrary length, and simplify the analysis by replacing the unit-circle approximations with singular-value approximations. We obtain these improvements because the unit-circle approximations, though similar in nature to singular-value approximations, do not satisfy property 2 described above. Although our results do not directly translate to an improved pseudorandom generator, it is an interesting question whether longer walks could somehow be used to improve the seed length.

As described in Section 3.2, Theorem 7 implies a $O(\sqrt{w} \cdot \lambda)$ total variation distance bound, which improves upon the $O(w^4 \cdot \sqrt{\lambda})$ total variation bound of Cohen et al. [6]. However, Cohen et al. [6] prove their result using bounds on the Fourier tails over $(\mathbb{Z}/2\mathbb{Z})^t$ of permutation branching programs with alphabet size $d = 2$, differing significantly from our proof using singular-value approximations, which generalizes readily to $d > 2$.

References

1. AmirMahdi Ahmadinejad. *Computing stationary distributions: perron vectors, random walks, and ride-sharing competition.* PhD thesis, Stanford University, Stanford, California, 2020.
2. AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision Estimation of Random Walks in Small Space. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306, November 2020. ISSN: 2575-8454. doi:10.1109/FOCS46700.2020.00123.
3. AmirMahdi Ahmadinejad, John Peebles, Aaron Sidford, and Salil Vadhan. Personal Communication, 2021.
4. Avraham Ben-Aroya and Amnon Ta-Shma. A Combinatorial Construction of Almost-Ramanujan Graphs Using the Zig-Zag Product. *SIAM Journal on Computing*, 40(2):267–290, January 2011. doi:10.1137/080732651.
5. Gil Cohen, Dor Minzer, Shir Peleg, Aaron Potechin, and Amnon Ta-Shma. Expander Random Walks: The General Case and Limitations. *Electronic Colloquium on Computational Complexity*, 2021. URL: https://eccc.weizmann.ac.il/report/2021/091/.
6. Gil Cohen, Noam Peri, and Amnon Ta-Shma. Expander random walks: a Fourier-analytic approach. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1643–1655, New York, NY, USA, June 2021. Association for Computing Machinery. doi:10.1145/3406325.3451049.
7. David Gillman. A Chernoff Bound for Random Walks on Expander Graphs. *SIAM Journal on Computing*, 27(4):1203–1220, August 1998. doi:10.1137/S0097539794268765.
8. Louis Golowich. A Berry-Esseen Theorem for Expander Walks. *Forthcoming*, 2022.
9. Louis Golowich and Salil Vadhan. Pseudorandomness of Expander Random Walks for Symmetric Functions and Permutation Branching Programs. *Electronic Colloquium on Computational Complexity*, 2022. URL: https://eccc.weizmann.ac.il/report/2022/024/.

10 Venkatesan Guruswami. Guest column: error-correcting codes and expander graphs. *ACM SIGACT News*, 35(3):25–41, September 2004. doi:10.1145/1027914.1027924.

11 Venkatesan Guruswami and Vinayak M. Kumar. Pseudobinomiality of the Sticky Random Walk. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 48:1–48:19, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2021.48.

12 Alexander D. Healy. Randomness-Efficient Sampling within NC1. *computational complexity*, 17(1):3–37, April 2008. doi:10.1007/s00037-007-0238-5.

13 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.

14 William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom Generators for Unbounded-Width Permutation Branching Programs. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2021.7.

15 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 '94, pages 356–364, New York, NY, USA, May 1994. Association for Computing Machinery. doi:10.1145/195058.195190.

16 Akhil Jalan and Dana Moshkovitz. Near-Optimal Cayley Expanders for Abelian Groups. *arXiv:2105.01149 [cs]*, May 2021. arXiv:2105.01149v1.

17 A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, September 1988. doi:10.1007/BF02126799.

18 Grigorii Aleksandrovich Margulis. Explicit constructions of expanders. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973. Publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization.

19 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy Waves, the Zig-Zag Graph Product, and New Constant-Degree Expanders. *The Annals of Mathematics*, 155(1):157, January 2002. doi:10.2307/3062153.

20 Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 238–251, Montreal Canada, June 2017. ACM. doi:10.1145/3055399.3055408.

21 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, December 2012. doi:10.1561/0400000010.

Influence in Completely Bounded Block-Multilinear Forms and Classical Simulation of Quantum Algorithms

Nikhil Bansal
University of Michigan, Ann Arbor, MI, USA

Makrand Sinha
Simons Institute, Berkeley, CA, USA
University of California Berkeley, CA, USA

Ronald de Wolf
QuSoft, CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Abstract

The Aaronson-Ambainis conjecture (Theory of Computing '14) says that every low-degree bounded polynomial on the Boolean hypercube has an influential variable. This conjecture, if true, would imply that the acceptance probability of every d-query quantum algorithm can be well-approximated almost everywhere (i.e., on almost all inputs) by a poly(d)-query classical algorithm. We prove a special case of the conjecture: in every completely bounded degree-d block-multilinear form with constant variance, there always exists a variable with influence at least $1/\text{poly}(d)$. In a certain sense, such polynomials characterize the acceptance probability of quantum query algorithms, as shown by Arunachalam, Briët and Palazuelos (SICOMP '19). As a corollary we obtain efficient classical almost-everywhere simulation for a particular class of quantum algorithms that includes for instance k-fold Forrelation. Our main technical result relies on connections to free probability theory.

2012 ACM Subject Classification Theory of computation → Quantum query complexity; Theory of computation → Oracles and decision trees

Keywords and phrases Aaronson-Ambainis conjecture, Quantum query complexity, Classical query complexity, Free probability, Completely bounded norm, Analysis of Boolean functions, Influence

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.28

Related Version *arXiv Version*: `https://arxiv.org/abs/2203.00212`

Funding *Nikhil Bansal*: Supported in part by the NWO VICI grant 639.023.812.
Makrand Sinha: Supported by a Simons-Berkeley postdoctoral fellowship.
Ronald de Wolf: Partially supported by the Dutch Research Council (NWO/OCW), as part of the Quantum Software Consortium programme (project number 024.003.037), and through QuantERA ERA-NET Cofund project QuantAlgo (680-91-034).

Acknowledgements We thank Scott Aaronson, Srinivasan Arunachalam, Jop Briët, Shachar Lovett and Ryan O'Donnell for helpful comments and pointers to the literature.

1 Introduction

This paper is motivated by quantum query complexity and its relation to classical query complexity. Query complexity has been the context in which many of the main quantum algorithms have been developed, including Shor's [32] (building on [33]) and Grover's [21]. It has the added advantage that we actually know how to prove good lower bounds on query complexity, in contrast to a setting like circuit complexity.

COMPUTATIONAL COMPLEXITY CONFERENCE

Quantum query complexity is closely connected to the study of bounded polynomials (or forms) on the Boolean hypercube. The key to this connection is that the amplitudes of the final state of a d-query quantum algorithm are polynomials of degree at most d in the bits of the input x, and therefore its acceptance probability $p(x)$ is a polynomial of degree at most $2d$. This observation was made by Beals, Buhrman, Cleve, Mosca and de Wolf [12], who used it to show that the bounded-error quantum query complexity and classical query complexity are polynomially related for any *total* Boolean function. Since then a long line of research [7, 3, 13, 8, 34, 4, 10, 31] has tried to pinpoint the exact polynomial dependence as well as studied the relationship with other measures of complexity of a Boolean function (e.g., sensitivity, certificate complexity, and others [27, 15, 11]).

On the other hand, quantum algorithms can offer a huge (superexponential) advantage for partial functions, which are only defined on a subset of the Boolean hypercube; there are many known examples of partial functions whose classical query complexity is much larger than their quantum query complexity, for instance k-fold Forrelation and its variants [2, 34, 10, 31]. This means that the acceptance probability $p(x)$ of a quantum algorithm cannot always be efficiently approximated by a classical algorithm, since otherwise quantum algorithms could offer only a polynomial speedup for any function, be it total or partial.

However, we can set our sights lower, and ask whether it is possible to classically efficiently approximate $p(x)$ on *almost all* inputs. The following conjecture, which first appeared in [1] and is attributed there to folklore, says that we can.

▶ **Conjecture 1** (Folklore). *The acceptance probability of any d-query quantum algorithm on n-bit inputs can be estimated up to additive error ϵ on a $1-\delta$ fraction of the inputs by a classical query algorithm making* $\mathrm{poly}(d, 1/\epsilon, 1/\delta)$ *queries.*

This conjecture is one expression of the general idea that quantum computers can only give significant speedup (in terms of queries, circuit complexity, or other things) on very structured problems, i.e., when the input to the problem has a particular structure, for instance some periodicity or specific correlations between different parts of the input. For generic unstructured inputs, the conjecture says that only a limited quantum speedup can be expected. This conjecture motivates and is implied by the following conjecture due to Aaronson and Ambainis [1]:

▶ **Conjecture 2** (Aaronson-Ambainis conjecture). *Let $f : \{\pm 1\}^n \to [0,1]$ be a degree-d multilinear polynomial. Then, the maximum influence among all variables in f is at least* $\mathrm{poly}(\mathrm{Var}[f], 1/d)$.

The above conjecture poses a fundamental structural question about bounded polynomials on the hypercube and is a notable open problem in the analysis of Boolean functions. Conjecture 2 is known to hold if the function is Boolean-valued (this follows from [24, 29]). For bounded polynomials, [1] observed that the results of Dinur, Friedgut, Kindler and O'Donnell [19] imply that the conjecture holds with at least an exponential dependence in d. Montanaro [25] proved a special case of the conjecture for *block-multilinear forms* where all coefficients have the same magnitude[1]. Defant, Mastyło and Perez [18] generalized this to bounded polynomials where all Fourier coefficients have the same magnitude and showed that the conjecture holds with an $\exp(\sqrt{d \log d})$ dependence. O'Donnell and Zhao [30] showed that it is sufficient to prove the conjecture for so-called *one-block decoupled* polynomials.

[1] This argument can be generalized to the case when $n^{\Omega(d)}$ coefficients have the same magnitude and the rest are zero, as noted in [25] where the observation is attributed to Ambainis.

In this work, our motivation is to study Conjecture 2 for polynomials that represent the acceptance probability of quantum algorithms. Such polynomials have a lot more structure – as shown by Arunachalam, Briët and Palazuelos [9], they can be represented in terms of *completely bounded block-multilinear forms* (as described in the next section) and conversely, such forms even characterize quantum algorithms in a certain sense (see Section 4). As such here we focus on understanding influences in such polynomials.

1.1 Our results

A degree-d block-multilinear form $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ mapping $\{\pm 1\}^{n\times d}$ to $\mathbb{R}$ is a polynomial where the variables are partitioned into d blocks of n variables each, and each monomial contains at most one variable from each block. Formally, $\boldsymbol{x}_b = (x_b(1), \ldots, x_b(n)) \in \{\pm 1\}^n$ constitutes the b^{th} block of variables and

$$f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = \mathbb{E} f + \sum_{m=1}^{d} \sum_{\substack{|\boldsymbol{b}|=m \\ |\boldsymbol{i}|=m}} \widehat{f}_{\boldsymbol{b},\boldsymbol{i}} \cdot \; x_{b_1}(i_1) x_{b_2}(i_2) \cdots x_{b_m}(i_m), \tag{1}$$

where the tuple $\boldsymbol{b} = (b_1, \ldots, b_m)$ satisfies $1 \le b_1 < \ldots < b_m \le d$ and $\boldsymbol{i} \in [n]^m$ is an m-tuple. Note that m is determined from the size of the tuple $\boldsymbol{b}$, so we just write $\widehat{f}_{\boldsymbol{b},\boldsymbol{i}}$ above.

Since each non-constant monomial contains at most one variable from each block and the ordering of the blocks is fixed, a degree-d block-multilinear form $f : \{\pm 1\}^{n\times d} \to \mathbb{R}$ can be naturally viewed as a non-commutative polynomial in matrix variables with the constant term replaced with $\mathbb{E} f$ times the identity. Denoting the non-commutative polynomial as $f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)$ where each $\boldsymbol{U}_b = (U_b(1), \ldots, U_b(n))$ is a block of non-commutative variables, the completely bounded norm[2] $\|f\|_{\text{cb}}$ of the form f is defined as

$$\|f\|_{\text{cb}} = \sup \left\{ \|f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)\|_{\text{op}} \;\middle|\; N \in \mathbb{N}, U_b(i) \in \mathbb{C}^{N\times N}, \|U_b(i)\|_{\text{op}} \le 1, b \in [d], i \in [n] \right\}.$$

The supremum above is always attained and can be computed by solving a semidefinite program as shown by Gribling and Laurent [20]. One can also equivalently restrict the supremum in the definition above to unitary matrices since the convex hull of the set of unitary matrices is the unit operator-norm ball. Moreover, $\|T\|_\infty \le \|T\|_{\text{cb}}$ where $\|T\|_\infty = \max_{x\in\{\pm 1\}^{n\times d}} |T(x)|$, so forms that are completely bounded are also bounded on the hypercube.

Our main result is a proof of the Aaronson-Ambainis conjecture for block-multilinear forms that are completely bounded. To state our result, we recall that the influence of a variable $x_b(i)$ on f is

$$\mathrm{Inf}_{b,i}(f) = \mathbb{E} \left|\partial_{b,i} f(X)\right|^2,$$

where X is uniform in $\{\pm 1\}^{n\times d}$ and $\partial_{b,i} f(x)$ is the discrete derivative (see Section 2). Denoting by $\mathrm{MaxInf}(f) = \max_{b\in[d], i\in[n]} \mathrm{Inf}_{b,i}(f)$ the maximum influence of any variable in f and by $\mathrm{Var}[f]$ the variance of f on the hypercube, we show:

[2] The completely bounded norm originates in the theory of operator algebras. In the literature, this norm is sometimes defined for homogeneous block-multilinear forms only, but here we extend the definition to non-homogeneous block-multilinear forms.

▶ **Theorem 3.** *Let f be a degree-d block-multilinear form with $\|f\|_{\mathrm{cb}} \leq 1$. Then, we have*

$$\mathrm{MaxInf}(f) \geq \frac{(\mathrm{Var}[f])^2}{e(d+1)^4}.$$

The main technical ingredient in the proof of Theorem 3 is a new influence inequality for *homogeneous* block-multilinear forms that relates the completely bounded norm to the influences.

▶ **Theorem 4.** (Non-commutative root-influence inequality). *Let f be a homogeneous degree-d block-multilinear form. Then, for blocks $b \in \{1, d\}$,*

$$\|f\|_{\mathrm{cb}} \geq \frac{1}{\sqrt{e(d+1)}} \sum_{i=1}^{n} \sqrt{\mathrm{Inf}_{b,i}(f)}.$$

In general, the completely bounded norm can change if we permute the blocks, and the theorem above only gives a bound in terms of the influences of the variables in the leftmost and rightmost blocks.

The inequality also easily implies the special case of Theorem 3 for homogeneous forms, with a better dependence on d, as follows:

$$\begin{aligned}\|f\|_{\mathrm{cb}} &\geq \frac{1}{\sqrt{e(d+1)}} \sum_{i=1}^{n} \sqrt{\mathrm{Inf}_{b,i}(f)} \\ &\geq \frac{1}{\sqrt{e(d+1)}} \sum_{i=1}^{n} \frac{\mathrm{Inf}_{b,i}(f)}{\sqrt{\mathrm{MaxInf}(f)}} \geq \frac{\mathrm{Var}[f]}{\sqrt{e(d+1) \cdot \mathrm{MaxInf}(f)}},\end{aligned}$$

where the last inequality follows since for any homogeneous block-multilinear form the sum of influences of variables in *any* one block equals $\mathrm{Var}[f]$ (see (7) in the preliminaries). Then, if $\|f\|_{\mathrm{cb}} \leq 1$, it follows that

$$\mathrm{MaxInf}(f) \geq \frac{(\mathrm{Var}[f])^2}{e(d+1)}.$$

The non-homogeneous case (Theorem 3) requires a bit more care and we use the inequality as an intermediate step to prove Theorem 3 with a worse polynomial dependence on d.

Combined with the results of [1], we obtain that completely bounded forms can be well-approximated by classical query algorithms (decision trees) on most inputs.

▶ **Corollary 5.** *Let $\epsilon, \delta > 0$ and let $f : \{\pm 1\}^{n \times d} \to \mathbb{R}$ be a degree-d block-multilinear form with $\|f\|_{\mathrm{cb}} \leq 1$. Then, there is a deterministic classical algorithm that makes $O(d^5 \epsilon^{-8} \delta^{-5})$ queries and approximates $f(x)$ up to an additive error ϵ on $1 - \delta$ fraction of the inputs $x \in \{\pm 1\}^{n \times d}$.*

1.1.1 Application to quantum algorithms

We consider quantum query algorithms of the type shown in Figure 1. Any such algorithm has black-box access to the input $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ where $\boldsymbol{x}_b \in \{\pm 1\}^n$ for each $b \in [d]$, via a phase oracle. In other words, the algorithm can apply the unitary $O_{\boldsymbol{x}_b} = \mathrm{Diag}(\boldsymbol{x}_b)$ for each $b \in [d]$.

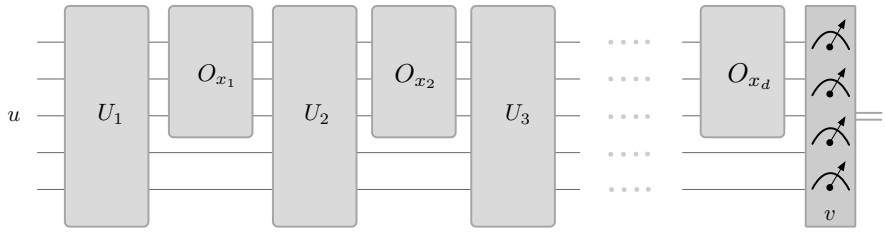


Figure 1 Quantum algorithms considered in Section 1.1.1.

The algorithm starts in some arbitrary quantum state[3] $u \in \mathbb{R}^n$, makes d quantum (phase) queries to oracles $O_{\boldsymbol{x}_b}$ for each $b \in [d]$, and succeeds according to a projective measurement that measures the projection of the final state onto some fixed state $v \in \mathbb{R}^n$. The algorithm is restricted to use each oracle $O_{\boldsymbol{x}_b}$ at most once (which means that each query is made to a disjoint input block not queried previously). The inner product of the state v with the final state at the end of the algorithm is given by the following degree-d block-multilinear form $T : \{\pm 1\}^{n\times d} \to \mathbb{R}$,

$$T(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = uU_1(O_{\boldsymbol{x}_1} \otimes I_s)U_2(O_{\boldsymbol{x}_2} \otimes I_s)U_3 \cdots (O_{\boldsymbol{x}_d} \otimes I_s)v, \tag{2}$$

and the acceptance probability of the algorithm on input $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$ is $T(x)^2$.

The connection between such algorithms and completely bounded norm comes from the following proposition in [9].

▶ **Proposition 6** ([9], Theorem 3.2). *Let $T : \{\pm 1\}^{n\times d} \to \mathbb{R}$ be a degree-d block-multilinear form given by (2). Then, $\|T\|_{\mathrm{cb}} \leq 1$.*

Using this connection, applying Corollary 5 to T implies the following almost-everywhere simulation result for quantum algorithms of the type mentioned above.

▶ **Corollary 7.** *The acceptance probability of any d-query quantum algorithm of the type shown in Figure 1 can be estimated up to an additive error ϵ on $1 - \delta$ fraction of the inputs in $\{\pm 1\}^{n\times d}$ by a classical query algorithm making $O(d^5\epsilon^{-8}\delta^{-5})$ queries.*

Note that quantum algorithms of the type considered in the above theorem can already exhibit super-exponential separation over classical algorithms in the query complexity model. For instance, problems like k-fold Forrelation (for $k = O(1)$) or its variants exhibit a $O(1)$ vs $n^{1-1/k}$ separation [10, 31] between the quantum and classical query complexities. In contrast however, Corollary 7 implies that such quantum algorithms can always be simulated almost-everywhere with only a polynomial overhead.

We remark that a general quantum algorithm can query the same block multiple times. Such an algorithm can be converted to an algorithm of the type given in Figure 1 by taking all the blocks $\boldsymbol{x}_b$ to be the same (and allowing some bits to be fixed to constants). However, the corresponding polynomial (2) that results from such an algorithm is not block-multilinear and thus handling this more general case remains an interesting open problem. We discuss the challenges involved in more detail in Section 4.

[3] Throughout this paper, we will assume that all unitaries and states used in the quantum algorithm are real, which one may assume without loss of generality (see e.g. [6]).

1.2 Proof overview

We first consider the case of homogeneous forms and explain the key ideas that go towards proving Theorem 4. We can write a homogeneous block-multilinear form in the following way,

$$f(\boldsymbol{x}_1,\dots,\boldsymbol{x}_d) = \sum_{i_1,\dots,i_d\in[n]} \widehat{f}_{i_1,\dots,i_d}\; x_1(i_1)x_2(i_2)\cdots x_d(i_d)$$
$$= \sum_{i=1}^{n} x_1(i) \underbrace{\left(\sum_{i_2,\dots,i_d\in[n]} \widehat{f}_{i,\dots,i_d}\; x_2(i_2)\cdots x_d(i_d)\right)}_{:= f_i(\boldsymbol{x}_2,\dots,\boldsymbol{x}_d)}.$$

For a first attempt, let us try to show that $\|f\|_{\mathrm{cb}}$ must be large by picking x from the discrete cube $\{\pm1\}^{n\times d}$ as follows: for each block b except the first block, we choose $\boldsymbol{x}_b$ uniformly and independently from $\{\pm1\}^n$, and for the first block we take $x_1(i) = \mathrm{sign}(f_i(\boldsymbol{x}_2,\dots,\boldsymbol{x}_d))$. Taking expectation, this gives us that

$$\|f\|_{\mathrm{cb}} \geq \sum_{i=1}^{n} \mathbb{E}|f_i| \geq 2^{-d/2}\sum_{i=1}^{n}\|f_i\|_2,$$

where the second inequality follows from the multilinear Khintchine inequality[4] which gives us an exponential dependence in d. Note that $\|f_i\|_2^2 = \mathrm{Inf}_{1,i}(f)$ for each i, thus we get that

$$\|f\|_{\mathrm{cb}} \geq 2^{-d/2}\sum_{i=1}^{n}\sqrt{\mathrm{Inf}_{1,i}(f)}. \tag{3}$$

The above also gives a lower bound on $\|f\|_\infty$ which is also a lower bound on $\|f\|_{\mathrm{cb}}$. However, the exponential dependence in d is necessary for the sup-norm as the following example shows.

Example. Consider the following block-multilinear form closely related to the address function. Let $n = 2^d$ and for $a = (a_1,\dots,a_d) \in \{0,1\}^d$, let $\mathrm{addr}(a)$ denote the unique integer in $[n]$ whose binary expansion equals a. Define the degree-$(d+1)$ homogeneous block-multilinear form $f : \{\pm1\}^{n\times(d+1)} \to \{\pm1\}$ as follows,

$$f(\boldsymbol{x}_1,\dots,\boldsymbol{x}_d,\boldsymbol{x}_{d+1}) = \sum_{a\in\{0,1\}^d} g_a(\boldsymbol{x}_1,\dots,\boldsymbol{x}_d)\cdot x_{d+1}(\mathrm{addr}(a)), \tag{4}$$

where $g_a(\boldsymbol{x}_1,\dots,\boldsymbol{x}_d) : \{\pm1\}^{n\times d} \to \{-1,0,1\}$ is defined as

$$g_a(\boldsymbol{x}_1,\dots,\boldsymbol{x}_d) = \left(\frac{x_1(1)+(-1)^{a_1}x_1(2)}{2}\right)\cdots\left(\frac{x_d(1)+(-1)^{a_d}x_d(2)}{2}\right).$$

Note that f only depends on the first two variables in the blocks $\boldsymbol{x}_1,\dots,\boldsymbol{x}_d$ (which we refer to as the address blocks) and all the variables in the last block $\boldsymbol{x}_{d+1}$ (which we

[4] The multilinear Khintchine inequality states that $\mathbb{E}|f| \geq 2^{-d/2}\|f\|_2$ for a homogeneous degree-d block-multilinear form f. A similar conclusion $\mathbb{E}|f| \geq 3^{-d}\|f\|_2$ holds for any degree-d polynomial f on the hypercube and can be derived from the $(4,2)$-hypercontractive inequality $(\mathbb{E}f^4)^{1/4} \leq 3^{d/2}\|f\|_2$ and using that $\mathbb{E}[|S|] \geq \mathbb{E}[S^2]^{3/2}/\mathbb{E}[S^4]^{1/2}$ for any random variable S by Hölder's inequality.

refer to as the data block). Moreover, $g_a(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) \in \{\pm 1\}$ iff the parity of bits in the address blocks matches with a, that is $x_b(1)x_b(2) = (-1)^{a_b}$ for every $b \in [d]$, and $g_a(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = 0$ otherwise.
It follows that $\|f\|_\infty = 1$, as for any setting of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$ exactly one term in the summation in (4) survives. However, for $i = \mathrm{addr}(a) \in [n]$,

$$\mathrm{Inf}_{d+1,i}(f) = \mathbb{E}[|g_a|^2] = 2^{-d} = \frac{1}{n},$$

thus $\sum_{i=1}^{n} \sqrt{\mathrm{Inf}_{d+1,i}(f)} = \sqrt{n} = 2^{d/2}$.

On the other hand, $\|f\|_{\mathrm{cb}} \geq 2^{d/2}$ in the example above – this can be checked by plugging in the following values on the complex unit circle (one-dimensional unitaries): $x_b(1) = 1, x_b(2) = i$ for each $b \in [d]$ and choosing the data block $\boldsymbol{x}_{d+1}$ so that all the magnitudes add up. Thus, one can hope that the freedom to choose large matrices can still allow us to show something like inequality (3) for the completely bounded norm with a polynomial dependence on d, instead of exponential.

Lower bounding $\|f\|_{\mathrm{cb}}$ using Haar random unitaries

Our key observation is that a non-commutative analog of the above strategy works very well. In particular, substituting $N \times N$ Haar random unitaries $U_2(1), \ldots, U_2(n), \ldots, U_d(1), \ldots, U_d(n)$ for the blocks $x_2, \ldots, x_d$ and choosing the block x_1 depending on the polar decomposition of $f_i(\boldsymbol{U}_2, \ldots, \boldsymbol{U}_d)$ allows one to obtain a much larger lower bound on the completely bounded norm $\|f\|_{\mathrm{cb}}$, losing only a polynomial rather than an exponential factor in d.

To obtain quantitative bounds, we need to understand the operator norm of low-degree polynomials of Haar random unitaries. A standard way to upper bound the expected operator norm of random matrices is via the trace method: computing the expected (normalized) trace of the matrix $(AA^*)^m$ for large enough m, and then taking mth root, gives a good control of the operator norm $\|A\|_{\mathrm{op}}$. Since the entries of a Haar random unitary are not independent of one another, it is hard to get a handle on the expected trace directly. A powerful method to understand such quantities is via free probability theory, which considers what happens when the dimension of the matrices $N \to \infty$. In this case, large random matrices behave like free operators, which live on an infinite-dimensional space with a corresponding "trace". We rely on a limiting theorem of Collins and Male [16] who, by strengthening a result of Voiculescu [35], show that the operator norm of a polynomial of Haar random unitaries converges to the operator norm of the polynomial of certain infinite-dimensional operators, called *free Haar unitaries*; thus it suffices to study such free operators.

In free probability theory, such quantities have been studied for a long time (since the work of Haagerup [22]), and we rely on a result of Kemp and Speicher [23] who generalized Haagerup's inequality and showed that for free Haar unitaries one can obtain much better bounds for the operator norm using the usual trace method. In particular, one gets that almost surely as $N \to \infty$, we have

$$\|f_i(\boldsymbol{U}_2, \ldots, \boldsymbol{U}_d)\|_{\mathrm{op}} \leq \mathrm{poly}(d)\|f_i\|_2,$$

in the non-commutative setting. Crucially, the improvement comes because free operators are much more constrained, and many terms that arise while looking at higher moments using the trace method in free probability are zero. One can keep close track of the non-zero terms by using careful combinatorial counting involving what are called *non-crossing partitions*.

Using the above, one can obtain Theorem 4 with the strategy described above using the polar decomposition. The non-homogeneous case requires a bit more technical care, but the key underlying idea is the same.

2 Preliminaries

Notation. Throughout this paper, $[d]$ denotes the set $\{1, 2, \ldots, d\}$. For a random vector (or bit-string) z in $\mathbb{R}^n$, we will use z_i or $z(i)$ to denote the i-th coordinate of z, depending on whether we need to use the subscript for another index. We shall use $\boldsymbol{i} = (i_1, \ldots, i_d)$ for a d-tuple of indices. For a d-tuple $\boldsymbol{i}$, we write $|\boldsymbol{i}| = d$ to denote the size of the tuple.

For a matrix $M \in \mathbb{C}^{N\times N}$, we denote by M^* its conjugate transpose. Given a string $x \in \mathbb{R}^N$, the $N \times N$ diagonal matrix with x on the diagonal is denoted by $\mathrm{Diag}(x)$. The *normalized trace* of an $N \times N$ matrix M is defined as $\mathrm{tr}_N(M) = \frac{1}{N}\left(\sum_{i=1}^N M_{ii}\right)$. The operator norm of a matrix M is denoted by $\|M\|_{\mathrm{op}}$. The left (resp. right) polar decomposition (V, P) of a square matrix M is a factorization of the form $M = VP$ (resp. $M = PV$) where V is a unitary matrix and P is a positive semidefinite matrix – such a factorization always exists for any square matrix, and can be obtained easily from the singular-value decomposition of M. An $N \times N$ matrix U is called a Haar random unitary if it is distributed according to the Haar measure on the Unitary group $\mathbb{U}(N)$.

Random variables are typically denoted by capital letters (e.g., X). We write $\mathbb{E}[f(X)]$ and $\mathrm{Var}[f(X)]$ to denote the expectation and variance of the random variable $f(X)$; if $f : \{\pm1\}^m \to \mathbb{R}$ then we abbreviate it to $\mathbb{E}f$ and $\mathrm{Var}[f]$, where the expectation and variance are taken with respect to the uniform measure on the discrete cube $\{\pm1\}^m$.

Fourier Analysis on the Discrete Cube

We give some basic facts about Fourier analysis on the discrete cube and refer to the book [28] for more details. Every function $f : \{\pm1\}^m \to \mathbb{R}$ can be written uniquely as a sum of monomials $\chi_S(x) = \prod_{i\in S} x_i$,

$$f(x) = \sum_{S\subseteq[m]} \widehat{f}(S)\chi_S(x), \tag{5}$$

where $\widehat{f}(S) = \mathbb{E}[f(X)\chi_S(X)]$ is the Fourier coefficient with respect to the uniform $X \in \{\pm1\}^m$. The monomials $\chi_S(x) = \prod_{i\in S} x_i$ form an orthonormal basis for real-valued functions on $\{\pm1\}^m$, called the *Fourier basis*. Parseval's identity implies that for uniform $X \in \{\pm1\}^m$,

$$\mathbb{E}f^2 = \sum_{S\subseteq[m]} \widehat{f}(S)^2, \text{ and } \mathrm{Var}[f] = \sum_{S\subseteq[m]:S\neq\emptyset} \widehat{f}(S)^2.$$

For a function on the hypercube, we define $\|f\|_2^2 := \mathbb{E}f^2$ which can also be viewed as the sum of squared Fourier coefficients because of Parseval's identity.

The discrete derivative of a function on the hypercube $\{\pm1\}^m$ is given by

$$\partial_i f(x) = \frac{1}{2}(f(x^{i\to1}) - f(x^{i\to-1})),$$

where $x^{i\to b}$ is the same as x except that the i-th coordinate is set to b. It is easily checked that $\partial_i f(x)$ coincides with the real partial derivative $\dfrac{\partial}{\partial x_i}$ of the real multilinear polynomial given by (5).

For a real-valued function $f : \{\pm 1\}^m \to \mathbb{R}$, the influence of a variable x_i on f is defined as

$$\mathrm{Inf}_i(f) = \mathbb{E}|\partial_i f|^2 = \sum_{S \subseteq [m] : i \in S} \widehat{f}(S)^2.$$

Block-multilinear Forms

A degree-d block-multilinear form $f : \{\pm 1\}^{n \times d} \to \mathbb{R}$ is given by

$$f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = \mathbb{E} f + \sum_{m=1}^{d} \sum_{\substack{|\boldsymbol{b}|=m \\ |\boldsymbol{i}|=m}} \widehat{f}_{\boldsymbol{b},\boldsymbol{i}} \cdot \; x_{b_1}(i_1) x_{b_2}(i_2) \cdots x_{b_m}(i_m), \tag{6}$$

where the tuple $\boldsymbol{b} = (b_1, \ldots, b_m)$ satisfies $1 \leq b_1 < \ldots < b_m \leq d$ and $\boldsymbol{i} \in [n]^m$ is an m-tuple. The expectation that yields the constant term is uniform over $\{\pm 1\}^{n \times d}$. Note that m is determined from the size of the tuple $\boldsymbol{b}$, so we just write $\widehat{f}_{\boldsymbol{b},\boldsymbol{i}}$ above.

From Parseval's identity, the variance of f and the influence of $x_b(i)$ on f (where $b \in [d]$ and $i \in [n]$) are respectively given by

$$\mathrm{Var}[f] = \sum_{m=1}^{d} \sum_{\substack{|\boldsymbol{b}|=m \\ |\boldsymbol{i}|=m}} \widehat{f}^2_{\boldsymbol{b},\boldsymbol{i}} \quad \text{and} \quad \mathrm{Inf}_{b,i}(f) = \sum_{m=1}^{d} \sum_{\substack{|\boldsymbol{b}|=m : b \in \boldsymbol{b} \\ |\boldsymbol{i}|=m : i_b = i}} \widehat{f}^2_{\boldsymbol{b},\boldsymbol{i}}.$$

From the above, it follows that for any block $b \in [d]$,

$$\sum_{i \in [n]} \mathrm{Inf}_{b,i}(f)^2 \leq \mathrm{Var}[f] \leq \sum_{b \in [d], i \in [n]} \mathrm{Inf}_{b,i}(f)^2 \tag{7}$$

where the first inequality is an equality if f is a homogeneous degree-d block-multilinear form. For any $b \in [d]$, we write $\mathrm{MaxInf}_b(T) = \max\{\mathrm{Inf}_{b,i}(T) \mid i \in [n]\}$ to denote the maximum influence of any variable in the block $\boldsymbol{x}_b$.

Note that if f is a degree-d block-multilinear form and if we fix some of the input bits to ± 1, then the resulting function g is also a degree-d block-multilinear form with the same blocks $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d$, but it does not depend on the variables that were fixed. It is also easy to see that $\|g\|_{\mathrm{cb}} \leq \|f\|_{\mathrm{cb}}$ because while computing $\|g\|_{\mathrm{cb}}$ we may restrict the matrix variables to $\pm I$ if that particular variable was set to ± 1. In other words, completely bounded norm does not increase under restrictions.

3 Influence in Completely Bounded Block-multilinear Forms

In this section we prove the non-commutative root-influence inequality (Theorem 4), the special case of the Aaronson-Ambainis conjecture given in Theorem 3, and also briefly mention how the simulation result in Corollary 5 follows from Theorem 3 and the results in [1]. We first need some preliminaries from free probability theory.

3.1 Low-degree polynomials of Haar random unitaries

As discussed in the proof overview, we require bounds on the operator norm (as well as normalized trace) of low-degree polynomials of random unitaries and these follow from known results in free probability theory. Here we explain these connections and also prove some auxillary lemmas needed for the proof of Theorem 4 and Theorem 3.

Let $z_{\boldsymbol{i}}$ denote the non-commutative monomial $z_{i_1} z_{i_2} \cdots z_{i_d}$ for a d-tuple $\boldsymbol{i} = (i_1, \ldots, i_d) \in [t]^d$ and let $p(z_1, \ldots, z_t)$ be a non-commutative polynomial in the variables $z_1, \ldots, z_t$. We are interested in computing the operator norm $\|\cdot\|_{\mathrm{op}}$ and the normalized trace tr_N of the polynomial $p(z_1, \ldots, z_t)$ (or its higher moments) when substituting $N \times N$ Haar random unitaries for the variables z_i.

As explained previously, the theory of free probability gives us tools that allow us to compute the above in the limit $N \to \infty$. In particular, Voiculescu [35] showed that the (normalized) trace of polynomials in Haar random unitaries and their conjugates converges to the trace of the same polynomial evaluated on certain infinite-dimensional operators called *Haar unitaries* that satisfy a non-commutative notion of independence called *free independence*. This was strengthened by Collins and Male [16] who showed that such convergence also holds for the operator norm. A short primer on free probability is given in Appendix A.1, but for now one can think of $\mathcal{A}$ as a self-adjoint algebra of bounded linear operators on a Hilbert space and φ as a trace functional for such operators in the statement given below.

▶ **Theorem 8** ([35, 16]). *Let $p(z_1, \ldots, z_{2t})$ be a non-commutative polynomial in $\mathbb{R}\langle z_1, \ldots, z_{2t}\rangle$. If $U_1, \ldots, U_t$ are $N \times N$ Haar random unitaries, then almost surely,*

$$\begin{aligned} \lim_{N\to\infty} \mathrm{tr}_N[p(U_1, \ldots, U_t, U_1^*, \ldots, U_t^*)] &= \varphi[p(u_1, \ldots, u_t, u_1^*, \ldots, u_t^*)], \\ \lim_{N\to\infty} \|p(U_1, \ldots, U_t, U_1^*, \ldots, U_t^*)\|_{\mathrm{op}} &= \|p(u_1, \ldots, u_t, u_1^*, \ldots, u_t^*)\|, \end{aligned}$$

where $u_1, \ldots, u_t$ are free Haar unitaries in a C^*-probability space $(\mathcal{A}, \varphi)$ and $\|\cdot\|$ is the norm for the underlying C^*-algebra.

Using the above result it suffices to consider free Haar unitaries in a C^*-probability space to compute the operator norm and trace of polynomials of random unitaries. For a non-commutative polynomial $p(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|\le d} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$, denoting by $\|p\|_2 = \left(\sum_{|\boldsymbol{i}|\le d} |c_{\boldsymbol{i}}|^2\right)^{1/2}$, one can show the following easily using techniques from free probability.

▶ **Lemma 9.** *Let $p(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|\le d} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$ be a non-commutative degree-d polynomial in $\mathbb{R}\langle z_1, \ldots, z_t\rangle$ and $u_1, \ldots, u_t$ be free Haar unitaries in a C^*-probability space $(\mathcal{A}, \varphi)$. Then,*

$$\varphi[p(u_1, \ldots, u_t)(p(u_1, \ldots, u_t))^*] = \|p\|_2^2.$$

The above implies that $\mathrm{tr}_N[p(U_1, \ldots, U_t)(p(U_1, \ldots, U_t))^*]$ converges to $\|p\|_2^2$ almost surely as $N \to \infty$. We shall defer the proof of Lemma 9 to Appendix A, but to aid our intuition we note here that since the U_i's are independent $N \times N$ Haar random unitaries, the expected value

$$\mathbb{E}\left[\mathrm{tr}_N[p(U_1, \ldots, U_t)(p(U_1, \ldots, U_t))^*\right] = \|p\|_2^2,$$

and from concentration of measure, it is natural to expect that it converges to the expected value.

Similarly, to compute the operator norm of $p(U_1, \ldots, U_t)$ for Haar random unitaries one can instead study the norm of the polynomial evaluated on free Haar unitaries. Such bounds are easier to prove using the trace method since free independence imposes strong restrictions on the non-commutative moments. For instance, if U_1 and U_2 are independent $N \times N$ Haar random matrices, then $\mathbb{E}[\mathrm{tr}_N(U_1 U_2 U_1^* U_2^*)]$ is non-zero (albeit quite small), while the corresponding trace evaluated on free Haar unitaries u_1 and u_2 is zero, that is $\varphi(u_1 u_2 u_1^* u_2^*) = 0$. Thus, computing the trace $\varphi[p(u_1, \ldots, u_t, u_1^*, \ldots, u_t^*)]$ reduces to handling the combinatorics of the patterns of u_i's and u_i^*'s.

In particular, we will rely on the following result that follows from the work of Kemp and Speicher [23] who consider the operator norm of homogeneous polynomials evaluated on free R-diagonal operators, a class that includes free Haar unitaries as well. We also remark that a bound where the right-hand side below is worse by a multiplicative $O(d^{1/2})$ factor also follows from the work of Haagerup[5] [22] who proved it in another context, predating even the introduction of free probability theory.

▶ **Theorem 10** ([23]). *Let $p(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|=d} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$ be a homogeneous non-commutative degree-d polynomial in $\mathbb{R}\langle z_1, \ldots, z_t\rangle$ and $u_1, \ldots, u_t$ be free Haar unitaries in a C^*-probability space. Then,*

$$\|p(u_1, \ldots, u_t)\| \leq \sqrt{e(d+1)} \cdot \|p\|_2,$$

where the left-hand side denotes the norm in the underlying C^*-algebra.

For completeness, we introduce the necessary free probability background and some combinatorial details in Appendix A, and we present the fairly short proof of Theorem 10 (from [23]) there in a self-contained way. We shall need to extend the above bound to non-homogeneous polynomials. Let $p(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|\leq d} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$ and let $p_k(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|=k} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$ denote the degree-k homogeneous part of p. Writing $p_k = p_k(u_1, \ldots, u_t)$ for $0 \leq k \leq d$ and $p = p(u_1, \ldots, u_t)$, it follows from the triangle inequality, Theorem 10, and Cauchy-Schwarz, that

$$\begin{aligned}\|p\| &\leq \sum_{k=0}^{d} \|p_k\| \leq \sum_{k=0}^{d} \sqrt{e(k+1)}\|p_k\|_2 \\ &\leq \sqrt{e}\left(\sum_{k=0}^{d}(k+1)\right)^{1/2}\left(\sum_{k=0}^{d}\|p_k\|_2^2\right)^{1/2} \leq \sqrt{e}(d+1)\cdot\|p\|_2.\end{aligned}$$

Thus, we essentially get the same bound as in the homogeneous case, at the expense of an additional $O(d^{1/2})$ factor.

Collecting all the above we have the following as a direct consequence:

▶ **Theorem 11.** *Let $p(z_1, \ldots, z_t) = \sum_{|\boldsymbol{i}|\leq d} c_{\boldsymbol{i}} z_{\boldsymbol{i}}$ be a non-commutative degree-d polynomial in $\mathbb{R}\langle z_1, \ldots, z_t\rangle$ and $U_1, \ldots, U_t$ be independent $N \times N$ Haar random unitaries. Then, the following holds almost surely,*

$$\lim_{N\to\infty} \mathrm{tr}_N[p(U_1, \ldots, U_t)(p(U_1, \ldots, U_t))^*] = \|p\|_2^2,$$

and

$$\lim_{N\to\infty} \|p(U_1, \ldots, U_t)\|_{\mathrm{op}} \leq \sqrt{e}(d+1)\cdot\|p\|_2,$$

Moreover, the factor $(d+1)$ in the operator norm bound can be improved to $\sqrt{d+1}$ if additionally p is also assumed to be homogeneous.

Based on the above theorem, we prove the following key lemma which captures the polar decomposition strategy mentioned in the earlier proof overview (Section 1.2). This will serve as the key ingredient in the proof of Theorem 3 and Theorem 4.

[5] We note that Haagerup considered the more general case of polynomials in both u_i's and u_i^*'s.

▶ **Lemma 12.** *Let p be a non-commutative degree-d polynomial in $\mathbb{R}\langle y_1, \ldots, y_m, z_1, \ldots, z_t\rangle$ given by*

$$p(y_1, \ldots, y_m, z_1, \ldots, z_t) = \sum_{i=1}^{m} y_i q_i(z_1, \ldots, z_t) + q_0(z_1, \ldots, z_t).$$

Then, for every $\delta > 0$, there exist an integer N and $N \times N$ unitaries $V_1, \ldots, V_m, W_1, \ldots, W_t$ such that

$$\|p(V_1, \ldots, V_m, W_1, \ldots, W_t)\|_{\mathrm{op}} \geq \frac{1}{\sqrt{e}(d+1)} \sum_{i=1}^{m} \|q_i\|_2 - \delta.$$

Moreover, the factor in front can be improved to $(e(d+1))^{-1/2}$ if p is homogeneous as well.

We note that the definition of completely bounded norm allows us to plug in matrices of arbitrarily large dimensions, so the exact dependence on δ is not relevant for our purposes since we can always make it arbitrarily small.

Proof. For an arbitrary integer N, let us pick independent $N \times N$ Haar random unitaries $W_1, \ldots, W_t$ which we substitute for the variables $z_1, \ldots, z_t$, respectively, and let $M_i = q_i(W_1, \ldots, W_t)$ be the corresponding random matrices. Then, for any tuple of matrices $V_1, \ldots, V_m$ that we could substitute for the variables $y_1, \ldots, y_m$, we have that

$$p(V_1, \ldots, V_m, W_1, \ldots, W_t) = \sum_{i=1}^{m} V_i M_i + M_0.$$

Theorem 11 and union bound imply that as $N \to \infty$, with probability 1 all the following events simultaneously hold:

- $\|M_i\|_{\mathrm{op}} \leq \sqrt{e}(d+1) \cdot \|q_i\|_2$ for each i,
- $\mathrm{tr}_N(M_i^* M_i) = \|q_i\|_2^2$ for each i, where $\mathrm{tr}_N(M)$ is the normalized trace.

To show that the operator norm must be large, let us fix a sufficiently large N and a choice of $N \times N$ unitaries $W_1, \ldots, W_t$ such that M_i satisfies $\|M_i\|_{\mathrm{op}} \leq \sqrt{e}(d+1) \cdot \|q_i\|_2 + \epsilon$ and $\mathrm{tr}_N(M_i^* M_i) \geq \|q_i\|_2^2 - \epsilon$ for each $0 \leq i \leq m$, where ϵ can be made arbitrarily small by increasing N. For $0 \leq i \leq m$, let $M_i = U_i P_i$ be the left polar decomposition of M_i, where U_i is a unitary matrix and P_i is a positive semidefinite matrix.

We select the tuple of unitary matrices $V_1, \ldots, V_m$ that we substitute for the variables $y_1, \ldots, y_m$ to be $V_i = U_0 U_i^*$ for $i \in [m]$. With this we have that $\|p(V_1, \ldots, V_m, W_1, \ldots, W_t)\|_{\mathrm{op}}$ is at least

$$\begin{aligned}
\Big\|M_0 + \sum_{i=1}^{m} V_i M_i\Big\|_{\mathrm{op}} &= \Big\|U_0 P_0 + \sum_{i=1}^{m} U_0 U_i^* U_i P_i\Big\|_{\mathrm{op}} \\
&= \Big\|U_0 P_0 + \sum_{i=1}^{m} U_0 P_i\Big\|_{\mathrm{op}} \\
&= \Big\|P_0 + \sum_{i=1}^{m} P_i\Big\|_{\mathrm{op}} \geq \mathrm{tr}_N\Big(P_0 + \sum_{i=1}^{m} P_i\Big) \geq \mathrm{tr}_N\Big(\sum_{i=1}^{m} P_i\Big),
\end{aligned}$$

where the last equality follows since the operator norm is unitarily invariant and the last two inequalities follow from the positive semidefiniteness of the P_i's.

For every positive semidefinite matrix P, we have that $\mathrm{tr}_N(P) \geq \mathrm{tr}_N(P^2)/\|P\|_{\mathrm{op}}$. Hence,

$$\|p(V_1, \ldots, V_m, W_1, \ldots, W_t)\|_{\mathrm{op}} \geq \sum_{i=1}^{m} \frac{\mathrm{tr}_N(P_i^2)}{\|P_i\|_{\mathrm{op}}}.$$

By our choice of M_i, we have that $\mathrm{tr}_N(P_i^2) = \mathrm{tr}_N(M_i^* M_i) \geq \|q_i\|_2^2 - \epsilon$ and $\|P_i\|_{\mathrm{op}} = \|M_i\|_{\mathrm{op}} \leq \sqrt{e}(d+1)\|q_i\|_2 + \epsilon$. Since ϵ can be made arbitrarily small by increasing N, it follows that

$$\|p(V_1, \ldots, V_m, W_1, \ldots, W_t)\|_{\mathrm{op}} \geq \frac{1}{\sqrt{e}(d+1)} \sum_{i=1}^{m} \|q_i\|_2 - \delta,$$

for large enough N. The improved bound for the homogeneous case follows directly by plugging the bound of Theorem 11 into the above proof. ◀

3.2 Non-commutative root-influence inequality

For clarity in the proofs below, we remind our convention that all tuples or blocks are denoted with boldface fonts (e.g. $\boldsymbol{U}_1$ or $\mathbf{A}$), while a single element is denoted without boldface (e.g. $U_1(i)$ or A_i or A). Before proceeding with the proof, we restate the statement for convenience.

▶ **Theorem 4.** (Non-commutative root-influence inequality). *Let f be a homogeneous degree-d block-multilinear form. Then, for blocks $b \in \{1, d\}$,*

$$\|f\|_{\mathrm{cb}} \geq \frac{1}{\sqrt{e(d+1)}} \sum_{i=1}^{n} \sqrt{\mathrm{Inf}_{b,i}(f)}.$$

Proof of Theorem 4. Since f is homogeneous, we can write

$$\begin{aligned} f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) &= \sum_{i_1, \ldots, i_d \in [n]} \widehat{f}_{i_1, \ldots, i_d}\; x_1(i_1) x_2(i_2) \cdots x_d(i_d) \\ &= \sum_{i=1}^{n} x_1(i) \underbrace{\left(\sum_{i_2, \ldots, i_d \in [n]} \widehat{f}_{i, \ldots, i_d}\; x_2(i_2) \cdots x_d(i_d) \right)}_{:= f_i(\boldsymbol{x}_2, \ldots, \boldsymbol{x}_d)}. \end{aligned}$$

In this case, it follows from (7) that for each $i \in [n]$, we have

$$\mathrm{Var}[f_i] = \|f_i\|_2^2 = \mathrm{Inf}_{1,i}(f) \text{ and } \mathrm{Var}[f] = \sum_{i=1}^{n} \mathrm{Inf}_{1,i}(f). \tag{8}$$

Let us denote the corresponding non-commutative block-multilinear polynomials by $f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)$ and $f_i(\boldsymbol{U}_2, \ldots, \boldsymbol{U}_d)$ where $\boldsymbol{U}_b = (U_b(1), \ldots, U_b(n))$ denotes the b^{th} block of non-commutative variables. To show a lower bound on $\|f\|_{\mathrm{cb}}$ it suffices to exhibit a collection of square matrices $\{U_b(i)\}_{b \in [d], i \in [n]}$ with operator norm at most 1, such that $\|f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)\|_{\mathrm{op}}$ is large.

Applying Lemma 12 for the homogeneous case (with $p = f$, $q_i = f_i$ for $i \in [n]$, and $q_0 = 0$), it follows that for every $\delta > 0$ there exists an integer N and a choice of tuples $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d$ of $N \times N$ unitaries such that

$$\|f\|_{\mathrm{cb}} \geq \|f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)\|_{\mathrm{op}} \geq \frac{1}{\sqrt{e(d+1)}} \sum_{i \in [n]} \|f_i\|_2 - \delta \overset{(8)}{=} \frac{1}{\sqrt{e(d+1)}} \left(\sum_{i=1}^{n} \sqrt{\mathrm{Inf}_{1,i}(f)} \right) - \delta.$$

Taking $\delta \to 0$, we get the statement of the lemma. The proof for the inequality when $b = d$ is the last block follows similarly by using the right polar decomposition analogue of Lemma 12. ◀

3.3 Aaronson-Ambainis Conjecture for non-homogeneous forms

In this section, we prove Theorem 3, which requires handling non-homogeneous forms. The proof will be similar to the proof of Theorem 4 but we will need to be careful about certain details.

Proof of Theorem 3. Any block-multilinear polynomial $f(x_1, \ldots, x_d)$ can be written as

$$f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) = \mathbb{E}f + \sum_{b \in [d]} f_b(\boldsymbol{x}_b, \boldsymbol{x}_{b+1}, \ldots, \boldsymbol{x}_d),$$

where f_b consists of all monomials of f that start with a variable in the b^{th} block $\boldsymbol{x}_b$. Note that f_b depends only on the variables in blocks $\boldsymbol{x}_b, \boldsymbol{x}_{b+1}, \ldots, \boldsymbol{x}_d$. Moreover, it follows from Parseval that

$$\mathrm{Var}[f] = \sum_{b \in [d]} \|f_b\|_2^2 = \sum_{b \in [d]} \mathrm{Var}[f_b], \tag{9}$$

so there exists a block $\beta \in [d]$ such that $\mathrm{Var}[f_\beta] \geq \frac{1}{d}\mathrm{Var}[f]$.

Since f_β contributes a lot to the variance, it is natural to try to find an influential variable in the block $\boldsymbol{x}_\beta$. Towards this end, we pull out the variables $x_\beta(i)$ and write

$$f_\beta(\boldsymbol{x}_\beta, \ldots, \boldsymbol{x}_d) = \sum_{i \in [n]} x_\beta(i) f_{\beta,i}(\boldsymbol{x}_{\beta+1}, \ldots, \boldsymbol{x}_d),$$

for block-multilinear polynomials $f_{\beta,i}(\boldsymbol{x}_{\beta+1}, \ldots, \boldsymbol{x}_d)$. Note that some of the $f_{\beta,i}$'s could be identically zero, so let us define S to be the set of those i such that $f_{\beta,i}$ is non-zero. We note that

$$\|f_{\beta,i}\|_2^2 = \mathrm{Inf}_{\beta,i}(f_\beta) \leq \mathrm{Inf}_{\beta,i}(f), \tag{10}$$

which implies (using Parseval's identity) that

$$\frac{1}{d}\mathrm{Var}[f] \leq \mathrm{Var}[f_\beta] = \sum_{i \in S} \|f_{\beta,i}\|_2^2 = \sum_{i \in S} \mathrm{Inf}_{\beta,i}(f_\beta). \tag{11}$$

Denote the corresponding non-commutative block-multilinear polynomials by $f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)$, $f_b(\boldsymbol{U}_b, \ldots, \boldsymbol{U}_d)$, and $f_\beta(\boldsymbol{U}_{\beta+1}, \ldots, \boldsymbol{U}_d)$ where $\boldsymbol{U}_b = (U_b(1), \ldots, U_b(n))$ denotes the b^{th} block of non-commutative variables. To show a lower bound on $\|f\|_{\text{cb}}$ it suffices to exhibit a collection of square matrices $\{U_b(i)\}_{b \in [d], i \in [n]}$ with operator norm at most 1 such that $\|f(\boldsymbol{U}_1, \ldots, \boldsymbol{U}_d)\|_{\text{op}}$ is large.

We set the matrices in blocks $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_{\beta-1}$ to be zero (that is, the all-zero matrix $\mathbf{0}$). Note that with this choice all polynomials $f_b(\boldsymbol{U}_b, \ldots, \boldsymbol{U}_d)$ where $b < \beta$ vanish and the non-commutative polynomial becomes

$$\begin{aligned} &f(\mathbf{0}, \ldots, \mathbf{0}, \boldsymbol{U}_\beta, \boldsymbol{U}_{\beta+1}, \ldots, \boldsymbol{U}_d) \\ &\quad = \sum_{i \in S} U_\beta(i) f_{\beta,i}(\boldsymbol{U}_{\beta+1}, \ldots, \boldsymbol{U}_d) + \sum_{b=\beta+1}^{d} f_b(\boldsymbol{U}_b, \boldsymbol{U}_{b+1}, \ldots, \boldsymbol{U}_d) + \mathbb{E}f, \end{aligned}$$

which is a non-commutative polynomial of the form considered in Lemma 12 (with $m = |S|$, $q_i = f_{\beta,i}$ and $q_0 = \sum_{b=\beta+1}^{d} f_b + \mathbb{E}f$). Thus, by Lemma 12 for every small $\delta > 0$ there exists an integer N and a choice of $N \times N$ matrices for the blocks $\boldsymbol{U}_\beta, \ldots, \boldsymbol{U}_d$ such that

$$\begin{aligned}\|f\|_{\mathrm{cb}} &\geq \|f(\mathbf{0},\ldots,\mathbf{0},\boldsymbol{U}_\beta,\boldsymbol{U}_{\beta+1},\ldots,\boldsymbol{U}_d)\|_{\mathrm{op}} \\ &\geq \frac{1}{\sqrt{e}(d+1)}\sum_{i\in S}\|f_{\beta,i}\|_2 - \delta \overset{(10)}{=} \frac{1}{\sqrt{e}(d+1)}\left(\sum_{i\in S}\sqrt{\mathrm{Inf}_{\beta,i}(f_\beta)}\right) - \delta \\ &\overset{(11)}{\geq} \frac{1}{\sqrt{e}(d+1)}\left(\frac{\sum_{i\in S}\mathrm{Inf}_{\beta,i}(f_\beta)}{\sqrt{\mathrm{MaxInf}(f)}}\right) - \delta \overset{(10)}{\geq} \frac{1}{\sqrt{e}(d+1)^2}\left(\frac{\mathrm{Var}[f]}{\sqrt{\mathrm{MaxInf}(f)}}\right) - \delta\end{aligned}$$

Taking $\delta \to 0$ and using the assumption that $\|f\|_{\mathrm{cb}} \leq 1$, we obtain the statement of the theorem:

$$1 \geq \|f\|_{\mathrm{cb}} \geq \frac{1}{\sqrt{e}(d+1)^2}\cdot\frac{\mathrm{Var}[f]}{\sqrt{\mathrm{MaxInf}(f)}} \implies \mathrm{MaxInf}(f) \geq \frac{(\mathrm{Var}[f])^2}{e(d+1)^4}. \qquad \blacktriangleleft$$

3.4 Approximating completely bounded forms with decision trees

In this section, we briefly mention how to obtain Corollary 5. Aaronson and Ambainis [1, Theorem 3.3] showed that querying the most influential variable reduces the variance of the function f, and if that influence is lower bounded by a polynomial in $\mathrm{Var}[f]/d$, then after $\mathrm{poly}(d)$ queries (the exact quantitative dependence can be read off from their proof), the variance of the function becomes small enough so that it can be approximated almost-everywhere by its expectation. Since the family of degree-d block-multilinear forms with completely bounded norm at most one is closed under restrictions, one can apply Theorem 3 repeatedly. This gives us Corollary 5.

4 Discussion and Open Problems

To prove Conjecture 1 in full generality, one would need to consider arbitrary quantum query algorithms: such an algorithm operating on an input $z \in \{\pm 1\}^m$ always makes queries to the same oracle O_z (with a control qubit possibly). One can always convert any such algorithm to the type given in Figure 1 by replacing the oracle O_z used at each step b with a new oracle $O_{\boldsymbol{x}_b}$ where $\boldsymbol{x}_b \in \{\pm 1\}^{m+1}$. The execution of the original algorithm can then be recovered by substituting $\boldsymbol{x}_b = (z, 1)$ for every $b \in [d]$. As such one can always obtain a completely bounded block-multilinear form associated with any quantum query algorithm. Conversely, the work [9] shows that the existence of a degree-$2d$ homogeneous block-multilinear form $F : \{\pm 1\}^{(m+1)\times 2d}$ with completely bounded norm at most one also implies the existence of a d-query quantum algorithm whose bias is given by $F((z,1),\ldots,(z,1))$ on every input $z \in \{\pm 1\}^m$.[6] Thus, completely bounded homogeneous block-multilinear forms fully characterize quantum query algorithms in this sense.

In many works in quantum query complexity that concern worst-case complexity, understanding completely bounded or bounded block-multilinear polynomials is sufficient to prove lower bounds as well as give worst-case classical simulation results (i.e. for all inputs), see for instance [2, 14]. However, a transformation that converts a general quantum query algorithm to the type shown in Figure 1 is not conducive to the almost-everywhere results considered in this paper, as the size of the input domain increases exponentially and the number of relevant inputs (i.e. where each $\boldsymbol{x}_b$ is set to the same $(z, 1)$) becomes an exponentially small fraction of the new domain.

[6] Note, however, that the polynomial in the variables z obtained after this substitution may not be block-multilinear.

It thus remains an intriguing open problem to see if the characterization of [9] can be used to make further progress on Conjecture 1. One can also hope to make progress on Conjecture 1 without relying on the connection via influences – recently, Aaronson, Ingram and Kretschmer [5] managed to directly prove Conjecture 1 for the special case where the quantum algorithm queries a sparse oracle, without first proving a special case of Conjecture 2.

Another interesting direction is to show that the Aaronson-Ambainis conjecture holds for bounded block-multilinear polynomials, that is, polynomials whose sup-norm on the Boolean hypercube is at most one. While this by itself does not suffice for the application to quantum algorithms as explained above, it might pave the way towards Conjecture 2 in full generality. Lastly, the free-probability toolbox has already found several applications in quantum information theory (see e.g. [36, 17]), and we hope this work will stimulate more applications elsewhere as well.

References

1 Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups. *Theory of Computing*, 10(6):133–166, 2014.

2 Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. *SIAM Journal on Computing*, 47(3):982–1038, 2018.

3 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 863–876, 2016.

4 Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of Huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing*, pages 1330–1342, 2021. `arXiv:2010.12629`.

5 Scott Aaronson, DeVon Ingram, and William Kretschmer. The acrobatics of BQP. *CoRR*, abs/2111.10409, 2021. `arXiv:2111.10409`.

6 Leonard M. Adleman, Jonathan Demarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

7 Andris Ambainis. Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences*, 72(2):220–238, 2006.

8 Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *Journal of the ACM*, 64(5):32:1–32:24, 2017.

9 Srinivasan Arunachalam, Jop Briët, and Carlos Palazuelos. Quantum query algorithms are completely bounded forms. *SIAM Journal on Computing*, 48(3):903–925, 2019.

10 Nikhil Bansal and Makrand Sinha. k-Forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1303–1316, 2021.

11 Howard Barnum, Michael E. Saks, and Mario Szegedy. Quantum query complexity and semi-definite programming. In *Proceedings of 18th Annual IEEE Conference on Computational Complexity*, pages 179–193, 2003.

12 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.

13 Shalev Ben-David, Pooya Hatami, and Avishay Tal. Low-sensitivity functions from unambiguous certificates. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*, volume 67 of *LIPIcs*, pages 28:1–28:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

14 Sergey Bravyi, David Gosset, Daniel Grier, and Luke Schaeffer. Classical algorithms for forrelation, 2021. `arXiv:2102.06963`.

15 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

16 Benoît Collins and Camille Male. The strong asymptotic freeness of Haar and deterministic matrices. *Annales Scientifiques de l'ENS*, (4) 47, fascicule 1:147–163, 2014. `arXiv:1105.4345`.

17 Benoît Collins and Ion Nechita. Random matrix techniques in quantum information theory. *Journal of Mathematical Physics*, 57(1):015215, 2016.

18 Andreas Defant, Mieczyslaw Mastylo, and Antonio Pérez. On the Fourier spectrum of functions on Boolean cubes. *Mathematische Annalen*, 374:653–680, 2018.

19 Irit Dinur, Ehud Friedgut, Guy Kindler, and Ryan O'Donnell. On the Fourier tails of bounded functions over the discrete cube. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 437–446, 2006.

20 Sander Gribling and Monique Laurent. Semidefinite programming formulations for the completely bounded norm of a tensor, 2019. arXiv:1901.04921.

21 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996.

22 Uffe Haagerup. An example of a non nuclear C^*-algebra, which has the metric approximation property. *Inventiones Mathematicae*, 50:279–293, 1978/79.

23 Todd Kemp and Roland Speicher. Strong Haagerup inequalities for free R-diagonal elements. *Journal of Functional Analysis*, 251(1):141–173, 2007. `arXiv:math/0512481`.

24 Gatis Midrijanis. On randomized and quantum query complexities, 2005. `arXiv:quant-ph/0501142`.

25 Ashley Montanaro. Some applications of hypercontractive inequalities in quantum information theory. *Journal of Mathematical Physics*, 53(12):122206, 2012.

26 Alexandru Nica and Roland Speicher. *Lectures on the Combinatorics of Free Probability*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2006.

27 Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

28 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

29 Ryan O'Donnell, Michael E. Saks, Oded Schramm, and Rocco A. Servedio. Every decision tree has an influential variable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 31–39, 2005.

30 Ryan O'Donnell and Yu Zhao. Polynomial bounds for decoupling, with applications. In *Proceedings of 31st Conference on Computational Complexity*, volume 50 of *LIPIcs*, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

31 Alexander A. Sherstov, Andrey A. Storozhenko, and Pei Wu. An optimal separation of randomized and quantum query complexity. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1289–1302, 2021.

32 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

33 Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

34 Avishay Tal. Towards optimal separations between quantum and randomized query complexities. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science*, pages 228–239, 2020.

35 Dan Voiculescu. A strengthened asymptotic freeness result for random matrices with applications to free entropy. *International Mathematics Research Notices*, 1998:41–63, 1998.

36 Z. Yin, A. W. Harrow, M. Horodecki, M. Marciniak, and A. Rutkowski. Random and free observables saturate the Tsirelson bound for CHSH inequality. *Physical Review A*, 95(032101), 2017. `arXiv:1512.00223`.

A Free Probability Primer

There are many excellent books on free probability theory. In particular, we refer to the book [26] for more details than the brief introduction given here.

A.1 Preliminaries

C^*-algebras

Let $\mathcal{A}$ be a unital C^*-algebra. For our purposes, we can think of this as an algebra of bounded operators on a complex Hilbert space which is self-adjoint ($a \in \mathcal{A}$ implies $a^* \in \mathcal{A}$), closed in the operator norm $\|\cdot\|$, and contains the identity ($\mathbf{1} \in \mathcal{A}$). A faithful trace φ on $\mathcal{A}$ is a continuous linear functional $\varphi : \mathcal{A} \to \mathbb{C}$ that is unital ($\varphi(\mathbf{1}) = 1$), positive $\varphi(aa^*) \geq 0$, and $\varphi(aa^*) = 0$ iff $a = 0$.

The pair $(\mathcal{A}, \varphi)$ where $\mathcal{A}$ is a unital C^*-algebra and φ is a faithful trace on $\mathcal{A}$ is called a *C^*-probability space*. Elements of $\mathcal{A}$ are called non-commutative random variables. An example of a finite-dimensional C^*-probability space is the class $(M_n(\mathbb{C}), \mathrm{tr}_n)$, which is the class of $n \times n$ complex matrices with the normalized trace functional defined as $\mathrm{tr}_n(M) = \frac{1}{n}\sum_{i=1}^{n} M_{ii}$. General C^*-probability spaces allow us to extend these definitions to infinite-dimensional operators, which are needed to define a non-commutative analog of independence called *free independence*. Faithfulness of the trace φ then ensures that $\|a\| = \lim_{m\to\infty} \varphi((aa^*)^m)^{1/2m}$ (see [26, Proposition 3.17]). In particular, this allows one to compute the norm $\|\cdot\|$ by using the trace method and taking higher powers of the trace functional φ, as we will see later.

An illustrative example of an infinite-dimensional C^*-algebra is the following: let G be an infinite discrete group with the identity element $\mathbf{1}$ and let $\ell^2(G)$ be the associated complex Hilbert space. The canonical orthonormal basis for $\ell^2(G)$ is given by the standard basis vectors $\{e_g \mid g \in G\}$. The left-regular representation of G maps elements of G to bounded linear operators on $\ell^2(G)$ and is defined by its action on the standard basis vectors,

$$\rho(g)e_h = e_{gh}, \ \forall h \in G.$$

The operator norm closure of the linear span of $\{\rho(g) \mid g \in G\}$ forms a unital algebra $\mathcal{A}$ (called the reduced C^*-algebra of G) and together with the faithful trace φ defined as

$$\varphi\left(\sum_g \alpha_g \rho(g)\right) = \alpha_{\mathbf{1}}$$

gives a C^*-probability space.

Free Independence

Let $(\mathcal{A}, \varphi)$ be a C^*-probability space and let $\{\mathcal{A}_i\}_{i=1}^n$ be unital $*$-subalgebras of $\mathcal{A}$. They are said to be *free* (or *freely independent*) if for all $k \in [n]$, for all indices $i_1, \ldots, i_k \in [n]$, and for all $a_1 \in \mathcal{A}_{i_1}, \ldots, a_k \in \mathcal{A}_{i_k}$ satisfying $\varphi(a_1) = \ldots = \varphi(a_k) = 0$, the joint *free moment*,

$$\varphi(a_1 \cdots a_k) = 0$$

whenever $j_1 \neq j_2, j_2 \neq j_3, \ldots, j_{k-1} \neq j_k$, that is, the free moments vanish when all the neighboring elements in the sequence $a_1, \ldots, a_k$ come from subalgebras with distinct indices, for example, $\varphi(a_1 a_2 a_1^* a_2^* a_3 a_2) = 0$.

Non-commutative random variables $a_1, \ldots, a_n \in (\mathcal{A}, \varphi)$ are said to be free if the sub-algebras $\{\mathcal{A}_i\}_{i=1}^n$ are free, where $\mathcal{A}_i$ is the unital $*$-subalgebra generated by a_i (the linear span of all monomials $a_i^{\epsilon_1} a_i^{\epsilon_2} \cdots a_i^{\epsilon_r}$ where $\epsilon_1, \ldots, \epsilon_r \in \{1, *\}$ and $r \in \mathbb{N} \cup \{0\}$). Note that the corresponding unital C^*-subalgebras obtained by taking the norm closure of each $\mathcal{A}_i$ are also freely independent in this case (see [26, Exercise 5.23]).

We remark that the set of free non-commutative random variables is an empty set if the underlying C^*-probability space is finite (for instance $(M_n(\mathbb{C}), \mathrm{tr}_n)$), so to find non-trivial examples one needs to work with infinite-dimensional C^*-probability spaces.

Free Haar Unitaries and Free Groups

Let $(\mathcal{A}, \varphi)$ be a C^*-probability space. An element $u \in \mathcal{A}$ is a *Haar unitary* if it is a unitary, i.e. $uu^* = u^*u = \mathbf{1}$, and if $\varphi(u^k) = 0$ for all non-zero integers k. A family $S = \{u_1, \ldots, u_n\} \in \mathcal{A}$ in a C^*-probability space $(\mathcal{A}, \varphi)$ is called a *free Haar unitary family* if each $u \in S$ is a Haar unitary and if $u_1, \ldots, u_n$ are free. For notational convenience, let us define $S^* = \{u_1^*, \ldots, u_n^*\}$ to be the set of corresponding adjoints.

One can give a concrete construction of free Haar unitaries in terms of the free group. The *free group* F_n with a generating set S of size n is an infinite discrete group constructed as follows: a word is defined to be product of elements of $S \cup S^*$ with $\perp$ denoting the empty word that contains no symbols. A word is called reduced if it does not contain a sub-word of the form gg^* or g^*g for $g \in S$. Given a word that is not reduced, the process of repeatedly removing such sub-words until it becomes reduced is called reduction. The free group F_n consists of all reduced words that can be built from the symbols in $S \cup S^*$ with the group operation being a product of words followed by reduction. The identity is the empty word $\perp$.

Let ρ denote the left-regular representation of the free group F_n. Then, writing $S = \{s_1, \cdots, s_n\}$, the family $u_i = \rho(s_i)$ for $i \in [n]$, is a free Haar unitary family in the C^*-probability space given by the reduced C^*-algebra of the group F_n. This connection to the free group also allows us to give a very precise condition when the trace φ evaluated on a non-commutative monomial in the u_i's vanishes. For a d-tuple $\boldsymbol{i} = (i_1, \ldots, i_d) \in [m]^d$, let $u_{\boldsymbol{i}}$ denote the non-commutative monomial $u_{i_1} \cdots u_{i_d}$ and write $u_{\boldsymbol{i}}^* = (u_{\boldsymbol{i}})^* = u_{i_d}^* \cdots u_{i_1}^*$. Let $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_t, \boldsymbol{j}_1, \ldots, \boldsymbol{j}_t$ each be a d-tuple in $[m]^d$ and consider the degree-$2td$ non-commutative monomial $w = u_{\boldsymbol{i}_1} u_{\boldsymbol{j}_1}^* u_{\boldsymbol{i}_2} u_{\boldsymbol{j}_2}^* \cdots u_{\boldsymbol{i}_t} u_{\boldsymbol{j}_t}^*$. Note that a degree-$2td$ monomial w corresponds to an ordered $2td$-tuple of variables. To illustrate, if $t = 1, m = 3$ and $\boldsymbol{i}_1 = (1, 2, 3)$ and $\boldsymbol{j}_1 = (2, 2, 1)$, then $w = u_1 u_2 u_3 (u_2 u_2 u_1)^* = u_1 u_2 u_3 u_1^* u_2^* u_2^*$ and corresponds to the ordered tuple $(u_1, u_2, u_3, u_1^*, u_2^*, u_2^*)$. We can also interpret w as a word in the free group by applying the reduction rules. Then the next proposition follows from the definitions of free independence and Haar unitaries.

▶ **Proposition 13.** *$\varphi(w) = 1$ iff w reduces to identity in the free group F_n, and $\varphi(w) = 0$ otherwise.*

For a monomial w that reduces to identity in the free group, the procedure for reducing a monomial w as above first removes some adjacent pair u_k (at index i) and u_k^* (at index j), then removes another adjacent pair u_l and u_l^* in the resulting word and so on and so forth until we reach the empty word. In particular, this reduction procedure produces a pairing of the set $[2td]$ where the index i and j are paired up iff the variables at indices i and j in the monomial w are u_k and u_k^* (for some k). Moreover, this pairing is what is called a *non-crossing* pairing defined below (see Figure 2). Note that a monomial could be reduced to identity in different ways, so there could be many such non-crossing pairings for a given monomial w.

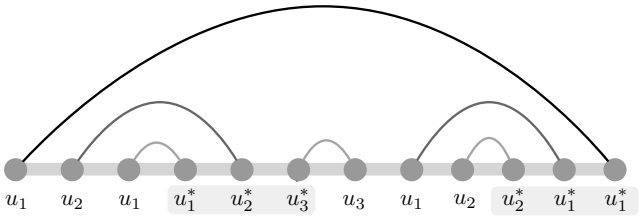


Figure 2 A non-crossing $*$-pairing resulting from the reduction of a word to identity in the free group.

Non-crossing Pairings

For any even integer n, let $\mathcal{P}_2(n)$ denote the set of all pairings of n, that is, the set of all partitions of $[n]$ where each block is of size two. Let $\mathcal{NC}_2(n) \subseteq \mathcal{P}_2(n)$ denote the set of all pairings of $[n]$ that are non-crossing, *i.e.* pairings which do not contain blocks $\{i_1, i_3\}, \{i_2, i_4\}$ such that $i_1 < i_2 < i_3 < i_4$.

For integers d, m, we divide the set $[2dm]$ into $2m$ consecutive blocks of d elements each and color consecutive blocks alternatively with red and blue. Formally, for $i \in [2m]$, the elements $\{(i-1)d+1, \ldots, id\}$ are colored red if i is odd and blue if i is even. We define $\mathcal{NC}_2^*(d, m) \subseteq \mathcal{NC}_2(2dm)$ to be the set of those non-crossing pairings of $[2dm]$ which only pair up elements of different colors. We call any pairing in $\mathcal{NC}_2^*(d, m)$ a $*$-pairing.

We shall need the following combinatorial fact about the number of $*$-pairings (see [23, Corollary 3.2]).

▶ **Lemma 14.** *For all d, m, the number of $*$-pairings $|\mathcal{NC}_2^*(d, m)|$ equals the Fuss-Catalan number*

$$C_{d,m} = \frac{1}{m}\binom{m(d+1)}{m-1} = O\left(\frac{(d+1)^{m(d+1)}}{\left(d+\frac{1}{m}\right)^{md+1}}\right).$$

A.2 Proofs of Lemma 9 and Theorem 11

Proof of Lemma 9. Writing $u_{\boldsymbol{i}}^* = (u_{\boldsymbol{i}})^*$ for a tuple $\boldsymbol{i}$ and using linearity of φ, we have that

$$\varphi[p(u_1, \ldots, u_t)(p(u_1, \ldots, u_t))^*] = \sum_{|\boldsymbol{i}|, |\boldsymbol{j}| \leq d} c_{\boldsymbol{i}} c_{\boldsymbol{j}} \varphi(u_{\boldsymbol{i}} u_{\boldsymbol{j}}^*).$$

From Proposition 13, the term $\varphi(u_{\boldsymbol{i}} u_{\boldsymbol{j}}^*)$ is 1 iff $u_{\boldsymbol{i}} u_{\boldsymbol{j}}^*$ reduces to identity in the free group F_t with generators $u_1, \ldots, u_t$. For the right-hand side above, this only happens when $\boldsymbol{i} = \boldsymbol{j}$ and thus these are the only non-zero terms. Thus,

$$\varphi[p(u_1, \ldots, u_t)(p(u_1, \ldots, u_t))^*] = \sum_{|\boldsymbol{i}| \leq d} |c_{\boldsymbol{i}}|^2. \qquad \blacktriangleleft$$

Below we present the argument of Kemp and Speicher [23]. Our exposition follows their proof closely but we adapt it to our context.

Proof of Theorem 10. We have that $\|p\| = \lim_{m \to \infty} \left(\varphi((pp^*)^m)\right)^{1/(2m)}$ by the faithfulness of the trace φ. Writing $u_{\boldsymbol{j}}^* = (u_{\boldsymbol{j}})^*$ for a tuple $\boldsymbol{j}$, we can compute

$$\varphi((pp^*)^m) = \sum_{\substack{|\boldsymbol{i}_1| = \ldots = |\boldsymbol{i}_m| = d \\ |\boldsymbol{j}_1| = \ldots = |\boldsymbol{j}_m| = d}} c_{\boldsymbol{i}_1} \cdots c_{\boldsymbol{i}_m} c_{\boldsymbol{j}_1} \cdots c_{\boldsymbol{j}_m} \varphi(u_{\boldsymbol{i}_1} u_{\boldsymbol{j}_1}^* \cdots u_{\boldsymbol{i}_m} u_{\boldsymbol{j}_m}^*).$$

Since $u_1, \ldots, u_t$ are free Haar unitaries, Proposition 13 implies that $\varphi(u_{\boldsymbol{i}_1} u_{\boldsymbol{j}_1}^* \cdots u_{\boldsymbol{i}_m} u_{\boldsymbol{j}_m}^*)$ is 1 iff the word $u_{\boldsymbol{i}_1} u_{\boldsymbol{j}_1}^* \cdots u_{\boldsymbol{i}_m} u_{\boldsymbol{j}_m}^*$ reduces to identity in the free group F_t, and is 0 otherwise. Moreover, if the word corresponding to the index $(\boldsymbol{i}_1, \boldsymbol{j}_1, \ldots, \boldsymbol{i}_m, \boldsymbol{j}_m)$ reduces to identity, then there exists a $*$-pairing $\pi \in \mathcal{NC}_2^*(d,m)$ which matches only variables with the same indices. We call any such $*$-pairing π consistent with the $2dm$-tuple $(\boldsymbol{i}_1, \boldsymbol{j}_1, \ldots, \boldsymbol{i}_m, \boldsymbol{j}_m)$ and denote this by the indicator function $\mathbf{1}[\pi, \boldsymbol{i}_1, \boldsymbol{j}_1, \ldots, \boldsymbol{i}_m, \boldsymbol{j}_m]$.

The above implies that we may bound

$$\varphi(u_{\boldsymbol{i}_1} u_{\boldsymbol{j}_1}^* \cdots u_{\boldsymbol{i}_m} u_{\boldsymbol{j}_m}^*) \leq \sum_{\pi \in \mathcal{NC}_2^*(d,m)} \mathbf{1}[\pi, \boldsymbol{i}_1, \boldsymbol{j}_1, \ldots, \boldsymbol{i}_m, \boldsymbol{j}_m],$$

where the inequality occurs because there could be multiple $*$-pairings consistent with a tuple. We thus have that

$$\begin{aligned}\varphi((pp^*)^m) &\leq \sum_{\substack{|\boldsymbol{i}_1|=\ldots=|\boldsymbol{i}_m|=d \\ |\boldsymbol{j}_1|=\ldots=|\boldsymbol{j}_m|=d}} |c_{\boldsymbol{i}_1} \cdots c_{\boldsymbol{i}_m} c_{\boldsymbol{j}_1} \cdots c_{\boldsymbol{j}_m}| \sum_{\pi \in \mathcal{NC}_2^*(d,m)} \mathbf{1}[\pi, \boldsymbol{i}_1, \boldsymbol{j}_1, \cdots, \boldsymbol{i}_m, \boldsymbol{j}_m] \\ &= \sum_{\pi \in \mathcal{NC}_2^*(d,m)} \sum_{\substack{|\boldsymbol{i}_1|=\ldots=|\boldsymbol{i}_m|=d \\ |\boldsymbol{j}_1|=\ldots=|\boldsymbol{j}_m|=d}} |c_{\boldsymbol{i}_1} \cdots c_{\boldsymbol{i}_m} c_{\boldsymbol{j}_1} \cdots c_{\boldsymbol{j}_m}| \mathbf{1}[\pi, \boldsymbol{i}_1, \boldsymbol{j}_1, \cdots, \boldsymbol{i}_m, \boldsymbol{j}_m].\end{aligned}$$

If a term corresponding to a fixed $*$-pairing π is non-zero, then the list of indices $(\boldsymbol{i}_1, \ldots, \boldsymbol{i}_m)$ is the same as $(\boldsymbol{j}_1, \ldots, \boldsymbol{j}_m)$ up to the exact ordering. Let us relabel $(\boldsymbol{i}_1, \ldots, \boldsymbol{i}_m) = (a_1, \ldots, a_{dm})$ and $(\boldsymbol{j}_1, \ldots, \boldsymbol{j}_m) = (b_1, \ldots, b_{dm})$ and let $c_{a_1,\ldots,a_{dm}} = c_{\boldsymbol{i}_1} \ldots c_{\boldsymbol{i}_m}$ and $c_{b_1,\ldots,b_{dm}} = c_{\boldsymbol{j}_1} \cdots c_{\boldsymbol{j}_m}$. Since π gives a non-crossing bijection between the two lists $(a_1, \ldots, a_{dm})$ and $(b_1, \ldots, b_{dm})$, it holds that $c_{b_1,\ldots,b_{dm}} = c_{\pi(a_1),\ldots,\pi(a_{dm})}$. Thus, the above sum is

$$\begin{aligned}\varphi((pp^*)^m) &\leq \sum_{\pi \in \mathcal{NC}_2^*(d,m)} \sum_{a_1,\ldots,a_{dm}} |c_{a_1,\ldots,a_{dm}}| \cdot |c_{\pi(a_1),\ldots,\pi(a_{dm})}| \\ &\leq \sum_{\pi \in \mathcal{NC}_2^*(d,m)} \left(\sum_{a_1,\ldots,a_{dm}} |c_{a_1,\ldots,a_{dm}}|^2 \right)^{1/2} \left(\sum_{a_1,\ldots,a_{dm}} |c_{\pi(a_1),\ldots,\pi(a_{dm})}|^2 \right)^{1/2},\end{aligned}$$

where the inequality follows from Cauchy-Schwarz. The two internal summations are exactly the same since the summation is over all dm tuples of indices and π is a bijection. Switching back to the old indexing scheme, the internal summation then equals

$$\sum_{a_1,\ldots,a_{dm}} |c_{a_1,\ldots,a_{dm}}|^2 = \sum_{|\boldsymbol{i}_1|=\ldots=|\boldsymbol{i}_m|=d} |c_{\boldsymbol{i}_1} \cdots c_{\boldsymbol{i}_m}|^2 = \left(\sum_{|\boldsymbol{i}|=d} |c_{\boldsymbol{i}}|^2 \right)^m.$$

Overall, we have

$$\varphi((pp^*)^m) \leq |\mathcal{NC}_2^*(d,m)| \left(\sum_{|\boldsymbol{i}|=d} |c_{\boldsymbol{i}}|^2 \right)^m.$$

Using Lemma 14 to bound the number of $*$-pairings,

$$|\mathcal{NC}_2^*(d,m)| = C_{d,m} = \frac{1}{m} \binom{m(d+1)}{m-1} = O\left(\frac{(d+1)^{m(d+1)}}{(d+\frac{1}{m})^{md+1}} \right).$$

Thus, taking the m-th root in the limit $m \to \infty$ yields

$$\|p\|^2 = \lim_{m\to\infty} \varphi((pp^*)^m)^{1/m} = \frac{(d+1)^{d+1}}{d^d} \left(\sum_{|\boldsymbol{i}|=d} |c_{\boldsymbol{i}}|^2 \right) \leq e(d+1) \left(\sum_{|\boldsymbol{i}|=d} |c_{\boldsymbol{i}}|^2 \right).$$

This completes the proof of the theorem. ◀

On Randomized Reductions to the Random Strings

Michael Saks
Department of Mathematics, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

Rahul Santhanam
Department of Computer Science, University of Oxford, UK

Abstract

We study the power of randomized polynomial-time non-adaptive reductions to the problem of approximating Kolmogorov complexity and its polynomial-time bounded variants.

As our first main result, we give a sharp dichotomy for randomized non-adaptive reducibility to approximating Kolmogorov complexity. We show that any computable language L that has a randomized polynomial-time non-adaptive reduction (satisfying a natural honesty condition) to $\omega(\log(n))$-approximating the Kolmogorov complexity is in $\mathsf{AM} \cap \mathsf{coAM}$. On the other hand, using results of Hirahara [28], it follows that every language in NEXP has a randomized polynomial-time non-adaptive reduction (satisfying the same honesty condition as before) to $O(\log(n))$-approximating the Kolmogorov complexity.

As our second main result, we give the first negative evidence against the NP-hardness of polynomial-time bounded Kolmogorov complexity with respect to randomized reductions. We show that for every polynomial t', there is a polynomial t such that if there is a randomized time t' non-adaptive reduction (satisfying a natural honesty condition) from SAT to $\omega(\log(n))$-approximating K^t complexity, then either $\mathsf{NE} = \mathsf{coNE}$ or E has sub-exponential size non-deterministic circuits infinitely often.

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

Keywords and phrases Kolmogorov complexity, randomized reductions

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.29

Funding *Michael Saks*: Supported in part by the Simons Foundation under Grant 332622.
Rahul Santhanam: Partially funded by EPSRC New Horizons Grant EP/V048201/1.

Acknowledgements We thank the anonymous reviewers for useful comments. The second author benefited from conversations with Shuichi Hirahara and Rahul Ilango.

1 Introduction

Meta-complexity studies the complexity of computational problems that are themselves about complexity, such as the Minimum Circuit Size Problem MCSP which asks if a Boolean function represented by its truth table has circuits of a given size, and the problem K^{poly} of determining the polynomial-time bounded Kolmogorov complexity of a string. One of the main open questions in meta-complexity is: Are MCSP and related problems such as K^{poly} NP-hard? This is a question with a long history; indeed, Levin is reported [15] to have delayed publication of his seminal NP-completeness results [40] because he hoped to show that MCSP was NP-complete. More than 50 years on, the question remains unresolved, despite much effort [41, 12, 29, 34, 36, 35], and MCSP is one of the few remaining natural problems in NP for which there is no clear evidence either of NP-hardness or of non-NP-hardness.

Why is it so difficult to show MCSP is NP-hard? This has been investigated in a series of works [37, 43, 33, 30, 14, 13, 44]. The theme of this line of works is that hardness of MCSP under various kinds of deterministic reducibility is related to our ability to prove longstanding complexity conjectures. The basic idea is simple: since we can generate negative instances of

SAT efficiently, an efficient honest reduction from SAT to MCSP would allow us to generate negative instances of MCSP efficiently, which would enable us to prove circuit lower bounds for EXP. There are many refinements of this idea in the aforementioned series of works, and in particular [42] unconditionally rule out hardness of MCSP under polylogarithmic-time projections, a class of reductions that is sufficient to show hardness for all known natural NP-complete problems.

Note that the difficulty identified above relies on the *determinism* of the reductions. Indeed, truth tables of hard Boolean functions are easy to generate for randomized algorithms - a uniformly random string is likely to be hard! Therefore, previous works shed no light on whether MCSP and related problems can be shown to be hard under *randomized* reductions. This question is at the heart of our work.

▶ **Question 1.** *How powerful are randomized reductions to meta-complexity problems?*

We note that several works showing hardness results for meta-complexity problems [6, 8, 11, 34, 36, 28] do employ randomness in their reductions, which is a further reason to consider Question 1. As far as we are aware, there is no evidence in the literature against solving all of NP with efficient randomized reductions to meta-complexity problems such as MCSP [1], K^{poly} or even the problem of computing Kolmogorov complexity, even for the case of many-one reductions.

In fact, the hardness results for meta-complexity problems mentioned above [6, 8, 11, 34, 36, 28] are even *robust* to close approximations of the complexity measure of interest, in the sense that the reduction continues to work even if the oracle only knows a close approximation to the complexity measure rather than an exact value. This motivates us to consider Question 1 more broadly in terms of *robustness to approximation* for randomized reductions. It is known, for instance [37, 6], that if one-way functions exist, then circuit size of a Boolean function with truth table size N is even hard to approximate to within an $N^{1-\epsilon}$ factor for any $\epsilon > 0$. One might hope that such strong inapproximability results continue to hold under the weaker assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$. Moreover, an intriguing recent line of work [22, 27, 45] on worst-case to average-case reductions for meta-complexity problems seem to *require* an inapproximability assumption to infer a conclusion on average-case hardness. So, for example, if our goal is to show that NP has worst-case to average-case reductions using these results, we need to show NP-hardness of *approximating* circuit size or K^{poly} complexity.

1.1 Our Results

Our two main results address Question 1 by showing that efficient randomized non-adaptive reductions to the meta-complexity problems of additively approximating the Kolmogorov complexity and additively approximating the polynomial-time bounded Kolmogorov complexity are surprisingly weak, even when the approximation term is super-logarithmic[2]. We first state the results informally, and then describe them in more detail.

[1] As pointed out by an anonymous reviewer, the results of [43] *unconditionally* rule out NP-hardness of MCSP under a very special form of randomized reduction: *projections* computable in *sublinear* time. The techniques of [43] do not seem immediately applicable to reductions that are not projections, or where the bits of the output take at least linear time to compute.

[2] Note that a smaller approximation term makes a negative result stronger.

▶ **Theorem 1** (Informal). *If there is a "nice" randomized non-adaptive poly-time reduction from L to approximating the Kolmogorov complexity within some super-logarithmic additive term, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

▶ **Theorem 2** (Informal). *Under plausible complexity assumptions, if there is a "nice" randomized non-adaptive fixed poly-time reduction from L to approximating the K^{poly} complexity within some super-logarithmic additive term, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Here "nice" has a fairly standard meaning: the reduction is required to be polynomially honest, meaning that it doesn't make queries which are too small. In the statement of Theorem 2, "fixed poly-time" means that the reduction runs within a time bound that is independent of the polynomial time bound used in the definition of K^{poly}. As direct corollaries of Theorem 1 and Theorem 2, we get that under plausible complexity assumptions, neither additively approximating the Kolmogorov complexity to a super-logarithmic term nor additively approximating the polynomial-time bounded Kolmogorov complexity to a super-logarithmic term is NP-hard under efficient randomized reductions.

All our results in this paper have to do with non-adaptive reductions - analysing adaptive reductions is an interesting open question. Indeed, it is not even known whether hardness of MCSP under general *deterministic* adaptive reductions leads to any new complexity lower bounds, or has any other surprising consequences[3].

1.1.1 Randomized Reductions to Approximating Kolmogorov Complexity

As a first step, we consider Question 1 when the meta-complexity problem is to approximate Kolmogorov complexity. Several known reductions to meta-complexity problems such as MCSP [6, 8, 11, 34, 36], especially those that exploit ideas from pseudorandomness, continue to work when the meta-complexity oracle is substituted with a Kolmogorov complexity oracle. Moreover, the power of reductions to Kolmogorov complexity has been studied extensively in its own right [7, 5, 21, 32, 10, 9, 4, 28]. Question 4.8 of Allender's survey [2] explicitly asks about the power of reductions to approximating Kolmogorov complexity.

When considering reductions to a Kolmogorov complexity[4] oracle, there is an ambiguity: Kolmogorov complexity K_U is defined with respect to some universal Turing machine U. In several works [5, 21, 10, 9, 4], the following approach has been taken to resolve the ambiguity. A language L is said to be reducible to Kolmogorov complexity if L is reducible to K_U for *every* universal machine U. In this paper, we take a different approach, as proposed by Allender [2]: we consider reductions to *additively approximating* Kolmogorov complexity rather than reductions to exact Kolmogorov complexity. In practice, reductions rarely take advantage of the exact Kolmogorov complexity of a queried string, so this doesn't lose us too much in terms of modelling known reductions. Moreover, by varying the approximation parameter, we can ensure robustness with respect to the universal machine, and even with respect to the precise notion of Kolmogorov complexity. For instance, reducibility to $\omega(1)$-additively approximating Kolmogorov complexity is robust to the universal machine, using

[3] Some partial results in this direction appear in [44]. Analogous questions about the power of adaptivity in the setting of worst-case to average-case reductions [23, 20] have remained open despite significant effort.

[4] Most works in the literature consider the Kolmogorov random strings as oracle, or else the *overgraph* for Kolmogorov complexity [3]. Without loss of generality, we instead consider the *functional* oracle that when asked a query q, returns the Kolmogorov complexity of q. Note that the Kolmogorov random strings and the overgraph are both reducible to the functional oracle why simple deterministic non-adaptive reductions.

the fact that for any two universal machines U and U', K_U and $K_{U'}$ are at most a constant apart. Reducibility to $O(\log(n))$-additively approximating Kolmogorov complexity is robust to whether we consider the standard version of Kolmogorov complexity or the prefix free version[5], using the fact that for any string, the Kolmogorov complexity and prefix-free Kolmogorov complexity are at most a logarithmic term apart.

Intuitively, a deterministic non-adaptive reduction cannot make use of the Kolmogorov randomness oracle in an effective way, because queries generated by the reduction on compressible inputs have low Kolmogorov complexity. However, if the reduction is allowed to be *randomized*, this limitation disappears, because the reduction can take advantage of randomness to produce queries of high complexity. Hirahara [28] recently showed that every language in NEXP (non-deterministic exponential time) is randomized polynomial-time non-adaptively reducible to the Kolmogorov random strings. This implies a strong positive answer to Question 1 when the meta-complexity problem is to approximate Kolmogorov complexity, and the approximation gap is *small*.

We show, somewhat counter-intuitively, that randomized polynomial-time non-adaptive reductions to *approximating* the Kolmogorov complexity are inherently limited when the approximation gap is $\omega(\log(n))$. Indeed, languages with such reductions (satisfying a natural honesty condition) are in $\mathsf{AM} \cap \mathsf{coAM}$. As far as we are aware, the previous best complexity upper bound known for languages that are randomized polynomial-time non-adaptively reducible to $\omega(\log(n))$-approximating the Kolmogorov complexity was EXPSPACE [10].

Indeed, we get a sharp *complexity dichotomy* based on the approximation gap, by combining our results with those of [28]:

▶ **Theorem 3.** *If a language L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to $\omega(\log(n))$-additively approximating Kolmogorov complexity such that F is polynomially honest and has inverse polynomial advantage, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. On the other hand, every $L \in \mathsf{NEXP}$ has a randomized polynomial-time non-adaptive reduction making q queries to $O(\log(n))$-additively approximating Kolmogorov complexity that is polynomially honest and has constant advantage.*

Theorem 3 is a more precise version of Theorem 1.

A polynomially honest reduction is one where any query on input x is of size $|x|^{\Omega(1)}$. We do have a more general result that is operative even when queries are only guaranteed to have super-constant size, but we adopt the formulation of Theorem 3 here because it is more easily stated. The honesty condition is a very natural one that is satisfied in most reductions of which we are aware. It appears for technical reasons, which we describe in Section 1.2. The *advantage* of a reduction is ϵ if it is correct with probability at least $1/2 + \epsilon$; thus the condition on advantage in the statement of Theorem 3 is the mildest reasonable one.

Our result yields an interesting phenomenon in terms of the tradeoff between *adaptivity* of the reduction and the *robustness to approximation* of the reduction. As mentioned before, it is shown in [28] that approximating Kolmogorov complexity when the approximation term is sufficiently small, i.e., logarithmically bounded, is hard for NEXP under randomized non-adaptive reductions. When the randomized reduction is allowed to be adaptive, it follows from work of [6] that approximating Kolmogorov complexity, even for large *multiplicative* approximation gap $|q|^{1-\epsilon}$ for any $\epsilon > 0$, is hard for PSPACE. Theorem 3 shows that we

[5] The prefix free version of Kolmogorov complexity considers only prefix-free Turing machines, i.e., Turing machines M such that if M halts on x, M does not halt on xy for any non-empty y. This is more natural in certain contexts.

cannot get a hardness result that combines the best of both worlds: if we require the reduction to be non-adaptive, and in addition the approximation gap is large, then the power of the reduction decreases significantly.

Theorem 3 has an application to *non-adaptive black-box* constructions of hitting set generators from worst-case hardness assumptions. Recall that a hitting set generator with seed length $s(n) < n$ is an efficiently computable function from $s(n)$ bits to n bits such that the range of the generator hits every dense set that is computable by polynomial-size circuits. Motivated by connections to uniform hardness-randomness tradeoffs, [26] showed that if there is a randomized non-adaptive reduction from a language L to avoiding the range of an exponential-time computable hitting set generator with seed length $< n/4$, then $L \in \mathsf{BPP}^{\mathsf{NP}}$. Recently, [31] improved this by showing that if there is a randomized non-adaptive reduction from a language L to avoiding the range of an exponential-time computable hitting set generator with seed length $(1-\epsilon)n$ for some $\epsilon > 0$, then $L \in \mathsf{AM} \cap \mathsf{coAM}$. Since a Kolmogorov complexity oracle can be used to avoid the range of any computable hitting set generator (by using the fact that outputs of the hitting set generator have non-trivial Kolmogorov complexity), we get the following immediate corollary:

▶ **Corollary 4.** *If a language L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to avoiding the range of a computable hitting set generator with seed length $n - \omega(\log(n))$ such that F is polynomially honest, then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Corollary 4 is not directly comparable to the main result of [31] because of the honesty condition, but modulo this condition, it shows an *optimal* limitation on reductions to avoiding the range of hitting set generators in terms of the seed length. We note that [28] shows that for every language $L \in \mathsf{NEXP}$, L randomized polynomial-time non-adaptively reduces to avoiding the range of a $\mathsf{EXP}^{\mathsf{NP}}$-computable hitting set generator with seed length $n - O(\log(n))$. Under standard derandomization assumptions [38], $\mathsf{AM} \cap \mathsf{coAM} = \mathsf{NP} \cap \mathsf{coNP}$, which is strictly contained in NEXP.

As discussed in [31], close connections between hitting-set generators and average-case hardness mean that Corollary 4 also relates to a long line of work ruling out non-adaptive black-box worst-case to average-case reductions for NP [23, 20, 1].

1.1.2 Randomized Reductions to Approximating Polynomial-Time Bounded Kolmogorov Complexity

The main application of Theorem 3 is to the question of NP-hardness of *meta-complexity* problems. As an immediate consequence of Theorem 3, even computing an approximation of Kolmogorov complexity with small gap is not NP-hard under (honest) randomized non-adaptive reductions, unless there is a surprising complexity collapse.

▶ **Corollary 5.** *Suppose there is a randomized polynomial-time non-adaptive reduction from SAT to $\omega(\log(n))$-additively approximating Kolmogorov complexity that is polynomially honest. Then the Polynomial Hierarchy collapses.*

Corollary 5 follows from Theorem 3 by using the fact that if SAT is in $\mathsf{AM} \cap \mathsf{coAM}$, then the Polynomial Hierarchy collapses [46].

As far as we are aware, Corollary 5 gives the first negative implication of the assumption that approximating Kolmogorov complexity is NP-hard under randomized non-adaptive reductions. Several recent works on hardness of meta-complexity problems [34, 36, 35] are only able to handle small approximation gap - Corollary 5 provides one possible explanation of this (in cases when the reduction is not refined enough to distinguish between Kolmogorov complexity and other complexity measures).

Corollary 5 does not immediately imply that NP-hardness of K^{poly} under randomized reductions is unlikely, as there is no known efficient non-adaptive reduction from K^{poly} to Kolmogorov complexity. By working quite a bit harder, we are able to show the following result.

▶ **Theorem 6.** *Suppose that there is a polynomially bounded function $t' : \mathbb{N} \to \mathbb{N}$ such that for all large enough $t = \text{poly}(t')$, there is a randomized time t' non-adaptive reduction that is polynomially honest, has fixed query length and inverse polynomial advantage, from SAT to $\omega(\log(m))$-additively approximating K^t complexity. Then either* E *has non-deterministic circuits of size $2^{o(n)}$ infinitely often, or* NE = coNE.

Theorem 6 is a more precise version of Theorem 2.

This appears to be the first negative evidence against NP-completeness of a meta-complexity problem in NP with respect to *randomized* non-adaptive reductions in the unrelativized setting[6]. Note that previous results on consequences of hardness with respect to deterministic reductions of MCSP do not suggest that such reductions are *unlikely*, but rather that they are *hard to show*. In contrast, the consequent of Theorem 6 would be considered unlikely by many complexity theorists.

A subtlety in the statement of Theorem 6 is that the negative evidence depends on the running time of the reduction being smaller than the time bound for Kolmogorov complexity. Typically, reductions from SAT to NP-complete L run in quasi-linear time. Theorem 6 suggests that the picture is very different for K^{poly} - if it is indeed NP-complete under randomized polynomial-time reductions, then the reductions are likely to need a lot of time.

1.2 Proof Techniques

We sketch the ideas behind the proofs of our main results: Theorem 3 and Theorem 6. We first introduce some notation, as per Section 2. Given a function $\beta : \mathbb{N} \to \mathbb{N}$, we say that two functional oracles $O : \Sigma^* \to \mathbb{N}$ and $O' : \Sigma^* \to \mathbb{N}$ are β-close if $|O(q) - O'(q)| \geq \beta(|q|)$ for every string q. In the following, we use β-closeness with $\beta(n) = \omega(\log(n))$.

Suppose M is an oracle machine implementing the reduction hypothesized in Theorem 3. At a high level, we'd like to construct an explicit computable oracle K' such that :

1. K' is β-close to K
2. There is an Arthur-Merlin protocol that given a string q and integer a to $K'(q)$, allows Merlin to prove to Arthur that a is "close" to $K'(q)$.

Suppose we have such an oracle K'. The hypothesis that M is a non-adaptive reduction to β-approximating Kolmogorov complexity with inverse polynomial advantage implies that $M^{K'}$ computes L with advantage $\epsilon = 1/n^{O(1)}$. We then hope to use the second condition above to show that it is possible to give an AM protocol that allows Merlin to give a proof of the output value of $M^{K'}(x)$ by having Merlin provide answers to all of the queries and then prove them to Arthur.

The proof roughly follows this strategy, but there are several obstacles that require modification. We don't know how to construct the desired oracle K'. Instead we consider a generalized notion of oracle called a *context-sensitive* oracle. This is an oracle O whose

[6] A result of Ko [39] states there is no relativizing NP-hardness proof for K^{poly}, but this seems to say little about the unrelativized case. Also, a result of Hirahara and Watanabe [30] shows that 1-query randomized reductions to MCSP that are "oracle independent" only exist for languages in AM ∩ coAM. We do not need the "oracle indepedent" restriction and our results hold for any number of queries, as long as they are non-adaptive.

answer to a query q can depend not just on q but also on the "context" within which q is generated, i.e., the oracle machine M making the query and the input x on which M makes the query. In what follows we use oracle to mean a possibly context-sensitive oracle and refer to an ordinary oracle as *context-insensitive*.

We will define a context-sensitive oracle which we call J that will be used in place of K' in the above outline. Here is a rough description of J. Fix the machine M and input x. Suppose that the machine M makes k queries to K, where the choice of queries is a function of the auxiliary random string ρ. For $i \in \{1, \ldots, k\}$, let $q_i(\rho)$ denote the ith query when the random string is ρ. Let $\pi(q)$ denote the probability (over ρ and random $i \in \{1, \ldots, k\}$) that $q_i(\rho) = q$. The context-sensitive oracle $J_x^M(q)$ is the ceiling of $\log 1/\pi(q)$. It is not hard to show that, for fixed M and x, $J(q)$ is β-close to $K(q)$ for all but a small fraction of q.

We want to show that (a) M^J computes the same language as M^K, and (b) the language computed by M^J is in $\mathsf{AM} \cap \mathsf{coAM}$. We can't show either of these, but we are able to state and prove small modifications to these assertions that suffice to give the desired conclusion.

We would like that M^J computes the same language as M^K. By hypothesis $L = M^{K'}$ for every *context-insensitive* oracle K' that is a β-approximation to K with advantage ϵ, but J is context-sensitive. Nevertheless, we are able to show that for all sufficiently long inputs x, the probability (with respect to the randomness of M) that $M^J(x) \neq M^K(x)$ is at most $\epsilon/2$, and since M^K computes L with advantage ϵ, we conclude that M^J computes L on all sufficiently long inputs with advantage at least $\epsilon/2$. To do this we fix a "sufficiently long" input x. We describe how to construct a context insensitive oracle K' (depending on x) that satisfies (i) if M^J and $M^{K'}$ are run on x then with probability at least $1 - \epsilon/2$ over the auxiliary random string ρ, J and K give the same answers on all of the queries $q_1(\rho), \ldots, q_k(\rho)$ and therefore the probability that M^J and $M^{K'}$ give different output on x is at most $\epsilon/2$, and (ii) K' is β-close to K, which implies that $M^{K'}$ computes L with advantage ϵ. Therefore $M^J(x)$ gives the correct output with advantage $\epsilon/2$.

We conclude that M^K and M^J compute the same language except for finitely many strings. We can modify M^J to fix those strings.

For the second part of the proof we aim to show that one can simulate the computation of M^J in $\mathsf{AM} \cap \mathsf{coAM}$. The J oracle can be viewed as solving an approximate counting problem: For a given query q, how many pairs (ρ, i) have $q_i(\rho) = q$, and such counting problems can be approximated in $\mathsf{AM} \cap \mathsf{coAM}$ [25, 20].

However, there is a major difficulty in translating this intuition into a simulation of M^J in $\mathsf{AM} \cap \mathsf{coAM}$. We are not guaranteed that the Merlin-Arthur protocol provides *consistent* answers that depend only on x and M. In order to get around this diffculty, our main idea is to use the following perturbation argument.

We define a family of oracles $J[\gamma]$ for $\gamma \in [0, 1)$. $J[\gamma]$ is a shifted version of J; it is the ceiling of the logarithm of $\log(1/(1+\gamma)\pi(x))$. We show that given M, there is a γ (in fact a random γ works with high probability) such that $M^{J[\gamma]}$ can be computed in $\mathsf{AM} \cap \mathsf{coAM}$.

In order to use this we show a stronger version of the first step: for all sufficiently long inputs x, for all $\gamma \in [0, 1)$, the probability that $M^J[\gamma]$ disagrees with M^K is at most $\epsilon/2$.

The Arthur-Merlin protocol we use bears significant similarities to the ideas of [23, 20, 31], however we need to work a bit harder to accommodate the perturbation argument.

Why does the honesty condition comes into our results? Our argument works by contradiction and involves using the fact that the first x on which the simulation fails has low Kolmogorov complexity. But this complexity bound is in terms of $|x|$, while the approximation condition on the Kolmogorov complexity oracle is in terms of the query length $|q|$. Thus it is important that there is some way of connecting $|x|$ and $|q|$ when q is a query

asked on x, and this is what the honesty condition guarantees. In fact our more general Theorem 14 is applicable even when the reduction is not polynomially honest, but it does require that query length is at least super-constant.

We now proceed to sketch the ideas behind Theorem 6. A natural idea is to try to *derandomize* the reduction using a standard derandomization hypothesis, and then argue that efficient deterministic non-adaptive reductions are unlikely. However, this doesn't work for the following reason: in order to derandomize the reduction using a standard derandomization hypothesis, we need to fool the test that checks, given fixed x and random string r, that the input x is consistent with the output $f(x, r)$ of the reduction. This test involves running machines both for the langage from which we are reducing and the language to which we are reducing, and a naive implementation takes at least non-deterministic time t (to simulate K^t). Hence the derandomized reduction will run in time $\text{poly}(t)$, which is greater than the time bound for Kolmogorov complexity, and we do not know how to argue against this possibility.

Instead, we use Theorem 3. The idea is to show that, under a standard derandomization hypothesis, oracle access to K^t can be replaced by oracle access to K, without affecting the correctness of the reduction. We prove a new lemma stating that for with high probability over samples from a distribution D samplable in time t', the K^t complexity of the sample q is at most $O(\log(|q|))$ apart from the K complexity. This lemma uses the derandomization hypothesis that E requires exponential-size non-deterministic circuits, and the time bound t is polynomially bounded in t' once the parameters of the derandomization hypothesis are fixed. The derandomization hypothesis is required to get a hitting set generator whose seed length has optimal dependence on the error parameter [19]. Our lemma implicitly improves a result of Antunes and Fortnow [16], which requires a stronger derandomization hypothesis.

However, we are unable to implement this idea for an arbitrary language L that reduces to K^t. The idea *does* work for unary languages however, and we get that every unary language in NP is in $\mathsf{AM} \cap \mathsf{coAM}$, applying Theorem 3. The derandomization hypothesis can now be applied one more time to get a simulation in $\mathsf{NP} \cap \mathsf{coNP}$. By a standard upward translation argument, we get that $\mathsf{NE} = \mathsf{coNE}$. Thus, under the assumption on reducibility, either the derandomization hypothesis fails or $\mathsf{NE} = \mathsf{coNE}$, both of which are unlikely or, at the very least, surprising consequences.

2 Preliminaries

2.1 Basic Complexity Notions

We refer to the book by Arora and Barak [18] for definitions of standard complexity classes.

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 say that L is a tally language if it is contained in Σ^* for some single-letter alphabet Σ.

2.2 Oracle Turing Machines

We consider randomized oracle Turing machines (OTMs). Our OTMs have a read-only input tape, a write-only output tape, one or more work tapes, a random tape, and an oracle tape. The alphabet for all tapes is $\{0, 1\}$. If the TM never makes oracle queries we say it is *oracle-free*.

We refer to an ordered pair (M, x) where M is a Turing machine and x is the input as a *context*.

We assume without loss of generality that a polynomial-time bounded randomized Turing Machine M comes equipped with a poly-time computable function $r^M : \{0,1\}^* \longrightarrow \mathbb{N}$ where r_n^M is a polynomial upper bound on the number of random bits generated by the machine on input x. On input x the TM starts by generating an auxiliary random string ρ of length r_x^M which is written on the random tape, and the program operates deterministicially on the pair x, ρ.

Normally, an oracle is a function O from a query set Q to an answer set A. We also need a more general notion of a *context-sensitive* oracle in which the response depends on the query q and the context (machine M and input x that is calling the oracle). We write $O_x^M(q)$ for the output of oracle O to query q.

Unless otherwise specified all oracles in this paper have query set $Q = \{0,1\}^*$ and answer set $A = \mathbb{N}$.

An OTM M *instantiated by oracle* O, denoted M^O is the program that uses oracle O to answer any queries. If O is computable, then M^O is *computably instantiated*, and can be identified with an oracle-free Turing machine, even in the case that O is a context-sensitive oracle.

The randomized OTMs we consider are *decision machines* which means that they always halt and output either 1 (identified with ACCEPT) or 0 (identified with REJECT). The language $L(M^O)$ recognized by an (possibly randomized) instantiated decision OTM M^O is the set of x for which the probability that $M(x)$ accepts is greater than $1/2$.

Let $\epsilon : \mathbb{N} \longrightarrow [0,1]$. We say that L *is accepted by* M^O *with advantage* ϵ if every accepted string x is accepted with probability at least $(1 + \epsilon(|x|))/2$ and every rejected string is accepted with probability at most $(1 - \epsilon(|x|))/2$. Note that computing with advantage $\epsilon = 1$ means that $M^O(x)$ always outputs $L(x)$.

An algorithm that accepts L with advantage ϵ can be converted to an algorithm with advantage $(1-\epsilon')$ by repeating the algorithm $O(\log(1/\epsilon') + (1/\epsilon))^2$ times and taking majority vote.

We restrict attention to OTMs that are *nonadaptive*. A nonadaptive OTM comes equipped with:

- A computable function $k^M : \{0,1\}^* \longrightarrow \mathbb{N}$. For $x \in \{0,1\}^*$, k_x^M is the number of oracle queries that M makes on input x. For convenience (and with no significant loss of generality) we assume that k_n^M is a power of 2.
- A computable function q^M that takes as input a string x and an integer $i \in \{1, \ldots, k_{|x|}^M\}$, and (if the OTM is randomized) a random string ρ of length $r_{|x|}^M$, and outputs an element of $\{0,1\}^*$. $q_x^M(i,\rho)$ is the ith query asked by M on input x when the random string is ρ.
- A computable function OUT^M that takes as input x and the sequence of query answers $(a_1, \ldots a_{k_x^M})$ and (if the OTM is randomized) the random string ρ and outputs either 0 or 1. $\text{OUT}_x^M(a_1, \ldots, a_{k_x^M}, \rho)$ is the output of the algorithm on input x, random string ρ when the query answers are $r_1, \ldots, r_{k_x^M}$.

In the above definition the nonadaptive OTM M is deterministic if r_x^M is identically 0. Thus the function q_x^M depends only on the query index i.

Q_x^M denotes the set of all $q \in \{0,1\}^*$ such that on input x, M asks query q with positive probability. Q_x^M is a finite set of size at most $2^{r_x^M} k_x^M$ since there are $2^{r_x^M}$ choices for ρ and each results in k_x^M queries.

We say that M is α-*honest*, for $\alpha : \mathbb{N} \longrightarrow \mathbb{N}$, provided that on any input x, all oracle queries have length at least $\alpha(|x|)$. If $\alpha(n) = n^\tau$ for some constant $\tau > 0$ we say that M is *polynomially honest*. Conventionally, a polynomially honest oracle algorithm is referred to simply as *h*onest, but it will be useful for us to distinguish different kinds of honesty.

If M is α-honest for some α that tends to ∞ we say that M is *weakly honest*. We say that M is *fixed query length* if the length of each query of M on input x depends only on $|x|$.

2.3 Approximate Oracles and Robust Oracle Algorithms

Recall that in this paper we consider oracles that take as input a string and output an integer. We now introduce two related concepts for an oracle (possibly context-sensitive) O:

- Approximation of O by another oracle O' that is "suitably close".
- An oracle algorithm M instantiated by O that is robust with respect to replacing O by any O' that is close enough to O.

The closeness of two oracles is measured by a nonnegative valued *closeness function* β with two arguments, a string q and natural number n. The value of β at these arguments is written $\beta_n(q)$. We say that O and P are *β-close with respect to M* provided that for all inputs x to M, and queries $q \in Q_x^M$, $|O_x^M(q) - P_x^M(q)| \leq \beta_{|x|}(q)$.

Let M be a randomized oracle algorithm, O be an oracle and L be the language accepted by M^O. Let β be a closeness function and let $\epsilon : \mathbb{N} \longrightarrow [0,1]$. We say that M is *(β, ϵ)-robust* with respect to oracle O provided that for every oracle O' that is β-close to O, $M^{O'}$ accepts L with advantage ϵ. We say that M is *β-robust* if it is $(\beta, 1)$-robust, which means that for every oracle O' that is β-close to O, $M^{O'}(x)$ always outputs the correct answer $L(x)$.

2.4 Descriptions and Kolmogorov complexity

Throughout we assume a fixed pairing function $< \cdot >$ and a standard encoding of Turing machines M by binary strings, with $\tilde{M}$ denoting the encoding of M.

We fix a universal Turing machine U that for any Turing machine M and input x takes as input $< \tilde{M}, x >$ and outputs $M(x)$. The Kolmogorov complexity $K(q)$ of q (with respect to U), denoted $K(q)$, is the minimum length of y such that $U(y) = q$. We define the Kolmogorov complexity of a machine M to be $K(\tilde{M})$. We have:

▶ **Proposition 7.** *For every computable function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ there is a constant C_f such that if $q = f(z)$ then $K(q) \leq K(z) + C_f$.*

In particular, for f being the identity function ID we have

▶ **Proposition 8.** *For all strings q, $K(q) \leq |q| + C_{ID}$.*

▶ **Proposition 9.** $\sum_{q \in \{0,1\}^*} \frac{2^{-K(q)}}{K(q)^2} \leq 2$.

Proof. Letting S_k be the set of strings of Kolmogorov complexity exactly k, we can rewrite the above sum as:

$$\sum_{k \geq 1} \sum_{q \in S_k} \frac{2^{-k}}{k^2}$$

Since $|S_k| \leq 2^k$, this is at most $\sum_{k \geq 1} 1/k^2 < 2$. ◀

The *canonical order* on $\{0,1\}^*$ is the order in which x precedes y if $|x| < |y|$ or $|x| = |y|$ and x is lexicographically less than y.

▶ **Proposition 10.** *Let L be a computable language that contains infinitely many strings.*

1. *Let $\Lambda : \mathbb{N} \longrightarrow \mathbb{N}$ be any computable unbounded function. Let x_j be the first string (according to the canonical order) satisfying $|x_j| \geq \Lambda(j)$ and $x \in L$. There is an integer C (depending on L and Λ) such that for all integers j, $K(x_j) \leq \log(j) + C$.*

2. *Let $\kappa : \mathbb{N} \longrightarrow \mathbb{Z}$ be any computable nondecreasing function that tends to ∞. Let L be a computable language that contains infinitely many strings. Then L contains a string x for which $K(x) < \kappa(|x|)$.*

Proof. Let M be a Turing Machine that computes L and N be a Turing machine that computes Λ. We can construct a machine M' that on input j outputs x_j. M first evaluates $\Lambda(j)$ and then enumerates strings x of length at least $\Lambda(j)$ in the canonical order and stops when it finds the first string in L. M' is a program whose length is a constant (depending on $|N|$ and $|M|$) and so the pair M', j has a description of length $\log(j) + C$.

For the second part, given κ, let Λ be the function where $\Lambda(j)$ is the least integer m such that $2\log(j) < \kappa(m)$. The function Λ is computable. Applying the first part, there is a constant C (depending on L and Λ) such that for all j, x_j is a string of length at least $\Lambda(j)$ such that $K(x_j) \leq \log(j) + C$. Choose $j = 2^C$. Then x_j has length at least $\Lambda(j)$, and $K(x_j) \leq \log(j) + C \leq 2\log(j) < \kappa(\Lambda(j)) \leq \kappa(|x_j|)$, as required. ◀

We also need the followng proposition giving symmetry of information for Kolmogorov complexity.

▶ **Proposition 11** (Symmetry of information). *Let x, y be strings. Then $K(x; y) = K(x) + K(y|x) + O(\log(K(x, y)))$.*

2.5 Resource-Bounded Kolmogorov Complexity

We study various notions of resource-bounded Kolmogorov complexity, and associated decision problems.

In the R_K problem, the instance is a string x, and the question is whether $K(x) \geq |x|/2$. Given a function $g : \mathbb{N} \to \mathbb{N}$, the R_K problem with gap g is the promise problem whose NO instances are strings x with $K(x) \geq |x|/2$ and whose YES instances are strings x with $K(x) < |x|/2 - g(|x|)$. Given a function $\alpha : \mathbb{N} \to \mathbb{R}$ such that $\alpha(n) \geq 1$ for all n, the R_K problem with multiplicative gap α is the promise problem whose NO instances are strings x with $K(x) \geq |x|/2$ and whose YES instances are strings x such that $K(x) < |x|/(2\alpha(|x|))$.

Given a time bound $t : \mathbb{N} \to \mathbb{N}$, the K^t complexity of a string is defined as follows: $K(x) = min\{|p| : U(p)$ halts and outputs x within t steps$\}$.

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 $\text{poly}(|x|)s^{O(s)}$.

3 Simulating M^K: the deterministic case

In this section we consider deterministic oracle machines M such that $M^{K'}$ computes L for any context-sensitive oracle K' that is suitably close to K and show (under suitably assumptions) that L is especially simple: either in P or P/poly. Allender, Buhrman, Friedman and Loff [4] use related ideas to show that any computable language that is non-adaptively reducible to K^t for some fixed time bound t is in P/poly. Their task is somewhat simpler because the reduction is to time-bounded Kolmogorov complexity for some fixed time bound; we, on the other hand, need to exploit the robustness of our presumed reduction with respect to approximation.

▶ **Theorem 12.** *Let $\beta = \beta_n(q)$ be a closeness function that is computable in time polynomial in n and $|q|$. Let L be a decidable language and M be a polynomial time deterministic oracle Turing Machine such that L is computed by $M^{K'}$ for any K' that is β-close to K. Let C and r be constants so that the running time of M on input x of length at most n is at most Cn^r.*

1. *Suppose that $\beta_n(q) = (2r+1)\log(n) + \omega_n(1)$. Then $L \in \mathsf{P}$.*
2. *If M is polynomially honest and $\beta_n(q) = \omega(\log(|q|))$ then $L \in \mathsf{P}$.*
3. *If $\beta_n(q) = \delta K(q)$ for some constant $\delta > 0$ then $L \in \mathsf{P/poly}$.*

Proof. Let L and M be as hypothesized.

For each string x, recall that k_x is the number of queries asked by M on input x. Let $Q_x = \{q_x(1), \ldots, q_x(k_x)\}$ be the set of queries asked on input x. Each of the queries $q_x(i)$ can be computed given i,x and M and so $K(q) \leq \log(k_x) + 2K(x) + O(1)$ for any $q \in Q_x$. Since $k_x \leq C|x|^r$ we have that for all x and all $q \in Q_x$:

$$K(q) \leq K(x) + 2r\log(|x|) + O(1). \tag{1}$$

(The factor 2 multiplying r is more than is needed, but is used to keep the expressions simple.)

The proofs of the first and third parts of the theorem have the following structure. In each part we make a specific definition for an oracle U, define B to be the set of strings x such that $M^U(x) \neq L(x)$. If B is finite, we can modify M to the machine $\hat{M}$ which on input x checks whether $x \in B$ and if so outputs $L(x)$ and otherwise runs M^U. The machine $\hat{M}^U$ also computes L. In the first part U is a just the oracle that always outputs 0, and so the resulting computation is in P. In the third part, U is a more complicated oracle, but the operation of U when M^U is run on input of length n can be implemented efficiently given an advice string A_n of polynomial length, and so the resulting computation is in $P/poly$.

So it will suffice to prove that B is finite. We will make use of the following:

▶ **Lemma 13.** *Let z be a string such that for all queries $q \in \{q_z(1), \ldots, q_z(k_z)\}$ we have $|U_z(q) - K(q)| \leq \beta_{|z|}(q)$. Then on input z, M^U outputs $L(z)$. In particular, $z \notin B$.*

Proof. Given such a z define the context-sensitive oracle U' as follows: $U'_y(q)$ is equal to $U_y(q)$ if $y = z$ and $q \in \{q_z(1), \ldots, q_z(k_z)\}$ and is equal to $K(y)$ otherwise. It follows immediately from the hypothesis on z that the oracle U' is β-close to K and therefore by the hypothesis of the theorem $M^{U'}$ computes L correctly on all inputs. Also $M^{U'}$ and M^U behave identically on input z, so $M^U(z)$ outputs $L(z)$. ◀

To prove that B is finite we assume for contradiction that B is infinite. Let F be the subset of $x \in B$ such that x is lexicographically minimum among all strings $y \in B$ of length x. Then F is also infinite. We then prove:

(*) For any sufficiently large $x \in F$, $|K(q) - U_x(q)| \leq \beta_{|x|}(q)$ for every query $q \in Q_x$.

We then select $z \in F$ so that $|z|$ is sufficiently large for this to happen. But then Lemma 13 contradicts that $z \in B$.

We now carry out this strategy for the first and third part of the Theorem.

For the first part of the Theorem, as mentioned, we define U so that $U_x(q) = 0$ for any input x and query q. Note that M^U is a polynomial time oracle-free computation.

Defining B as above, it suffices to show that B is finite. Suppose for contradiction that B is infinite, and define F as above. As noted above, it suffices to show (*) above. Note that for $x \in F$, $K(x) = \log|x| + O(1)$ since x can be described by the machine M, the oracle-free

machine N that computes L (since L is computable) and the number $|x|$. Using (1), for any $q \in Q_x$ we have $K(q) \leq (2r+1)\log(|x|)+O(1)$. By the hypothesis on β, and for x sufficiently large we get that for all $q \in Q_z$, $K(q) \leq \beta_{|x|}(q)$ and therefore $|U(q) - K(q)| \leq \beta_{|x|}(q)$.

The second part of the theorem follows from the first part. Assume that M is polynomially honest. Then there is a constant $\gamma > 0$ so that on input of length n, all queries have size at least n^γ. By hypothesis, $\beta_n(q) = \omega(\log(|q|)) = \omega(\log(n))$, and thus the hypothesis of the first part is satisfied.

For the third part, given δ and r as hypothesized let $D = (2r+2)/\delta$. Define the (context sensitive) oracle U as follows: on input x and oracle query q, $U_x(q)$ outputs $\min(K(q), D\log(n)($. Again we need to prove that the set B is finite. Assuming this is true, we can modify the machine M^V to the machine $\hat{M}^V$ which on input x first checks whether $x \in B$, and if so outputs $L(x)$ and otherwise runs $M^W[x]$. Now $\hat{M}^V$ can be implemented in $\mathsf{P/poly}$ since for any input x of size n, the action of V on queries asked during the computation $M^V(x)$ can be specified by an advice string A_n of polynomial size. Indeed, let S_n be the set of strings of length at most cn^r that satisfy $K(q) < D\log(n)$ and let A_n be the set of ordered pairs $(q, K(q))$ for $q \in S_n$. (Note that during the computation of $M^V(x)$ given a query q, $V_x(q) = D\log(n)$ unless $q \in S_n$, in which case the second part of the ordered pair specifies the query answer.)

For use below, we observe that $K(A_x) \leq D\log(|x|)+O(1)$ since $A_{|x|}$ can be reconstructed by knowing the number $T \leq 2n^D$ of programs of length less than $D\log(|x|)$ that output a string in $S_{|x|}$. (Given T, $S_{|x|}$ can be determined by interleaving the execution of all programs of length less than $D\log(n)$ until T of them halt with an output of length at most cn^r. S_n is then the set of such output strings, and for each string $q \in S_n$, $K(q)$ is the length of the shortest program that output q.)

It remains to prove that B is finite, and again we suppose for contradiction that B is infinite, and define the infinite set F as above. Again we will show that (*) is satisfied. For $x \in F$, $K(x) = \log|x| + K(A_{|x|}) + O(1)$ since since x can be described by the machine M, the oracle-free machine N that computes L (since L is computable) the advice string $A_{|x|}$ (which allows simulation of the oracle V during the computation of $M(x)$) and the number $|x|$. From (1), for any $q \in Q_x$ we have $K(q) \leq (2r+1)\log(|x|) + K(A_x) + O(1) \leq (2r+1+D)\log(|x|) + O(1)$ which is at most $D(1+\delta)\log(|x|)$ provided that $|x|$ is sufficiently large. For any such x, if $K(q) < D\log(|x|)$, then $U_x(q) = K(q)$ so that $|U_x(q) - K(q)| = 0$, or $K(q) \geq D\log(|x|)$, in which case $U_x(q) = D\log(|x|)$ and $K(q) \in [D\log(|x|), D(1+\delta)\log(|x|)]$ and so $|U_x(q) - K(q)| \leq \delta D\log(|x|) \leq \delta K(q)$, establishing (*) as required. ◀

4 Simulating robust K-oracle computation in $AM \cap coAM$

Throughout this section we make the following assumptions.

H1 M is a randomized nonadaptive decision OTM

H2 M^K recognizes the computable language L.

H3 $\beta = \beta_n(q)$ is a computable closeness function.

H4 $\epsilon : \mathbb{N} \longrightarrow [0,1]$.

H5 M is (β, ϵ)-robust with respect to oracle K.

H6 $\alpha : \mathbb{N} \longrightarrow \mathbb{N}$ is an unbounded non-decreasing function.

Also, let k_n^M denote the maximum of k_x^M over x of length n.

Our overall goal is to prove the following theorem:

▶ **Theorem 14.** *Suppose $M, \alpha, \beta, \epsilon$ satisfy hypotheses [H1-H6]. Suppose also that $\beta_n(q) = \log(k_n^M/\epsilon(n)) + 2\log K(q) + \alpha(n) + 5$. Then both L and $\bar{L}$ have constant round Arthur-Merlin protocols for membership whose running time is polynomial in the running time of M and $1/\epsilon$. In particular if M runs in polynomial time then $L \in AM \cap coAM$.*

The rather cumbersome condition on β arises from the proof. It is more natural if the closeness function depends only on the query q and not on the length of the input x to the calling algorithm.

We prove Theorem 14 in two parts. First, in this section, we construct a parameterized family $\{J[\gamma] : \gamma \in [0,1)\}$ of computable context-sensitive oracles J based on simple combinatorial properties of the machine M and show that there exists an integer ℓ such that under [H1]-[H6], $M[\ell]^{J[\gamma]}$ computes L with advantage $\epsilon/2$ for every $\gamma \in [0,1)$.

Then, in the next section, we show that if L satisfies the conclusion of the first part, then it has an efficient AM protocol and coAM protocol.

4.1 Some additional preliminaries

Let (M, x) be a context. As usual, for a set or quantity Z that depends on the context, we denote this dependence by Z_x^M, but we often suppress this dependence and write simply Z. For example we write r for r_x^M and k for k_x^M.

Let $\Gamma = \Gamma_x^M = \{(\rho, i) : (\rho, i) \in \{0,1\}^r \times \{1, \ldots, k\}\}$. Under the assumption stated earlier that k is a power of 2 we have that $\log|\Gamma| = r + \log k$ is an integer, and we can identify Γ with the set $\{0,1\}^{r+\log k}$.

It will be convenient to introduce notation for some elementary functions of positive real numbers.

For a nonnegative integer s:

- ϕ is the function given by $\phi(s) = \lceil \log(\frac{\Gamma}{s}) \rceil = \log(|\Gamma|) - \lfloor \log s \rfloor$.
- μ is the function given by: $\mu(s)$ is the least nonnegative number γ such that $s(1+\gamma)$ is a power of 2. Note that $\mu(s) \in [0,1)$.

It is easy to check:

▶ **Proposition 15.** *For $\gamma \in [0,1)$ and $s > 0$, $\phi((1+\gamma)s) = \phi(s)$ if $\gamma < \mu(s)$ and $\phi((1+\gamma)s) = \phi(s) - 1$ if $\gamma \geq \mu(s)$.*

The following technical fact will be needed later.

▶ **Proposition 16.** *Let $s, t \in \mathbb{N}$ and $\tau \in [0,1)$ with $t \in [s/(1+\tau), s(1+\tau)]$. Let $\gamma \in [0,1)$ be such that $\gamma \notin [\mu(s) - 2\tau, \mu(s) + 2\tau] \cup [1 - 2\tau, 1) \cup [0, 2\tau)$. Then $\phi((1+\gamma)s)) = \phi((1+\gamma)t)$.*

Proof. We prove the contrapositive. Suppose $\phi((1+\gamma)s) \neq \phi((1+\gamma)t)$. Then $\lfloor \log((1+\gamma)s) \rfloor \neq \lfloor \log((1+\gamma)t) \rfloor$. This means that there is an integral power of 2 between them, which means that the interval $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ contains an integral power of 2, say 2^w. Note that $s \leq 2^{w+1} < 8s$. This implies that $(1+\mu(s))s = 2^{w+1}$, or $(1+\mu(s))s = 2^w$, or $(1+\mu(s))s = 2^{w-1}$.

Case 1. Suppose $(1+\mu(s))s = 2^{w+1}$. This means $s > 2^w$, and since $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ contains 2^w, we have that $\tau > \gamma$. Hence $\gamma \in [0, 2\tau)$.

Case 2.Suppose $(1+\mu(s))s = 2^w$. Then $(1+\mu(s))s$ belongs to the interval $[(1+\gamma)s/(1+\tau), (1+\gamma)(1+\tau)s]$ so $1+\mu(s)$ belongs to the interval $[(1+\gamma)/(1+\tau), (1+\gamma)(1+\tau)] \subseteq [1+\gamma-2\tau, 1+\gamma+2\tau]$ which implies $\gamma \in [\mu(s) - 2\tau, \mu(s) + 2\tau]$.

Case 3. Suppose $(1+\mu(s))s = 2^{w-1}$. Then $2 \leq 2(1+\mu(s)) = 2^w/s \leq (1+\gamma)(1+\tau) \leq 1+\gamma+2\tau$ which implies $\gamma \geq 1 - 2\tau$. ◀

4.2 Replacing K by the oracle $J[\gamma]$

As described earlier the context (M, x) determines a function $q_x^M : \Gamma \longrightarrow \{0,1\}^*$ where $q_x^M(\rho, i)$ is the ith query asked by M on input x when the random string is ρ.

For a query q, we define $S(q) = S_x^M(q)$ to be the set $\{(\rho, i) \in \Gamma : q_x^M(\rho, i) = q\}$.

Let $s(q) = |S(q)|$ and let $\pi(q) = \pi_x^M(q) = \frac{s_x^M(q)}{|\Gamma|}$. This is the probability, with respect to (ρ, i) chosen uniformly from Γ that $q(\rho, i) = q$. Note that for the function ϕ defined earlier we have $\phi(s(q)) = \lceil \log \frac{1}{\pi(q)} \rceil$.

For each $\gamma \in [0, 1)$, define the function $J[\gamma]$ on queries by:

$$J[\gamma](q) = J[\gamma]_x^M(q) = \phi((1+\gamma)s(q)).$$

If $\gamma = 0$ we often write $J(q)$ for $J[\gamma](q)$.

From the definition we observe that for all $\gamma \in [0, 1)$:

$$J[\gamma](q) \in [\log(\frac{1}{\pi(q)}) - 1, \log(\frac{1}{\pi(q)}) + 1]. \tag{2}$$

We view $J[\gamma]$ as a context-sensitive oracle and we will see below that $J[\gamma]_x^M$ can be used in place of K in a suitably robust oracle algorithm. The following Lemma relates $J[\gamma]$ to K:

▶ **Lemma 17.** *Let M be a randomized oracle algorithm and x a string. Recall that Q_x^M is the set of queries q such that M on input x asks the query q with positive probability.*

1. *There are constants C_0, C_1 such that for any string $q \in Q_x^M$, $K(q) - J[\gamma]_x^M(q) \leq C_0 K(x) + C_1$.*
2. *For $u > 0$ let $P_x^M(u)$ be the set of strings in Q_x^M such that $J[\gamma]_x^M(q) - K(q) \geq u + 2\log K(q)$. Then $\pi_x^M(P_x^M(u)) \leq 2^{3-u}$.*

Proof. Fix M and x. We omit the superscript M and subscript x from $J[\gamma]$, π, P and Q.

For the first part, order the elements of Q lexicographically by the triple $(1/\pi(q), |q|, q)$. For a query q, let $p(q)$ be the number of $q' \in Q$ that precede q in this order. Given the Turing machine M and input x we can construct a Turing machine of size $O(K(x)) + O(1)$ that takes as input an integer $h \in \{0, \ldots, |Q| - 1\}$ and outputs q such that $p(q) = h$.

Therefore $K(q) \leq O(K(x)) + \log p(q) + O(1)$. Since there are at most $1/\pi(q)$ members of Q that precede q in the order, we have $p(q) \leq (1/\pi(q))$ and therefore $\log p(q) \leq \log(1/\pi(q)) \leq J[\gamma](q) + 1$. Therefore $K(q) - J[\gamma](q) \leq O(K(x)) + O(1)$, as required.

For the second part, (2) implies $\pi(q) \leq 2^{1-J[\gamma](q)}$. For $q \in P(u)$ we therefore have $\pi(q) \leq 2^{1-K(q)-u}/K(q)^2$. Summing over all $q \in P(u)$ this is at most $2^{1-u} \sum_{q \in P(u)} 2^{-K(q)}/K(q)^2$ which is at most 2^{2-u} by Proposition 9. ◀

4.3 Replacing a K oracle by a $J[\gamma]$ oracle

In this section we prove:

▶ **Theorem 18.** *Assume hypotheses [H1]-[H6] and that $\beta_{|x|}(q) = \log(k_x^M/\epsilon(|x|)) + 2\log K(q) + \alpha(|x|) + 5$. Then there is an integer ℓ such that for all $\gamma \in [0, 1]$, $M[\ell]^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$, where $M[\ell]$ is the OTM that outputs $L(x)$ on any input x of length less than ℓ and otherwise executes M on x. Letting $N = M[\ell]$, there is an OTM N such that $N^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$.*

Proof. Fix M. Say that x is bad for γ if $M^{J[\gamma]}(x)$ outputs $L(x)$ with probability less than $\frac{1}{2}(1+\epsilon(|x|)/2)$ and let $B[\gamma]$ be the set of strings that are bad for γ. Let $B = \cup_{\gamma\in[0,1)}B[\gamma]$.

If B is finite, let ℓ be the length of the largest string in B. Then $M[\ell]$ satisfies the desired conclusion. So we assume that B is infinite and derive a contradiction.

Note that $x \in B$ holds if and only if one of the infinitely many conditions $x \in B[\gamma]$ holds. The following Proposition reduces this to a finite set of conditions.

▶ **Proposition 19.** *Let $x \in \{0,1\}^*$ and let $\Gamma_x = \{\mu(s(q)) : q \in Q_x^M\}$ (where Q_x^M is the (finite) set of queries that are asked with nonzero probability by M on input x and μ is the function defined in the definition of $J[\gamma]$). Then $x \in B$ if and only if $x \in B[\gamma]$ for some $\gamma \in \Gamma_x$.*

Proof. For each $\gamma \in [0,1)$ let $\gamma^- = \max\{\mu(s(q)) : q \in Q_x^M, \mu(s(q)) \le \gamma\}$. By the definition of $J[\gamma]$, we have $J[\gamma](q) = J[\gamma^-](q)$ for all $q \in Q$. Therefore $M^{J[\gamma]}(x)$ behaves identically to $M^{J[\gamma-](x)}$ and so $x \in B$ if and only if there exists $\gamma \in \Gamma_x$ such that $x \in B[\gamma]$. ◀

Let us define γ_x to be the least $\gamma \in \Gamma_x$ for which $x \in B[\gamma_x]$.

The set Γ_x is computable from x, and for each $\gamma \in \Gamma_x$ we can computably determine the probability that $M^{J[\gamma]}(x)$ agrees with $L(x)$. Therefore there is an (instantiated) program of size $|M| + O(1)$ that on input x, tests whether $x \in B$.

Let $\beta^*(n)$ be the minimum over all q and all $n' \ge n$ of $\beta_{n'}(q)$. To obtain the desired contradiction we will prove:

▶ **Lemma 20.** *For any string x there is an oracle $K[x]$ satisfying:*

- *$K[x]$ is β-close to K and therefore the probability that $M^{K[x]}$ on input x differs from M^K on input x is at most $(1-\epsilon(|x|))/2$.*
- *If $K(x) < (\beta^*(|x|) - C_1)/C_0$ (where C_0, C_1 are given in Lemma 17) then the probability that $M^{K[x]}$ on input x differs from M^J on input x is at most $\epsilon(|x|)/4$.*

The lemma implies any string x satisfying $K(x) < (\beta^*(|x|) - C_1)/C_0$ is not in B, since the probability that M^J on input x disagrees with M^K on input x can be upper bounded by $(1-\epsilon(|x|)/2 + \epsilon(|x|)/4 \le (1-\epsilon(|x|)/2)/2$. But by applying the second part of Proposition 10 to the language B and $\kappa(n) = (\beta^*(n) - C_1)/C_0$ there is a string $x \in B$ such that $K(x) < (\beta(|x|) - C_1)/C_0$, which yields the desired contradiction. Note that κ is an unbounded non-decreasing function as required for the application of Proposition 10 by the definition of β^* and the fact that α is an unbounded non-decreasing function.

So it remains to prove Lemma 20. Given x, we define the oracle $K[x]$ as follows. Recall that Q_x^M is the set of queries q that are made by M with non-zero probability when the input is $x(w)$. Let Q_{CLOSE} be the set of $q \in Q_x^M$ for which $|J[\gamma]_x^M(q) - K(q)| \le \beta_{|x|}(q)$, let $Q_>$ be the set of $q \in Q_x^M$ for which $J[\gamma_x]_x^M(q) > K(q) + \beta_{|x|}(q)$ and let $Q_<$ be the set of $q \in Q_x^M$ for which $J[\gamma_x](q) < K(q) - \beta_{|x|}(q)$ Define the oracle $K[x]$ as follows: $K[x](q) = J[\gamma_x](q)$ if $q \in Q_{\text{CLOSE}}$ and $K[x](q) = K(q)$ otherwise.

By definition $K[x]$ is β-close to K and therefore by hypothesis $M^{K[x]}$ computes L with advantage ϵ. In particular $M^{K[x]}$ on input x differs from $L(x)$ with probability at most $(1-\epsilon(|x|))/2$. This completes the first part of the lemma.

For the second part of the lemma, assume that x is such that $K(x) < (\beta(|x|) - C_1)/C_0$ (where C_0, C_1 are given in Lemma 17).

If all of the queries asked by M on input x are in Q_{CLOSE} then all query answers during the computation $M^{J[\gamma]}(x)$ are identical to those during the computation $M^{K[x]}(x)$, and therefore the output is the same. .

So it now suffices to prove an upper bound of $\epsilon(x)/4$ on the probability that at least one of the selected queries on input x belongs to $Q_< \cup Q_>$.

From the hypothesis, we have $K(x) \leq (\beta(|x|) - C_1)/C_0$. We now show that this implies that $Q_< = \emptyset$. By the first part of Lemma 17, for all queries $q \in Q_x^M$ we have $K(q) - J_x^M(q) \leq C_0 K(x) + C_1$ which is at most $\beta(|x|)$ and so $q \notin Q_<$.

Hence we only need to upper bound the probability that, on input x, at least one query belongs to $Q_>$. For brevity let $k = k_x^M$. For $i \in \{1, \ldots, k\}$ let p_i be the probability that the ith query is in $Q_>$. Let $\sigma = \sum_i p_i$, it suffices to show that $\sigma \leq \epsilon(|x|)/4$. Note that σ/k is the chance that a random query i (with respect to the random bits and the choice of $i \in \{1, \ldots, k\}$) is in $Q_>$. By the second part of Lemma 17, with $u = \beta(|x|) - 2\log(K(q))$, $p \leq 2^{2+2\log(K(q))-\beta(|x|)}$. Using the hypothesis on $\beta(|x|)$ this is at most $\epsilon(|x|)/4k$ and therefore $\sigma \leq \epsilon(|x|)/4$, as required. ◀

5 The Arthur-Merlin protocol

The starting point for this section is the conclusion of Theorem 18:

> **Hypothesis AM(N).** N is an oracle machine such such that for all $\gamma \in [0, 1)$, $N^{J[\gamma]}(x)$ computes L with advantage $\epsilon/2$.

We will show that this implies that there is an Arthur-Merlin protocol for L whose running time is in the running time of N.

We make use of two known AM protocols. The input to both protocols includes:

- A function $f : \{0,1\}^w \longrightarrow \{0,1\}^*$ given by a circuit C_f. The function f induces a probability distribution π_f on $\{0,1\}^*$, with $\pi_f(q)$ equal to $|f^{-1}(q)|/2^2$, which is the probability that a uniformly chosen element of $\{0,1\}^w$ maps to q.
- $\zeta, \delta > 0$ are input parameters for the protocol.

▶ **Lemma 21** (Entropy Estimation Protocol, [24]). *There is a constant round AM-protocol that takes as input a circuit C_f as above, an integer h and an error parameter δ, runs in time $|C_f| poly(\log(1/\delta))$, and outputs* ACCEPT *or* REJECT *with the following conditions*

1. *If $|h - H(\pi_f)| \leq 1$ then Merlin has a strategy that causes Arthur to reject with probability at most δ.*
2. *If $|h - H(\pi_f)| \geq 2$ then the probabilty that the verification is accepted is at most δ.*

▶ **Lemma 22** (Lower Bound protocol, [25]). *There is a constant round protocol with input C_f, input sequences $q_1, \ldots, q_m \in (\{0,1\}^*)^m$ and $(s_1, \ldots, s_m) \in \mathbb{N}^m$, and error parameters (ζ, δ) both in $(0, 1)$ that runs in time $C_f poly(m, 1/\zeta, \log(1/\delta))$, and outputs either* ACCEPT *or* REJECT *according to the following:*

1. *If $s_i \geq |f^{-1}(q_i)|$ for every i, then Merlin has a strategy such that the protocol rejects with probability at most δ.*
2. *If $s_i \leq |f^{-1}(q_i)|/(1 + \zeta)$ for some i then for any strategy of Merlin the probabilty that the verification is accepted is at most δ.*

We fix some notation for this section:

- x denotes the string whose membership in L is to be determined.
- $r = r_x^N$ is the length of the random string generated by N on input x.
- $k = k_x^N$ is the number of queries asked by N on input x. We assume (with no significant loss of generality) that $\log k$ is an integer.
- For $\rho \in \{0,1\}^r$, $q_1(\rho), \ldots, q_k(\rho)$ denotes the queries asked by N on input x.
- f denotes the function on $\{0,1\}^{r+\log k}$ that maps (ρ, i) to $q_i(\rho)$.
- Let $\pi = \pi_f$ be the probability distribution on $\{0,1\}^*$ induced by f, which is given by $\pi_f = |f^{-1}(q)|/2^{r+\log k}$.

- $Q = Q_x^N$ is the image of the function f, which is the set of queries that are asked with positive probability.
- For $\rho \in \{0,1\}^r$, $a_1[\gamma](\rho), \ldots, a_k[\gamma](\rho)$ is the sequence of answers from the oracle $J[\gamma]$ to the queries $q_1(\rho), \ldots, q_k(\rho)$.
- When the random string is ρ, the output of the algorithm is $\text{OUT}(a_1[\gamma](\rho), \ldots, a_k[\gamma](\rho); \rho)$. We denote this by $z[\gamma](\rho)$.
- $\epsilon = \epsilon_{|x|}^N$.

We note the following technical fact:

▶ **Proposition 23.** *Let π be a probability distribution over $\{0,1\}^w$.*

1. *If q is selected according to π, $\mathrm{E}[\log(1/\pi(q))] = H(\pi)$ where $H(\pi)$ is the binary entropy of π.*
2. *Suppose $q_1, \ldots, q_t$ are selected independently according to π. Then*

$$\mathrm{Prob}[\frac{1}{t}\sum_{i=1}^{t} \log(1/\pi(q_i)) - H(\pi)| > \zeta] \leq \frac{w^2}{t\zeta^2}.$$

Proof. The first part follows directly from the definition of $H(\pi)$. For the second part, let $Z_i = \log(1/\pi(q_i))$ and let $Z = \frac{1}{t}\sum Z_i$. Note that $\mathrm{E}[Z] = \frac{1}{t}\sum \mathrm{E}[Z_i] = H(\pi)$ by the first part. Also, note that since Z is an average of t i.i.d. random variables, it's variance is $\frac{1}{t}\mathrm{Var}(Z_i) \leq \frac{w^2}{t}$ since a random variable taking values in $[0,w]$ has variance at most w^2. By Chebyshev's inequality, for any random variable X, the probability that $|Z - \mathbb{E}[Z]| > \zeta$ is at most $Var(Z)/\zeta^2 \leq \frac{w^2}{t\zeta^2}$. ◀

In describing the Arthur-Merlin protocol we refer to the actions of "Honest Merlin" which is the ideal behavior for Merlin (which Merlin may not follow).

The protocol makes use of several parameters, whose values are determined by the analysis. For reference we list these parameters here with their values:

- t is the *sample complexity*, and it set to $t = 32000\frac{\max(r,1)^2k^2}{\epsilon(|x|)^2}$.
- τ is the *safety parameter*, which is set to $\tau = \frac{\epsilon(|x|)}{16(4k+2)}$
- ν is the *lower bound parameter*, which is set to $\nu = \frac{\epsilon(|x|)^2}{20000k^2}$.
- D is the *precision parameter* and is a positive integer. We choose $D = \lceil \frac{16(4k+2)}{\epsilon(|x|)} \rceil$.

It is straightforward to verify that these parameter values satisfy the following conditions:

▶ **Proposition 24.**

$$\begin{aligned}
\tau &\leq \frac{\epsilon(|x|)}{16(4k+2)} \\
\nu &\leq \tau\epsilon(|x|)/128k \\
t &\geq \frac{256k}{\epsilon(|x|)}\log\frac{16}{\epsilon(|x|)} \\
t &\geq 16\frac{(r+\log k)^2}{\epsilon(|x|)}. \\
t &\geq \frac{320k}{\epsilon(|x|)\tau}. \\
D &\geq \frac{1}{\tau}. \\
D &\geq \frac{32(k+1)}{\epsilon(|x|)}
\end{aligned}$$

The inequalities stated in this Proposition are needed in the analysis of the protocol. The values of the parameters were chosen to ensure these conditions.

We are now ready to state the protocol:

Step 1. Merlin provides an integer h. Arthur and Merlin perform the Entropy estimation protocol of Lemma 21 for the probability distribution π and input h with error parameter $\delta = \epsilon(|x|)/16$. (If Merlin is honest, then Merlin sets h so that $|h - H(\pi)| \leq 1$, and adopts the strategy from the first part of Lemma 21 that makes the rejection probability at most $\epsilon(|x|)/16$.)

Step 2. Random strings $\rho_1, \ldots, \rho_t$, each of length r, are generated using public coins. Let χ be the $t \times k$ matrix with $\chi_{i,j} = f(\rho_i, j)$, the jth query asked when the random string is ρ_i.

Step 3. Merlin provides a t by k positive integer matrix B (Honest Merlin sets B to be equal to the matrix B^* whose i, j entry is $|f^{-1}(\chi_{i,j})|$.)

Notational Remark. It is convenient here to define some matrices and vectors that are determined from B. For $\gamma \in [0, 1)$ the $t \times k$ matrix $A[\gamma]$ is given $A[\gamma]_{i,j} = \phi((1+\gamma)B_{i,j})$, and the vector $z[\gamma] \in \{0,1\}^t$ is given by $z[\gamma]_i = \textsc{out}(A[\gamma]_{i,1}, \ldots, A[\gamma]_{i,k})$. The matrix $A^*[\gamma]$ and vector $z^*[\gamma]$ are obtained analogously with B replaced by B^*. Note that row i of $A^*[\gamma]$ is exactly the sequence of query answers provided by oracle $J[\gamma]$ when the random string is ρ_i and $z^*[\gamma]_i$ is the output of $N^{J[\gamma]}(x)$ for random string is ρ_i.

Step 4. For each row index i, Arthur randomly chooses $\hat{j}(i) \in \{1, \ldots, k\}$. Define

$$\sigma(B) = \frac{1}{t} \sum_i (r - \log(B_{i,\hat{j}(i)})).$$

Arthur computes $\sigma(B)$ approximately to determine an integer H' that belongs to the interval $[\sigma(B) - 1, \sigma(B) + 1]$. Arthur rejects this step if $|H' - h| > 3$ and accepts this step otherwise.

Step 5. Arthur-Merlin perform the lower bound protocol of Lemma 22 for the above-mentioned function f that maps (ρ, i) to $q_i(\rho)$, with error parameters $(\zeta, \delta) = (\nu, \epsilon(|x|)/16)$. The input $(q_1, \ldots, q_m), (s_1, \ldots, s_m)$ to the protocol is the sequence of tk entries of the matrix χ and the corresponding sequence of entries of the matrix B. (If Merlin is Honest then since $B = B^*$, $s_i = |f^{-1}(q_i)|$ for every i and Merlin follows the strategy from the first part of Lemma 22 so that the probability of rejection is at most $\epsilon(|x|)/16$.)

Step 6. Arthur selects a uniformly random $\hat{\lambda} \in \{1, \ldots, D\}$ and a uniformly random $\hat{i} \in \{1, \ldots, t\}$. $\hat{z}$ is defined to be $z[\lambda/D]_{\hat{i}}$. For future reference we also define $\hat{z}^*$ to be $\hat{z}$ if B is replaced by B^*.

Step 7. If any of the steps 1,4 or 5 result in REJECT, then the protocol outputs FAILURE. Otherwise the output is $\hat{z}$.

This protocol outputs either FAILURE or 0 or 1. The main theorem of this subsection is:

▶ **Theorem 25.** *Under the hypotheses AM(N) given earlier, the above protocol satisfies:*

1. *If Merlin behaves honestly, then the protocol outputs $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/16$.*
2. *For any behavior of Merlin, the probability that the protocol outputs FAILURE or outputs $L(x)$ is at least $\frac{1}{2} + \epsilon(|x|)/16$.*

▶ **Corollary 26.** *Both L and $\bar{L}$ can be recognized by AM protocols that run in time polynomial in the $1/\epsilon$ and the running time of N.*

Proof. The protocol obtained by changing a FAILURE output to 0 (resp. 1) is an AM protocol (resp. coAM protocol) for L. ◀

Proof of Theorem 25. We first note the following:

▶ **Proposition 27.** *The value $\hat{z}^*$ determined in Step 6 in the case that $B = B^*$ agrees with $L(x)$ with probability at least $\frac{1}{2} + \frac{\epsilon(|x|)}{4}$.*

Proof. The value $\hat{z}^*$ determined in Step 6 is equal to the output of $N^{J[\lambda/D]}$ on input x when the random string is $\rho_{\hat{i}}$. Since $\rho_{\hat{i}}$ is a uniformly random string, the hypothesis on N implies that this is equal to $L(x)$ with probability at least $\frac{1}{2} + \frac{\epsilon(|x|)}{4}$. ◀

We now prove the first part of the Theorem. For $s \in \{1, 4, 5\}$ define the event R_s to be the event that the test in Step s outputs REJECT.

Assume Merlin behaves honestly, and therefore $B = B^*$. By the above proposition, the probability that the output of the protocol is $L(x)$ is at least:

$$\frac{1}{2} + \frac{\epsilon(|x|)}{4} - \text{Prob}[R_1 \vee R_4 \vee R_5] \geq \frac{1}{2} + \frac{\epsilon(|x|)}{4} - \text{Prob}[R_1] - \text{Prob}[R_4] - \text{Prob}[R_5].$$

As noted in the description of Steps 1 and 5, $\text{Prob}[R_1]$ and $\text{Prob}[R_5]$ are each at most $\epsilon(|x|)/16$. So it suffices to show that $\text{Prob}[R_4] \leq \epsilon(|x|)/16$, which we now do.

Step 4 is rejected only if $|H' - h| > 3$. We have:

$$|H' - h| \leq |H' - \sigma(B^*)| + |\sigma(B^*) - H(\pi)| + |H(\pi) - h|.$$

$|H' - \sigma(B^*)| \leq 1$ by the choice of H' and the fact that $B = B^*$, and $|H(\pi) - h| \leq 1$ since Merlin is assumed to be honest. Therefore $\text{Prob}[R_4] \leq \text{Prob}[|\sigma(B^*) - H(\pi)| > 1$. So it suffices to prove:

▶ **Proposition 28.** $\text{Prob}[|\sigma(B^*) - H(\pi)| > 1] \leq \epsilon(|x|)/16$.

Proof. By the definition of B^*, $B^*_{i,\hat{j}} = 2^{r+\log k}\pi(\chi_{i,\hat{j}(i)})$ and so $r + \log k - \log(B^*_{i,\hat{j}(i)}) = \log(1/\pi(\chi_{i,\hat{j}(i)}))$ and so $\sigma(B^*) = \frac{1}{t}\sum_{i=1}^{t} \log(1/\pi(\chi_{i,\hat{j}(i)}))$. Since $\chi_{1,\hat{j}(1)}, \ldots, \chi_{t,\hat{j}(t)}$ is a sequence of independent samples from the distribution π, Proposition 23 implies

$$\text{Prob}[\frac{1}{t}\sum_i \log(1/\pi(\chi_{i,\hat{j}(i)})) - H(\pi)| > 1] < \frac{(r + \log k)^2}{t} \leq \epsilon(|x|)/16,$$

where the last inequality used Proposition 24. ◀

This completes the proof of the first part of the theorem.

We now turn to the proof of the second part. We consider an arbitrary strategy for Merlin. We break up into cases according to properties of the matrix B.

We divide into three cases.

Case 1. There is at least one entry (i, j) with $B_{i,j} < B^*_{i,j}/(1 + \nu)$. By the property of the Lower bound protocol performed in Step 5, this will reject with probability at least $1 - \epsilon(|x|)/16 \geq \frac{1}{2}) + \epsilon(|x|)/4$.

Case 2. $B_{i,j} \geq B^*_{i,j}/(1+\nu)$ for all entries (i,j) and there are more than $\epsilon(|x|)t/16$ entries with $B_{i,j} > (1+\tau)B^*_{i,j}$. We claim that Step 4 will reject with probability at least $1-\epsilon(|x|)/8 \geq 1/2+\epsilon/4$. For this we give a lower bound on the probability $H' - H(\pi)$ exceeds 3. We have

$$\begin{aligned} H' - H(\pi) &= \sigma(B) - \sigma(B^*) - (\Sigma(B) - H') - (H(\pi) - \sigma(B^*)) \\ &\geq \sigma(B) - \sigma(B^*) - |\sigma(B) - H'| - |H(\pi) - \sigma(B^*)|. \end{aligned}$$

By the choice of H', $|\sigma(B) - H'| \leq 1$. Therefore:

$$\begin{aligned} \text{Prob}[H' - H(\pi) > 3] &\geq \text{Prob}[(\sigma(B) - \sigma(B^*) \geq 5) \wedge (|\sigma(B^*) - H(\pi)| < 1)] \\ &\geq 1 - \text{Prob}[\sigma(B) < \sigma(B^*) + 5] - \text{Prob}[|\sigma(B^*) - H(\pi)| \geq 1]. \end{aligned}$$

The desired lower bound will follow by showing that each of the two probabilities on the right is at most $\epsilon(|x|)/16$. Proposition 28 shows this for the second probability. So we now bound the first probability.

For $i \in \{1,\ldots,t\}$ define:

- $V_i = \{j \in \{1,\ldots,k\} : \log B_{i,j} \geq (1+\tau)B^*_{i,j}$. (Note that $\sum_i V_i \geq \epsilon(|x|)t/16$ by the case assumption.)
- Y_i is the 0-1 indicator of the event that $\hat{j}(i) \in V_i$, and $Y = \sum_i Y_i$.

We now make two claims.

Claim 1. $\sigma(B) - \sigma(B^*) \geq \tau Y - \nu t$.

Claim 2. $\text{Prob}[Y < \frac{1}{2}\frac{\epsilon(|x|)t}{16k}] \leq \epsilon(|x|)/16$.

If $Y \geq \frac{1}{2}\frac{\epsilon(|x|)t}{16k}$, then by the conditions on τ and ν given by Proposition 24 we have that $\tau Y - \nu t \geq 5$ and so

$$\text{Prob}[\sigma(B) - \sigma(B^*) < 5] \leq \text{Prob}[Y < \frac{1}{2}\frac{\epsilon(|x|)t}{16k}] \leq \epsilon(|x|)/16.$$

To prove the first claim, note that for i such that $Y_i = 1$, we have

$$\log(B_{i,\hat{j}(i)}) \geq \log((1+\tau)B^*_{i,\hat{j}(i)} \geq \log(B^*_{i,\hat{j}(i)}) + \tau$$

(since $\log(1+\tau) \geq \tau$ for $\tau \in [0,1]$). For i such that $Y_i = 0$, we have $B_{i,\hat{j}(i)} \geq B_{i,\hat{j}(i)}/(1+\nu)$ which implies $\log B_{i,\hat{j}(i)} \geq \log B_{i,\hat{j}(i)} - 2\nu$ since $\log(1+\nu) \leq 2\nu$). Therefore $\sigma(B) - \sigma(B^*) \geq \tau Y - \nu t$.

For the second claim, note that that $\text{E}[Y] = \sum_i \text{E}[Y_i] = \sum_i \frac{|V_i|}{k} \geq \frac{\epsilon(|x|)t}{16k}$, by the case assumption. The "multiplicative Chernoff bound" implies that if Y is a sum of independent 0-1 random variables, then $\text{Prob}[Y \leq \frac{1}{2}\text{E}[Y]] \leq e^{-\text{E}[Y]/8}$. Applying this here we obtain $\text{Prob}[Y \leq \frac{\epsilon(|x|)t}{32k} \leq e^{-\epsilon(|x|)t}256k \leq \epsilon(|x|)/16$ by Proposition 24.

Case 3. $B_{i,j} \geq B^*_{i,j}/(1+\nu)$ for all entries (i,j) and there are at most $\epsilon(|x|)t/16$ entries with $B_{i,j} > (1+\tau)B^*_{i,j}$.

By Proposition 27, $\hat{z}^*$ is equal to $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/4$. We show that under the case assumption the probability that $\hat{z} \neq \hat{z}^*$ is at most $\epsilon(|x|)/8$ which will show that $\hat{z}$ is equal to $L(x)$ with probability at least $\frac{1}{2} + \epsilon(|x|)/8$.

We say that a row index $i \in \{1,\ldots,t\}$ is *unsafe* if there is at least one j such that $B_{i,j} > (1+\tau)B^*_{i,j}$ and is *safe* otherwise. We identify the following "bad" events:

B1 The selected row index $\hat{i}$ is unsafe.

B2 The selected $\hat{\lambda}$ satisfies $\hat{\lambda}/D \in [0,2\tau) \cup [1-2\tau,1) \cup \bigcup_{j=1}^{k}[\mu(B^*_{\hat{i},j}) - 2\tau, \mu(B^*_{\hat{i},j}) + 2\tau]$.

We claim that if neither bad event happens then $\hat{z} = \hat{z}^*$. Since [B1] doesn't happen, $\hat{i}$ is safe. Together with the case assumption and the fact that $\nu \leq \tau$ this implies that for each $j \in [k]$, $B_{\hat{i},j} \in [B^*_{\hat{i},j}/(1+\tau), B^*_{\hat{i},j}(1+\tau)]$. Since [B2] doesn't happen, then for all $j \in \{1,\ldots,k\}$, the hypotheses of Proposition 16 hold for $(s,t) = (B_{\hat{i},j}, B^*_{\hat{i},j})$ from which we conclude that for all $j \in \{1,\ldots,k\}$

$$A[\lambda/D]_{\hat{i},j} = \phi((1+\lambda/D)B_{\hat{i},j})) = \phi((1+\lambda/D)B^*_{\hat{i},j})) = A^*[\lambda/D]_{\hat{i},j},$$

which implies that $\hat{z}[\lambda/D] = \hat{z}^*[\lambda/D]$.

So now it will be enough to show that Prob$[B1]$ and Prob$[B2]$ are both at most $\epsilon(|x|)/16$. Prob$[B1]$ is equal to the fraction of unsafe rows, which is at most $\epsilon(|x|)/16$ by the case assumption.

Since λ is chosen uniformly from $\{1,\ldots D\}$, Prob$[B2]$ is the fraction of integers in $\{1,\ldots,D\}$ that belong to the set $[0,2D\tau) \cup [D(1-2\tau), D) \cup \bigcup_{j=1}^{k} [D(\mu(B^*_{\hat{i},j}) - 2\tau), D(\mu(B^*_{\hat{i},j}) + 2\tau)]$

Now we use the fact that the number of integers in an interval $[a,b)$ is less than $b-a+1$ and an interval $[a,b]$ is at most $b-a+1$. Therefore the number of integers in $[0,2D\tau) \cup [D(1-2\tau), D)$ is less than $4\tau D+2$ and the number of integers in each set of the big union is at most $4\tau D+1$. Summing up over all intervals the number of integers in the union is at most $(4k+4)\tau D+k+1$. Since $\tau D \leq 1/4$ by Proposition 24, this is at most $2k+2$. So the probability that λ is in this set is at most $(2k+2)/D$ which is at most $\epsilon(|x|)/16$ by Proposition 24. ◀

6 A Dichotomy

We first make Theorem 14 more concrete by identifying some interesting parameter settings. One natural setting is where the number of queries is polynomial and the advantage is inverse polynomial.

▶ **Theorem 29.** *Suppose L is computable and there is a randomized polynomial-time non-adaptive reduction F from L to $\omega(\log(|q|))$-additively approximating Kolmogorov complexity such that F is polynomially honest and has inverse polynomial advantage. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Proof. We apply Theorem 14 with $k^M_n = \mathrm{poly}(n)$, $\epsilon(n) = 1/\mathrm{poly}(n)$, and $\alpha(n) = O(\log(n))$. Hence we have $\beta_n(q) = O(\log(n))$. Since F is polynomially honest, we have that $\beta_n(q) = O(\log(q))$ for every $q \in Q^M_x$ for x of length n. Hence by the conclusion of Theorem 14, we have $L \in \mathsf{AM} \cap \mathsf{coAM}$. ◀

Another natural setting is where the number of queries and the advantage are both constant. In this case, we can get a conclusion from a smaller upper bound on the approximation gap.

▶ **Theorem 30.** *Suppose L is computable and there is a computable function $\eta = \omega(1)$ such that there is a randomized polynomial-time non-adaptive reduction F from L to $(2\log(K(q)) + \eta(|q|))$-additively approximating Kolmogorov complexity such that F makes $O(1)$ queries, is weakly honest and has advantage $\Omega(1)$. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$.*

Proof. Let $\lambda : \mathbb{N} \to \mathbb{N}$ be an unbounded function such that F is λ-honest. When applying Theorem 14, we choose $k^M_n = O(1)$ and $\epsilon(n) = \Omega(1)$. We also choose α appropriately so that $\beta_n(q) \geq 2\log(K(q)) + \eta(|q|)$ - this is possible since F is weakly honest, by ensuring $\alpha(\lambda^{-1}) = o(\eta)$. Once more, applying Theorem 14 gives us the desired condlusion. ◀

Theorem 29 supplies the first part of our dichotomy result. We require the following result of Hirahara [28] for the second part of our dichotomy. This is Theorem 6.3 in [28], where it is stated without mentioning the honesty property of the reduction. The proof of Theorem 6.3 in [28] does yield a polynomially honest reduction.

▶ **Theorem 31** ([28]). *For any constant $c \in \mathbb{N}$, there exists an $\mathsf{EXP}^{\mathsf{NP}}$ computable function $G = \{G_n\}$, where G_n maps $n - c\log(n)$ bits to n bits, such that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to distinguishing G from uniform.*

▶ **Corollary 32.** *For every constant $c \in \mathbb{N}$ and any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time non-adaptive reduction F from L to $c\log(|q|)$-additively approximating Kolmogorov complexity such that F is polynomially honest and has constant advantage.*

Proof. Let $c \in \mathbb{N}$, and let K' be any oracle that $c\log(|q|)$-additively approximates Kolmogorov complexity. Wlog we assume $c \geq 3$ - note that establishing the conclusion for a constant c also establishes it for any constant $c' \leq c$. Applying Theorem 31, we have that there is a computable function $G = \{G_n\}$, where G_n maps $n - 3c\log(n)$ bits to n bits such that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to D, where D is any language distinguishing G from the uniform distribution. We define $D(q) = 1$ if $K'(q) < |q| - 1.5c\log(|q|)$. Note that any output of the generator G can be described with $\log(n) + n - 3c\log(n) + O(1)$ bits, and hence has Kolmogorov complexity $< n - 2.5c\log(n)$ for large enough n. This implies that for every n-bit output y of the generator G_n, $K'(y) < n - 1.5c\log(n)$. On the other hand, with probability at least $1 - 1/n^{c/2}$ over $y \sim U_n$. we have that $K(y) \geq n - 0.5c\log(n)$, and hence $K'(y) \geq n - 1.5c\log(n)$. Thus K' does distinguish the output of G_n from uniform. Since D' can be computed with a single deterministic query to K' of the same length as the input, we have that for any $L \in \mathsf{NEXP}$, there is a randomized polynomial-time polynomially honest non-adaptive reduction from L to K'. Since this holds for any K' that $c\log(|q|)$-additively approximates Kolmogorov complexity, the result follows. ◀

Theorem 29 together with Corollary 32 establishes the dichotomy result in Theorem 3.

7 On Randomized Reductions to K^t

In this section, we show limitations on randomized nonadaptive reductions to K^t for polynomially bounded t. We first need to define the notion of a *hitting set generator*.

▶ **Definition 33.** *Given integers n and $\ell < n$ and rational error parameter $\epsilon \in [0,1]$ with $\epsilon \geq 1/2^n$, a hitting set generator $H_{n,\epsilon}$ for size n with seed length ℓ and error ϵ is a function from $\ell(n)$ bits to n bits, such that for any circuit C_n of size n, if C_n accepts at least an ϵ fraction of inputs of length n, we have that $\Pr_{z\in\{0,1\}^\ell}[C_n(H_{n,\epsilon}(z)) = 1] > 0$. Given a function $\ell : \mathbb{N} \times [0,1] \to \mathbb{N}$, we say that a sequence $H_{n,\epsilon}$ of functions is a polynomial-time computable hitting set generator with seed length ℓ if for each integer n and rational $\epsilon \in [0,1]$ such that $\epsilon \geq 1/2^n$, $H_{n,\epsilon}$ is a hitting set generator for size n with error ϵ and seed length $\ell(n,\epsilon)$, and moreover $H_{n,\epsilon}$ can be computed in time $\mathrm{poly}(n, \log(1/\epsilon))$ when given n and $\lceil\log(1/\epsilon)\rceil$ in unary.*

A key component of our proof is a construction of hitting set generators with optimal dependence on ϵ in the seed length [19].

▶ **Theorem 34** ([19]). *If* E *requires non-deterministic circuits of size* $2^{\Omega(n)}$*, then there is a polynomial-time computable hitting set generator* H *with seed length* $O(\log(n)) + \log(1/\epsilon)$.

It is crucial for our results that the constant factor in front of the $\log(1/\epsilon)$ term is 1; if we could afford an arbitrary constant factor there, we would only need an assumption on hardness for deterministic circuits. We note that the result of [19] is stated for ϵ with a fixed dependence on n, but it is easy to see that their proof works also if $\log(1/\epsilon)$ is given as an independent parameter to the algorithm computing the hitting set generator.

We next use Theorem 34 to argue that under a standard derandomization assumptions, the Kolmogorov complexity of a typical sample from a polynomial-time samplable distributions does not differ by much from the time-bounded Kolmogorov complexity. We require that the time bound when measuring time-bounded Kolmogorov complexity is large enough compared to the time required for sampling.

▶ **Lemma 35.** *Let* $t' : \mathbb{N} \to \mathbb{N}$ *be a polynomially bounded function, and* $\{D_m\}$ *be a sequence of distributions samplable in time* $t'(m)$*, where for each* $m \in \mathbb{N}$*,* D_m *is supported on* m*-bit strings. Suppose that* E *requires non-deterministic circuits of size* $2^{\Omega(n)}$*. Then there is* $t : \mathbb{N} \to \mathbb{N}$ *such that* $t = \text{poly}(t')$ *and* $\Pr_{x \sim D_m}[K^t(x) - K(x) = \omega(\log(m))] = 1/m^{\omega(1)}$.

Proof. Let $\{D_m\}$ be a sequence of distributions, where D_m is supported on m-bit strings, sampled by an algorithm A running in time $t'(m)$. Using the standard simulation of deterministic time by circuit size, A can be simulated by a sequence of circuits $\{C'_m\}$, where C'_m is of size at most $t'(m)^2$ for large enough m. Also, let $f = \omega(\log(m))$ be an arbitrary function.

Let z be a fixed string of length at most $n = 2t'(m)^2$ that is assigned probability ϵ by D_m. We argue that the polynomial-time bounded Kolmogorov complexity of z is at most $\log(1/\epsilon) + O(\log(n))$, under the assumption that E required non-deterministic circuits of size $2^{\Omega(n)}$. Indeed, under the given derandomization assumption, there is hitting set generator $H = \{H_{n,\epsilon}\}$ with seed length $O(\log(n)) + \log(1/\epsilon))$ computable in polynomial time as per Definition 33. Let B be an algorithm that computes H in polynomial time.

We show that there must be at least one output r of $H_{n,\epsilon}$ such that C'_m outputs z on input r. Indeed, consider a circuit C_n that on input r' of length at most n, runs $C'_m(r')$ and accepts iff the answer is z. Since z has probability ϵ according to D_m and C'_m samples from D_m, we have that C_n accepts with probability at least ϵ on a uniformly random input. Since $H_{n,\epsilon}$ is a hitting set generator, we have that at least one output r of the hitting set generator is such that $C'_m(r) = z$.

Let y be the seed of length $O(\log(n)) + \log(1/\epsilon)$ for which $H_{n,\epsilon}(y) = r$. We describe z using the code of B, the string y and descriptions of n and $\lceil \log(1/\epsilon) \rceil$ in binary. This composite description is of size at most $O(\log(n)) + \log(1/\epsilon)$, and z can be computed in time $t = \text{poly}(n, \log(1/\epsilon))$ from this description by running the algorithm B with parameters n and $\log(1/\epsilon)$ and input y to produce a string r, and then running A on r to produce z. Hence $K^t(z) = O(\log(n)) + \log(1/\epsilon)$. Let C be an explicit constant such that $K^t(z) \leq C \log(m) + \log(1/\epsilon)$ for large enough n.

In the next part of our argument, we show that the cumulative probability according to D_m of strings with additive gap at least $f(m)$ between Kolmogorov complexity and t-time bounded Kolmogorov complexity is $1/m^{\omega(1)}$, establishing our result.

Partition all binary strings of length m into $2m$ groups as follows. The group $S_i, i \in [2m-1]$ contains all strings of length m sampled with probability at most $1/2^{i-1}$ and greater than $1/2^i$ from D_m. The group S_m contains all strings of length m sampled with probability at most $1/2^{2m-1}$. We will obtain our upper bound on the likelihood of a large gap between K and K^t complexity by using a union bound on the groups S_i.

Consider any group $S_i, i \in [2m-1]$. Let $g(m) = f(m) - C\log(m)$. Since $f(m) = \omega(\log(m))$, so is $g(m)$. Call a string x of length m i-bad if $x \in S_i$ and $K(x) \leq i - g(m)$. By the upper bound on Komogorov complexity, we have that there are at most $2^{i-g(m)} = 2^i/m^{\omega(1)}$ i-bad strings. Since each i-bad string is in S_i, it has probability at most $1/2^{i-1}$ of being sampled according to D_m, and therefore the cumulative probability of i-bad strings according to D_m is at most $1/2^{i-1} \cdot 2^i/m^{\omega(1)} = 1/m^{\omega(1)}$.

By the argument earlier in the proof, each string in S_i has K^t complexity at most $C\log(m)+\log(1/2^i) = C\log(m)+i$. Hence any string $x \in S_i$ such that $K^t(x)-K(x) \geq f(m)$ is i-bad, and the cumulative probability of all such strings according to D_m is $1/m^{\omega(1)}$. By a union bound over $i \in [2m-1]$, we have that the cumulative probability of all strings with gap at least $f(m)$ between Kolmogorov complexity and t-time bounded Kolmogorov complexity and that occur with probability greater than $1/2^{2m}$ in D_m is $1/m^{\omega(1)}$. Since there are only 2^m m-bit strings, the cumulative probability according to D_m of strings from S_{2m} is at most $2^{-m} = 1/m^{\omega(1)}$. Hence, by another union bound, $\Pr_{x\sim D_m}[K^t(x) - K(x) = \omega(\log(m))] = 1/m^{\omega(1)}$, as desired. ◀

We note that a weaker version of Lemma 35, where the hypothesis is that E requires Σ_2^p-oracle circuits of truly exponential size, is implicit in work of Antunes and Fortnow [16].

We require a couple of other well-known results. The first is a proposition shown using a straightforward padding argument.

▶ **Proposition 36.** *If every tally language in* $\mathsf{NTIME}(n)$ *is in* coNP*, then* $\mathsf{NE} = \mathsf{coNE}$*.*

The second is a standard derandomization result.

▶ **Theorem 37** ([38])**.** *If there is an* $\epsilon > 0$ *such that* E *does not have non-deterministic circuits of size* $2^{\epsilon n}$*, then* $\mathsf{AM} = \mathsf{NP}$*.*

Now we have the tools to state and prove the main result of this section, which also appears as Theorem 6 in the Introduction.

▶ **Theorem 38.** *Suppose that there is a polynomially bounded function* $t' : \mathbb{N} \to \mathbb{N}$ *such that for all large enough* $t = \mathrm{poly}(t')$*, there is a randomized time* t' *non-adaptive reduction that is polynomially honest, has fixed query length and inverse polynomial advantage, from SAT to* $\omega(\log(m))$*-additively approximating* K^t *complexity. Then either* E *has non-deterministic circuits of size* $2^{o(n)}$ *infinitely often, or* $\mathsf{NE} = \mathsf{coNE}$*.*

Proof. Suppose there is a randomized time t' non-adaptive reduction R from SAT to $O(\log(n))$-additively approximating K^t complexity with the properties specified in the statement of the Theorem, where $t = \mathrm{poly}(t')$ and the precise polynomial dependence is to be specified later. We use Lemma 35 to show that under the assumption that E does not have non-deterministic circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$, R also reduces any tally language in $\mathsf{NTIME}(n)$ to $\omega(\log(m))$-additively approximating K complexity, and then apply Theorem 29.

Indeed, let $L \in \mathsf{NTIME}(n)$ be a tally language. By the Cook-Levin theorem, there is a m-reduction from L to SAT running in time $n\,\mathrm{polylog}(n)$. This reduction is polynomially honest and has fixed query length. By composing this reduction with the reduction R in the assumption, we get that there is a randomized time t'' non-adaptive reduction R' from L to $\omega(\log(m))$-additively approximating K^t complexity, where the reduction is polynomially honest and has fixed query length, and where $t'' = t(n\,\mathrm{polylog}(n))$ is polynomially bounded.

Now we apply Lemma 35 to the sequence of distributions D_m sampled by running the randomized reduction R' on input 1^m and outputting a random query [7]. Applying Lemma 35 and a union bound, we have that with probability $1 - 1/m^{\Omega(1)}$ over the randomness r of R', all queries z asked on randomness r have $|K(z) - K^t(z)| = O(\log(m))$, and since the reduction has inverse polynomial advantage, replacing an oracle that $\omega(\log(m))$-additively approximates K^t complexity with one that $\omega(\log(m))$-additively approximates K complexity will preserve the answer of the reduction for each string 1^m. Hence we have that the hypothesis of Theorem 29 is satisfied, and we get that $L \in \mathsf{AM} \cap \mathsf{coAM}$.

Using the assumption that E does not have non-deterministic circuits of size $2^{\epsilon n}$ once more and applying Theorem 37, we have that $L \in \mathsf{NP} \cap \mathsf{coNP}$. Since this is the case for every $L \in \mathsf{NTIME}(n)$, we can use Proposition 36 and conclude that $\mathsf{NE} = \mathsf{coNE}$, as desired. ◀

8 Future Directions

There are two obvious directions to pursue. The first is to extend our negative results to other meta-complexity problems such as MCSP. As mentioned before, [43] unconditionally rule out NP-hardness of MCSP under randomized sublinear-time projections, but nothing seems to be known about less specialized forms of reduction.

The second is to obtain evidence against NP-hardness of meta-complexity problems with respect to randomized adaptive reductions. An anonymous reviewer suggests that it might be possible to use ideas from [17] to obtain interesting consequences from NP-hardness of approximating Kolmogorov reductions with respect to randomized reductions with bounded adaptivity.

References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710, 2006.

2 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.

3 Eric Allender. The new complexity landscape around circuit minimization. In *Proceedings of 14th International Conference on Language and Automata Theory and Applications*, volume 12038 of *Lecture Notes in Computer Science*, pages 3–16. Springer, 2020.

4 Eric Allender, Harry Buhrman, Luke Friedman, and Bruno Loff. Reductions to the set of random strings: The resource-bounded case. *Log. Methods Comput. Sci.*, 10(3), 2014.

5 Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the kolmogorov-random strings? *Ann. Pure Appl. Log.*, 138(1-3):2–19, 2006.

6 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

7 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

8 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 25–32, 2014.

[7] Technically speaking, it might be unclear if Lemma 35 applies here, as the sampled strings do not in general have length m, but the proof of Lemma 35 continues to work as long as the sampled strings have length $m^{\Omega(1)}$

9 Eric Allender, George Davie, Luke Friedman, Samuel Hopkins, and Iddo Tzameret. Kolmogorov complexity, circuits, and the strength of formal theories of arithmetic. *Chic. J. Theor. Comput. Sci.*, 2013, 2013.

10 Eric Allender, Luke Friedman, and William I. Gasarch. Limits on the computational power of random strings. *Inf. Comput.*, 222:80–92, 2013.

11 Eric Allender, Joshua A. Grochow, and Cristopher Moore. Graph isomorphism and circuit size. *CoRR*, abs/1511.08189, 2015. `arXiv:1511.08189`.

12 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008.

13 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.

14 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.

15 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.

16 Luis Filipe Coelho Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 298–303. IEEE Computer Society, 2009.

17 Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proceedings of 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 211–220, 2008.

18 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1st edition, 2009.

19 Sergei Artemenko, Russell Impagliazzo, Valentine Kabanets, and Ronen Shaltiel. Pseudorandomness when the odds are against you. In *Proceedings of the 31st Conference on Computational Complexity*, volume 50 of *LIPIcs*, pages 9:1–9:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

20 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

21 Harry Buhrman, Lance Fortnow, Michal Koucký, and Bruno Loff. Derandomizing from random strings. In *Proceedings of the 25th Annual Conference on Computational Complexity*, CCC '10, 2010.

22 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic learning from tolerant natural proofs. In *Approximation, Randomization, and Combinatorial Optimization*, pages 35:1–35:19, 2017.

23 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

24 Oded Goldreich and Salil P. Vadhan. On the complexity of computational problems regarding distributions. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650, pages 390–405. Springer, 2011.

25 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 59–68. ACM, 1986.

26 Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In *Proceedings of 12th International Workshop on Randomness and Computation*, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482. Springer, 2008.

27 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018.

28 Shuichi Hirahara. Unexpected hardness results for kolmogorov complexity under uniform reductions. In *Proceedings of 52nd Annual Symposium on Theory of Computing*, pages 1038–1051, 2020.

29 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.

30 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.

31 Shuichi Hirahara and Osamu Watanabe. On nonadaptive security reductions of hitting set generators. In *Proceedings of 24th International Conference on Randomization and Computation*, volume 176 of *LIPIcs*, pages 15:1–15:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

32 John M. Hitchcock. Limitations of efficient reducibility to the kolmogorov random strings. *Comput.*, 1(1):39–43, 2012.

33 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.

34 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and ac^0[p]. In *Proceedings of 11th Innovations in Theoretical Computer Science Conference*, volume 151 of *LIPIcs*, pages 34:1–34:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

35 Rahul Ilango. Constant depth formula and partial function versions of MCSP are hard. In *Proceedings of 61st IEEE Annual Symposium on Foundations of Computer Science*, pages 424–433, 2020.

36 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. Np-hardness of circuit minimization for multi-output functions. In *Proceedings of 35th Computational Complexity Conference*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

37 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Symposium on Theory of Computing* (STOC), pages 73–79, 2000.

38 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.

39 Ker-I Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM Journal on Computing*, 20(5):962–986, 1991.

40 Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):115–116, 1973.

41 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

42 Cody Murray and Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Conference on Computational Complexity* (CCC), pages 365–380, 2015.

43 Cody Murray and Ryan Williams. On the (non) np-hardness of computing circuit complexity. *Theory Comput.*, 13(1):1–22, 2017.

44 Michael Saks and Rahul Santhanam. Circuit lower bounds from np-hardness of MCSP under turing reductions. In Shubhangi Saraf, editor, *Proceedings of 35th Computational Complexity Conference*, volume 169 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

45 Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

46 Stathis Zachos. Probabilistic quantifiers and games. *J. Comput. Syst. Sci.*, 36(3):433–451, 1988. doi:10.1016/0022-0000(88)90037-2.

A Limitations of Randomized m-Reductions to Approximating Kolmogorov Complexity

In this section, we show that any language that has a low-error randomized polynomial-time m-reduction to a gap version of the Kolmogorov random strings is solvable in Statistical Zero Knowledge.

▶ **Theorem 39.** *Let L be any decidable language such that there is a randomized poly-time m-reduction with error $1/n^{\omega(1)}$ from L to R_K with gap g for some time-constructible $g = \omega(\log(n))$. Then $L \in \mathsf{SZK}$.*

Proof. Let L be a language as in the statement of the theorem. Let f be a randomized poly-time m-reduction with error $1/n^{\omega(1)}$ to R_K with gap g for some $g = \omega(\log(n))$.

We can assume without loss of generality that any query made by f on input x of length n is of length $p(n)$ for some polynomial p. Indeed, let $p(n)$ be an upper bound on the running time of f. We modify the reduction by "padding" any query made by f to length $m = 2p(n)$ as follows. Let c be the constant in the $O(\cdot)$ in Proposition 11. If the query q is of length k, we sample a string $y = r0^{(m-k)/2-2c\log(n)}$ where $r \in \{0,1\}^{(m-k)/2+2c\log(n)}$ is chosen uniformly at random, and output the query qy. Observe that by Proposition 11, with probability $1 - 1/n^{\omega(1)}$, if q is a YES instance of R_K, then qy is a YES instance of R_K, and if q is a NO instance of R_K, then qy is a NO instance of R_K.

For each $x \in \{0,1\}^*$, $f(x)$ is a random variable supported on $\{0,1\}^{2p(|x|)}$. We claim that the following holds. If $x \notin L$, the random variable $f(x)$ has entropy at least $p(|x|) - g(2p(|x|))/2 + 1$, and if $x \in L$, the random variable $f(x)$ has entropy at most $p(|x|) - g(2p(|x|))/2 - 1$.

We first show that the claim implies that $L \in \mathsf{SZK}$, and then establish the claim. To see that the claim implies $L \in \mathsf{SZK}$, we use the fact that Entropy Difference is in SZK. Namely, there is a statistical zero-knowledge protocol that, given a circuit C sampling a distribution over m bits and a parameter s, accepts when the entropy of C is at least $s+1$ and rejects when the entropy of C is at most $s-1$. Note that a circuit C sampling $f(x)$ can easily be computed from x. Since SZK is closed under poly-time m-reductions, we have that $L \in \mathsf{SZK}$, by reducing to Entropy Difference with parameter $m/2 - g(m)/2$.

Next we establish the claim. Assume for the sake of contradiction that the claim fails for infinitely many inputs. Let $\{n_i\}$ be an infinite sequence of numbers such that the claim fails for at least one input of length n_i for each i, and let x_i be the first input of length n_i for which the claim fails. Note that $K(x_i) = O(\log(n_i))$. The reason is that we can describe x_i by a program which has n_i in binary encoded into it, and then searches for the first string z of length n_i such that the claim fails for z, i.e., either $z \in L$ and the entropy of $f(z)$ is greater than $p(|z|) - g(2p(|z|)/2 - 1$, or $z \notin L$ and the entropy of $f(z)$ is smaller than $p(|z|) - g(2p(|z|))/2 + 1$.

Let i be large enough and x_i be the first string of length n_i for which the claim fails. Let $m = 2p(n_i)$. Either $x_i \in L$ and the entropy of $f(x_i)$ is greater than $m/2 - g(m)/2 - 1$, or $x_i \notin L$ and the entropy of $f(x_i)$ is smaller than $m/2 - g(m)/2 + 1$. We show that either case leads to a contradiction.

In the first case, the random variable $f(x_i)$ has entropy greater than $m/2 - g(m)/2 - 1$. Then with probability at least $1/m$ over $y \in \{0,1\}^m$ sampled from $f(x_i)$, $K(y) > m/2 - g(m)$. Indeed, if not, the entropy of $f(x_i)$ is at most $1/m * m + (1 - 1/m) * (m/2 - g(m)) \leq m/2 - g(m) + 1 < m/2 - g(m)/2 - 1$, which contradicts the lower bound on the entropy of $f(x_i)$. But if $K(y) > m/2 - g(m)$ with probability at least $1/m$, the reduction cannot be correct for $x_i \in L$, as correctness of the reduction would imply $K(y) \leq m/2 - g(m)$ with probability $1 - 1/n^{\omega(1)}$ (as we could answer NO for all queries y with $K(y) > m/2 - g(m)$).

In the second case, the random variable $f(x_i)$ has entropy at most $m/2 - g(m)/2 + 1$. We show that in this case, with probability $1/n^{\Omega(1)}$, y sampled from $f(x_i)$ has $K(y) < m/2$, which again contradicts the correctness of the reduction.

We show that with probability $1/n^{\Omega(1)}$ over y sampled from $f(x_i)$, $K(y|x_i) < m/2 - \omega(\log(m))$, from which the previous line follows using Proposition 11 and the fact that $K(x_i) = O(\log(n_i)) = O(\log(m))$. Indeed let k be the entropy of $f(x_i)$ and let $y_1 \dots y_{2^{k+\log(m)}}$ be the first $2^{k+\log(m)}$ strings in the support of $f(x_i)$ in order of decreasing probability, and let Y be the set of these strings. Conditioned on x_i, each such string can be described with a program of size $k + \log(m) + O(\log(m))$, which is $m/2 - \omega(\log(m))$, using the facts that $k \leq m/2 - g(m)/2 + 1$ and $g(m) = \Omega(\log(m))$. We claim that with probability at least $1/m$, y sampled from $f(x_i)$ belongs to Y. Indeed, if not, we have that with probability greater than $1 - 1/m$, y sampled from $f(x_i)$ has probability at most $2^{-(k+\log(m))}$, and hence the contribution to the entropy of $f(x_i)$ from such strings is at least $(1 - 1/m)(k + \log(m)) > k$, which is impossible.

Thus we have that with probability $1/m$, y sampled from $f(x_i)$ belongs to Y, and that each string in Y has Kolmogorov complexity smaller than $m/2$. But this contradicts the correctness of the reduction, as x_i is a NO instance, and hence with probability $1 - 1/n^{\omega(1)}$, y sampled from $f(x_i)$ should have $K(y) \geq m/2$ (as otherwise we could answer YES for all queries y with $K(y) < m/2 - g(m)$). ◀

Interactive Oracle Proofs of Proximity to Algebraic Geometry Codes

Sarah Bordage
LIX, CNRS UMR 7161, Ecole Polytechnique, Institut Polytechnique de Paris, France
Inria, Palaiseau, France

Mathieu Lhotel
Laboratoire de Mathématiques de Besançon, UMR 6623 CNRS,
Université de Bourgogne Franche-Comté, France

Jade Nardi
Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France

Hugues Randriam
ANSSI, Paris, France
Institut Polytechnique de Paris, Télécom Paris, Palaiseau, France

Abstract

In this work, we initiate the study of proximity testing to Algebraic Geometry (AG) codes. An AG code $C = C(\mathcal{X}, \mathcal{P}, D)$ over an algebraic curve $\mathcal{X}$ is a vector space associated to evaluations on $\mathcal{P} \subseteq \mathcal{X}$ of functions in the Riemann-Roch space $L_{\mathcal{X}}(D)$. The problem of testing proximity to an error-correcting code C consists in distinguishing between the case where an input word, given as an oracle, belongs to C and the one where it is far from every codeword of C. AG codes are good candidates to construct probabilistic proof systems, but there exists no efficient proximity tests for them. We aim to fill this gap.

We construct an Interactive Oracle Proof of Proximity (IOPP) for some families of AG codes by generalizing an IOPP for Reed-Solomon codes, known as the FRI protocol [9]. We identify suitable requirements for designing efficient IOPP systems for AG codes. Our approach relies on a neat decomposition of the Riemann-Roch space of any invariant divisor under a group action on a curve into several explicit Riemann-Roch spaces on the quotient curve. We provide sufficient conditions on an AG code C that allow to reduce a proximity testing problem for C to a membership problem for a significantly smaller code C'.

As concrete instantiations, we study AG codes on Kummer curves and curves in the Hermitian tower. The latter can be defined over polylogarithmic-size alphabet. We specialize the generic AG-IOPP construction to reach linear prover running time and logarithmic verification on Kummer curves, and quasilinear prover time with polylogarithmic verification on the Hermitian tower.

2012 ACM Subject Classification Theory of computation → Error-correcting codes; Theory of computation → Interactive proof systems

Keywords and phrases Algebraic geometry codes, Interactive oracle proofs of proximity, Proximity testing

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.30

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2020/165/`

Funding This work was funded in part by the grant ANR-21-CE39-0009 - BARRACUDA from the French National Research Agency.
Sarah Bordage: Supported by the Chair "Blockchain & B2B Platforms", led by l'*X* – *École Polytechnique* and the *Fondation de l'École Polytechnique*, sponsored by *Capgemini*, *NomadicLabs* and *Caisse des Dépôts*.
Jade Nardi: Suported by the French government "Investissements d'Avenir" program ANR-11-LABX-0020-01.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 30; pp. 30:1–30:45

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

Acknowledgements The authors are very appreciative to the anonymous reviewers whose comments and suggestions helped to improve and clarify this manuscript. The third author thanks Marc Perret for his precious advice in the early days of this project. The authors are grateful to Daniel Augot for suggesting to work on this project and for many valuable discussions. They also thank Eli Ben-Sasson and Alessandro Chiesa for their helpful insights.

1 Introduction

Let $C \subset \Sigma^S$ be an evaluation code with evaluation domain S of size n and alphabet Σ. For $u \in \Sigma^S$, if $\Delta(u, C) > \delta$, we say that u is δ-far from C and δ-close otherwise. We address the problem of proximity testing to a code C, *i.e.* given a code C and assuming a verifier has oracle access to a function $f : S \to \Sigma$, distinguish between the case where $f \in C$ and f is δ-far from C. In this paper, we focus on the case where C is an AG code. An algebraic geometry (AG) code $C = C(\mathcal{X}, \mathcal{P}, D)$ is a vector space formed by evaluations on a set $\mathcal{P} \subset \mathcal{X}$ of functions in the Riemann-Roch space $L_{\mathcal{X}}(D)$. We address this problem in the Interactive Oracle Proof model [15], which has demonstrated to be particularly promising for the design of proof systems in the past few years.

Context of this work. Under the generic term of *arithmetization* [34], algebraic techniques for constructing proof systems using properties of low-degree polynomials have emerged from the study of interactive proofs [5, 24]. Arithmetization techniques have been enhanced and fruitfully applied to other broad families of proof systems since then, including probabilistically checkable proofs (PCPs, [6, 3, 2]). Loosely speaking, in a probabilistic proof system for a binary relation $\mathcal{R}$, the arithmetization process transforms any instance-witness pair (x, w) into a word that belongs to a certain error-correcting code C if $(x, w) \in \mathcal{R}$, and is very far from C otherwise.

Since the seminal works of Kilian [31] and Micali [38], a lot of efforts have been put into making PCPs efficient enough to obtain *practical* sublinear non-interactive arguments for delegating computation. In search of reducing the work required to generate such probabilistic proofs, as well as the communication complexity of succinct arguments based on them, Interactive Oracle Proofs (IOPs, [15, 42]) have been introduced as a common generalization of PCPs, IPs and IPCPs [28]

Considering for the first time univariate polynomials instead of multivariate ones, [19, 23] constructed a PCP with quasilinear proof length and constant query complexity. Since then, efficient transparent and zero-knowledge non-interactive arguments have been designed by relying on Reed-Solomon (RS) codes, including [1, 10, 14, 13, 30, 22, 49]. At some point, aforementioned sublinear arguments require a proximity test for RS codes.

As a solution, one can use an IOP of Proximity for Reed-Solomon codes (an IOP of Proximity [12] is the natural extension to the IOP model of a PCP of Proximity). In an IOP of Proximity (IOPP) for an error-correcting code, a verifier is given as input a code, has oracle access to a function (a purported codeword) and interacts with a prover. After the interaction, the verifier accepts if the function is indeed a codeword, and rejects with high probability if the function is far from any codeword. We defer the formal definition of an IOPP to Section 1.1.

The FRI protocol is a prover-efficient IOP of Proximity for testing proximity to Reed-Solomon codes evaluated over well-chosen evaluation points (introduced by [9] and further improved in [18, 16, 11]). It admits linear prover time, logarithmic verifier time and logarithmic query complexity. While being sub-optimal for some parameters, the FRI

protocol is highly-efficient in practice and is a crucial tool in systems deployed in the real-world. For instance, [12, 43] proposed IOPP constructions for RS codes with constant query complexity whereas the FRI protocol has logarithmic query complexity.

The main drawback of RS codes is that they must have an alphabet size larger than their length. AG codes [25], as evaluations of a set of functions at some designated rational points on a given curve, extend the notion of Reed-Solomon codes and inherit many of their interesting properties. Therefore, replacing RS codes with AG codes is not only natural but has also led to improvements in the past. Examples of cryptographic applications of AG codes include public key cryptography, distributed storage, secret sharing and multi-party computation. A feature for a family of codes that facilitates arithmetization is a multiplication property [37], namely the fact that the component-wise multiplication of two codewords results in codewords in a code whose minimum distance is still good. This multiplication property actually emulates multiplication of low-degree polynomials. Algebraic geometry codes not only feature this multiplication property but may also have arbitrary large length given a fixed finite field $\mathbb{F}$, unlike RS codes.

Limitations of Reed-Solomon codes. We identify two limitations of using RS codes in IOPs.

As mentioned earlier, RS codes are the simplest case of AG codes, but possess an inherent limitation: the alphabet size must be larger than the block length of the code. Therefore, practical IOP-based succinct arguments are designed over *large fields*.

The second limitation is related to the algebraic structure of the field. RS-IOPPs [19, 9] require the set $D \subset \mathbb{F}$ of evaluation points to have a special structure. Concretely, the field must contain a subgroup of *large smooth order*, typically a power of 2 which is larger than the size of the non-deterministic computation to be verified. Depending on the applications of succinct non-interactive arguments, a base field might already be imposed. This is for instance the case for standard digital signature schemes. For computations whose size exceeds the order of the largest smooth subgroup of the field, RS-IOPPs known to date can no longer be used.

We observe that lowering the size of field elements may not significantly shorten the length of IOP-based succinct non-interactive arguments (see [15]). There are, however, other reasonable motivations to replace RS codes with AG ones. We explain below how AG codes could circumvent the limitations of RS codes.

Why can AG codes be useful? First, working over smaller fields lowers the cost of field operations[1]. For concrete efficiency, complexity measures such as prover time and verifier time are closely examined. Reducing significantly the size of the alphabet would have a direct impact on the binary cost of arithmetic operations. Smaller fields could enhance efficiency of proof systems since arithmetization of general circuits would be more efficient. Moreover, on the prover side, the bit complexity of encoding codewords might be smaller.

A popular belief is that encoding with AG codes is an heavy task. It is surely true in general, but there are explicit families of AG codes for which there are quasilinear time encoding algorithms [8]. We discuss more about encoding in Section 1.3. On another note, putting forward applications of AG codes can motivate the study of fast encoding algorithms for AG codes, in particular in the computer algebra community.

[1] Consider the application of checking the correct execution of a size n computation. Then an RS-based IOP for this problem will work over a field of size $\Omega(n)$. This means that a single addition of two field elements will cost $\Omega(\log n)$ operations. If the IOP is instead based on a code with polylogarithmic-size alphabet, the cost of a single addition is only $\Omega(\log(\text{polylog}(n))$.

One may be concerned by the overhead of reducing the alphabet size when targeting a soundness error less than $2^{-\kappa}$. Notice that it is possible to sample enough bits of randomness from an extension field when needed, or to repeat only some parts of the protocol (see [44, 9]). For instance, the soundness error of our IOPP is bounded from below by $|\mathbb{F}|^{-1}$. Reaching the targeting soundness requires to repeat the interactive phase of the IOPP s times, inducing a factor $s \simeq \frac{\kappa}{\log|\mathbb{F}|}$ multiplicative overhead for the prover. A rough estimation of bit complexities does not show evidence of a significant overhead. Overall, a proof system supporting small fields might be more efficient: any part of the protocol which does not contribute to the soundness error could benefit from cheaper field operations.

In addition, AG codes offer more flexibility on the choice of the field. For computations of size n, we propose AG codes for which the field is not required to admit an n-th root of unity (unlike RS-IOPP on a prime field). Specifically, for AG codes over Kummer curves, the base field needs only to have a N-th root of unity, where N divides n. For AG codes over curves in a Hermitian tower (which admits a polylogarithmic-size alphabet), *our IOPP does not involve any assumption on the alphabet*, except that it must be a degree-2 extension of a field $\mathbb{F}_q$, where q is any prime power.

Finally, the question of whether there exist concretely efficient IOPPs for AG codes is motivated by both a theoretical and practical perspective.

1.1 Definition of an IOPP for a code

We are specifically interested in public-coin IOP of Proximity (IOPP) for a family of evaluation codes $\mathscr{C}$, thereby we specify our definition for this particular setting. An IOPP (P, V) for a code C is a pair of randomized algorithms, where both P (the prover) and V (the verifier) receive as explicit input the specification of a code $C \in \mathscr{C}$, $C \subseteq \Sigma^S$. We define the input size to be $n = |S|$. Furthermore, a purported codeword $f : S \to \Sigma$ is given as explicit input to P and as an oracle to V. The prover and the verifier interact over at most $\mathsf{r}(n)$ rounds. During this conversation, P seeks to convince V that the purported codeword f belongs to the code C.

At each round, the verifier sends a message chosen uniformly and independently at random, and the prover answers with an oracle. Verifier's queries to the prover's messages are generated by public randomness and performed after the end of the interaction with the prover. Thus, such an IOPP is in particular a *public-coin* protocol (or Arthur-Merlin [5]).

Let us denote $\langle \mathsf{P} \leftrightarrow \mathsf{V} \rangle \in \{\mathsf{accept}, \mathsf{reject}\}$ the output of V after interacting with P. The notation V^f means that f is given as an oracle input to V. We say that a pair of randomized algorithms (P, V) is an IOPP system for the code $C \subseteq \Sigma^S$ with *soundness error* $s : (0, 1] \to [0, 1]$, if the following conditions hold:

Perfect completeness: If $f \in C$, then $\Pr[\langle \mathsf{P}(C, f) \leftrightarrow \mathsf{V}^f(C) \rangle = \mathsf{accept}] = 1$.

Soundness: For any function $f \in \Sigma^S$ such that $\delta := \Delta(f, C) > 0$ and any unbounded malicious prover P^*, $\Pr[\langle \mathsf{P}^* \leftrightarrow \mathsf{V}^f(C) \rangle = \mathsf{accept}] \leq s(\delta)$.

The length of any prover message is expressed in the number of symbols of an alphabet $\mathsf{a}(n)$. The sum of lengths of prover's messages defines the proof length $\mathsf{l}(n)$ of the IOPP. The query complexity $\mathsf{q}(n)$ is the total number of queries made by the verifier to both the purported codeword f and the oracle sent by the prover during the interaction. The prover complexity $t_p(n)$ is the time needed to generate prover messages during the interaction (which does not include the input function f). The verifier complexity $t_v(n)$ is the time spent by the verifier to make her decision when queries and query-answers are given as inputs.

1.2 Our results

In this section, we provide and overview of the three contributions of this paper. In all this work, we state complexities in field elements and field operations, where the field is the alphabet of the considered code. Asymptotic complexities are relative to the length of the code.

- The first one is a clear criterion for constructing IOPPs with linear proof length and sublinear query complexity for AG codes. Our hope with this result is to open up new possibilities for designing efficient probabilistic proof systems based on families of AG codes with constant rate and constant distance.
- The second contribution is a concrete instantiation for AG codes defined over Kummer-type curves. This IOPP has strictly linear prover time and strictly logarithmic verification (counted in field operations). Thus, we give a strict generalization of the FRI protocol for codes of length n over an alphabet of size roughly $n^{2/3}$.
- The third one is a concrete instantiation for AG codes defined over a tower of Hermitian curves. Considering recursive towers enables to construct an IOPP for AG codes with *polylogarithmic-size* alphabet. For those codes, we give an IOPP with quasilinear prover time and polylogarithmic verification (counted in field operations).

Efficiency of our two AG-IOPP instantiations leverages the fact that proximity testing for these families of AG code can be reduced to a proximity test for a small RS code.

Generic criterion for constructing AG-IOPPs

Let $\mathcal{X}$ be a curve defined over a finite field $\mathbb{F}$, D a divisor on the curve $\mathcal{X}$ and $\mathcal{P} \subset \mathcal{X}(\mathbb{F})$. This defines an AG code $C = C(\mathcal{X}, \mathcal{P}, D)$. We construct a sequence of curves

$$\mathcal{X} \coloneqq \mathcal{X}_0 \xrightarrow{\pi_0} \mathcal{X}_1 \xrightarrow{\pi_1} \mathcal{X}_2 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_{r-1}} \mathcal{X}_r,$$

so that $\mathcal{X}_{i+1}$ arises as the quotient of the curve $\mathcal{X}_i$ by some automorphism subgroup Γ_i under the quotient map π_i.

Using these consecutive projection maps, we construct a sequence of AG codes $(C_i)_{0 \leq i \leq r}$ of decreasing length to turn the proximity test of the function $f^{(0)} = f$ to C_0 into a membership test of a function $f^{(r)}$ in C_r. We show that such a procedure is possible if there is a large enough (with respect to the length of the code C_0) group $\mathcal{G}$ in the automorphism group of $\mathcal{X}$ and under some hypotheses on the divisor D (overviewed in Section 2.2, detailed in Section 4). An AG code fulfilling all the required conditions is called *foldable*.

Assuming that an AG code $C(\mathcal{X}, \mathcal{P}, D)$ of blocklength n is foldable, we show that there is an $O(\log n)$-round IOPP for it, with linear proof length, sublinear query complexity and constant soundness (see Theorem 42).

In general, we observe that the larger is the group $\mathcal{G}$ acting on $\mathcal{X}_0$ compared to n, the smaller are the query complexity and the verifier decision complexity of the protocol.

However, we notice that the hypothesis on the size of $\mathcal{G}$ is not a necessary condition for constructing an IOPP with sublinear verification. For instance, if the curve $\mathcal{X}_r$ is isomorphic to the projective line $\mathbb{P}^1$, we can continue to recurse in order to reduce even more the size of the proximity testing problem. We propose two interesting families of AG codes for which it is the case.

Concrete IOPP for AG codes on Kummer curves

When $\mathcal{X}$ is a Kummer curve of the form $y^N = f(x)$, we show how to choose $\mathcal{P}$ and D to make the AG code $C = C(\mathcal{X}, \mathcal{P}, D)$ foldable. We benefit from the action of the group $\mathbb{Z}/N\mathbb{Z}$ on $\mathcal{X}$ that yields a quotient curve $\mathcal{X}/(\mathbb{Z}/N\mathbb{Z})$ isomorphic to the *projective line*. This enables us to define a sequence of codes $(C_i)_{0 \le i \le s}$ such that $C_0 = C$ and the code C_s is a *Reed-Solomon code* of dimension $(\deg D)/N + 1$.

▶ **Theorem 1** (Informal, see Theorem 43). *Let $C = C(\mathcal{X}, \mathcal{P}, D) \subset \mathbb{F}^{\mathcal{P}}$ be a foldable AG code defined over a Kummer curve $\mathcal{X}$ of equation $\mathcal{X} : y^N = f(x)$ such that $\deg f = N\ell - 1$ for some integer $\ell > 0$ and N is a smooth integer, coprime with $|\mathbb{F}|$. Assume $\mathbb{F}$ contains a primitive N-th root of unity. The block length $n := |\mathcal{P}|$ is a multiple of N and satisfies $n < \ell N^2 |\mathbb{F}|^{1/2}$. Let $f : \mathcal{P} \to \mathbb{F}$ be a purported codeword. For every proximity parameter $\delta \in (0,1)$ and soundness $\varepsilon \in (0,1)$, there exists a public-coin IOPP system (P, V) for C with perfect completeness and the following properties:*

rounds	$\mathsf{r}(n)$	$< \log n$,
proof length	$\mathsf{l}(n)$	$= O(n)$,
query complexity	$\mathsf{q}(n)$	$= O(\log n)$,
prover complexity	$\mathsf{t_p}(n)$	$= O(n)$,
verifier decision complexity	$\mathsf{t_v}(n)$	$= O(\log n)$.

It is worth noting that the Hermitian curve defined over $\mathbb{F}_{q^2}$ by $y^{q+1} = x^q + x$ satisfies the hypotheses of the previous theorem. It is well known to be *maximal*, i.e. it has the maximum number of rational points with respect to its geometry. Besides, we recall that Hermitian codes over alphabet Σ support block length up to $|\Sigma|^{3/2}$.

Concrete IOPP for AG codes on towers of Hermitian curves

We recall that a tower of curves consists of an infinite sequence of curves

$$\mathbb{P}^1 = \mathcal{X}_0 \leftarrow \mathcal{X}_1 \leftarrow \ \ldots \leftarrow \mathcal{X}_n \leftarrow \ldots$$

such that the number of rational points of the n^{th} curve tends to infinity as n tends to infinity. Towers of curves play a prominent role in the history of AG codes as they define codes with outstanding length and correction capacity [47, 7].

The Hermitian tower is an example of the widely studied Artin-Scheier extensions [33, 45]. In this case, the curve $\mathcal{X}_i$ arises as the quotient of the curve $\mathcal{X}_{i+1}$ above modulo the action of a group of order q of the finite field $\mathbb{F}_{q^2}$, the curve $\mathcal{X}_0$ being isomorphic to the projective $\mathbb{P}^1$. Therefore, one can test proximity to an AG code from one of the curves $\mathcal{X}_n$ by going down along the tower and then testing proximity to a RS code, whose degree can be expressed explicitly in terms of the initial AG code.

Beyond supporting polylogarithmic-size alphabet, AG codes over the Hermitian tower happen to be more naturally "foldable". In particular, no additional assumptions on the alphabet are required.

We write polylog(n) for functions that are in $O\left(\log^k(n)\right)$ for some k.

▶ **Theorem 2** (Informal, see Theorem 46). *Let $C = C(\mathcal{X}, \mathcal{P}, D) \subset \mathbb{F}^{\mathcal{P}}$ be a foldable AG code over an alphabet $\mathbb{F}$ of size $|\mathbb{F}| = \Omega(\log^k(n))$ for some constant k. We denote $n = |\mathcal{P}|$. Let $f : \mathcal{P} \to \mathbb{F}$ be a purported codeword. For every proximity parameter $\delta \in (0,1)$ and soundness $\varepsilon \in (0,1)$, there exists a public-coin IOPP system (P, V) for C with perfect completeness and the following properties:*

rounds	$\mathsf{r}(n)$	$< \log n$,
proof length	$\mathsf{l}(n)$	$= O(n)$,
query complexity	$\mathsf{q}(n)$	$= \mathrm{polylog}(n)$,
prover complexity	$\mathsf{t_p}(n)$	$= \widetilde{O}(n)$,
verifier decision complexity	$\mathsf{t_v}(n)$	$= \mathrm{polylog}(n)$.

1.3 More on the practicality of AG codes

When constructing a proximity test for a code, it is assumed that the purported codeword is given as input to the prover. Thus, the prover complexity is computed thereof. While we heavily rely on the group of automorphisms of the curve for proving the existence of an efficient IOPP for "foldable" AG codes, we emphasize that the work of the prover and the verifier during the protocol is essentially to perform some univariate polynomial interpolation tasks, with very small degree. In particular, neither the prover nor the verifier of the IOPP system needs to run an encoding algorithm for AG codes.

However, keeping applications to code-based IOP constructions in mind, the running time of the IOP prover is bounded from below by the time needed to encode codewords during arithmetization. Fast encoding for AG codes is not the most widely studied computational task, and is often a concern when suggesting constructions based on AG codes[2].

This is a reason why we focus our study on families of AG codes that are particularly likely to lead to practical implementations, as we argue next. Specifically, our study includes the two following subfamilies of one-point AG codes over small alphabets with constant rate and distance.

- The first family includes one-point AG codes over Kummer-type curves, and in particular the notorious Hermitian curve. [8] proposed an encoding algorithm with quasilinear complexity $\widetilde{O}(n)$. Roughly speaking, [8] method consists in translating the encoding task into a bivariate polynomial multipoint-evaluation problem. Assuming that the evaluation points are well-structured, they view a bivariate polynomial in $\mathbb{F}[X, Y]$ as a polynomial in $\mathbb{F}[X][Y]$ in order to evaluate it thanks to two univariate multipoint evaluations. It is the same idea as the one for computing m-dimensional FFT from m (univariate) FFTs.
- The second family of one-point AG codes arises from curves on the Hermitian tower and has an alphabet size polylogarithmic in the block length of the code. It is very likely that those codes could also be encoded in quasilinear time, by iteratively applying the encoding method proposed by [8].

We also point out that bases for Riemann-Roch spaces related to these codes are explicitly known.

1.4 Related work

We discuss works related to AG-based proximity testing. We emphasize that the motivation behind existing works was only theoretical. In particular, the PCP techniques being used are too complex to be implemented for verifying meaningful computations.

In 2013, [17] constructed a PCP with linear proof length and sublinear query complexity for boolean circuit satisfiability by relying on AG codes. More precisely, for any $\varepsilon > 0$ and instances of size n, their PCP has length $2^{O(1/\varepsilon)}n$ and query complexity n^{ε}. When aiming

[2] In general, the asymptotic cost of the task of encoding an arbitrary linear code of length n is $O(n^2)$ (using a generator matrix for the code).

at optimal proof length and query complexity as small as possible, this result remains the state-of-the-art PCP construction. By using AG codes, the authors of [17] reduced the field size to a constant, which avoids a logarithmic blowup in proof bit-length (occurring e.g. in [19] when using univariate polynomials of degree m to encode binary strings of length m). In [17], the authors pointed out that they are not able to apply proof composition [3] to reduce the query complexity of their PCP because the decision complexity of the PCP verifier is too large (polynomial in the query complexity).

Improving on [17], [12] proposed an IOP for boolean circuit satisfiability with linear proof length and constant query complexity. The IOP of [12] invoked the sumcheck protocol [34] on $O(1)$-wise tensor product of AG codes, which exponentially deteriorates the rate of the base code. Then, they use Mie's PCP of Proximity for non-deterministic languages [39] to test proximity to the tensored code. Both constructions benefit from the use of AG codes to get constant size alphabet and linear proof bit-lengths.

A recent work of [43] constructed an IOPP for any deterministic language which can be decided in time $\mathrm{poly}(n)$ and space $n^{o(1)}$. In particular, [43, Corollary 3.6] can be applied to test proximity to AG codes. This IOPP outperforms our construction on some parameters: it has constant round and query complexities, and proof length is slightly less than n. However, it is unlikely that [43]'s IOPP leads to a concrete implementation, which is a motivation for our work. Indeed, prover running time is polynomial, and the inner IOPP used for achieving constant query complexity via proof composition is the heavy PCPP of [39]. Mie's PCPP is a theoretical and complex tool used to achieve constant query (e.g in [12, 13, 20]), but it is seen as impractical[3].

By contrast, we exhibit two explicit families of AG codes for which we are able to construct a proximity test with linear prover running time and logarithmic verification (for the first one) and quasilinear prover time with polylogarithmic verification (for the second one). The main point however, is that our construction is undoubtedly much simpler to implement: the most complex task of the prover and the verifier is simply to perform univariate interpolations (with very small degrees). The technical difficulties are in analyzing the conditions allowing the construction, but the protocol itself is very similar to the FRI protocol. IOPP inspired by the FRI protocol have the inherent barrier of logarithmic query complexity. However, in practice, it is still the most efficient proximity test for Reed-Solomon codes known to date.

2 Technical Overview

Our IOPP construction relies on the generalization of the FRI protocol to AG codes. Let us first recall some ideas behind the construction of FRI protocol (see e.g. [18] for a detailed presentation). Then we shall describe how we tailor these ideas and which difficulties arise to construct our IOPP for AG codes over curves of positive genus.

2.1 The FRI protocol for RS proximity testing

Let k be a positive integer and $\rho \in (0,1)$ such that $\rho = 2^{-k}$. The FRI protocol allows to check proximity to the Reed-Solomon code

$$\mathsf{RS}\left[\mathbb{F}, \mathcal{P}, \rho\right] \coloneqq \left\{ f \in \mathbb{F}^{\mathcal{P}} \mid \deg f < \rho \cdot |\mathcal{P}| \right\}$$

[3] Proposing an alternative to [39] which does not involve heavy PCP machinery would allow to narrow the gap between the best constructions known in theory and the most efficient ones used in practice.

by testing proximity to $\mathsf{RS}[\mathbb{F}, \mathcal{P}', \rho]$ with $|\mathcal{P}'| < |\mathcal{P}|$. The FRI protocol considers a family of linear maps $\mathbb{F}^{\mathcal{P}} \to \mathbb{F}^{\mathcal{P}'}$ which randomly "fold" any function in $\mathbb{F}^{\mathcal{P}}$ into a function in $\mathbb{F}^{\mathcal{P}'}$. We present in a simplified way three key ingredients that enable the FRI protocol to work.

a) *Splitting of polynomials.* For any polynomial f of degree $\deg f < \rho n$, there exist two polynomials g, h of degree $< \frac{1}{2}\rho n$ such that

$$f(x) = g\left(x^2\right) + x \cdot h\left(x^2\right). \tag{1}$$

One may view such a decomposition as the result of the splitting of the space of polynomials of degree less than ρn into two copies of the space of polynomials of degree less than $\rho n/2$.

b) *Randomized folding.* Choose $\mathcal{P}$ to be a multiplicative group of order 2^r generated by $\omega \in \mathbb{F}$. Then, define $\mathcal{P}' = \langle \omega^2 \rangle = \{x^2 \mid x \in \mathcal{P}\}$. Set $\pi : \mathbb{F} \to \mathbb{F}$ to be the map defined by $\pi(x) = x^2$, observe that $\pi(\mathcal{P}) = \mathcal{P}'$. Moreover, $|\mathcal{P}'| = |\mathcal{P}|/2$. The structure of the evaluation domain will allow to reduce the problem of proximity to one of half the size at each round of interaction.
Based on the decomposition (1), define a *folding operator* $\mathbf{Fold}[\cdot, z] : \mathbb{F}^{\mathcal{P}} \to \mathbb{F}^{\mathcal{P}'}$ for any $z \in \mathbb{F}$ as follows:

$$\mathbf{Fold}[f, z] := g + zh.$$

If $\deg f < \rho n$, both functions $g : \mathcal{P}' \to \mathbb{F}$ and $h : \mathcal{P}' \to \mathbb{F}$ belong to $\mathsf{RS}[\mathbb{F}, \mathcal{P}', \rho]$. Then for any random challenge $z \in \mathbb{F}_q$, the operator $\mathbf{Fold}[\cdot, z]$ maps $\mathsf{RS}[\mathbb{F}, \mathcal{P}, \rho]$ into $\mathsf{RS}[\mathbb{F}, \mathcal{P}', \rho]$.

c) *Distance preservation after folding.* Except with small probability over z, we have that if $\Delta(f, \mathsf{RS}[\mathbb{F}, \mathcal{P}, \rho]) \geq \delta$, then

$$\Delta\left(\mathbf{Fold}[f, z], \mathsf{RS}[\mathbb{F}, \mathcal{P}', \rho]\right) \geq (1 - o(1))\delta.$$

The protocol then goes as follows: the verifier sends a random challenge $z \in \mathbb{F}$ and the prover answers with an oracle function $f' : \mathcal{P}' \to \mathbb{F}$, which is expected to be equal to $\mathbf{Fold}[f, z] : \mathcal{P}' \to \mathbb{F}$. At the next round, f' becomes the function to be "folded", and the process is repeated for r rounds. Each round reduces the problem by half, eventually leading to a function $f^{(r)}$ evaluated over a small enough evaluation domain. This induces a sequence of Reed-Solomon codes of strictly decreasing length. The code rate remains unchanged, and so does the relative minimum distance. The final test consists in testing that $f^{(r)}$ belongs to the last RS code.

Perfect completeness follows from Item b. Prover and verifier efficiencies of the FRI protocol come from the possibility of determining any value of $\mathbf{Fold}[f, z]$ at a point $y \in \mathcal{P}'$ with exactly two values of f, namely on the set $\pi^{-1}(\{y\})$. Consequently, a single test of consistency between f and f' requires only two queries to f and one query to f'.

Soundness of the protocol relies notably on Item c. It is proved using results about distance preservation under random linear combinations, that could be roughly stated as follows: "*Let $V \subset \mathbb{F}_q^n$ be a linear code and $g, h \in \mathbb{F}_q^n$. As long as δ is small enough, if we have $\Delta(g + zh, V) \leq \delta$ for enough values $z \in \mathbb{F}_q$, then both g and h are δ'-close to V, where $\delta' = (1 - o(1))\delta$.*" (see [9, 18, 16, 11]). Based on that, one can deduce that if $\mathbf{Fold}[f, z] = g + zh$ is δ-close to V for enough values of z, then both g and h are δ'-close from V. The idea of the proof of Item c is to exhibit a codeword which is δ-close from f, based on the decomposition of Item a.

▶ Remark 3. We point out that Item c holds because the functions g and h appearing in the decomposition (1) have *exactly* the same degree. This arises from the crucial fact that the FRI protocol considers only RS code of dimension a power of 2. This means that the RS code is defined by polynomials of degree at most an *odd* bound.

Let us give a glimpse of what happens when f is expected to have degree at most an even integer, say $2d$. The degrees of the functions g and h appearing in the decomposition (1) of f are respectively $\deg g \leq d$ and $\deg h \leq d-1$. Therefore, if $\deg f \leq 2d$, then $g + zh$ corresponds to a polynomial of degree $\leq d$. However, knowing that $g + zh$ is a polynomial of degree $\leq d$ with high probability on z only tells us that both g and h are of degree $\leq d$, which is not enough to deduce that f has degree $\leq 2d$ and not $2d+1$. It is worth noting that words corresponding to a polynomial of degree $2d+1$ are among the *farthest* words from the RS code of degree $\leq 2d$. In the univariate case, one can overcome this obstacle by supposing not only $\deg g, \deg h \leq d$ but also $\deg(\nu h) \leq d$ for a degree-1 polynomial function ν. This implies that $\deg h < d$, hence $\deg f \leq 2d$.

2.2 Our IOPP for AG proximity testing

Let $\mathcal{X}$ be a curve defined over a finite field $\mathbb{F}$ and $C = C(\mathcal{X}, \mathcal{P}, D)$ be an AG code. We aim to adapt the three ingredients of the FRI protocol to the AG context.

Group actions and Riemann-Roch spaces. The splitting of the polynomial f into an even and an odd part in (1) comes from the action of a multiplicative group of order 2 on the evaluation set $\mathcal{P}$. This observation is also true with the actual FRI protocol, which sets π to be an affine subspace polynomial. This phenomenon is likely to occur in a more general framework.

As soon as a group Γ of order p acts on the curve $\mathcal{X}$, its action naturally extends onto the functions on $\mathcal{X}$. Let us denote by π the canonical projection $\pi : \mathcal{X} \to \mathcal{X}/\Gamma$. If we are able to write the Riemann-Roch space associated to D as following

$$f = \sum_{j=0}^{p-1} \mu^j f_j \circ \pi \text{ with } f_j \in L_{\mathcal{X}/\Gamma}(E_j), \tag{$\star$}$$

for some function μ on the curve $\mathcal{X}$ and some divisors E_j on the quotient curve that are explicitly expressed in terms of the divisor D, then we can mimic the decomposition (1) used in the FRI protocol.

Now assume that no point of $\mathcal{P}$ is fixed by Γ, *i.e.* for every $P \in \mathcal{P}$ and $\gamma \in \Gamma$, $\gamma \cdot P \neq P$. Set $\mathcal{P}' = \pi(\mathcal{P})$. Polynomial interpolation enables the determination of $f_j(P)$ for any point $P \in \mathcal{P}'$ with exactly p values of f, namely on the set $\pi^{-1}(\{P\})$. This means that the decomposition ($\star$) can be written for any function in $\mathbb{F}^{\mathcal{P}}$, not only for elements of $L_{\mathcal{X}}(D)$.

Folding operators. From the decomposition ($\star$) above, we want to define a family of folding operators $(\mathbf{Fold}[\cdot, z])_{z \in \mathbb{F}}$ from $\mathbb{F}^{\mathcal{P}}$ to $\mathbb{F}^{\mathcal{P}'}$ and a code $C' = C(\mathcal{X}/\Gamma, \mathcal{P}', D')$ such that $\mathbf{Fold}[\cdot, z](C) \subseteq C'$.

In a first approach, one could choose to define the folding operators similarly to the FRI protocol by setting for $z \in \mathbb{F}$,

$$\mathbf{Fold}[f, z] = \sum_{j=0}^{p-1} z^j f_j$$

where the functions f_j come from the decomposition $(\star)$ of $f \in \mathbb{F}^{\mathcal{P}}$. With this definition, the code C' has to be associated to a divisor D' on $\mathcal{X}/\Gamma$ such that each Riemann-Roch space $L_{\mathcal{X}/\Gamma}(E_j)$ can be embedded into $L_{\mathcal{X}/\Gamma}(D')$.

The best scenario is when the divisor D yields a decomposition of $L_{\mathcal{X}}(D)$ as p "copies" of the same Riemann-Roch space, as it is the case with Reed-Solomon codes of even dimension. Unfortunately, to the best of our knowledge, it is unlikely that all divisors E_j involved in the decomposition $(\star)$ of f are the same (or even equivalent) *if $\mathcal{X}$ is not the projective line*. We are then facing an issue analogous to the one described in Remark 3 on $\mathbb{P}^1$.

Therefore, such a choice of the folding operators does not guarantee the soundness of our protocol. We thus aim to adapt the idea at the end of Remark 3 to the AG setting. We introduce some *balancing* functions ν_j such that, for every $f_j \in C'$, if the product $\nu_j f_j$ also lies in C', then the function f_j belongs to the desired Riemann-Roch space $L_{\mathcal{X}/\Gamma}(E_j)$. Defining such a balancing function ν_j is tantamount to specify its pole order at the points supporting the divisor D'. The existence of all the functions ν_j thus depends on the *Weierstrass semigroup* of these points (see [26, Section 6.6] for definition) and does not hold for any divisor D'. If such balancing functions exist for a divisor D', we say that D' is *compatible* with D. Finding a convenient divisor D' compatible with a given divisor D is definitely the trickiest part in defining the folding operators properly.

Once we have a divisor D' that is D-compatible, we shall embed additional terms in the folding operators to take account of the balancing functions. We shall use more randomness so as not to double the degree in z to avoid a loss in soundness. For $(z_1, z_2) \in \mathbb{F}^2$, we set

$$\textbf{Fold}\left[f, (z_1, z_2)\right] = \sum_{j=0}^{p-1} z_1^j f_j + \sum_{j=0}^{p-1} z_2^{j+1} \nu_j f_j.$$

We prove that $\textbf{Fold}\left[\cdot, (z_1, z_2)\right](C) \subseteq C'$, the function $\textbf{Fold}\left[f, (z_1, z_2)\right] \in \mathbb{F}^{\mathcal{P}'}$ can be locally computed from p values of f, and $\textbf{Fold}\left[\cdot, (z_1, z_2)\right]$ preserves the distance to the code.

Finding a decomposition $(\star)$. Such a decomposition exists for a cyclic group $\Gamma = \langle\gamma\rangle$ whose order is coprime with the characteristic. A result of Kani [29] states that, in this case, there exists a function μ on $\mathcal{X}$ satisfying $(\star)$ such that $\gamma \cdot \mu = \zeta\mu$ where ζ is a primitive root of unity of order $|\mathcal{G}|$. Moreover, the divisor E_j can be explicitly written in terms of the divisor D and the function μ.

Alternatively, if a basis of $L_{\mathcal{X}}(D)$ is explicitly known, we may also be able to exhibit such a decomposition without invoking Kani's theorem for some cyclic group of order divisible by the characteristic. This is exactly the strategy we use to design an IOPP for AG codes along the Hermitian tower.

Soundness preservation. The soundness of the protocol depends on the relative minimum distance of the codes C and C'. Ideally , we would like the rates of the codes C and C' to be roughly equal to prevent the relative minimum distance from dropping. In particular, we need $L_{\mathcal{X}/\Gamma}(D')$ to be not too large with respect to the components $L_{\mathcal{X}/\Gamma}(E_j)$.

A natural idea would be to choose D' as the divisor E_j with the largest Riemann-Roch space. However, balancing functions only exists for some well-chosen divisors D', whose degree can be significantly larger than the degree of the divisor E_j in $(\star)$. Therefore, the divisors D and D' have to be carefully chosen to prevent the minimum distance from collapsing.

Sequence of "foldable" AG codes. With the goal of iterating the folding process in mind, we assume that the base curve $\mathcal{X}_0 \coloneqq \mathcal{X}$ is endowed with a *suitable* acting group $\mathcal{G}$ that we decompose into smaller groups $\Gamma_0, \Gamma_1, \ldots, \Gamma_{r-1}$ to fragment its action and create intermediary quotients

$$\mathcal{X}_0 \xrightarrow{\pi_0} \mathcal{X}_1 \xrightarrow{\pi_1} \mathcal{X}_2 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_{r-1}} \mathcal{X}_r,$$

where the morphism $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ is the quotient map by Γ_i. For instance, a sufficient condition on the group $\mathcal{G}$ to have such a sequence is the *solvability*.

A code $C = C(\mathcal{X}, \mathcal{P}, D)$ is said to be a *foldable AG code* (Definition 8) if we are able to construct a sequence of AG codes $C_i \coloneqq C(\mathcal{X}_i, \mathcal{P}_i, D_i)$ that support a family of randomized folding operators $\mathbf{Fold}\,[\cdot, \boldsymbol{z}] : \mathbb{F}^{\mathcal{P}_i} \to \mathbb{F}^{\mathcal{P}_{i+1}}$ with the desirable properties for our IOPP (i.e. $\mathbf{Fold}\,[\cdot, \boldsymbol{z}]\,(C_i) = (C_{i+1})$, local computability, distance preservation to the code). Moreover, to ensure that the last code C_r has sufficiently small length and to obtain an IOPP with sublinear query complexity, we require the size of $\mathcal{G}$ to be greater than $|\mathcal{P}|^e$ for a certain $e \in (0, 1)$. Details are provided in Section 4.

3 Preliminaries

We start with some reminders on important terms and notations related to the theory of AG codes. We refer readers to [48, 45] for further details on these notions. We will always use $\mathbb{F}$ to denote a finite field.

3.1 Functions and divisors on algebraic curves

Let $\mathcal{X}$ be an algebraic curve defined over a field $\mathbb{F}$. Let $\overline{\mathbb{F}}$ be an algebraic closure of the field $\mathbb{F}$. We denote by $\mathcal{X}(\mathbb{F})$ the set of its $\mathbb{F}$-rational points and $\mathrm{Aut}(\mathcal{X})$ its automorphism group.

A *divisor* D on $\mathcal{X}$ is a formal sum of points $D = \sum n_P P$. We say that the divisor D is *effective* if $n_P \geq 0$ for every point P. The *support* of D, denoted by $\mathrm{Supp}(D)$, is the set of points P for which the coefficient n_P is non zero. We will always consider *rational* divisors, whose support only consists in $\mathbb{F}_q$-rational points. We define the *degree* of D equals $\deg D \coloneqq \sum n_P$.

The set of divisors on the curve $\mathcal{X}$ forms an additive group, denoted by $\mathrm{Div}(\mathcal{X})$. It is endowed with a partial order relation $\leq$ such that $D \leq D'$ if $D' - D$ is effective. An element f of the function field $F \coloneqq \mathbb{F}(\mathcal{X})$ of the curve $\mathcal{X}$ defines a divisor

$$\mathrm{div}_F(f) = \sum_{P \in \mathcal{X}} v_P(f) P$$

where $v_P(f)$ is the valuation of the function f at the point P. The index F will be omitted when the context is clear.

We denote by $\mathrm{div}_0(f)$ (respectively $\mathrm{div}_\infty(f)$) the positive (respectively negative) part of the principal divisor $\mathrm{div}(f)$, *i.e.*

$$\mathrm{div}_0(f) = \sum_{\substack{P \in \mathcal{X} \\ v_P(f) > 0}} v_P(f) P \quad \text{and} \quad \mathrm{div}_\infty(f) = \sum_{\substack{P \in \mathcal{X} \\ v_P(f) < 0}} v_P(f) P$$

so that $\mathrm{div}(f) = \mathrm{div}_0(f) - \mathrm{div}_\infty(f)$. The divisors $\mathrm{div}_0(f)$ and $\mathrm{div}_\infty(f)$ correspond to the loci of zeroes and poles respectively.

Let $\phi : \mathcal{X} \to \mathcal{X}'$ be a map between two algebraic curves. It induces a *pull-back* map $\phi^* : \mathbb{F}(\mathcal{X}') \to \mathbb{F}(\mathcal{X})$ defined by $\phi^* f = f \circ \phi$ for $f \in \mathbb{F}(\mathcal{X}')$. For $D = \sum_P n_p P \in \mathrm{Div}(\mathcal{X})$, the *push-forward* of D is the divisor on $\mathcal{X}'$ defined by $\pi_*(D) = \sum_P n_P \phi(P)$.

The *Riemann-Roch space* of a divisor $D \in \mathrm{Div}(\mathcal{X})$ is the $\mathbb{F}$-vector space defined by

$$L_{\mathcal{X}}(D) = \{f \in \mathbb{F}(\mathcal{X}) \mid \mathrm{div}_F(f) + D \geq 0\} \cup \{0\}.$$

The subscript specifying the curve in $L_{\mathcal{X}}(D)$ is omitted when it is clear from the context. If $D' \leq D'$, then $L_{\mathcal{X}}(D) \subseteq L_{\mathcal{X}}(D')$.

As usual, given a real number α, $\lfloor \alpha \rfloor$ denotes the biggest integer less than or equal to α and $\lceil \alpha \rceil$ the smallest integer bigger than or equal to α.

▶ **Definition 4.** *Let $D = \sum n_P P \in \mathrm{Div}(\mathcal{X})$. For any positive integer n, we denote by $\left\lfloor \frac{1}{n} D \right\rfloor \in \mathrm{Div}(\mathcal{X})$ the divisor defined by*

$$\left\lfloor \frac{1}{n} D \right\rfloor := \sum \left\lfloor \frac{n_P}{n} \right\rfloor P.$$

3.2 Algebraic geometry codes

Throughout this paper, the term *code* will refer to a *linear code*, i.e. a linear subspace of $\mathbb{F}^n$, where n is the length of the code.

Take $D \in \mathrm{Div}(\mathcal{X})$ and $\mathcal{P} \subset \mathcal{X}(\mathbb{F})$ of size $n := |\mathcal{P}|$ such that $\mathrm{Supp}(D) \cap \mathcal{P} = \emptyset$. The *Algebraic Geometry (AG) code* $C = C(\mathcal{X}, \mathcal{P}, D)$ is defined as the image under the evaluation map

$$\mathrm{ev} : L(D) \to \mathbb{F}^n.$$

The integer n is called the *length* of C. The *dimension* of C is defined as its dimension as $\mathbb{F}$-vector space. We denote by $\Delta(C)$ the relative minimum distance of C, i.e.

$$\Delta(C) = \min \{\Delta(c, c') \mid c, c' \in C \text{ and } c \neq c'\}.$$

In particular, AG codes on $\mathcal{X} = \mathbb{P}^1$ correspond to Reed-Solomon codes. The AG code C is said to be *one-point* if the support of D consists in a single point.

By the Riemann-Roch theorem, if $\deg D \geq 2g - 1$ where g is the genus of the curve $\mathcal{X}$, then $\dim L_{\mathcal{X}}(D) = \deg D - g + 1$. Moreover, if $\deg D < n$, the evaluation map is injective and the Riemann-Roch theorem gives the dimension of the associated AG code. In this case, the minimum distance is bounded from below by $n - \deg D$.

The divisor D will always be chosen so that the map ev is injective. Therefore, the elements of $\mathbb{F}^n$ will be regarded as functions in $\mathbb{F}^{\mathcal{P}}$, and elements of C simply as functions in the Riemann-Roch space $L(D)$.

3.3 Group and action

A finite group $\mathcal{G}$ is said to be *solvable* if there exists a sequence of subgroups of $\mathcal{G}$

$$\mathcal{G} = \mathcal{G}_0 \rhd \mathcal{G}_1 \rhd \cdots \rhd \mathcal{G}_r = 1,$$

such that $\mathcal{G}_{i+1}$ is a normal subgroup of $\mathcal{G}_i$ and each factor group $\mathcal{G}_i / \mathcal{G}_{i+1}$ is abelian. Such a sequence is called a *normal series*. If $\mathcal{G}$ is solvable, its cardinality equals the product of the sizes of the factor groups. Additionally, a normal series of $\mathcal{G}$ is a *composition series* of $\mathcal{G}$ if

each $\mathcal{G}_{i+1}$ is a maximal proper normal subgroup of $\mathcal{G}_i$. In the case where $\mathcal{G}$ is a finite group, we have that $\mathcal{G}$ is solvable if and only if $\mathcal{G}$ has a composition series whose factor groups are cyclic groups of prime order.

Let $\mathcal{X}$ be an algebraic curve. A group Γ is said to *act on the curve* $\mathcal{X}$ if Γ is a subgroup of the automorphism group $\mathrm{Aut}(\mathcal{X})$. The *stabilizer* of a point $P \in \mathcal{X}$ is the subgroup

$$\Gamma_P = \{\gamma \in \Gamma \mid \gamma \cdot P = P\} \subset \Gamma.$$

A divisor $D = \sum_P n_P P \in \mathrm{Div}(\mathcal{X})$ is said to be Γ-*invariant* is $n_P = n_{\gamma \cdot P}$ for all $P \in \mathcal{X}$ and $\gamma \in \Gamma$.

The action of Γ on $\mathcal{X}$ gives a projection $\pi : \mathcal{X} \to \mathcal{X}/\Gamma$ onto the quotient curve $\mathcal{X}/\Gamma$. A point $Q \in \mathcal{X}/\Gamma$ is called a *ramification point* if the number of preimages of Q by π is not equal to $|\Gamma|$. Equivalently, Q is a ramification point if one of its preimages has a non-trivial stabilizer.

4 Setting of AG codes compatible with proximity test

In this section, we display a workable setting for the construction of an IOPP system (P, V) to test whether a given function $f : \mathcal{P} \to \mathbb{F}$ is close to the evaluation of a function in a given Riemann-Roch space. As the idea is to iteratively reduce the problem of testing proximity to $C(\mathcal{X}, \mathcal{P}, D)$ to testing proximity to a smaller AG code, we introduce a sequence of suitable AG codes of decreasing length.

4.1 Sequence of curves

Fix a curve $\mathcal{X}$ defined over $\mathbb{F}$, a finite solvable group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ and a sequence

$$\boldsymbol{\mathcal{G}} \coloneqq (\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_r)$$

such that

$$\mathcal{G} = \mathcal{G}_0 \rhd \mathcal{G}_1 \rhd \cdots \rhd \mathcal{G}_r = 1, \tag{2}$$

is a normal series for the group $\mathcal{G}$.

For $i \in \{0, \ldots, r-1\}$, we denote by Γ_i the (abelian) factor group $\Gamma_i \coloneqq \mathcal{G}_i/\mathcal{G}_{i+1}$ and by p_i the order of Γ_i. We have that the cardinality of $\mathcal{G}$ equals

$$|\mathcal{G}| = \prod_{i=0}^{r-1} |\Gamma_i| = \prod_{i=0}^{r-1} p_i.$$

The group Γ_0 acts on $\mathcal{X}_0 \coloneqq \mathcal{X}$, as a factor group of $\mathcal{G}$. We thus define the quotient curve $\mathcal{X}_1 \coloneqq \mathcal{X}_0/\Gamma_0$. The group Γ_1 acts trivially on the orbits under Γ_0. Repeating the process for every $i \in \{0, \ldots, r-1\}$ defines a sequence of curves recursively as follows:

$$\mathcal{X}_0 := \mathcal{X} \text{ and } \mathcal{X}_{i+1} \coloneqq \mathcal{X}_i/\Gamma_i.$$

We set $F_i \coloneqq \mathbb{F}(\mathcal{X}_i)$ and we denote by $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ the canonical projection modulo the action of Γ_i. Even if the sequence of curves (3) depends on the derived series (2) of $\mathcal{G}$, the last curve $\mathcal{X}_r$ is always isomorphic to the quotient $\mathcal{X}/\mathcal{G}$:

$$\mathcal{X}_0 \xrightarrow{\pi_0} \mathcal{X}_1 \xrightarrow{\pi_1} \cdots \xrightarrow{\pi_i} \mathcal{X}_i \xrightarrow{\pi_{i+1}} \mathcal{X}_{i+1} \longrightarrow \cdots \xrightarrow{\pi_{r-1}} \mathcal{X}_r. \tag{3}$$

(with Γ_0 acting on $\mathcal{X}_0$, Γ_1 on $\mathcal{X}_1$, Γ_i on $\mathcal{X}_i$, and Γ_{i+1} on $\mathcal{X}_{i+1}$)

▶ **Definition 5.** *A sequence of curves constructed as above will be called a* $(\mathcal{X}, \boldsymbol{\mathcal{G}})$-sequence.

4.2 Sequence of codes

Let $(\mathcal{X}_i)$ be a $(\mathcal{X}, \mathcal{G})$-sequence. For any $i \in \{0, \dots, r-1\}$, the factor group Γ_i which acts on the curve $\mathcal{X}_i$ is abelian of order p_i.

For $i \in \{0, \dots, r\}$, we aim to define an AG code $C_i \subset \mathbb{F}^{\mathcal{P}_i}$ associated to a divisor $D_i \in \mathrm{Div}(\mathcal{X}_i)$ and an evaluation set $\mathcal{P}_i$. The rest of this subsection is dedicated to the choice of the sets $\mathcal{P}_i$ and the divisors D_i.

4.2.1 Evaluation points

From a set $\mathcal{P}_0 \subset \mathcal{X}(\mathbb{F})$, we want to recursively define a sequence of set of points $(\mathcal{P}_i)$ such that $P_i \subseteq \mathcal{X}_i(\mathbb{F})$ and $\mathcal{P}_{i+1} = \pi_i(\mathcal{P}_i)$.

For our protocol, we need for each $i \in \{0, \dots, r-1\}$ that every point in $\mathcal{P}_{i+1}$ admits exactly p_i preimages under π_i. Since the last curve $\mathcal{X}_r$ is isomorphic to the quotient $\mathcal{X}/\mathcal{G}$, it is necessary and sufficient that the first set $\mathcal{P}_0 \subset \mathcal{X}_0$ is a union of $\mathcal{G}$-orbits of size $|\mathcal{G}|$, *i.e.* that $\mathcal{G}$ acts freely on $\mathcal{P}_0$.

4.2.2 Divisors

Fix a divisor $D_0 \in \mathrm{Div}(\mathcal{X}_0)$ that is globally Γ_0-invariant. This way, the support of D_0 does not meet the set $\mathcal{P}_0$. For the sake of simplicity, we will assume that D_0 is in fact supported by Γ_0-fixed points. To make our protocol complete and sound, we need the sequence of divisors (D_i) to have the following properties:

- the divisor D_i is supported by Γ_i-invariant points;
- for each divisor D_i, its associated Riemann-Roch space admits a nice decomposition (given in (4) below);
- at each step, a divisor D_{i+1} needs to be compatible with D_i and the decomposition of $L_{\mathcal{X}_i}(D_i)$ in the sense of Definition 7 below.

▶ **Definition 6.** *Let $i \in \{0, \dots, r-1\}$. Fix a divisor $D_i \in \mathrm{Div}(\mathcal{X}_i)$ and a function $\mu_i \in F_i$. We say that μ_i* partitions *$L_{\mathcal{X}_i}(D_i)$ (with respect to the action of Γ_i) if*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{p_i-1} \mu_i^j \pi_i^* L_{\mathcal{X}_{i+1}}(E_{i,j}) \tag{4}$$

with

$$E_{i,j} \coloneqq \left\lfloor \frac{1}{p_i} \pi_{i*}(D_i + j \,\mathrm{div}_{F_i}(\mu_i)) \right\rfloor \in \mathrm{Div}(\mathcal{X}_{i+1}) \text{ for } j \in \{0, \dots, p_i - 1\}\,, \tag{5}$$

where the floor function of a divisor is given in Definition 4.

▶ **Definition 7.** *Let $i \in \{0, \dots, r-1\}$. Fix a divisor $D_i \in \mathrm{Div}(\mathcal{X}_i)$ and a function $\mu_i \in F_i$ such that μ_i partitions $L_{\mathcal{X}_i}(D_i)$. A divisor $D_{i+1} \in \mathrm{Div}(\mathcal{X}_{i+1})$ is said to be* compatible with *(D_i, μ_i) if both assertions hold.*

1. *for every $j \in \{0, \dots, p_i - 1\}$, $E_{i,j} \le D_{i+1}$,*
2. *for every $j \in \{0, \dots, p_i - 1\}$, there exists a function $\nu_{i+1,j} \in \mathbb{F}(\mathcal{X}_{i+1})$ such that*

$$\mathrm{div}_\infty(\nu_{i+1,j}) = D_{i+1} - E_{i,j}. \tag{6}$$

The functions $\nu_{i+1,j}$ are called balancing functions.

In Definition 7, the first requirement implies that $L(E_{i,j}) \subseteq L(D_{i+1})$. The second one means that a function f_j belongs to $L(E_{i,j})$ if and only if the product $\nu_{i+1,j} f_j$ lies in $L(D_{i+1})$.

We have now described all the key components to formally define the notion of foldable codes.

▶ **Definition 8** (Foldable AG codes). *Let $C = C(\mathcal{X}, \mathcal{P}, D)$ be an AG-code. This code is said to be* foldable *if the following conditions are satisfied.*

1. *There exists a finite solvable group $\mathcal{G} \in \mathrm{Aut}(\mathcal{X})$ that acts freely on $\mathcal{P}$: a normal series of $\mathcal{G}$ (2) provides a $(\mathcal{X}, \boldsymbol{\mathcal{G}})$-sequence of curves $(\mathcal{X}_i)$;*
2. *There exists $e \in (0,1)$ such that $|\mathcal{G}| > |\mathcal{P}|^e$;*
3. *There exist some sequences of functions (μ_i) and divisors (D_i) where $\mu_i \in F_i, D_0 = D$ and $D_i \in \mathrm{Div}(\mathcal{X}_i)$ such that, for every $i \in \{0, \dots, r-1\}$, all the following properties hold:*
   a. *the divisors D_i are supported by Γ_i-fixed points,*
   b. *the function μ_i partitions $L_{\mathcal{X}_i}(D_i)$ (Definition 6),*
   c. *the function μ_i maps distinct points in the same Γ_i-orbit to distinct values: for every $\gamma_i \in \Gamma_i$ and $P \in \mathcal{P}_i$, we have $\mu_i(P) = \mu_i(\gamma_i(P))$ if and only if γ_i is the identity map,*
   d. *D_{i+1} is (D_i, μ_i)-compatible (Definition 7).*

The second requirement given in Definition 7 is definitely compelling and requires some geometric knowledge about the curves $\mathcal{X}_i$. Indeed, on a general curve, not every effective divisor is the poles locus of a function. Characterizing which effective divisors arise this way is at the heart of the Weierstrass gaps theory. Nonetheless, the existence of the balancing functions $\nu_{i+1,j}$ happens to be the main ingredient in Lemma 41, which will take a prominent role in the design of our IOPP.

To prevent the relative minimum distance of the code C_{i+1} from collapsing and thence ensure a good soundness of the protocol designed in Section 7, one may be tempted to take D_{i+1} as one of the divisors $E_{i,j}$ (5) that appear in the decomposition (4) of $L_{\mathcal{X}_i}(D)$. However the Weierstrass gaps theory indicates that balancing functions exist only when choosing a (D_i, μ_i)-compatible divisor D_{i+1} whose degree may be unexpectedly substantial (an illustration of this situation will be given in Example 20). Therefore, to broaden the spectrum of foldable codes, we do not make this additional hypothesis.

Condition 3c in Definition 8 is necessary for the completeness of our protocol. If the divisors D_i have degree sufficiently large, it is always fulfilled, as proven in the following lemma.

▶ **Lemma 9.** *Let us assume that a function $\mu_i \in F_i$ partitions $L_{\mathcal{X}_i}(D_i)$ with respect to the action of Γ_i for some divisor $D_i \in \mathrm{Div}(\mathcal{X}_i)$. Let us denote by g_i the genus of the curve $\mathcal{X}_i$. If $\deg(D_i) \geq 2g_i + 1$, then μ_i takes different values at points lying in the same orbit under Γ_i.*

Proof. Let us assume by contradiction that there exists two distinct points P and Q on $\mathcal{X}_i$ such that $\pi_i(P) = \pi_i(Q)$ and $\mu_i(P) = \mu_i(Q)$. By Definition 6, every function f in $L(D_i)$ can be written $f = \sum \mu_i^j f_j \circ \pi_i$ some some functions f_j on the quotient curve $\mathcal{X}_{i+1}$. Then, for every $f \in L(D_i)$, we have $f(P) = f(Q)$. In particular, this means that a function of $L(D_i)$ vanishes at P if and only if its vanishes at Q. Therefore, the Riemann-Roch spaces $L(D_i - P)$ and $L(D_i - P - Q)$ are equal, which contradicts the Riemann-Roch theorem, since $\deg(D_i - P - Q) \geq 2g_i - 1$. ◀

4.3 RS codes are foldable AG codes

Reed-Solomon codes are AG codes on the projective line $\mathbb{P}^1$. Moreover the Riemann-Roch space $L_{\mathbb{P}^1}(dP_\infty)$ on $\mathbb{P}^1$ for $P_\infty = [0:1]$ is isomorphic to the set of polynomials of degree (less than or equal to) d.

In this case, the decomposition (4) is nothing but the splitting of a polynomial into an even part and an odd part, which plays a crucial role in the FRI protocol when the characteristic is not 2.

To make both points of view coincide, let us consider the involution $\gamma : [X_0 : X_1] \mapsto [-X_0 : X_1]$. It generates a group isomorphic to $\mathbb{Z}/2\mathbb{Z}$ and the quotient of $\mathbb{P}^1$ by this group is obtained as the image by $\pi : [X_0 : X_1] \mapsto [X_0^2 : X_1^2]$.

The divisor $D \coloneqq dP_\infty$ is invariant under γ. Let us choose μ as the function $x = \frac{X_0}{X_1}$. We have $\mathrm{div}(x) = P_0 - P_\infty$ with $P_0 = [1 : 0]$. Noticing that $\pi_*(P_\infty) = P_\infty$ and $\pi_*(P_0) = P_0$, we get

$$\left\lfloor \frac{1}{2}\pi_*(D + (x)) \right\rfloor = \left\lfloor \frac{1}{2}((d-1)P_\infty + P_0) \right\rfloor = \left\lfloor \frac{d-1}{2} \right\rfloor P_\infty,$$

and the Riemman-Roch space $L_{\mathbb{P}^1}(dP_\infty)$ is split into two parts:

$$L_{\mathbb{P}^1}(dP_\infty) = \pi^* L_{\mathbb{P}^1}\left(\left\lfloor \frac{d}{2} \right\rfloor P_\infty\right) + x\pi^* L_{\mathbb{P}^1}\left(\left\lfloor \frac{d-1}{2} \right\rfloor P_\infty\right).$$

We recover the decomposition of a polynomial of degree d into even and odd parts of respective degrees $\left\lfloor \frac{d}{2} \right\rfloor$ and $\left\lfloor \frac{d-1}{2} \right\rfloor$.

▶ Remark 10. The function μ is not unique: any odd polynomial of x would make a suitable choice for μ.

Now, let us remark that the RS code

$$V \coloneqq \left\{ f \in \mathbb{F}^{\mathcal{P}}; \deg f \leq d \right\} = C(\mathbb{P}^1, \mathcal{P}, dP_\infty)$$

is a foldable AG code, for any $\mathcal{P} \subset \mathbb{F}$ of size $|\mathcal{P}| = 2^r$ for a certain integer r and any degree bound d. We shall then retrieve the construction of the RS proximity test of [9].

Firstly, the finite solvable group $\mathbb{Z}/2^r\mathbb{Z}$ of size $|\mathcal{P}|$ acts on $\mathbb{P}^1$ via $[X_0 : X_1] \mapsto [X_0, \xi X_1]$, where ξ is a primitive 2^r-th root unity. It clearly fulfils the two first items of Definition 8. When considering its composition series

$$\mathbb{Z}/2^r\mathbb{Z} \rhd \mathbb{Z}/2^{r-1}\mathbb{Z} \rhd \cdots \rhd 1 \tag{7}$$

and the action of the corresponding factor group $\Gamma = \langle \gamma \rangle \simeq \mathbb{Z}/2\mathbb{Z}$, we obtain a trivial sequence of curves $(\mathcal{X}_i)$ with $\mathcal{X}_i = \mathbb{P}^1$ for all i. Next, consider the sequence (μ_i) with $\mu_i = \mu = x \coloneqq \frac{X_1}{X_0}$, then $\gamma\mu = -\mu$. Set $d_0 \coloneqq d$, and for any $i \in \{0, \ldots, r-1\}$, $d_{i+1} \coloneqq \left\lfloor \frac{d_i}{2} \right\rfloor$. Note that there exists $r' < r$ such that $d_{r'}, \ldots, d_r$ are all equal to 0. Setting $D_i = \left\lfloor \frac{d_i}{2} \right\rfloor P_\infty$, we have Γ_i-invariant divisors fulfilling the compatibility condition given in Definition 7, by letting $\nu_{i+1,j}$ to be the constant function equal to 1 if $\left\lfloor \frac{d_i}{2} \right\rfloor = \left\lfloor \frac{d_i - 1}{2} \right\rfloor$, and $\nu_{i+1,j} : x \mapsto x$ otherwise.

4.4 Splitting Riemann-Roch spaces according to a cyclic group of automorphisms

The first requirement to make a sequence of codes foldable is the splitting the Riemann-Roch spaces as in (4), which mimics the decomposition in odd and even parts of univariate polynomials. Under some additional hypotheses, a decomposition like (4) always exists. Let us detail this framework.

Let $\mathcal{X}$ be a smooth irreducible curve over a field $\mathbb{F}$ and let Γ be a *cyclic* group of order m generated by an element γ that acts on $\mathbb{F}(\mathcal{X}) \times \overline{\mathbb{F}}$. Assume that m and the characteristic of $\mathbb{F}$ are coprime and consider $\zeta \in \overline{\mathbb{F}}$ a primitive m^{th} root of unity, lying in some algebraic closure $\overline{\mathbb{F}}$ of $\mathbb{F}$.

Set $\mathcal{Y} := \mathcal{X}/\Gamma$ and $\pi : \mathcal{X} \to \mathcal{Y}$ be the canonical projection morphism.

Fix a Γ-invariant divisor $D \in \mathrm{Div}(\mathcal{X})$. We want to exhibit a relation between the Riemann-Roch space $L_{\mathcal{X}}(D)$ and some Riemann-Roch spaces on $\mathcal{Y}$. The group Γ acts on the vector space $L_{\mathcal{X}}(D)$ via $\gamma \cdot f = f \circ \gamma$. By representation theory,

$$L_{\mathcal{X}}(D) = \bigoplus_{j=0}^{m-1} L_{\mathcal{X}}(D)_j,$$

where $L_{\mathcal{X}}(D)_j \coloneqq \{g \in L_{\mathcal{X}}(D) \mid \gamma \cdot g = \zeta^j g\}$.

One of the key ingredients of this section is a theorem due to Kani [29], which we reformulate here in the case where Γ is cyclic.

▶ **Theorem 11** ([29]). *Let $\Gamma = \langle \gamma \rangle$ be a cyclic group that acts on $\mathbb{F}(\mathcal{X}) \times \overline{\mathbb{F}}$. Assume that the order $m = |\Gamma|$ is coprime with $|\mathbb{F}|$. Then the two following statements hold.*

1. *There exists a function $\mu \in \overline{\mathbb{F}}(\mathcal{X})$ such that $\gamma \cdot \mu = \zeta\mu$.*
2. *For any Γ-invariant divisor $D \in \mathrm{Div}(\mathcal{X})$, when considering the Riemman-Roch spaces over the algebraic closure $\overline{\mathbb{F}}$, we have*

$$L_{\mathcal{X}}(D)_j \otimes \overline{\mathbb{F}} \simeq \mu^j \pi^* \left(L_{\mathcal{Y}} \left(\left\lfloor \frac{1}{m} \pi_* \left(D + j \,\mathrm{div}(\mu) \right) \right\rfloor \right) \otimes \overline{\mathbb{F}} \right). \tag{8}$$

▶ Remark 12. If the function μ is defined over the base field $\mathbb{F}$ then the decomposition (8) is valid when considering $\mathbb{F}$-vector spaces:

$$L_{\mathcal{X}}(D)_j \simeq \mu^j \pi^* \left(L_{\mathcal{Y}} \left(\left\lfloor \frac{1}{m} \pi_* \left(D + j \,\mathrm{div}(\mu) \right) \right\rfloor \right) \right).$$

In practical instantiations, we are always able to choose μ defined over $\mathbb{F}$, even when ζ does not belong to $\mathbb{F}$. For instance, in the case of Kummer curves (see Section 5), the decomposition provided by Theorem 15 is always valid. However, as the evaluation set $\mathcal{P}$ needs to be formed of orbits of size $|\mathcal{G}|$, instantiations require this primitive root to belong to $\mathbb{F}$.

Let us highlight that Theorem 11 does not apply on the Hermitian tower, as the degree of the Artin–Schrieir extensions is not coprime with the characteristic.

To handle the divisors that appear in the decomposition above, we need to get a better grasp on the zeroes and the poles of the function μ. If the group action fixes the field of constants, the ramification points of π are zeroes or poles of the function μ, as stated in the following lemma.

▶ **Lemma 13.** *Assume that $\Gamma = \langle \gamma \rangle$ is a cyclic group of order m which fixes the m^{th} primitive root ζ. Let P be a point of X whose stabilizer Γ_P is non-trivial. Then $P \in \mathrm{Supp}(\mu)$.*

Proof. By hypothesis, there exists $j \in \{1, \ldots, m-1\}$ such that $\gamma^j \in \Gamma_P$. Then

$$\begin{aligned} (\gamma^j \cdot \mu)(P) &= \zeta^j \mu(P) && \text{by definition of } \mu \text{ in Th. 11,} \\ &= \mu(P) && \text{because } \gamma^j \in \Gamma_P. \end{aligned}$$

Since $\zeta^j \neq 1$, the point P is either a pole or a zero of μ. ◀

5 Foldable AG codes on Kummer curves

5.1 Preliminaries

Let us consider a Kummer curve over a finite field $\mathbb{F}$ defined by an equation of the form

$$\mathcal{X} : y^N = f(x) = \prod_{\ell=1}^{m}(x - \alpha_\ell) \tag{9}$$

where f is a degree-m separable polynomial of $\mathbb{F}[X]$, $\gcd(N, m) = 1$ and $\alpha_\ell \in \mathbb{F}$ for all $\ell \in \{1, \dots, m\}$. Let us denote by P_ℓ the affine point $(\alpha_\ell, 0)$ and P_∞ the unique point of $\mathcal{X}$ lying on the line at infinity.

Sequence of curves. Assume that $\gcd(N, |\mathbb{F}|) = 1$. The group $\mathbb{Z}/N\mathbb{Z}$ acts on $\mathcal{X}$ via the morphism $(x, y) \mapsto (x, \zeta y)$ where ζ is a primitive N^{th} root of unity. We assume that ζ belongs to $\mathbb{F}$.

The cyclic group $\mathbb{Z}/N\mathbb{Z}$ is solvable: writing the prime decomposition of $N = \prod_{i=0}^{s-1} p_i$ gives the following sequence of subgroups

$$\mathbb{Z}/N\mathbb{Z} \rhd \mathbb{Z}/N_1\mathbb{Z} \rhd \mathbb{Z}/N_2\mathbb{Z} \rhd \cdots \rhd \mathbb{Z}/N_{s-1}\mathbb{Z} \rhd 1, \tag{10}$$

where

$$N_i \coloneqq \prod_{j=i}^{s-1} p_j. \tag{11}$$

The i-th factor group Γ_i is isomorphic to the cyclic group of prime order $\mathbb{Z}/p_i\mathbb{Z}$. It is spanned by $\gamma_i : (x, y) \mapsto (x, \zeta_i y)$ where ζ_i is a primitive p_i^{th} root of unity.

Set $\mathcal{X}_0 \coloneqq \mathcal{X}$. By Section 4.1, the composition series (10) gives a sequence of curves $(\mathcal{X}_i)$ in which the i^{th} curve is defined by

$$\mathcal{X}_i : y^{N_i} = f(x) \tag{12}$$

and has genus

$$g_i = \frac{(N_i - 1)(m - 1)}{2}.$$

The last curve $\mathcal{X}_s$ has genus 0 and is isomorphic to the projective line $\mathbb{P}^1$. These successive quotients provide a sequence of projections $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$ defined by $\pi_i(x, y) = (x, y^{p_i})$:

$$\overset{\gamma_0}{\circlearrowright}\ \mathcal{X}_0 \xrightarrow{\pi_0} \dots \xrightarrow{\pi_i} \overset{\gamma_i}{\circlearrowright}\ \mathcal{X}_i \xrightarrow{\pi_{i+1}} \overset{\gamma_{i+1}}{\circlearrowright}\ \mathcal{X}_{i+1} \longrightarrow \dots \xrightarrow{\pi_{r-1}} \mathcal{X}_r \simeq \mathbb{P}^1.$$

▶ **Example 14.** The Hermitian curve defined over $\mathbb{F}_{q^2}$ by

$$\mathcal{X}_0 : y^{q+1} = x^q + x. \tag{13}$$

is a well-studied particular case of Kummer type curve. In this case, every curve in a $(\mathcal{X}, \mathcal{G})$-sequence is maximal over $\mathbb{F}_{q^2}$ [32, Proposition 6], i.e. $\left|\mathcal{X}_i(\mathbb{F}_{q^2})\right| = q^2 + 1 + 2g_i q$.

Stabilized points. Let us denote P_∞^i the unique point at infinity on the curve $\mathcal{X}_i$. One can easily check that

$$P_\infty^i := \begin{cases} (1:0:0) & \text{if } N > m \\ (0:1:0) & \text{otherwise.} \end{cases}$$

The points of $\mathcal{X}_0$ whose stabilizer under $\mathbb{Z}/N\mathbb{Z}$ is non-trivial are in fact fixed by $\mathbb{Z}/N\mathbb{Z}$ and consist precisely in $P_1, \dots, P_m$ and P_∞^i.

5.2 Decomposition of Riemann-Roch spaces

The theory of Kummer extensions provides us a decomposition like (4) at each level, with $\mu_i = y$ for every $i \in \{0, \dots, s-1\}$.

▶ **Theorem 15** ([35, Theorem 2.2]). *Let $D \in \mathrm{Div}(\mathcal{X}_i)$ that is Γ_i-invariant. Then*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{p_i-1} y^j L_{\mathcal{X}_{i+1}}\left(\left\lfloor \frac{1}{p_i}(\pi_i)_*(D_i + j\,\mathrm{div}_{F_i}(y))\right\rfloor\right).$$

Note that the function y maps distinct points in the same Γ_i-orbits onto different values, and thus satisfies the condition 3c of Definition 8.

An example of a sequence of y-compatible divisors. In order to exhibit a sequence of divisors (D_i) such that D_{i+1} is (D_i, y)-compatible for every $i \geq 0$, we need to handle the divisor associated to y and some other elementary functions on each curve $\mathcal{X}_i$, described for instance in [36].

▶ **Lemma 16** ([36]). *On $\mathcal{X}_i$ for every $i \in \{0, \dots, s-1\}$, we have*

1. $\mathrm{div}_{F_i}(x - \alpha_\ell) = N_i(P_\ell - P_\infty^i)$,
2. $\mathrm{div}_{F_i}(y) = P_1 + \cdots + P_m - mP_\infty^i$.

We now give sufficient conditions on the curve $\mathcal{X}_0$ and the first divisor D_0 to get a sequence of compatible divisors.

▶ **Lemma 17.** *Set $D_0 = \sum_{\ell=1}^{m} a_{0,\ell}P_\ell + b_0 P_\infty^0 \in \mathrm{Div}(\mathcal{X}_0)$.*

Assume that $m \equiv -1 \mod N$ and that the integers $a_{0,1}, \dots, a_{0,m}, b_0$ are all divisible by N. For every $i \in \{0, \dots, s-1\}$, set $D_{i+1} = \frac{D_i}{p_i}$. Then, the divisor D_{i+1} is (D_i, y)-compatible.

Proof. For $i \in \{1, \dots, s\}$, let us set $a_{i,\ell} = \frac{a_{i-1,\ell}}{p_{i-1}}$ and $b_i = \frac{b_{i-1}}{p_{i-1}}$ such that $D_i = \sum_{\ell=1}^m a_{i,\ell}P_\ell + b_i P_\infty^i$.

Fix $i \in \{0, \dots, s-1\}$. The divisor D_i is supported only by Γ_i-fixed points.

For any $j \in \{0, \dots, p_i - 1\}$, we have

$$E_{i,j} = \left\lfloor \frac{1}{p_i}\pi_{i*}(D_i + j\,\mathrm{div}_{F_i}(y))\right\rfloor = \sum_{\ell=1}^{m}\left\lfloor \frac{a_{i,\ell}+j}{p_i}\right\rfloor P_\ell + \left\lfloor \frac{b_i - jm}{p_i}\right\rfloor P_\infty^{i+1}.$$

Since N_i divides N, we have $m \equiv -1 \mod N_i$. Write $m = \kappa_i N_i - 1$ with $\kappa_i \geq 1$. The hypothesis on the integers $a_{0,1}, \dots, a_{0,m}, b_0$ entails

$$\left\lfloor \frac{a_{i,\ell}+j}{p_i}\right\rfloor = a_{i+1,\ell} + \left\lfloor \frac{j}{p_i}\right\rfloor = a_{i+1,\ell},$$
$$\left\lfloor \frac{b_i - jm}{p_i}\right\rfloor = b_{i+1} - \frac{j\kappa_i N_i}{p_i} + \left\lfloor \frac{j}{p_i}\right\rfloor = b_{i+1} - j\kappa_i N_{i+1}.$$

Then $E_{i,j} = D_{i+1} - j\kappa_i N_{i+1} P_\infty^{i+1}$. In particular, $D_{i+1} = E_{i,0}$ and $E_{i,j} \leq D_{i+1}$. Any $\nu_{i+1,j} \coloneqq (x-\alpha)^{\kappa_i j}$ with $\alpha \in \{\alpha_1, \ldots, \alpha_m\}$ gives the last condition on D_{i+1} for it to be (D_i, y)-compatible by Definition 7, i.e. $D_{i+1} - E_{i,j} = \mathrm{div}_\infty(\nu_{i+1,j})$. ◀

5.3 Family of foldable codes

We have gathered all the components to exhibit a foldable code on a family of Kummer curves.

▶ **Proposition 18.** *Let $\mathcal{X}_0$ be a Kummer curve defined by (9) with $m \equiv -1 \mod N$. Consider an evaluation set $\mathcal{P}_0 \subseteq \mathcal{X}_0(\mathbb{F}) \setminus \{P_1, \ldots, P_m, P_\infty^0\}$ formed by $\mathbb{Z}/N\mathbb{Z}$-orbits. Take $D_0 \in \mathrm{Div}(\mathcal{X}_0)$ satisfying hypothesis of Lemma 17. If $N > n^e$ for some $e \in (0,1)$, then the AG code $C = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ is foldable.*

The length of foldable codes over a Kummer curve as defined in (9) over $\mathbb{F}_q$ is bounded from above by $q + 1 + (N-1)(\kappa N - 2)\sqrt{q} - \kappa N$, using Hasse-Weil bound, write $m = \kappa N - 1$.

▶ Remark 19.

1. The primitive N^{th} root ζ needs to belong to the base field $\mathbb{F}$ to ensure that the set $\mathcal{P}_0$ is not empty.
2. The condition on the coefficients of D_0 can be loosened while the previous statement still holds. If $a_{0,1}, \ldots, a_{0,m}, b_0$ are divisible by $\prod_{i=0}^{s-2} p_i$ and not necessarily by p_{s-1}, we choose $a_{s,\ell} = \left\lceil \frac{a_{s-1,\ell}}{p_{s-1}} \right\rceil$ and $b_s = \left\lfloor \frac{b_{s-1}}{p_{s-1}} \right\rfloor$ for the coefficients of D_s. The last curve $\mathcal{X}_s$ being isomorphic to $\mathbb{P}^1$, the existence of balancing functions is trivial, if the first requirement of Definition 7 holds.

What happens outside these hypotheses? Lemma 17 provides sufficient conditions to make the code C_{i+1} as small as possible compared to C_i by choosing D_{i+1} among the divisors $E_{i,j}$, as required for a sequence of foldable codes by Definition 8. Let us have a look at what could happen when dropping these conditions.

▶ **Example 20.** Over $\mathbb{F}_8$, consider $y^N = x^m + x$ where $N = 9$ and $m = 5$. Then $m \not\equiv -1 \mod N$ and $N = p_0 p_1$ with $p_0 = p_1 = 3$. For $D_0 = 18P_\infty^0$, we have

$$E_{0,0} = \left\lfloor \frac{18}{3} \right\rfloor P_\infty^1 = 6P_\infty^1,$$
$$E_{0,1} = \left\lfloor \frac{18-5}{3} \right\rfloor P_\infty^1 = 4P_\infty^1,$$
$$E_{0,2} = \left\lfloor \frac{18-2\times 5}{3} \right\rfloor P_\infty^1 = 2P_\infty^1.$$

Choosing $D_1 = E_{0,0}$ would satisfy the first and the second conditions of Definition 7 to be (D_0, y)-compatible but not the third one. One can reasonably ask the support of D_1 to consist only of $\pi_0(P_\infty^0) = P_\infty^1$, as one-point codes are generally better understood. The Weierstrass gap theory on Kummer curves (e.g. [36, Theorem 3.2]) entails that if a function on $\mathcal{X}_1 : y^3 = x^5 + x$ has a pole locus of the form αP_∞^1, then $\alpha \in 3\mathbb{Z}_+ + 5\mathbb{Z}_+$. Therefore the smallest divisor of the form $D_1 = d_1 P_\infty^1$ that is (D_0, y)-compatible is $D_1 = 12P_\infty^1$. With such a choice of divisors, the code C_0 of dimension 15 is folded into the code C_1 of dimension 12 whereas the length of C_1 is the third of the length of C_0.

5.3.1 Explicit basis of the Riemann-Roch spaces

AG codes from the Kummer curve $\mathcal{X}$ associated to divisors as defined in Lemma 17 have been studied by Hu and Yang [27]. They provide a basis of the Riemann-Roch spaces in a combinatorial form.

▶ **Theorem 21** ([27, Theorem 5]). *Let $j, j_2, \dots, j_m$ be integers. We define*

$$E_{j,\, j_2,\dots,j_m} \coloneqq y^j \prod_{\ell=2}^{m} (x-\alpha_\ell)^{j_\ell}.$$

Consider $D = \sum_{\ell=1}^{m} a_\ell P_\ell + bP_\infty$. Set

$$\Omega_{a_1,\dots,a_m,b} \coloneqq \Bigg\{ (j,\, j_2,\dots,j_m) \mid j + a_1 \geq 0,\; j_\ell = \left\lceil \frac{-j-a_\ell}{N} \right\rceil \text{ for } \ell = 2,\dots,m \\ \text{and } mj + N(j_2+\cdots+j_m) \leq b \Bigg\}.$$

Then the elements $E_{j,\, j_2,\dots,j_m}$ for $(j,\, j_2,\dots,j_m) \in \Omega_{a_1,\dots,a_m,b}$ form a basis of $L_{\mathcal{X}}(D)$.

5.3.2 Parameters

To estimate the parameters of the code by using the Riemann-Roch theorem, we shall rely on the following result.

▶ **Lemma 22.** *Assume that $2(g_0-1) < \deg(D_0)$ (resp. $\deg(D_0) < n_0$). Then for every $i \in \{0,\dots,s\}$, $2(g_i-1) < \deg(D_i)$ (resp. $\deg(D_i) < n_i$).*

Proof. It is enough to notice that for every $i \in \{0,\dots,s-1\}$,

$$\deg D_{i+1} = \frac{\deg D_i}{p_i}, \qquad n_{i+1} = \frac{n_i}{p_i}, \qquad \text{and} \quad g_{i+1} \leq \frac{g_i}{p_i}.$$ ◀

In other words, if the degree of the first divisor is such that we can estimate the parameters of C_0 thanks to Riemann-Roch Theorem, then we handle the parameters of all the sequence of codes.

▶ **Proposition 23.** *If $\deg(D_0) < n_0$, then for every $i \in \{0,\dots,s\}$, the code C_i has length n_i and minimum relative distance $\Delta(C_i) = 1 - \frac{\deg D_0}{n_0}$. In particular, the RS code C_s has length $\frac{n_0}{N}$, dimension $\frac{\deg D_0}{N} + 1$ and relative minimum distance $1 - \frac{\deg D_0}{n_0}$.*

Moreover, if $2(g_0-1) < \deg(D_0)$, for every $i \in \{0,\dots,s\}$, the code C_i has dimension $\deg D_i - g_i + 1$.

Proof. The length of C_i is n_i by construction and its dimension is given by the Riemann-Roch theorem. So let us prove the statement concerning the relative minimum distance.

First notice that $n_i = p_i n_{i+1}$ and $\deg(D_i) = p_i \deg(D_{i+1})$ so $1 - \frac{\deg D_i}{n_i} = 1 - \frac{\deg D_0}{n_0}$. For $i = s$, the code C_s is a Reed-Solomon code of degree $0 \leq \deg(D_s) < n_s$ by Lemma 22 and has the expected relative minimum distance.

Now assume that $\Delta(C_{i+1})$ equals $1 - \frac{\deg D_0}{n_0}$ and let us prove that so does $\Delta(C_i)$. On the one hand, the divisor D_{i+1} corresponds to $E_{i,0}$ then for every $f \in C_{i+1}$, $f \circ \pi_i \in C_i$. In addition, the weight of $f \circ \pi_i$ in C_i is p_i times the weight of f in C_{i+1}. Since $n_i = p_i n_{i+1}$, we have $\Delta(C_i) \leq \Delta(C_{i+1})$. On the other hand, as $\deg(C_i) < n_i$, we have $\Delta(C_i) \geq 1 - \frac{\deg D_i}{n_i}$, which concludes the proof. ◀

6 Foldable AG codes along the Hermitian tower

6.1 Preliminaries

Sequence of curves. We consider the sequence of function fields $\mathcal{F} = (F_i)_{i\geq 0}$ over $\mathbb{F}_{q^2}$ that is defined recursively by $F_0 = \mathbb{F}_{q^2}(x_0)$ and $F_i = F_{i-1}(x_i)$ with equations

$$x_i^q + x_i = x_{i-1}^{q+1} \text{ for } i \geq 1. \tag{14}$$

Note that this tower of function field $\mathcal{F}$ corresponds to a tower of curves $(\mathcal{X}_i)_{i\geq 0}$ such that $F_i = \mathbb{F}_{q^2}(\mathcal{X}_i)$. One can view the curve $\mathcal{X}_i$ embedded in an i-dimensional affine space with variables $(x_1, \dots, x_i)$ defined by the equations (14).

For $i = 1$, the field F_1 is the function field of the Hermitian curve $\mathcal{H} := \mathcal{X}_1$ over $\mathbb{F}_{q^2}$.

Let $g_i := g(\mathcal{X}_i)$ denote the genus of the curve $\mathcal{X}_i$. An explicit formula was given by Pellikaan, Shen and Wee [41, Proposition 4]: we have $g_0 = 0$ and for $i \geq 1$,

$$g_i = \frac{1}{2}\left[(q^2-1)\left((q+1)^i - q^i\right) + 1 - q^i\right] = \frac{1}{2}\cdot\left(\sum_{k=1}^{i} q^{i+1}\cdot\left(1+\frac{1}{q}\right)^{k-1} + 1 - (1+q)^i\right). \tag{15}$$

For every $i \geq 0$, the number of $\mathbb{F}_{q^2}$–rational places in F_i is given by

$$\left|\mathcal{X}_i(\mathbb{F}_{q^2})\right| = q^{i+2} + 1.$$

We have an infinite sequence of curves $(\mathcal{X}_i)_{i\geq 0}$ as follows.

$$\dots \xrightarrow{\pi_{i+1}} \mathcal{X}_i \xrightarrow{\pi_i} \mathcal{X}_{i-1} \xrightarrow{\pi_{i-1}} \dots \xrightarrow{\pi_1} \mathcal{X}_0 \simeq \mathbb{P}^1.$$

(Γ_i acts on $\mathcal{X}_i$, Γ_{i-1} acts on $\mathcal{X}_{i-1}$)

▶ Remark 24. In the context of recursive towers, it is classical to **index the curves the other way** round compared to the notations used in Section 4. In the context of the Hermitian tower, the curve $\mathcal{X}_{i-1}$ is the quotient curve of the curve $\mathcal{X}_i$ under the action of the group Γ_i. To design foldable codes, we will fix a level of the tower, say $i_{\max}$, and consider codes over the curve $\mathcal{X}_{i_{\max}}$, which we will fold using the sequence of curves with *decreasing* indices $(\mathcal{X}_i)_{i_{\max}\geq i\geq 0}$.

This tower is a tower of Artin-Schreier extensions, which have been extensively studied (see for example [45]). Let us recall some classical results that will be useful to design foldable AG codes along this tower.

Automorphisms and projection maps. By definition of the Hermitian tower [45, Proposition 3.7.10], the Galois group of the extension F_i/F_{i-1} is the group of automorphisms defined by $(x_1, \dots, x_{i-1}, x_i) \mapsto (x_1, \dots, x_{i-1}, x_i + \alpha)$ where α runs in

$$\mathcal{S} = \left\{\alpha \in \mathbb{F}_{q^2} \mid \alpha^q + \alpha = 0\right\}.$$

Note that if we fixed a non–zero element $\alpha \in \mathcal{S}$, then for every $\beta \in \mathbb{F}_q$, $\alpha\beta$ lies in $\mathcal{S}$. So $\mathcal{S}$ is an additive group which is isomorphic to $\mathbb{F}_q$.

The quotient map $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i-1}$ consists in the projection onto the first i coordinates. For every $i \geq 0$, we set Π_i to be the composition of the first i quotient maps, *i.e.*

$$\Pi_i \coloneqq \pi_i \circ \pi_{i-1} \circ \cdots \circ \pi_0.$$

Behaviour of the point of infinity. In what follows, let us denote by $P_\infty^{(0)}$ the unique pole of the function x_0 in F_0, which corresponds to the point at infinity on the projective line $\mathcal{X}_0 = \mathbb{P}^1$.

▶ **Lemma 25** ([45, Proposition 3.7.8])**.** *Let $i \geq 1$. The place $P_\infty^{(0)}$ is totally ramified in F_i, which means that the preimage $\Pi_i^{-1}\left(\left\{P_\infty^{(0)}\right\}\right)$ consists in a unique place, denoted by $P_\infty^{(i)} \in \mathcal{X}_i$. Moreover, $P_\infty^{(0)}$ is the unique place that is ramified in the tower $\mathcal{F}$.*

The peculiar behaviour of the points $P_\infty^{(i)}$ in the tower encourages us to define a sequence of codes associated with divisors $D_i \in \mathrm{Div}(\mathcal{X}_i)$ of the form

$$D_i \coloneqq d_i P_\infty^{(i)} \text{ for } i \geq 1.$$

Let us focus on the principal divisors $\mathrm{div}_{F_i}(x_j)$ $(0 \leq j \leq i)$ and their valuation at the point $P_\infty^{(i)}$. Their properties follow from the study of the basic function field $F = \mathbb{F}_{q^2}(x, y)$ which is nothing but the Hermitian function field. It is a special case of Artin-Schreier extension of $\mathbb{F}_{q^2}(x)$ and is well known that we have

$$\mathrm{div}_F(y) = P^{(0)} - P_\infty^0.$$

▶ Remark 26. The role of the variables x and y is reversed compared to the Kummer model of the Hermitian curve, studied in the previous section.

Since each extension F_i/F_{i-1} corresponds to the same Artin-Schreier extension, and that $P_\infty^{(0)}$ is fully ramified in F_i/F_0, we can deduce the form of the divisor $\mathrm{div}_{F_i}(x_i)$, given in the next lemma. The valuation of the function $x_j \in F_{i-1}$ at $P_\infty^{(j)}$ follows from the extension degrees $[F_{i-1} : F_j] = q^{i-1-j}$ for $j < i$.

▶ **Lemma 27.** *The following two assertions hold.*

1. *For $i \geq 1$, denote by $P^{(i)}$ the unique common zero of the functions $x_0, ..., x_i$. Then we have*

$$\mathrm{div}_{F_i}(x_i) = (q+1)^i \left(P^{(i)} - P_\infty^{(i)}\right);$$

2. *Let $i \geq 1$. Then for $0 \leq j < i$, the valuation of the function $x_j \in F_{i-1}$ is given by*

$$v_{P_\infty^{(i-1)}}(x_j) = -q^{i-1-j}(q+1)^j.$$

Basis of the Riemann-Roch spaces associated to the divisor $d_i P_\infty^{(i)}$. For a given $i \geq 0$, the $P_\infty^{(i)}$ is the unique pole of the functions $x_0, ..., x_i$, which gives an explicit basis of the Riemann-Roch space associated to a multiple of $P_\infty^{(i)}$.

▶ **Lemma 28.** *For all $i \leq 1$ and $m \leq 1$, the Riemann-Roch space $L_{\mathcal{X}_i}(mP_\infty^{(i)})$ is formed by linear combinations of functions in the following set:*

$$\left\{ x_0^{a_0} \cdots x_i^{a_i} \mid 0 \leq a_0 \ , \ 0 \leq a_j \leq q-1 \text{ and } \sum_{j=0}^{i} a_j q^{i-j}(q+1)^j \leq m \right\}.$$

6.2 Construction of foldable AG codes

Fix a level $i_{\max}$ in the Hermitian tower. We want to construct foldable codes on the curve $\mathcal{X}_{i_{\max}}$ of the form

$$C(\mathcal{X}_{i_{\max}}, \mathcal{P}_{i_{\max}}, D_{i_{\max}}) \text{ where } \mathcal{P}_{i_{\max}} \subseteq \mathcal{X}_{i_{\max}}(\mathbb{F}_{q^2}) \setminus \{P_\infty^{(i_{\max})}\} \text{ and } D_{i_{\max}} = d_{i_{\max}} P_\infty^{(i_{\max})}.$$

We thus define a sequence of codes (C_i) as follow:

$$C_i \coloneqq C(\mathcal{X}_{i_{\max}-i}, \mathcal{P}_{i_{\max}-i}, D_{i_{\max}-i}) \text{ where } \mathcal{P}_{i-1} = \pi_i(\mathcal{P}_i) \text{ and } D_i = d_i P_\infty^{(i)}$$

In order to make sure $C_0 = C(\mathcal{X}_{i_{\max}}, \mathcal{P}_{i_{\max}}, D_{i_{\max}})$ is foldable, we need to describe the Riemann-Roch spaces on a certain step from Riemann-Roch spaces on *lower* curves. A priori Kani's theorem does not apply, so we will have to find a decomposition by hand. We deduce such a decomposition from the explicit basis of the Riemann-Roch space given in Lemma 28.

▶ **Proposition 29.** *Let $i \geq 0$. Set $D_i = d_i P_\infty^{(i)}$ for some integer d_i. Then*

$$L_{\mathcal{X}_i}(D_i) = \bigoplus_{j=0}^{q-1} x_i^j \pi_i^* \left(L_{\mathcal{X}_{i-1}}(E_{i,j}) \right)$$

with

$$E_{i,j} \coloneqq \left\lfloor \frac{1}{q} \pi_{i*} \left(D_i - j \cdot \mathrm{div}_{F_i}(x_i) \right) \right\rfloor \text{ for } 0 \leq j \leq q-1.$$

In other words, the function $x_i \in F_i$ partitions the divisor D_i in the sense of Definition 6.

Proof. By Lemma 28, the space $L_{\mathcal{X}_i}(D_i)$ is formed by linear combinations of $x_0^{a_0} \cdots x_i^{a_i}$ with non-negative exponents such that $0 \leq a_j \leq q-1$ for $j \neq 1$ and

$$\sum_{j=0}^{i} a_j q^{i-j} (q+1)^j \leq m.$$

As a_j runs in $\{0, \ldots, q-1\}$, the proof is concluded by noticing that the function $x_0^{a_0} \cdots x_{i-1}^{a_{i-1}} \in F_i$ lies in $L(D_i - j \cdot \mathrm{div}_{F_i}(x_i))$ which means that $x_0^{a_0} \cdots x_{i-1}^{a_{i-1}} \in F_{i-1}$ belongs to $L_{\mathcal{X}_{i-1}}(E_{i,j})$. ◀

To make D_{i-1} compatible with (D_i, x_i) (Definition 7), we need the existence of q balancing functions $\nu_{i-1,j} \in F_{i-1}$ (for every $0 \leq j \leq q-1$) such that

$$D_{i-1} - E_{i,j} = (\nu_{i-1,j})_\infty. \tag{16}$$

In our setup, we have

$$E_{i,j} = \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor P_\infty^{(i-1)}.$$

Thus, we need to "balance" the divisors

$$D_{i-1} - E_{i,j} = \left(d_{i-1} - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor \right) P_\infty^{(i-1)}.$$

We are led to study the Weierstrass semigroup of $P_\infty^{(i-1)}$, denoted by $\mathcal{H}\left(P_\infty^{(i-1)}\right)$. The generators of this semigroup can be found using Lemma 27. In fact, $P_\infty^{(i-1)}$ is the unique common pole of the functions $x_0, \ldots, x_{i-1} \in F_{i-1}$ and we know their exact valuation. Thus we have

$$\mathcal{H}\left(P_\infty^{(i-1)}\right) = \left\langle q^{i-1-k}(q+1)^k \ , \ 0 \leq k \leq i-1 \right\rangle_{\mathbb{N}}.$$

▶ Remark 30. In the spirit of the FRI protocol, one could be tempted to choose D_{i-1} as $E_{i,0}$. Such a choice would be valid in the sense of Definition 7 if and only for every $0 \leq j \leq q-1$

$$\left\lfloor \frac{d_i}{q} \right\rfloor - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor \in \mathcal{H}\left(P_\infty^{(i-1)}\right).$$

Unfortunately, when i increases, this condition is never satisfied.

To ensure that $\deg(D_{i-1} - E_{i,j})$ is never a Weierstrass gap for $P_\infty^{(i-1)}$, we increase the degree d_{i-1} of D_{i-1}.

▶ **Theorem 31.** *Let $i \geq 1$. Set $D_i = d_i P_\infty^{(i)}$ for some integer d_i and $D_{i-1} = d_{i-1} P_\infty^{(i-1)}$ where*

$$d_{i-1} \coloneqq \left\lfloor \frac{d_i}{q} \right\rfloor + 2g_{i-1}. \tag{17}$$

Then D_{i-1} is compatible with (D_i, x_i) (Definition 7).

Proof. By [45, Theorem 1.6.8], we know that

$$\max\left(\mathbb{N} \setminus \mathcal{H}\left(P_\infty^{(i-1)}\right)\right) \leq 2g_{i-1} - 1.$$

Then for every $0 \leq j \leq q-1$, the difference

$$m_{i,j} \coloneqq \deg(D_{i-1} - E_{i,j}) = \left(\left\lfloor \frac{d_i}{q} \right\rfloor - \left\lfloor \frac{d_i - j(q+1)^i}{q} \right\rfloor + 2g_{i-1}\right) \tag{18}$$

always belongs to the Weierstrass semigroup $\mathcal{H}\left(P_\infty^{(i-1)}\right)$. ◀

About the balancing functions. Since we know a $\mathbb{N}$-basis of the Weierstrass semigroup at $P_\infty^{(i-1)}$, we are able to explicit the form of the functions $\nu_{i-1,j}$. In particular, they can be chosen as the products of powers of the functions $x_0, ..., x_{i-1}$. More precisely, if $a_{i,j} \coloneqq (a_{i,j}(0), ..., a_{i,j}(i-1)) \in \mathbb{N}^i$ are integers such that

$$m_{i,j} = \sum_{k=0}^{i-1} a_{i,j}(k) \cdot q^{i-1-k}(q+1)^k, \tag{19}$$

then $m_{i,j} \in \mathcal{H}\left(P_\infty^{(i-1)}\right)$. The corresponding choice for the balancing function is then given by

$$\nu_{i-1,j} = \prod_{k=0}^{i-1} x_k^{a_{i,j}(k)}.$$

Note that finding a vector $a_{i,j} \in \mathbb{N}^i$ satisfying (19) leads to the study of the diophantine equation

$$m_{i,j} = \sum_{k=0}^{i-1} a_k \cdot q^{i-1-k}(q+1)^k$$

with i unknowns $a_k \in \mathbb{N}$, for which we know there exists at least a solution (and we only need one).

6.2.1 A family of foldable codes

We denote by $i_{\max}$ the level in the tower $(\mathcal{X}_i)_{i\geq 0}$, such that $\mathcal{X}_{i_{\max}}$ is the curve on which the code we want to test proximity is defined.

▶ **Proposition 32.** *Fix an integer $i_{\max}$. Set $\mathcal{P}_0 \subseteq \mathbb{P}^1(\mathbb{F}_{q^2}) \setminus \{P_\infty^0\}$ and define $\mathcal{P}_{i_{\max}}$ as the preimage of $\mathcal{P}_0$ under the morphism $\Pi_{i_{\max}}$. Fix an integer $d_{i_{\max}}$. Then the code $C(\mathcal{X}_{i_{\max}}, \mathcal{P}_{i_{\max}}, d_{i_{\max}} P_\infty^{(i)})$ is foldable.*

Proof. Beware that the sequence of curves is indexed decreasingly here, contrary to Section 4. In particular, the original code is defined over the curve $\mathcal{X}_{i_{\max}}$.

By definition of the Hermitian tower, there exists a solvable group $\mathcal{G}$ acting on $\mathcal{X}_{i_{\max}}$ that admits a normal series for which each factor group is isomorphic to the additive abelian group of $\mathbb{F}_q$. The action of $\mathcal{G}$ on $\mathcal{P}_{i_{\max}}$ is free, by definition of $\mathcal{P}_{i_{\max}}$. The cardinality of $\mathcal{P}_{i_{\max}}$ is equal to $|\mathcal{P}_0|\, q^{i_{\max}}$, hence $|\mathcal{G}| > |\mathcal{P}_{i_{\max}}|^e$ for some $e \in (0,1)$.

The third condition of Definition 8 follows from Theorem 31, noticing that for every $i \geq 0$, the function x_i maps different points in the same Γ_i-orbit onto different values. ◀

To control the dimension of foldable codes, we will focus on those of the form

$$C \coloneqq C\left(\mathcal{X}_{i_{\max}}, \mathcal{X}_{i_{\max}}(\mathbb{F}_{q^2}) \setminus \{P_\infty^{(i_{\max})}, (2\alpha+1)g_{i_{\max}})P_\infty^{(i_{\max})}\right) \tag{20}$$

for some $\alpha > 1/2$. In this case, we have $n_{i_{\max}} = q^{i_{\max}+2}$. We can determine a sufficient condition over $i_{\max}$ and α to get a constant rate.

▶ **Lemma 33.** *Let $R \in (0,1)$. Fix $\varepsilon \in (0,1)$. Set $i_{\max} \coloneqq q^\varepsilon$ and $\alpha \coloneqq Rq^{1-\varepsilon}$.*

The ratio of the dimension of the code C defined in (20) by its block length goes to R when q tends to infinity.

If $2(q^\varepsilon - 1) < q$, the relative minimum distance of C is bounded from below by $1 - R\left(1 + \frac{1}{q}\right)$.

Proof. If $\alpha > \frac{1}{2}$, the dimension of the code is equal to $(2\alpha+1)g_{i_{\max}} - g_{i_{\max}} + 1 = 2Rq^{1-\varepsilon} g_{i_{\max}} + 1$ by the Riemann-Roch Theorem. As R is fixed and q goes to infinity, we can assume that $\alpha > 1/2$ to compute the rate as

$$\lim_{q\to\infty} \frac{2Rq^{1-\varepsilon} g_{i_{\max}}}{q^{i_{\max}+2}}$$

for $i_{\max} = q^\varepsilon$. Lemma 54 in Appendix B clearly implies that this limit is equal to R.

Regarding the relative minimum distance, we use the Goppa bound: if

$$(2\alpha+1)g_{i_{\max}} < q^{i_{\max}+2},$$

then the relative minimum distance of C satisfies

$$\Delta(C) \geq 1 - \frac{(2\alpha+1)g_{i_{\max}}}{q^{i_{\max}+2}}.$$

By Proposition 53 in Appendix B, we have

$$\frac{g_{i_{\max}}}{q^{i_{\max}+2}} \leq \frac{i_{\max}}{2q}\left(1 + \frac{i_{\max}}{q}\right),$$

which gives the expected lower bound for our choice of α and $i_{\max}$. ◀

7 Folding operators for AG codes

Now that we have determined the needed properties of an AG-code to be foldable, we construct the fundamental building block of our IOPP by generalizing the so-called algebraic hash function of [18] to the AG codes setting, and we refer to it as the *folding operator*. Next, we provide a formal description of the IOPP system (P, V) and state the theorem capturing its efficiency properties.

7.1 Definition of folding operators

Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a code satisfying Definition 8. We consider its associated $(\mathcal{X}, \boldsymbol{\mathcal{G}})$-sequence of curves $(\mathcal{X}_i)$ and its sequence of divisors (D_i).

To test proximity of a function $f^{(0)} : \mathcal{P}_0 \to \mathbb{F}$ to C_0, we aim to inductively reduce the problem to a smaller one, consisting of testing proximity to the code $C_i = C(\mathcal{X}_i, \mathcal{P}_i, D_i)$. Broadly speaking, our goal is to define from any function $f^{(i)} : \mathcal{P}_i \to \mathbb{F}$ a function $f^{(i+1)} : \mathcal{P}_{i+1} \to \mathbb{F}$ such that the relative distance $\Delta(f^{(i+1)}, C_{i+1})$ is roughly equal to $\Delta(f^{(i)}, C_i)$.

Fix $i \in \{0, \dots, r-1\}$ and let $f : \mathcal{P}_i \to \mathbb{F}$ be an arbitrary function.

▶ **Notation 34** (Interpolation polynomial). *For each $P \in \mathcal{P}_{i+1}$, let us denote $S_P \coloneqq \pi_i^{-1}(\{P\})$ the set of p_i distinct preimages of P. Recall that the function μ_i satisfies Item 3c of Definition 8 and consider*

$$I_{f,P}(X) \coloneqq \sum_{j=0}^{p_i-1} X^j a_{j,P} \tag{21}$$

the univariate polynomial over $\mathbb{F}$ of degree less than p_i which interpolates the set of points

$$\left\{(\mu_i(\widehat{P}), f(\widehat{P})); \widehat{P} \in S_P\right\}.$$

Specifically, for all $\widehat{P} \in S_P$, we have

$$I_{f,P}(\mu_i(\widehat{P})) = f(\widehat{P}).$$

Then for every $j \in \{0, \dots, p_i - 1\}$, we define the function

$$f_j : \left\{ \begin{array}{ccc} \mathcal{P}_{i+1} & \to & \mathbb{F}, \\ P & \mapsto & a_{j,P}. \end{array} \right. \tag{22}$$

Given $f : \mathcal{P}_i \to \mathbb{F}$, the idea is to define p_i functions $f_j : \mathcal{P}_{i+1} \to \mathbb{F}$, where $|\mathcal{P}_{i+1}| = \frac{|\mathcal{P}_i|}{p_i}$ such that f corresponds to the evaluation of a function in $L(D_i)$ if and only if each f_j coincides with a function in $L(E_{i,j}) \subset L(D_{i+1})$. Instead of testing for each $j \in \{0, \dots, p_i - 1\}$ whether $f_j \in C_{i+1}$, we reduce those p_i claims to a single one, by taking a random linear combination of the f_j's, which we referred to as a *folding of f*. By linearity of the codes, such a combination of the f_j's belongs to C_{i+1} whenever $f \in C_i$ (see Proposition 37 below). However, for soundness analysis, one needs to ensure that no f_j corresponds to a function lying in $L(D_{i+1}) \setminus L(E_{i,j})$. Some safeguards are embedded into the folding operation by introducing the balancing functions $\nu_{i+1,j}$ from Definition 7 in the second term of the sum in (23).

▶ **Definition 35** (Folding operator). *For any $\boldsymbol{z} = (z_1, z_2) \in \mathbb{F}^2$, we define the* folding of f *to be the function* ***Fold***$[f, \boldsymbol{z}] : \mathcal{P}_{i+1} \to \mathbb{F}$ *such that*

$$\textbf{\textit{Fold}}[f, \boldsymbol{z}] \coloneqq \sum_{j=0}^{p_i-1} z_1^j f_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} f_j \tag{23}$$

where the functions f_j are defined in Equation (22) and the functions $\nu_{i+1,j}$ in Definition 7.

7.2 Properties of folding operators

Our aim is to prove that the folding operators satisfy three key properties: local computability, completeness, and distance preservation. This will enable us to invoke [4, Theorem 1] for the completeness and soundness of our AG-IOPP.

Given the p_i points $((\mu_i(\widehat{P}), f(\widehat{P})))_{\widehat{P}\in S_P}$, one can determine the coefficients $(a_{j,P})_{0\le j<p}$ of $I_{f,P}$ defined in (21) by polynomial interpolation. Recalling that for each $P \in \mathcal{P}_{i+1}$, we have $f_j(P) = a_{j,P}$, we get the following lemma. This lemma will allow to obtain fast prover time and verifier decision complexity.

▶ **Lemma 36** (Locality). *Let $\boldsymbol{z} \in \mathbb{F}^2$. For each $P \in \mathcal{P}_{i+1}$, the value of $\textbf{\textit{Fold}}[f, \boldsymbol{z}](P)$ can be computed with exactly p_i queries to f, namely at the points $\pi_i^{-1}(\{P\})$.*

▶ **Proposition 37** (Completeness). *Let $\boldsymbol{z} \in \mathbb{F}^2$. If $f \in C_i$, then $\textbf{\textit{Fold}}[f, \boldsymbol{z}] \in C_{i+1}$.*

Proof. Write $\boldsymbol{z} = (z_1, z_2)$. If $f \in C_i$, it coincides with a function of $L(D_i)$. By definition of the divisors $E_{i,j}$ and Theorem 11, there exist some functions $\widetilde{f}_j \in L(E_{i,j})$ such that

$$f = \sum_{j=0}^{p_i-1} \mu_i^j \widetilde{f}_j \circ \pi_i.$$

Let $P \in \mathcal{P}_{i+1}$. For any $\widehat{P} \in S_P$,

$$\textbf{Fold}\left[f, (\mu_i(\widehat{P}), 0)\right](P) = I_{f,P}(\mu_i(\widehat{P})) = f(\widehat{P}) = \sum_{j=0}^{p_i-1} \mu_i(\widehat{P})^j \widetilde{f}_j(P).$$

Moreover, for any $P \in \mathcal{P}_{i+1}$, polynomials $I_{f,P}(X)$ and $\textbf{Fold}[f, (X, 0)](P)$ in $\mathbb{F}[X]$ are of degree less than p_i and agree on $\left\{\mu_i(\widehat{P}); \widehat{P} \in S_P\right\}$ of size p_i, therefore they are equal. In particular,

$$\textbf{Fold}\left[f, (\mu_i(\widehat{P}), 0)\right](P) = \sum_{j=0}^{p_i-1} \mu_i(\widehat{P})^j f_j(P).$$

Thus, for all $P \in \mathcal{P}_{i+1}$,

$$\sum_{j=0}^{p_i-1} \mu_i(\widehat{P})^j (\widetilde{f}_j(P) - f_j(P)) = 0$$

and the polynomial

$$\sum_{j=0}^{p_i-1} X^j (\widetilde{f}_j(P) - f_j(P))$$

of degree less than p_i is zero on at least $\left|\left\{\mu_i(\widehat{P}); P \in \mathcal{P}_{i+1}\right\}\right| = p_i$ points. Hence, for every $j \in \{0, \dots, p_i - 1\}$, the function f_j defined in Equation (22) coincides with $\widetilde{f}_j$ and

$$\textbf{Fold}[f, \boldsymbol{z}] \coloneqq \sum_{j=0}^{p_i-1} z_1^j \widetilde{f}_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} \widetilde{f}_j$$

where $\widetilde{f}_j \in L(E_{i,j}) \subseteq L(D_{i+1})$ and $\nu_{i+1,j} f_j \in L(D_{i+1})$, by definition of the divisors $E_{i,j}$, D_{i+1} and the functions $\nu_{i+1,j}$ (see Definition 7). Thus each term of $\textbf{Fold}[f, \boldsymbol{z}]$ lies in the vector space C_{i+1}, which concludes the proof. ◀

We discuss the effect of the folding operation on a function which is far from the code. Roughly speaking, we want to show that, if f is δ-far from the code C_i, then the folding $\mathbf{Fold}[f, \boldsymbol{z}]$ of f is almost δ-far from the code C_{i+1} with high probability over $\boldsymbol{z} \in \mathbb{F}^2$. We start with the notion of weighted agreement.

▶ **Definition 38** (Weighted agreement)**.** *For any function $\eta \in [0,1]^{\mathcal{P}}$, we define the* η-agreement *of two functions $u, v \in \mathbb{F}^{\mathcal{P}}$ by*

$$\omega_\eta(u,v) \coloneqq \frac{1}{|\mathcal{P}|} \sum_{\substack{P \in \mathcal{P} \\ u(P)=v(P)}} \eta(P).$$

Given a subspace $V \subset \mathbb{F}^{\mathcal{P}}$ and $u \in \mathbb{F}^{\mathcal{P}}$, we set

$$\omega_\eta(u,V) \coloneqq \max_{v \in V} \omega_\eta(u,v).$$

Notice that since $\eta \in [0,1]^{\mathcal{P}}$, we have for any $V \subset \mathbb{F}^{\mathcal{P}}$ and any $u \in \mathbb{F}^{\mathcal{P}}$,

$$\omega_\eta(u,V) \leq 1 - \Delta(u,V). \tag{24}$$

Let us introduce some notations related to the Johnson list-decoding function. For any $\varepsilon \in (0,1]$, let $J_\varepsilon : [0,1] \to [0,1]$ be the function such that

$$J_\varepsilon(\lambda) = 1 - \sqrt{1 - (1-\varepsilon)\lambda},$$

and denote $J_\varepsilon^l = \underbrace{J_\varepsilon \circ \cdots \circ J_\varepsilon}_{l \text{ times}}$.

We now state a preliminary result concerning the weighted agreement on a low-degree parametrized curve. Proof of Proposition 39 builds upon the one of [18, Theorem 4.5] and is given in Appendix A.

▶ **Proposition 39.** *Let $\eta \in [0,1]^{\mathcal{P}}$ and $\varepsilon, \delta > 0$ such that and $\delta < J_\varepsilon^l(\lambda)$. Let $u_0, \ldots, u_{l-1} \in \mathbb{F}^{\mathcal{P}}$ such that*

$$\Pr_{z \in \mathbb{F}} \left[\omega_\eta \left(\sum_{i=0}^{l-1} z^i u_i, V \right) > 1 - \delta \right] \geq \frac{l-1}{|\mathbb{F}|} \left(\frac{2}{\varepsilon} \right)^{l+1}, \tag{25}$$

then there exists $T \subset \mathcal{P}$, and $v_0, \ldots, v_{l-1} \in V$ such that:

- $\sum_{P \in T} \eta(P) \geq (1 - \delta - \varepsilon)|\mathcal{P}|$
- *for each i, $u_{i|T} = v_{i|T}$.*

Here, for a function $u \in \mathbb{F}^{\mathcal{P}}$, $u_{|T} \in \mathbb{F}^T$ corresponds to the function obtained by restriction on $T \subset \mathcal{P}$.

As mentioned earlier, soundness analysis relies on the relation between the weighted agreement of f to C_i and the weighted agreement of the folding of f to C_{i+1}, constrained by the next corollary.

▶ **Corollary 40.** *Fix $i \in \{0, \ldots, r-1\}$. For a function $\eta : \mathcal{P}_i \to [0,1]$, define $\theta : \mathcal{P}_{i+1} \to [0,1]$ by*

$$\forall P \in \mathcal{P}_{i+1},\ \theta(P) \coloneqq \frac{1}{p_i} \sum_{\widehat{P} \in S_P} \eta(\widehat{P}).$$

Let λ_i be the minimal relative distance of C_i. Fix $\varepsilon \in (0,1)$ and

$$\delta < \min\left(J_\varepsilon^{p_i}(\lambda_i), \frac{1}{2}\left(\lambda_i + \frac{\varepsilon}{2}\right)\right).$$

For any function $f : \mathcal{P}_i \to \mathbb{F}$ such that $\omega_\eta(f, C_i) < 1 - \delta$, we have

$$\Pr_{\boldsymbol{z}\in\mathbb{F}^2}\left[\omega_\theta(\textbf{\textit{Fold}}\left[f, \boldsymbol{z}\right], C_{i+1}) > 1 - \delta + \varepsilon\right] \leq \frac{1}{|\mathbb{F}|}\left(p_i + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_i}.$$

Proving Corollary 40 requires the lemma stated next. We prove Corollary 40, then prove Lemma 41.

▶ **Lemma 41.** *Let $i \in \{0, \dots, r-1\}$, $D_i \in \mathrm{Div}(\mathcal{X}_i)$ and $\mu_i \in \mathbb{F}(\mathcal{X}_i)$ satisfying Definition (6). Consider a divisor $D_{i+1} \in \mathrm{Div}(\mathcal{X}_{i+1})$ that is (D_i, μ_i)-compatible in the sense of Definition 7.*

Fix $j \in \{0, \dots, p_i - 1\}$. Then a function $g \in \mathbb{F}(\mathcal{X}_{i+1})$ belongs to $L(E_{i,j})$ if and only if both functions g and $g\nu_{i+1,j}$ belong to $L(D_{i+1})$.

Proof of Corollary 40. Let $f : \mathcal{P}_i \to \mathbb{F}$ be an arbitrary function. According to Equation (22), there exist p_i function $f_j : \mathcal{P}_{i+1} \to \mathbb{F}$ such that for any $\boldsymbol{z} = (z_1, z_2) \in \mathbb{F}^2$,

$$\textbf{Fold}\left[f, \boldsymbol{z}\right] = \sum_{j=0}^{p_i-1} z_1^j f_j + \sum_{j=0}^{p_i-1} z_2^{j+1} \nu_{i+1,j} f_j.$$

Rewrite $\textbf{Fold}\left[f, \boldsymbol{z}\right]$ as a polynomial function in z_2:

$$\textbf{Fold}\left[f, \boldsymbol{z}\right] = f_{z_1} + z_2 f'_0 + z_2^2 f'_1 + \cdots + z_2^{p_i} f'_{p_i-1},$$

where

$$f_{z_1} \coloneqq \sum_{j=0}^{p_i-1} z_1^j f_j, \qquad f'_j \coloneqq \nu_{i+1,j} f_j.$$

Finally, set

$$K_0 \coloneqq \frac{p_i - 1}{|\mathbb{F}|}\left(\frac{4}{\varepsilon}\right)^{p_i} \qquad \text{and} \qquad K_1 \coloneqq \frac{p_i}{|\mathbb{F}|}\left(\frac{4}{\varepsilon}\right)^{p_i+1}.$$

Let us prove the corollary by contrapositive. We assume that

$$\Pr_{\boldsymbol{z}\in\mathbb{F}^2}\left[\omega_\theta(\textbf{Fold}\left[f, \boldsymbol{z}\right], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0 + K_1,$$

and thus

$$\Pr_{z_1\in\mathbb{F}}\left[\Pr_{z_2\in\mathbb{F}}\left[\omega_\theta(\textbf{Fold}\left[f, \boldsymbol{z}\right], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0\right] > K_1.$$

Fix $z_1 \in \mathbb{F}$ such that

$$\Pr_{z_2\in\mathbb{F}}\left[\omega_\theta(\textbf{Fold}\left[f, \boldsymbol{z}\right], C_{i+1}) > 1 - \delta + \varepsilon\right] > K_0.$$

By Proposition 39, there exist $v_{z_1}, v'_1, \dots, v'_{p_i-1} \in C_{i+1}$ and $\mathcal{T}' \subset \mathcal{P}$ such that

- $\sum_{P\in\mathcal{T}'} \theta(P) \geq (1 - \delta + \frac{\varepsilon}{2})\,|\mathcal{P}_{i+1}|$,
- $v_{z_1|\mathcal{T}'} = f_{z_1|\mathcal{T}'}$,

- for each $j \in \{1, \dots, p_i - 1\}$, $v'_{j\,|\mathcal{T}'} = f'_{j\,|\mathcal{T}'}$.

In particular,

$$\omega_\theta(f_{z_1}, C_{i+1}) \geq \omega_\theta(f_{z_1}, v_{z_1}) = \frac{1}{|\mathcal{P}_{i+1}|} \sum_{P \in \mathcal{T}'} \theta(P) \geq 1 - \delta + \frac{\varepsilon}{2}.$$

It means that

$$\Pr_{z_1 \in \mathbb{F}} \left[\omega_\theta(f_{z_1}, C_{i+1}) \geq 1 - \delta + \frac{\varepsilon}{2} \right] \geq \Pr_{z_1 \in \mathbb{F}} \left[\Pr_{z_2 \in \mathbb{F}} \left[\omega_\theta(\mathbf{Fold}\left[f, \boldsymbol{z}\right], C_{i+1}) > 1 - \delta + \varepsilon \right] > K_0 \right] > K_1.$$

The polynomial form of f_{z_1} in z_1 enables us to reapply Proposition 39: there exist $\mathcal{T} \subset \mathcal{P}$ and $v_0, v_1, \dots, v_{p_i - 1} \in C_{i+1}$ such that

- $\sum_{P \in \mathcal{T}} \theta(P) \geq (1 - \delta) \left|\mathcal{P}_{i+1}\right|$,
- for each $j \in \{0, \dots, p_i - 1\}$, $v_{j\,|\mathcal{T}} = f_{j\,|\mathcal{T}}$.

On $\mathcal{T}' \cap \mathcal{T}$, we thus have

$$v'_{j\,|\mathcal{T}' \cap \mathcal{T}} = f'_{j\,|\mathcal{T}' \cap \mathcal{T}} = (\nu_{i+1,j} f_j)_{|\mathcal{T}' \cap \mathcal{T}} = (\nu_{i+1,j} v_j)_{|\mathcal{T}' \cap \mathcal{T}}.$$

The cardinality of $\mathcal{T}' \cap \mathcal{T}$ satisfies

$$|\mathcal{T}' \cap \mathcal{T}| = |\mathcal{T}'| + |\mathcal{T}| - |\mathcal{T}' \cup \mathcal{T}| \geq \sum_{P \in \mathcal{T}'} \theta(P) + \sum_{P \in \mathcal{T}} \theta(P) - |\mathcal{P}_{i+1}| \geq (1 - 2\delta + \frac{\varepsilon}{2}) |\mathcal{P}_{i+1}|.$$

The assumption on δ ensures that $2\delta - \frac{\varepsilon}{2} < \lambda_{i+1}$ where λ_{i+1} is the minimal distance of C_{i+1}. Hence, for every $j \in \{0, \dots, p_i - 1\}$, the evaluations of v'_j and $\nu_{i+1,j} v_j$ on $\mathcal{P}_{i+1}$ are equals. They are codewords of C_{i+1}, thus this implies that both functions v_j and $\nu_{i+1,j} v_j$ belong to $L(D_{i+1})$. By Lemma 41, we get that the function v_j lies in $L(E_{i,j})$.

Now let us define $v : \mathcal{P}_i \to \mathbb{F}$ by

$$\forall Q \in \mathcal{P}_i,\ v(Q) \coloneqq \sum_{j=0}^{p_i - 1} \mu_i^j(Q) v_j \circ \pi_i(Q).$$

By definition of the divisors $E_{i,j}$ (5), the function v belong to $L(D_i)$. Now let us prove that it agrees with f on $S_{\mathcal{T}} \coloneqq \bigsqcup_{P \in \mathcal{T}} S_P$.

Let $P \in \mathcal{T}$ and $\widehat{P} \in S_P$.

$$\begin{aligned} f(\widehat{P}) = I_{f,P}(\mu_i(\widehat{P})) &= \sum_{j=0}^{p_i - 1} \mu_i(\widehat{P})^j f_j(P) && \text{by definition of } I_{f,P}, \\ &= \sum_{j=0}^{p_i - 1} \mu_i(\widehat{P})^j v_j \circ \pi_i(\widehat{P}) && \text{since } f_{j\,|\mathcal{T}} = v_{j\,|\mathcal{T}} \text{ and } P = \pi_i(\widehat{P}), \\ &= v(\widehat{P}). \end{aligned}$$

As a result, since $v \in C_i$, we can conclude that

$$\omega_\eta(f, C_i) \geq \omega_\eta(f, v) \geq \frac{1}{|\mathcal{P}_i|} \sum_{P \in \mathcal{T}} \sum_{\widehat{P} \in S_P} \eta(\widehat{P}) = \frac{1}{|\mathcal{P}_{i+1}|} \sum_{P \in \mathcal{T}} \theta(P) \geq 1 - \delta. \qquad \blacktriangleleft$$

Proof of Lemma 41. Assume that $g \in L(E_{i,j})$. Then Definition 7 ensures that g and $g\nu_{i+1,j}$ lie in $L(D_{i+1})$.

Conversely, assume that g and $g\nu_{i+1,j}$ belong to $L(D_{i+1})$ and write $D_{i+1} = \sum n_P P$. The hypotheses on g imply that

$$g \in L(D_{i+1}) \cap L(D_{i+1} - (\nu_{i+1,j})).$$

By [40, Lemma 2.6], the function g belongs to $L(D'_{i+1})$, where the divisor D'_{i+1} is defined by

$$D'_{i+1} \coloneqq \sum_P n'_P P \quad \text{where } n'_P \coloneqq \min(n_P, n_P + v_P(\nu_{i+1,j})).$$

Then $D'_{i+1} = D_{i+1} - \operatorname{div}_\infty(\nu_{i+1,j}) = E_{i,j}$ by the second item of Definition 7. ◀

7.3 IOPP for foldable AG codes

Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code over an alphabet $\mathbb{F}$. Given a family of folding operators defined as per Definition 35, [4] yields an IOPP for C_0, which is abstracted from the FRI protocol of [9]. We informally describe the IOPP system (P, V) for testing proximity of a function $f^{(0)} : \mathcal{P}_0 \to \mathbb{F}$ to C_0, then give its properties. A formal description will be provided in Section 8 for instantiations with concrete AG codes.

As in the FRI protocol, the IOPP is divided in two phases, referred to as COMMIT and QUERY. Before any interaction, P and V agree on:

- a $(\mathcal{X}, \boldsymbol{\mathcal{G}})$-sequence of curves $(\mathcal{X}_i)$, for which we denote the length of the composition serie of $\mathcal{G}$ by r.
- a sequence of codes (C_i) where for each $i \in \{0, \dots, r\}$, $C_i = (\mathcal{X}_i, \mathcal{P}_i, D_i)$ and $\mathcal{X}_i, \mathcal{P}_i$ and D_i are defined as per Section 4,
- a sequence of functions $(\mu_i) \in \mathbb{F}(\mathcal{X}_i)$ satisfying Definition 6,
- a sequence of balancing functions $(\nu_{i+1})_{0 \le i < r}$ of p_i-tuples of functions in $\mathbb{F}(\mathcal{X}_{i+1})$ such that $\nu_{i+1} = (\nu_{i+1,j})_{0<j<p_i}$ and $\nu_{i+1,j}$ satisfies (6).

We recall that the choice of a sequence $(\mathcal{X}_i)$ induces a sequence of projections $\pi_i : \mathcal{X}_i \to \mathcal{X}_{i+1}$.

- The COMMIT phase is an interaction over r rounds between P and V. For each round $i \in \{0, \dots, r-1\}$, the verifier samples a random challenge $\boldsymbol{z}^{(i)} \in \mathbb{F}^2$. As an answer, the prover gives oracle access to function $f^{(i+1)} : \mathcal{P}_{i+1} \to \mathbb{F}$, which is expected to be equal to $\mathbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right]$. To compute the values of $f^{(i+1)}$ on $\mathcal{P}_{i+1}$, an honest prover P exploits the fact that the folding of $f^{(i)}$ is locally computable (Lemma 36). Namely, for each $P \in \mathcal{P}_{i+1}$, P computes the coefficients $(a_{j,P})_{0 \le j < p}$ of $I_{f^{(i)},P} \in \mathbb{F}[X]$ from ${f^{(i)}}_{|S_P}$, evaluates $\nu_{i+1,j}$ at P, and sets

$$\mathbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right](P) \coloneqq \sum_{j=0}^{p_i-1} \left(z_1^{(i)}\right)^j a_{j,P} + \sum_{j=0}^{p_i-1} \left(z_2^{(i)}\right)^{j+1} \nu_{i+1,j}(P) a_{j,P}.$$

- During the QUERY phase, one of the two tasks of the verifier V is to check that each pair of successive oracle functions $(f^{(i)}, f^{(i+1)})$ is consistent. A standard idea is to check that the equality

$$f^{(i+1)} = \mathbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right] \tag{26}$$

holds at a random point in $\mathcal{P}_{i+1}$. By leveraging the local property of the folding operator, such a test requires only p_i queries to $f^{(i)}$ and 1 query to $f^{(i+1)}$. As in [9], we call this step of verification a *round consistency test*. The verifier begins by sampling at random

$Q_0 \in \mathcal{P}_0$ and once this is done, all the locations of the round consistency tests run inside the current **query test** are determined. More specifically, for each round i, V defines $Q_{i+1} \coloneqq \pi_i(Q_i)$ to be the random point where Equation (26) is checked. Through this process, the round consistency tests are correlated to improve soundness. Such a **query test** can be seen as a *global* consistency test, similar to the one of the FRI protocol. For the final test, V reads $f^{(r)} : \mathcal{P}_r \to \mathbb{F}$ in its entirety to test if $f^{(r)} \in C_r$.

▶ **Theorem 42.** *Let $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code of length $n \coloneqq |\mathcal{P}_0|$. By definition, C_0 admits a solvable group $\mathcal{G} \in \mathrm{Aut}(\mathcal{X}_0)$ such that $|\mathcal{G}| > n^e$ for a certain $e \in (0,1)$ and induces a sequence of codes (C_i). Set $p_{max} \coloneqq \max p_i$, $\lambda \coloneqq \min_i \Delta(C_i)$ and*

$$\gamma \coloneqq \min\left(J_\varepsilon^{p_{max}}(\lambda), \frac{1}{2}(\lambda + \frac{\varepsilon}{2})\right).$$

There is an IOPP system (P, V) for C_0 satisfying:

Perfect completeness: *If $f^{(0)} \in C_0$ and $f^{(1)}, \dots, f^{(r)}$ are honestly generated by the prover, the verifier outputs* accept *with probability 1.*

Soundness: *Assume $f^{(0)}$ is δ-far from C_0 and let $\varepsilon \in (0,1)$. With probability at least $1 - \mathit{err}_{commit}$ over the randomness of the verifier during the COMMIT phase, where*

$$\mathit{err}_{commit} \leq \frac{\log n}{|\mathbb{F}|}\left(p_{max} + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_{max}}$$

and for any oracles $f^{(1)}, \dots, f^{(r)}$ adaptively chosen by a possibly dishonest prover P^*, the probability that the verifier V outputs accept *after a single query test is at most*

$$\mathit{err}_{query}(\delta) \leq (1 - \min(\delta, \gamma) + \varepsilon \log n).$$

Overall, for any prover P^*, the soundness error $\mathit{err}(\delta)$ after t repetitions of the QUERY phase satisfies

$$\begin{aligned}\mathit{err}(\delta) &\leq \mathit{err}_{commit} + (\mathit{err}_{query}(\delta))^t \\ &< \frac{\log n}{|\mathbb{F}|}\left(p_{max} + \frac{4}{\varepsilon} - 1\right)\left(\frac{4}{\varepsilon}\right)^{p_{max}} + (1 - \min(\delta, \gamma) + \varepsilon \log n)^t.\end{aligned}$$

Moreover, the IOPP system is public-coin, has round complexity $\mathsf{r}(n) < \log n$, proof length $\mathsf{l}(n) < n$ and query complexity $\mathsf{q}(n) < t p_{max} \log n + n^{1-e}$.

Proof. Lemma 36, Proposition 37 and Corollary 40 satisfy the conditions of [4, Theorem 1]. Completeness and soundness are given by [4, Theorem 1]. Let us prove the rest of the theorem. Regarding round complexity, we have that

$$\prod_{i=0}^{r-1} p_i = \frac{n}{n_r},$$

where

$$n_r = |\mathcal{P}_r| = \frac{n}{|\mathcal{G}|} < n^{1-e}.$$

For every $i \in \{0, \dots, r-1\}$, $2 \leq p_i \leq p_{max}$. Therefore

$$\mathsf{r}(n) \leq \log_2 n - \log_2 n_r < \log_2 n.$$

For query complexity, notice that for $i \in \{0, \dots, r-2\}$, $f^{(i+1)}(Q_{i+1})$ is reused for the next round consistency test. Hence,

$$\mathsf{q}(n) = t\left(\sum_{i=0}^{r-1} p_i\right) + n^{1-e} \leq trp_{max} + n^{1-e}.$$

Finally, the total proof length $\mathsf{l}(n)$ is the sum of the lengths of all the oracles provided by P during the COMMIT phase, counted in field elements. Denoting $t_{i+1} \coloneqq \prod_{j=0}^{i} p_j$, we notice that

$$|\mathcal{P}_{i+1}| = \frac{|\mathcal{P}_i|}{p_i} = \frac{|\mathcal{P}_0|}{t_{i+1}}.$$

Thus, we have

$$\mathsf{l}(n) = \sum_{i=1}^{r} |\mathcal{P}_i| = \sum_{i=1}^{r} \frac{|\mathcal{P}_0|}{t_i} \leq n \sum_{i=1}^{r} \frac{1}{2^i} = n\left(1 - \frac{1}{2^r}\right) < n. \quad \blacktriangleleft$$

8 Proximity tests for AG codes on Kummer curves and Hermitian towers

When we instantiate the AG-IOPP proposed in Section 7.3 for the setting of Kummer curves (Section 5) and curves in the Hermitian tower (Section 6), we end up with a membership test to a RS code. An RS code is itself a foldable AG code (see Section 4.3). In order to lower verifier complexity, we can extend the AG-IOPP by replacing the final test by an IOPP for RS code. This enhanced AG-IOPP is examined in this section.

8.1 How to iterate the folding to reach a code of dimension 1

Let $C = C(\mathcal{X}, \mathcal{P}, \mathcal{G})$ be foldable in the sense of Definition 8 on a Kummer curve or on a curve in the Hermitian tower. In Sections 5 and 6, we defined s codes $(C_i)_{0 \leq i \leq s}$, where s is the the number of prime in the decomposition of N in the Kummer case and $s = i_{\max}$ for the Hermitian tower. The code $C_s = C(\mathbb{P}^1, \mathcal{P}', D')$ corresponds to a Reed-Solomon code $\mathsf{RS}[\mathbb{F}, \mathcal{P}', d] = \{f : \mathcal{P}_s \to \mathbb{F}; \deg f \leq d\}$, where the degree bound depends on the parameters of the original code C_0. Taking this into consideration, we want to iterate the folding operation until we get a RS code of dimension 1, as it is done in the FRI protocol [9].

As in Example 4.3, we set $d_0 = d$ and define $d_{i+1} = \left\lfloor \frac{d_i}{2} \right\rfloor$ for any integer i. Set s' the smallest integer such that $d_{s'} = 0$. Then, we consider the sequence of Reed-Solomon codes $(C_{s+i})_{1 \leq i \leq s'}$ when applying the construction described in Section 4 to the initial code C_s. Letting $r = s + s'$, we iteratively reduce the proximity test to the code C_0 to a membership test to the code C_r, which is a Reed-Solomon code of dimension 1. If $f^{(0)} \in C_0$, then $f^{(r)}$ is expected to be a constant function, and this can be tested in a trivial way. We can leverage the fact that C_r is a Reed-Solomon code to extend the protocol described in Section 7.3. We obtain a r-round IOPP system (P, V) for C_0, which is described below.

The prover P and the verifier V are given as input the description of the code C_0. The verifier V is given oracle access to a function $f^{(0)} : \mathcal{P}_0 \to \mathbb{F}$, which is also given as explicit input to the prover P.

COMMIT phase

1. For each round i from 0 to $r-1$:
   a. V picks uniformly at random $\boldsymbol{z}^{(i)}$ in $\mathbb{F}^2$ and sends it to P,
   b. P computes $f^{(i+1)} = \textbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right]$,
   c. If $i < r-1$: P gives oracle access to $f^{(i+1)} : \mathcal{P}_{i+1} \to \mathbb{F}$.
   d. If $i = r-1$: P commits to $\beta \in \mathbb{F}$ (if $f^{(0)} \in C_0$, then $f^{(r)}$ is supposed to be constant equal to β).

QUERY phase

1. Repeat t times the following **query test**:
   a. Pick $Q_0 \in \mathcal{P}_0$ uniformly at random.
   b. For $i = 0$ to $r-1$, run the following *round consistency test*:
      i. Define $Q_{i+1} \in \mathcal{P}_{i+1}$ by $Q_{i+1} = \pi_i(Q_i)$,
      ii. Query $f^{(i+1)}$ to get $f^{(i+1)}(Q_{i+1})$ and query $f^{(i)}$ at points $\widehat{Q} \in S_{Q_{i+1}}$, (if $i = r-1$, set $f^{(r)}(Q_r) = \beta$)
      iii. Compute the value $\textbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right](Q_{i+1})$,
      iv. If $i < r-1$: return **reject** if and only if $f^{(i+1)}(Q_{i+1}) \neq \textbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right](Q_{i+1})$
      v. If $i = r-1$: return **reject** if and only if $\beta \neq \textbf{Fold}\left[f^{(i)}, \boldsymbol{z}^{(i)}\right](Q_{i+1})$
2. Return **acccept**.

8.2 Properties of the AG-IOPP with Kummer curves

Assume $C_0 = C(\mathcal{X}_0, \mathcal{P}_0, D_0)$ is a foldable AG code of blocklength $n_0 = |\mathcal{P}_0|$ on a Kummer curve $\mathcal{X}_0$ (cf. Proposition 18). This means that $\mathcal{X}_0$ is defined by an equation $y^N = f(x)$, where $f \in \mathbb{F}[X]$ is a separable degree-m polynomial, $m \equiv -1 \mod N$, N is coprime with $|\mathbb{F}|$, $|\mathcal{P}_0| = \alpha N$ for some integer α, and $\deg D_0 < \alpha N$. Assume α is a power of 2 and N is a η-smooth integer for a small fixed parameter $\eta \in \mathbb{N}$.

Proposition 23 states that the relative minimum distances of the codes C_i are all equal to $\Delta(C_0) = 1 - \frac{\deg D_0}{\alpha N}$. Therefore, the ordering on the integers involved in the prime decomposition $\prod_{i=0}^{s-1} p_i$ of N does not impact the parameters of the protocol. Moreover, the code $C_s = C(\mathcal{X}_s, \mathcal{P}_s, D_s)$ corresponds to a RS code

$$C_s = \mathsf{RS}\left[\mathbb{F}, \mathcal{P}_s, \frac{\deg D_0}{N}\right] = \left\{f : \mathcal{P}_s \to \mathbb{F}; \deg f \leq \frac{\deg D_0}{N}\right\}$$

of blocklength $|\mathcal{P}_s| = \alpha$, which is itself a foldable AG code (see Example 4.3).

▶ **Theorem 43** (Kummer case). *Let $C = (\mathcal{X}_0, \mathcal{P}_0, D_0)$ be a foldable AG code on a Kummer curve satisfying the hypotheses of Proposition 18 with N a η-smooth integer. Denote $n = |\mathcal{P}_0|$. The IOPP (P, V) described in Section 8.1 has perfect completeness and soundness as stated in Theorem 42. Moreover, for t repetitions of the QUERY phase, we have:*

rounds complexity	$\mathsf{r}(n)$	$< \log n$,
proof length	$\mathsf{l}(n)$	$< n$,
query complexity	$\mathsf{q}(n)$	$\leq t\eta \log_2 n + 1$,
prover complexity	$\mathsf{t_p}(n)$	$= O_\eta(n)$,
verifier decision complexity	$\mathsf{t_v}(n)$	$= O_\eta(t \log n)$.

Proof. Noticing that the round complexity is now $\mathsf{r}(n) = s + s'$, straightforward calculations show that complexity, query complexity and proof length computed in the proof 42 still hold. Completeness and soundness also follow from [4]. We estimate prover complexity and verifier complexity below.

Prover complexity. Fix a round index $i < r - 1$. The balancing functions $\nu_{i+1,j} : \mathcal{P}_{i+1} \to \mathbb{F}$ can be precomputed since they do not depend on $f^{(i)}, \boldsymbol{z}^{(i)}$ (see Remark 44). To simplify notation, denote $f = f^{(i)}$. For any $\boldsymbol{z} = (z_1, z_2) \in \mathbb{F}^2$, computing the successive powers $(z_1^j, z_2^j)_{0 \le j < p_i}$ takes $2(p_i - 2)$ multiplications. For each $P \in \mathcal{P}_{i+1}$, an honest prover must compute the coefficients $(a_{j,P})_{0 \le j, <P}$ of the polynomial $I_{f,P}(X)$ of degree $\deg I_{f,P} < p_i$ from the interpolation set $\left\{(\mu_i(\widehat{P}), f(\widehat{P})) \mid \widehat{P} \in S_P\right\}$ of size p_i. Notice that $\mu_i = y$, so computing $\mu_i(\widehat{P})$ for $\widehat{P} \in S_P$ is done for free. Univariate interpolation for a polynomial of degree $< p_i$ can be done in $O(p_i^2)$ by Lagrange interpolation. Overall, one can honestly evaluate $\mathbf{Fold}\left[f, \boldsymbol{z}\right] : \mathcal{P}_{i+1} \to \mathbb{F}$ with $|\mathcal{P}_{i+1}|\, O(p_i^2)$ operations in $\mathbb{F}$. We showed previously that $\sum_{i=1}^{r-1} |\mathcal{P}_i| < n$, thus when summing over $r - 1$ rounds, we get that the cost of (honestly) generating the oracles $f^{(1)}, \ldots, f^{(r-1)}$ is $O_\eta(n)$.

Verifier decision complexity. Verifier complexity is inferred from the previous discussion about prover complexity. For each round, the verifier computes the successive powers of z_1 and z_2, interpolates $I_{f,P}$ for a point $P \in \mathcal{P}_{i+1}$ in $O(p_i^2)$ operations, then computes $\mathbf{Fold}\left[f, \boldsymbol{z}\right](P)$ in a number of operations which is independent of n. Hence, verifier complexity for repetition parameter t is $\mathsf{t}_\mathsf{v}(n) = O_\eta(t \log(n))$. ◀

▶ Remark 44. We give the cost of precomputing the evaluation tables of the balancing functions. Letting $\nu_{i+1,j}$ be as defined in proof of Lemma 17, the sequence of functions $(\nu_{i+1,j})_{0<j<p_i}$ can be evaluated at the same point $P \in \mathcal{P}_{i+1}$ in time $O(\log m + p_i)$ using exponentiation by squaring. Thus, the evaluations of $\nu_{i+1,1}, \ldots \nu_{i+1,p_i-1}$ on $\mathcal{P}_{i+1}$ are obtained with $O((\log m + p_i)\,|\mathcal{P}_{i+1}|)$ operations.

We give an example of an AG code over a Kummer curve where $p_{max} = 2$.

▶ **Example 45.** On $\mathbb{F}_{q^2}$ with $q = 2^{61} - 1$ (9^{th} Mersenne prime), we consider the curve

$$\mathcal{X}_0 : y^N = x^3 + x$$

where $N = 2^r$ with $r = 16$. It is maximal [46] of genus $g = N - 1$. We consider the code C_0 associated to $D_0 = 2^{17} P_\infty^0$ on an evaluation set $\mathcal{P}_0 \subset \mathcal{X}_0(\mathbb{F}_{q^2})$ of size $n = 2^{20}$. Its dimension equals $\dim C_0 = 2^{16} + 2$ and its relative minimum distance λ is bounded from below by $1 - 2^{-3}$. Take $\varepsilon = 2^{-6.55}$. By Theorem 42,

$$\mathsf{err}_{commit} \le \frac{\log(n)}{\left|\mathbb{F}_{q^2}\right|} \left(1 + \frac{4}{\varepsilon}\right) \left(\frac{4}{\varepsilon}\right)^2 \approx 2^{4.33+6+3\cdot 6.55-121} \le 2^{-91}$$

$$\mathsf{err}_{query}(\delta) \le (1 - \delta + \varepsilon \log(n))$$

where $1 - \delta = (1 - \lambda + \varepsilon)^{\frac{1}{3}} \le 0,51384$. Hence

$$\mathsf{err}_{query}(\delta) \le 0,51384 + \frac{20}{2^{6.55}} \approx 0,72728.$$

By running the QUERY phase with repetition parameter $t \ge 199$, we get $(\mathsf{err}_{query})^t \le 2^{-91}$ and $\mathsf{err}(\delta) \le 2^{-90}$. The last code C_r is a small Reed-Solomon code of length $n_r = 2^4$ and dimension 2. The total number of rounds of the IOPP is thus $r + 1$.

8.3 Properties of the AG-IOPP with towers of Hermitian curves

▶ **Theorem 46.** *Let $C = (\mathcal{X}, \mathcal{P}, D)$ be a foldable AG code with alphabet $\mathbb{F} = \mathbb{F}_{q^2}$ on a tower of Hermitian curves satisfying the hypotheses of Proposition 32. Letting $i_{\max}$ be the index of the curve $\mathcal{X}$ in the Hermitian tower $(\mathcal{X}_i)_{i\geq 0}$, the length $n = |\mathcal{P}|$ of C is at most $q^{i_{\max}+2}$. The IOPP (P, V) described in Section 8.1 has perfect completeness, and soundness as stated in Theorem 42. Moreover, we have:*

rounds complexity	$\mathsf{r}(n)$	$< \log n$,
proof length	$\mathsf{l}(n)$	$< n$,
query complexity	$\mathsf{q}(n)$	$\leq tq \log n + 1$,
prover complexity	$\mathsf{t_p}(n)$	$= O(n \cdot \mathsf{M}_{\mathbb{F}}(q) \log(q))$,
verifier decision complexity	$\mathsf{t_v}(n)$	$= O(\log n \cdot \mathsf{M}_{\mathbb{F}}(q) \log(q))$.

Proof. The proof follows from proof of Theorem 43, replacing η by q. Prover and verifier complexities are computed from the cost of computing the coefficients of a univariate polynomial of degree less than q from its evaluation on points forming an arithmetic progression in $\mathbb{F} = \mathbb{F}_{q^2}$. This interpolation task can be done in $\mathsf{M}_{\mathbb{F}}(q) \log q + O(\mathsf{M}_{\mathbb{F}}(q))$ base field operations [21], where $\mathsf{M}_{\mathbb{F}}(d)$ denotes the cost of multiplying two degree-d univariate polynomials in $\mathbb{F}[X]$. ◀

Given a foldable code as in Proposition 32, the IOPP constructs a sequence of codes as follow:

$$C_i \coloneqq C(\mathcal{X}_{i_{\max}-i}, \mathcal{P}_{i_{\max}-i}, D_{i_{\max}-i}) \text{ where } \mathcal{P}_{i-1} = \pi_i(\mathcal{P}_i) \text{ and } D_i = d_i P_\infty^{(i)}$$

with the integers d_i defined recursively by

$$d_{i-1} \coloneqq \left\lfloor \frac{d_i}{q} \right\rfloor + 2g(\mathcal{X}_{i-1}).$$

▶ Remark 47. Be careful about the indices: the indices for the codes on one hand and for the curves, the support and the degree on the other hand, are in reserved order. The original code to which we test proximity is C_0, defined over the curve $\mathcal{X}_{i_{\max}}$, and the last RS code is $C_{i_{\max}}$, which is defined over $\mathcal{X}_0 \simeq \mathbb{P}^1$.

Unlike the Kummer case, we have to increase the degree of divisor by twice the genus of the curve at each step to make sure the compatibility hypotheses of Definition 7 are valid. This has a counterpart: the dimension of the codes C_i decreases much slowly than their block length. A foldable code in the sense of Definition 8 may induce of a sequence of codes in which the last code $C_{i_{\max}}$ is trivial. In this case, the protocol would no longer be sound. We thus need to control the dimension of the code $C_{i_{\max}}$. This is the purpose of the remaining of this section.

▶ Remark 48. In light of the Kummer case in which the group $\mathbb{Z}/N\mathbb{Z}$ is factored as much as possible, if q is some prime power $q = p^\ell$, we could split the additive group $\mathbb{F}_q \simeq (\mathbb{Z}/pZ)^\ell$ acting at each level with ℓ intermediary rounds. In 36, any value taken by the folding of a function f would be determined by p values of f (instead of q values), and the verifier and the prover would perform polynomial interpolations of degree p. However, for each of these intermediary steps, we would have to make the new divisor grow to fulfill the compatibility conditions (Definition 7), as mentioned above. If the rate increases too much, the relative minimum distance drops and the total number queries to target a designated soundness may be tremendous. The loss in terms of soundness error per QUERY phase seems to be much more significant than the aforementioned advantages.

8.3.1 Bounding the rate of the underlying Reed-Solomon code

We aim to bound the dimension of the code Reed-Solomon code $C_{i_{\max}}$. Let us compute the degree $d_{i_{\max}}$ of the divisor $D_{i_{\max}}$ on $\mathbb{P}^1$.

▶ **Lemma 49.** *For $1 \leq j \leq i_{\max}$, we have*

$$d_{i_{\max}-j} \leq \left\lfloor \frac{d_{i_{\max}}}{q^j} \right\rfloor + \sum_{k=1}^{j} \left\lfloor \frac{2g_{i_{\max}-k}}{q^{j-k}} \right\rfloor + (j-1).$$

Proof. It follows from the definition of the degrees d_i given in (17) and by induction on j. ◀

From Lemma 49, we can get an upper bound on d_0.

▶ **Corollary 50.** *Let us assume that $2(i_{\max}-1) < q$. The degree d_0 of the divisor D_0 on $\mathbb{P}^1$ is bounded from above by*

$$d_0 \leq \left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6} \cdot (3q - 4 + 2i_{\max})\right)$$

Proof. By Lemma 49, we have the following bound over d_0:

$$d_0 \leq \left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + \sum_{i=0}^{i_{\max}-1} \left\lfloor \frac{2g_i}{q^i} \right\rfloor + i_{\max} - 1.$$

It is thus enough to estimate the sum $\sum_{k=0}^{i_{\max}-1} \left\lfloor \frac{2g_k}{q^k} \right\rfloor$. By Proposition 53,

$$\begin{aligned}\sum_{k=0}^{i_{\max}-1} \left\lfloor \frac{2g_k}{q^k} \right\rfloor &\leq \sum_{k=0}^{i_{\max}-1} (kq + k(k-1)) \\ &= (q-1) \cdot \frac{i_{\max}(i_{\max}-1)}{2} + \frac{i_{\max}(i_{\max}-1)(2i_{\max}-1)}{6} \\ &= \frac{i_{\max}(i_{\max}-1)}{2} \cdot \left(q - \frac{4}{3} + \frac{2i_{\max}}{3}\right),\end{aligned}$$

which gives the expected result. ◀

We aim to determine a sufficient condition on $i_{\max}$ that ensures that the RS code $C_{i_{\max}}$ is not trivial. Let us denote by n_0 the size of the support $\mathcal{P}_0$ of $C_{i_{\max}}$. It satisfies $n_0 \leq q^2$. The rate of $C_{i_{\max}}$ is equal to

$$\frac{d_0 + 1}{n_0}.$$

Also we have $n_{i_{\max}} := |\mathcal{P}_{i_{\max}}| = q^{i_{\max}} n_0$.

▶ **Corollary 51.** *Let us fix $\rho \in (0,1)$. If*

$$\left\lfloor \frac{d_{i_{\max}}}{q^{i_{\max}}} \right\rfloor + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6} \cdot (3q - 4 + 2i_{\max})\right) + 1 < \rho n_0,$$

then the rate of the RS code $C_{i_{\max}}$ is less than ρ.

8.3.2 Foldable codes with constant rate which are endowed with an IOPP with designed soundness

In this paragraph, we focus on foldable codes of the form

$$C_0 = C\left(\mathcal{X}_{i_{\max}}, \mathcal{X}_{i_{\max}}(\mathbb{F}_{q^2}) \setminus \{P_\infty^{(i_{\max})}, (2\alpha+1)g_{i_{\max}})P_\infty^{(i_{\max})}\right) \tag{20}$$

for some $\alpha > 1/2$, as in Section 6.2.1. The evaluation set $\mathcal{P}_{i_{\max}}$ is the whole set of rational points of $\mathcal{X}_{i_{\max}}$ minus the point of at infinity $P_\infty^{(i_{\max})}$, *i.e.* $n_{i_{\max}} = q^{i_{\max}+2}$.

▶ **Proposition 52.** *Let us fix $\rho \in (0,1)$. The rate of the RS code $C_{i_{\max}}$ below C_0 is less than ρ if*

$$2i_{\max}^3 + 3i_{\max}^2(2\alpha + q - 1) + i_{\max}(6\alpha(q-1)+7) - 6\rho q^2 < 0.$$

Proof. For q large enough, we can assume that $2i_{\max} - 1 < q$. Using Proposition 53, we get an upper bound over $d_{i_{\max}}$:

$$d_{i_{\max}} \leq (2\alpha+1)\frac{i_{\max}}{2}q^{i_{\max}}(q + (i_{\max}-1)).$$

From Corollary 51, a sufficient condition for the underlying RS code to have a rate less than ρ is

$$(2\alpha+1)\frac{i_{\max}}{2}(q+(i_{\max}-1)) + (i_{\max}-1)\left(1 + \frac{i_{\max}}{6}(3q - 4 + 2i_{\max})\right) + 1 < \rho q^2$$

Multiplying the inequality by 6, expanding and simplifying, we get our condition. ◀

Now assume that $i_{\max} = q^\varepsilon$ for $\varepsilon \in (0,1)$. In the constant rate regime described in Lemma 33, we have $\alpha i_{\max} = Rq$. The condition above becomes

$$2i_{\max}^3 + 3i_{\max}^2(q-1) + i_{\max}(6Rq+7) < 6q^2\left(\rho - R\left(1 - \frac{1}{q}\right)\right).$$

If $R\left(1-\frac{1}{q}\right) < \rho$, the right handside is positive. Let us give a rough estimation of the largest ε such that $i_{\max} = q^\varepsilon$ satisfies this inequality. The left handside begin equivalent to $3q^{1+2\varepsilon}$, we have $\varepsilon \simeq \frac{1}{2}(1 + \log_q\left(\rho - R\left(1-\frac{1}{q}\right)\right)$.

Table 1 displays some examples of level $i_{\max}$ and initial rate R of foldable codes for which the AG-IOPP reduce the proximity test to testing RS codes of rate ρ. In terms of the soundness of the protocol, it means that λ as defined in Theorem 42 is greater than $1-\rho$.

Table 1 Example of parameters of foldable codes of rate R along the Hermitian tower. Alphabet is $\mathbb{F}_q^2$ and block length is n. The last column gives a bound on the minimal distance of the RS code.

q	$i_{\max}$	n	R	$1-\rho >$
2^4	3	2^{20}	1/8	1/3
2^5	5	2^{35}		
2^4	4	2^{24}	1/16	1/3
2^5	3	2^{25}		3/4
	5	2^{35}		1/2
2^6	4	2^{36}		3/4
	5	2^{42}		2/3
	7	2^{54}		1/2
2^4	3	2^{20}	1/32	1/2

References

1 Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2087–2104. ACM, 2017. doi:10.1145/3133956.3134104.

2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. extended version of FOCS'92. doi:10.1145/278298.278306.

3 Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs; A New Characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 2–13. IEEE Computer Society, 1992. doi:10.1145/273865.273901.

4 Daniel Augot, Sarah Bordage, and Jade Nardi. Efficient multivariate low-degree tests via interactive oracle proofs of proximity for polynomial codes. *Electron. Colloquium Comput. Complex.*, page 118, 2021. URL: https://eccc.weizmann.ac.il/report/2021/118.

5 László Babai. Trading Group Theory for Randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985. doi:10.1145/22145.22192.

6 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991. doi:10.1145/103418.103428.

7 Alp Bassa, Peter Beelen, Arnaldo Garcia, and Henning Stichtenoth. An Improvement of the Gilbert–Varshamov Bound Over Nonprime Fields. *IEEE Transactions on Information Theory*, 60(7):3859–3861, 2014. doi:10.1109/TIT.2014.2316531.

8 Peter Beelen, Johan Rosenkilde, and Grigory Solomatov. Fast encoding of AG codes over c_{ab} curves. *IEEE Trans. Inf. Theory*, 67(3):1641–1655, 2021. doi:10.1109/TIT.2020.3042248.

9 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 14:1–14:17, 2018.

10 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable Zero Knowledge with No Trusted Setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019. doi:10.1007/978-3-030-26954-8_23.

11 Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed-Solomon Codes. *IACR Cryptol. ePrint Arch.*, 2020:654, 2020.

12 Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Interactive Oracle Proofs with Constant Rate and Query Complexity. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 40:1–40:15, 2017.

13 Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. Linear-Size Constant-Query IOPs for Delegating Computation. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 494–521. Springer, 2019. doi:10.1007/978-3-030-36033-7_19.

14 Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent Succinct Arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019.

15 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016. doi:10.1007/978-3-662-53644-5_2.

16 Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling Outside the Box Improves Soundness. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 5:1–5:32, 2020.

17 Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 320–329. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.42.

18 Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-Case to Average Case Reductions for the Distance to a Code. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 24:1–24:23, 2018.

19 Eli Ben-Sasson and Madhu Sudan. Short PCPs with Polylog Query Complexity. *SIAM J. Comput.*, 38(2):551–607, 2008. doi:10.1137/050646445.

20 Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. Zero-Knowledge Succinct Arguments with a Linear-Time Prover. *IACR Cryptol. ePrint Arch.*, page 1527, 2020. URL: https://eprint.iacr.org/2020/1527.

21 Alin Bostan and Eric Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21(4):420–446, 2005. doi:10.1016/j.jco.2004.09.009.

22 Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and Transparent Recursive Proofs from Holography. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 769–793. Springer, 2020. doi:10.1007/978-3-030-45721-1_27.

23 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.

24 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract). In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304. ACM, 1985. doi:10.1145/22145.22178.

25 Valerii Denisovich Goppa. Codes associated with divisors. *Problemy Peredachi Informatsii*, 13(1):33–39, 1977.

26 J. W. P. Hirschfeld, G. Korchmáros, and F. Torres. *Algebraic Curves over a Finite Field*. Princeton University Press, Princeton, 25 Mar. 2013. doi:10.1515/9781400847419.

27 Chuangqiang Hu and Shudi Yang. Multi-point codes over Kummer extensions. *Designs, Codes and Cryptography*, 86:211–230, 2018. doi:10.1007/s10623-017-0335-7.

28 Yael Tauman Kalai and Ran Raz. Interactive PCP. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 536–547. Springer, 2008. doi:10.1007/978-3-540-70583-3_44.

29 Ernst Kani. The Galois-module structure of the space of holomorphic differentials of a curve. *Journal für die reine und angewandte Mathematik*, 367:187–206, 1986. doi:10.1515/crll.1986.367.187.

30 Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. *IACR Cryptol. ePrint Arch.*, 2019:1400, 2019. URL: https://eprint.iacr.org/2019/1400.

31 Joe Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992. doi:10.1145/129712.129782.

32 Gilles Lachaud. Sommes d'Eisenstein et nombre de points de certaines courbes algébriques sur les corps finis. *C. R. Acad. Sci. Paris*, 305, January 1987.

33 Gilles Lachaud. Artin-Schreier curves, exponential sums, and coding theory. *Theoretical Computer Science*, 94(2):295–310, 1992. doi:10.1016/0304-3975(92)90040-M.

34 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89518.

35 Hiren Maharaj. Code Construction on Fiber Products of Kummer Covers. *Information Theory, IEEE Transactions on*, 50:2169–2173, October 2004. doi:10.1109/TIT.2004.833356.

36 Ariane M. Masuda, Luciane Quoos, and Alonso Sepúlveda. One- and Two-Point Codes over Kummer Extensions. *IEEE Transactions on Information Theory*, 62(9):4867–4872, 2016. doi:10.1109/TIT.2016.2583437.

37 Or Meir. IP = PSPACE Using Error-Correcting Codes. *SIAM J. Comput.*, 42(1):380–403, 2013. doi:10.1137/110829660.

38 Silvio Micali. Computationally-Sound Proofs. In Johann A. Makowsky and Elena V. Ravve, editors, *Proceedings of the Annual European Summer Meeting of the Association of Symbolic Logic, Logic Colloquium 1995, Haifa, Israel, August 9-18, 1995*, volume 11 of *Lecture Notes in Logic*, pages 214–268. Springer, 1995. doi:10.1007/978-3-662-22108-2_13.

39 Thilo Mie. Short PCPPs Verifiable in Polylogarithmic Time with O(1) Queries. *Annals of Mathematics and Artificial Intelligence*, 56(3–4):313–338, August 2009. doi:10.1007/s10472-009-9169-y.

40 Carlos Munuera and Ruud Pellikaan. Equality of geometric Goppa codes and equivalence of divisors. *Journal of Pure and Applied Algebra*, 90(3):229–252, 1993. doi:10.1016/0022-4049(93)90043-S.

41 Ruud Pellikaan, Ba zhong Shen, and Gerhard J. M. van Wee. Which linear codes are algebraic-geometric? *IEEE Trans. Inf. Theory*, 37:583–602, 1991. doi:10.1109/18.79915.

42 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62. ACM, 2016. doi:10.1145/2897518.2897652.

43 Noga Ron-Zewi and Ron D. Rothblum. Local Proofs Approaching the Witness Length [Extended Abstract]. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 846–857. IEEE, 2020. doi:10.1109/FOCS46700.2020.00083.

44 StarkWare. ethSTARK Documentation. *IACR Cryptol. ePrint Arch.*, page 582, 2021. URL: https://eprint.iacr.org/2021/582.

45 Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008.

46 Saeed Tafazolian and Fernando Torres. On the curve $y^n = x^m + x$ over finite fields. *Journal of Number Theory*, 145:51–66, 2014. doi:10.1016/j.jnt.2014.05.019.

47 M. A. Tsfasman, S. G. Vlăduţ, and Th. Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Math. Nachr.*, 109:21–28, 1982. doi:10.1002/mana.19821090103.

48 Michael Tsfasman, Serge Vladut, and Dmitry Nogin. *Algebraic Geometric Codes: Basic Notions*. American Mathematical Society, USA, 2007.

49 Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 859–876. IEEE, 2020. doi:10.1109/SP40000.2020.00052.

A Proof of Proposition 39

Proposition 39 is a weighted version of [18, Theorem 4.5]. We only highlight the changes to be made in the proof of [18, Theorem 4.5].

For $z \in \mathbb{F}$ and $(v_0, \dots, v_{l-1}) \in V^l$, let us set $v_z \coloneqq \sum_{i=0}^{l-1} z^i v_i$. Rewriting the proof of Theorem 4.5 [18] with

$$A \coloneqq \{z \in \mathbb{F} \mid \omega_\eta(u_z, V) > 1 - \delta\}$$

provides $v_0, \dots, v_{l-1} \in V$ and a set

$$C \coloneqq \{z \in \mathbb{F} \mid \omega_\eta(u_z, v_z) > 1 - \delta\} \subset A$$

with cardinality $|C| > \frac{l-1}{\varepsilon}$. Let us set $T \coloneqq \{P \in \mathcal{P} \mid u_{i|T} = v_{i|T} \text{ for all } i\}$. Therefore

$$\begin{aligned} 1 - \delta &< \frac{1}{|C|} \sum_{z \in C} \omega_\eta(u_z, v_z) \\ &= \frac{1}{|C| \times |\mathcal{P}|} \sum_{z \in C} \sum_{P \in \mathcal{P}} \eta(P) \mathbb{1}_{u_z(P) = v_z(P)} \\ &= \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} \eta(P) \frac{1}{|C|} \sum_{z \in C} \mathbb{1}_{u_z(P) = v_z(P)} \end{aligned}$$

Notice that if there exists $i \in \{0, \dots, l-1\}$ such that u_i which does not coincide with v_i, the number of $z \in \mathbb{F}$ such that $u_z(P) = v_z(P)$ is at most $l - 1$. Then

$$\begin{aligned} 1 - \delta &\leq \frac{1}{|\mathcal{P}|} \sum_{P \in T} \eta(P) + \frac{1}{|\mathcal{P}|} \sum_{P \in C \setminus T} \eta(P) \frac{l-1}{|C|} \\ &\leq \frac{1}{|\mathcal{P}|} \sum_{P \in T} \eta(P) + \varepsilon, \end{aligned}$$

which gives the first item of the proposition.

B Properties of the genera of the curves in the Hermitian tower

To estimate the parameters of the foldable codes we define along the Hermitian tower, we need to handle the genera of the curves in this tower. From the formulae (15), we deduce a bound on the genus of the curve $\mathcal{X}_i$ for small i (Proposition 53) and the asymptotic behaviour of the ratio of g_i by q^{i+2} for $i = q^\varepsilon$ when q goes to infinity (Lemma 54).

▶ **Proposition 53.** *For $i \geq 1$, we have*

$$g_i \leq \frac{q^{i+1}}{2} \sum_{k=1}^{i} \binom{i}{k} \frac{1}{q^{k-1}} \leq \frac{iq^{i+1}}{2} \sum_{k=1}^{i} \left(\frac{i}{q}\right)^{k-1} \leq \frac{i}{2} q^{i+1} + \frac{i(i-1)}{2} q^i,$$

the last inequality holding only if $2(i-1) < q$.

Proof. Starting from the second formula of (15), we can write

$$\begin{aligned} g_i &= \frac{1}{2} \cdot \left(q^{i+1} \sum_{k=1}^{i} \left(1+\frac{1}{q}\right)^{k-1} + 1 - (1+q)^i \right) \\ &\leq \frac{q^{i+1}}{2} \cdot q \cdot ((1+1/q)^i - 1) \\ &= \frac{q^{i+1}}{2} \cdot \sum_{k=1}^{i} \binom{i}{k} \frac{1}{q^{k-1}}, \end{aligned}$$

using that the term outside the geometric sum is non-positive. Note that if $k \geq 2$, then we can bound the binomial coefficients as follows

$$\binom{i}{k} = \frac{i(i-1)\cdots(i-k+1)}{k(k-1)\cdots 2} \leq \frac{i(i-1)^{k-1}}{2},$$

as the denominator is greater than 2 and the factors $i-1, i-2, ..., i-k+1$ are all lesser than $i-1$. Factoring and using this upper bound on the binomial coefficients, we get

$$g_i \leq \frac{q^{i+1}}{2} \cdot \left(i + \frac{i}{2} \sum_{k=2}^{i} \left(\frac{i-1}{q}\right)^{k-1} \right) = \frac{iq^{i+1}}{2} \left(1 + \frac{1}{2} \cdot \left(\frac{i-1}{q}\right) \cdot \sum_{k=2}^{i} \left(\frac{i-1}{q}\right)^{k-2} \right).$$

Assuming that $2(i-1) < q$, we can bound the sum $\sum_{k=2}^{i} \left(\frac{i-1}{q}\right)^{k-2}$ by 2, which concludes the proof. ◀

▶ **Lemma 54.** *Fix $\varepsilon \in (0,1)$ and set $i = q^\varepsilon$. Then*

$$\frac{g_i}{q^{i+2}} \underset{q\to\infty}{\sim} \frac{1}{2q^{1-\varepsilon}}.$$

Proof. From the first formula of (15), we get

$$\frac{2g i_{\max}}{q^{i_{\max}+2}} = \left(1 - \frac{1}{q^2}\right) \left[\left(1+\frac{1}{q}\right)^{q^\varepsilon} - 1 \right] + \frac{1}{q^{i_{\max}+2}} - \frac{1}{q^2}$$

Let us examine the asymptotic behaviour of $\left(1+\frac{1}{q}\right)^{q^\varepsilon}$ when q goes to infinity. Set $h = q^{-1}$.

$$\left(1+\frac{1}{q}\right)^{q^\varepsilon} = \exp\left(h^{1-\varepsilon} \cdot \frac{\log(1+h)}{h} \right) = \exp\left(h^{1-\varepsilon} \left(1 - \frac{h}{2} + o(h) \right) \right) = 1 + h^{1-\varepsilon} + o(h^{1-\varepsilon})$$

Therefore, we have

$$\left(1+\frac{1}{q}\right)^{q^\varepsilon} - 1 \sim \frac{1}{q^{1-\varepsilon}}.$$ ◀

Vanishing Spaces of Random Sets and Applications to Reed-Muller Codes

Siddharth Bhandari
Simons Institute for the Theory of Computing, Berkeley, CA, USA

Prahladh Harsha
Tata Institute of Fundamental Research, Mumbai, India

Ramprasad Saptharishi
Tata Institute of Fundamental Research, Mumbai, India

Srikanth Srinivasan
Aarhus University, Denmark

Abstract

We study the following natural question on random sets of points in $\mathbb{F}_2^m$:

> Given a random set of k points $Z = \{z_1, z_2, \ldots, z_k\} \subseteq \mathbb{F}_2^m$, what is the dimension of the space of degree at most r multilinear polynomials that vanish on all points in Z?

We show that, for $r \leq \gamma m$ (where $\gamma > 0$ is a small, absolute constant) and $k = (1-\varepsilon) \cdot \binom{m}{\leq r}$ for any constant $\varepsilon > 0$, the space of degree at most r multilinear polynomials vanishing on a random set $Z = \{z_1, \ldots, z_k\}$ has dimension exactly $\binom{m}{\leq r} - k$ with probability $1 - o(1)$. This bound shows that random sets have a much smaller space of degree at most r multilinear polynomials vanishing on them, compared to the worst-case bound (due to Wei (IEEE Trans. Inform. Theory, 1991)) of $\binom{m}{\leq r} - \binom{\log_2 k}{\leq r} \gg \binom{m}{\leq r} - k$.

Using this bound, we show that high-degree Reed-Muller codes ($\mathrm{RM}(m,d)$ with $d > (1-\gamma)m$) "achieve capacity" under the Binary Erasure Channel in the sense that, for any $\varepsilon > 0$, we can recover from $(1-\varepsilon) \cdot \binom{m}{\leq m-d-1}$ random erasures with probability $1 - o(1)$. This also implies that $\mathrm{RM}(m,d)$ is also efficiently decodable from $\approx \binom{m}{\leq m-(d/2)}$ random errors for the same range of parameters.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Reed-Muller codes, polynomials, weight-distribution, vanishing ideals, erasures, capacity

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.31

Funding *Siddharth Bhandari*: Research supported by the Simons-Berkeley Postdoctoral Fellowship.
Prahladh Harsha: Research done when the author was visiting the Simons Institute for the Theory of Computing. Research supported in part by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500 and the Swarnajayanti Fellowship.
Ramprasad Saptharishi: Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500 and the Ramanujan Fellowship of the DST.
Srikanth Srinivasan: Supported by Startup grant from Aarhus University.

Acknowledgements We thank the anonymous referees for several helpful comments.

1 Introduction

The Reed-Muller (RM) code is one of the most basic error-correcting codes studied in coding theory, first introduced by Muller [7] and Reed [9] in 1954. Stated in the language of polynomials, the RM code with parameters m and r for positive integers $m > r$, denoted by $\mathrm{RM}(m,r)$, is the code whose codewords are evaluations of m-variate multilinear polynomials

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 31; pp. 31:1–31:14

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

of degree at most r over the vector space $\mathbb{F}_2^m$. Despite these being one of the earliest codes discovered in coding theory, several properties of these codes (weight-distribution, capacity-achieving on any binary memory-less symmetric (BMS) channel) are not yet fully-understood.

A natural problem that arises while investigating the Shannon-capacity of RM code is the following:

Given a set Z of k points $Z = \{z_1, z_2, \ldots, z_k\} \subseteq \mathbb{F}_2^m$, what is the dimension of the space $\mathbb{I}_r(Z)$ of degree at most r multilinear polynomials that vanish on all points in Z?

To understand the connection to RM codes, we give an equivalent description of this problem in terms of the parity check matrix of the RM code. Let $E(m,r)$ be the $\binom{m}{\leq r} \times 2^m$-matrix whose columns are indexed by elements in $\mathbb{F}_2^m$ and rows by m-variate multilinear monomials of degree at most r. The z^{th} column of $E(m,r)$ for $z \in \mathbb{F}_2^m$ is the column vector $z^{(r)}$ consisting of evaluations of all multilinear m-variate degree $\leq r$ monomials at the point z. It is not hard to see that $E(m,r)$ is the parity check matrix of the Reed-Muller code $\mathrm{RM}(m, m-r-1)$. An equivalent formulation of the above question in terms of the matrix $E(m,r)$ is the following: given a set Z of k points $Z = \{z_1, z_2, \ldots, z_k\} \subseteq \mathbb{F}_2^m$, what is the rank of the $\binom{m}{\leq r} \times |Z|$ sub-matrix $E(m,r)_Z$ of $E(m,r)$ obtained by picking the columns corresponding to the points in Z.

While studying the generalized Hamming weights of RM codes over $\mathbb{F}_2$, Wei [15] proved worst-case bounds for the above problem. In its simplest form, these worst-case bounds show that if $|Z| = 2^\ell$ for some $0 \leq \ell \leq m$, then $\dim(\mathbb{I}_r(Z)) \leq \binom{m}{\leq r} - \binom{\ell}{\leq r}$. These worst-case bounds were then generalized to arbitrary fields by Keevash and Sudakov [5] and have had applications in combinatorics as well as theoretical computer science. Ben-Eliezer, Hod and Lovett [2] reproved the above form of this bound and used it to study the correlation of random polynomials with lower-degree polynomials. Nie and Wang [8] used these bounds bound to prove extensions of the Kakeya Theorem. Chen, De, and Vijayaraghavan [3] used these bounds to analyze their algorithms for learning mixtures of subspaces over finite fields. In the context of RM codes, Abbe, Shpilka and Wigderson [1] used these worst-case bounds to show that RM codes of high rate ($r = m - o(\sqrt{m/\log m})$) achieve Shannon-capacity over the BEC channel.

These bounds due to Wei only prove extremal (i.e., worst-case) bounds on $\dim(\mathbb{I}_r(Z))$. The tightness of this bound is witnessed when Z is a subcube of size 2^ℓ, where $\dim(\mathbb{I}_r(Z) = \binom{m}{\leq r} - \binom{\ell}{\leq r}$ which is considerably larger than $\binom{m}{\leq r} - |Z| = \binom{m}{\leq r} - 2^\ell$. We note that $\binom{m}{\leq r} - |Z|$ is a lower bound on the dimension of $\mathbb{I}_r(Z)$ since each point in Z can reduce the dimension by at most 1. It is thus natural to ask how large is $\dim(\mathbb{I}_r(Z))$ for sets Z other than a subcube. In particular, how does $\dim(\mathbb{I}_r(Z))$ compare to $\binom{m}{\leq r} - |Z|$ when Z is a random set of points of size $K < \binom{m}{\leq r}$. Our main result shows that for small r (more precisely, for $r \leq \gamma m$ for a small, but absolute, constant $\gamma > 0$) and $|Z| = (1-\varepsilon) \cdot \binom{m}{\leq r}$ for any constant $\varepsilon \in (0,1)$, a random set Z behaves very differently from a subcube and the dimension of $\mathbb{I}_r(Z)$ is as small as it can be, namely $\binom{m}{\leq r} - |Z|$.

▶ **Theorem 1.1** (dimension of degree-r vanishing space of random set)**.** *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let $K = (1-\varepsilon) \cdot \binom{m}{\leq r}$ and $r < \gamma_0 m$. Then,*

$$\Pr_{Z \subseteq \mathbb{F}_2^m}\left[\dim(\mathbb{I}_r(Z)) = \binom{m}{\leq r} - K\right] = 1 - o(1)$$

where $Z = \{z_1, \ldots, z_K\}$ is a uniformly random set of K distinct points in $\mathbb{F}_2^m$. In other words, the probability that the set of columns $\{z_1^{(r)}, z_2^{(r)}, \ldots, z_K^{(r)}\}$ is not linearly independent is $o(1)$, where $z^{(r)}$ denotes the vector consisting of evaluations of all multilinear m-variate degree $\leq r$ monomials at the point z.

Abbe, Shpilka and Wigderson [1] also proved a similar result for a smaller range of r, namely $r = o(\sqrt{m/\log m})$ using Wei's extremal bound. We note that if γ_0 is chosen sufficiently small such that $K \leq \binom{m}{\leq r} = o(2^{m/2})$, then it does not matter if the points $z_1, \ldots, z_k$ are chosen with or without replacement since the two distributions are then $o(1)$-indistinguishable. When the points are chosen with replacement, then one cannot expect to improve the range of r for which the above theorem holds (up to the choice of the constant γ_0) since if $K \gg 2^{m/2}$, then with high probability two of the z_i's are equal in which case $\mathbb{I}_r(Z)$ will definitely have dimension strictly larger than $\binom{m}{\leq r} - K$.

As one would expect, this average-case bound on $\dim(\mathbb{I}_r(Z))$ leads to a better understanding of the Shannon-capacity of RM codes over the high-rate regime.

Shannon-capacity of Reed-Muller codes

Reed-Muller codes were shown to achieve Shannon-capacity over the BEC channel in the extremal regimes (low rate $r = o(m)$ and high rate $r = m - o(m)$) by Abbe, Shpilka and Wigderson [1] and in the constant rate regime ($r = m/2 \pm O(\sqrt{m})$) by Kudekar et al [6]. More recently, Reeves and Pfister [10] showed that RM codes over the constant rate regime ($r = m/2 \pm O(\sqrt{m})$) achieve Shannon-capacity over any BMS channel. Despite all this progress, the general question of whether RM codes achieve capacity over the entire regime ($1 \leq r < m - 1$) and over any BMS channel remains open.

The results of Kudekar et al [6] and Reeves and Pfister [10] are obtained using Boolean function analysis while the results of Abbe, Shpilka and Wigderson for the Shannon-capacity over the BEC channel of RM codes in the low-rate regime ($r = o(m)$) were obtained using the weight-distribution bound of Kaufman, Lovett and Porat [4]. Subsequent improvements in the weight distribution due to Sberlo and Shpilka [13] led to the Shannon-capacity of RM codes over the BEC channel for a wider range of the degree parameter r, more precisely for $r = \gamma m$ for $\gamma < 1/70$.

The Shannon-capacity of RM over the high-rate regime ($r = m - o(\sqrt{m/\log m})$ were also proved by Abbe, Shpilka and Wigderson using Wei's worst-case bounds on $\dim(\mathbb{I}_r(Z))$. In particular, these latter results do not use the weight distribution of RM codes and hence the subsequent improvements in our understanding of the weight distribution of RM codes due to Samorodnitsky [11] and Sberlo-Shpilka [13] did not lead to an improved understanding of the Shannon-capacity of RM codes in the high-rate regime. We rectify this gap by giving an alternate bound for the Shannon-capacity in terms of the our average-case bound on $\dim(\mathbb{I}_r(Z))$ and widen the range of the degree parameter r for which high-rate RM codes achieve Shannon-capacity over the BEC channel.

▶ **Theorem 1.2** (high-degree RM codes under erasures). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let m, d be growing parameters with $d > m(1 - \gamma_0)$. Then, the code $\mathrm{RM}(m, d)$ can correct $K = (1 - \varepsilon)\binom{m}{\leq m-d-1}$ random errors with probability $1 - o(1)$.*

Abbe, Shpilka and Wigderson [1] and Saptharishi, Shpilka and Volk [12] showed how to reduce the resilience of certain RM codes under the BSC to the resilience of appropriate RM codes under the BEC. Using this reduction, we obtain the following corollary of the above theorem for the BSC channel.

▶ **Corollary 1.3** (high-degree RM codes under errors). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let m, r be growing parameters with $r < \gamma_0 m$. Then, the code $\mathrm{RM}(m, m-2r-2)$ can be efficiently decode from $K = (1-\varepsilon)\binom{m}{\leq r}$ random errors.*

See Section 4 for further discussion on Shannon-capacity and what it means for high-rate RM codes to achieve it over the BEC and BSC channels.

We remark that our proof methods work for any linear code, not necessarily the RM code, provided one has good bounds on the weight distribution of the dual code.

1.1 Proof Overview

Recall that Theorem 1.1 is a statement about the dimension of the degree-r vanishing space of a uniformly random subset Z of $\mathbb{F}_2^m$ of size K where $K = (1-\varepsilon) \cdot \binom{m}{\leq r}$. In the regime of parameters we are interested in (i.e., $r < \gamma_0 m$ for a small enough constant γ_0), this is more or less equivalent to[1] choosing K points $z_1, \ldots, z_K$ independently and uniformly from $\mathbb{F}_2^m$. We assume that Z is defined this way for the rest of this section.

To argue about the dimension of the degree-r vanishing space $\mathbb{I}_r(Z)$, we instead argue about the *size* S of $\mathbb{I}_r(Z)$. Note that since $\dim(\mathbb{I}_r(Z)) \geq D := \binom{m}{\leq r} - K$, this set has size *at least* 2^D, and has size at least 2^{D+1} if $\dim(\mathbb{I}_r(Z)) > D$. In light of this, it is sufficient to show that

$$\mathbb{E}[S] = \exp_2(D)(1 + o(1)).$$

Estimating $\mathbb{E}[S]$ turns out to be very closely related to a recent result of Sberlo and Shpilka [13], who prove strong results on the parameters of capacity-achieving Reed-Muller codes in the low-degree setting.

More precisely, we can easily see that the probability that a uniformly random polynomial $P \in \mathrm{RM}(m, r)$ belongs to $\mathbb{I}_r(Z)$ is exactly $(1 - \mathrm{wt}(P))^K$ where $\mathrm{wt}(P)$ is the *fractional Hamming weight of* P (i.e., the fraction of points where it does not vanish). Thus, we have

$$\mathbb{E}[S] = \sum_{P \in \mathrm{RM}(m,r)} (1 - \mathrm{wt}(P))^K.$$

Now, while there are polynomials $P \in \mathrm{RM}(m, r)$ of very small weight[2], *most* polynomials in $\mathrm{RM}(m, r)$ have weight close to $1/2$, which should indicate that the sum is close to 2^{-K} as required. However, a careful analysis is required, as the contribution of a polynomial P increases exponentially as its weight decreases.

It turns out that Sberlo and Shpilka [13] analyzed a very similar quantity in their recent work. More precisely they showed that for any constant $\delta > 0$ (and also a large range of sub-constant δ), we have

$$\sum_{P \in \mathrm{RM}(m,r) \setminus \{\mathbf{0}\}} (1 - \mathrm{wt}(P))^{(1+\delta)\cdot\binom{m}{\leq r}} = o(1).$$

While this is very closely related to our result, we are not able to recover the exact bound we need using this inequality.

---

[1] In the sense that the two distributions have small statistical distance.

[2] It is a standard fact (e.g. by the Schwartz-Zippel lemma) that the minimum weight of a non-zero polynomial from $\mathrm{RM}(m, r)$ is 2^{-r}.

However, the technical lemmas used to derive the above result are strong results on the *weight distribution* of $\mathrm{RM}(m, r)$, which are upper bounds on the number of polynomials of small weight. Using these upper bounds and carrying out the relevant computations yields Theorem 1.1.

The application to resilience under the BEC and BSC (Theorem 1.2 and Corollary 1.3) follow from Theorem 1.1 in a straight-forward manner from the works of Abbe, Shpilka and Wigderson [1] and Saptharishi, Shpilka and Volk [12].

Organisation

We discuss some notation and preliminaries in Section 2 and proceed to the proof of Theorem 1.1 in Section 3. We then present the applications to proving the resilience of RM codes under BEC and BSC in Section 4.

2 Notation and preliminaries

1. We use $[m]$ to represent the set $\{1, 2, 3, \ldots, m\}$ and the expression $\binom{m}{\leq r}$ to denote the sum
$$\binom{m}{0} + \cdots + \binom{m}{r}.$$
2. We denote by $\binom{[m]}{r}$ and $\binom{[m]}{\leq r}$ the sets $\{S \subseteq [m] \colon |S| = r\}$ and $\{S \subseteq [m] \colon |S| \leq r\}$ respectively.
3. We shall abuse notation and use $\mathrm{RM}(m, r)$ to denote the vector space of all m-variate multivariate polynomials in $\mathbb{F}_2[x_1, \ldots, x_m]$ of degree at most r. Throughout the paper, the parameter m will be unchanged and we will avoid mentioning it for brevity. Let $n = 2^m$.
4. For parameters m, r, define the matrix $E(m, r)$ as the $\binom{m}{\leq r} \times 2^m$-matrix whose columns are indexed by elements in $\mathbb{F}_2^m$ and rows by multlinear m-variate monomials of degree at most r and whose z-th column is the vector consisting of the *evaluation* of all multilinear m-variate degree $\leq r$ monomials on z.
5. For a polynomial $P \in \mathrm{RM}(m, r)$, we use $\mathrm{wt}(P)$ to denote the *fractional Hamming weight*:
$$\mathrm{wt}(P) := \frac{|\{z \in \mathbb{F}_2^m \ : \ P(z) \neq 0\}|}{2^m}$$
We shall say that a polynomial $P \in \mathrm{RM}(m, r)$ is η-biased if $|\mathrm{wt}(P) - 1/2| \geq \eta$. We shall also use $\mathrm{RM}_\eta(m, r)$ to denote the set of all η-biased polynomials in $\mathrm{RM}(m, r)$.
6. For a real number $\alpha \in (0, 1)$, we denote by $\mathrm{WtDist}_{m,r}(\alpha)$ the number of polynomials of (fractional) weight at most α in $\mathrm{RM}(m, r)$, i.e., $\mathrm{WtDist}_{m,r}(\alpha) := |\{P \in \mathrm{RM}(m, r) \colon \mathrm{wt}(p) \leq \alpha\}|$.
7. All complexity notations used in the paper are with respect to m as the growing parameter.

2.1 Weight distribution bounds

We use the following bounds on the weight-distribution of Reed-Muller codes due to Sberlo and Shpilka [13].

▶ **Theorem 2.1** (Sberlo and Shpilka [13]: bounds for low-weight codewords). *For any $\ell \geq 1$, we have*

$$\mathrm{WtDist}_{m,r}(2^{-\ell}) \leq \exp_2\left(O(m^4) + 17 \cdot (c_\gamma \ell + d_\gamma) \cdot \gamma^{\ell-1} \cdot \binom{m}{\leq r}\right)$$

where $c_\gamma = 1/(1-\gamma)$ and $d_\gamma = \frac{(2-\gamma)}{(1-\gamma)^2}$.

In particular, if $\gamma \leq (1/2)$, we have

$$\mathrm{WtDist}_{m,r}(2^{-\ell}) \leq \exp_2\left(O(m^4) + O(\ell \cdot \gamma^{\ell-1}) \cdot \binom{m}{\leq r}\right). \tag{1}$$

▶ **Theorem 2.2** (Sberlo and Shpilka [13]: bounds for medium-weight codewords). *Assume that $\gamma \in (0, (1/2) - \Omega(1))$. For a positive integer ℓ such that ℓ/m upper bounded by a small enough constant*[3]*, we have*

$$\mathrm{WtDist}_{m,r}\left(\frac{1}{2} - 2^{-\ell}\right) \leq \exp_2\left(O(m^4) + (1 - 2^{-c(\gamma,\ell)}) \cdot \binom{m}{\leq r}\right) \tag{2}$$

where $c(\gamma, \ell) = O(\max\{\gamma^2\ell, \gamma\})$.

3 Degree-r vanishing spaces and closures

We first define the notion of a *vanishing space*. This is similar to the notion of *vanishing ideals* in basic algebraic geometry but we refer to them as *vanishing space* instead to stress that we are studying them as a vector space and not an ideal.

▶ **Definition 3.1** (degree-r vanishing spaces). *For a set $Z \subseteq \mathbb{F}_2^m$, we use $\mathbb{I}_r(Z)$ to denote the* degree-r vanishing space *defined as*

$$\mathbb{I}_r(Z) := \{P \in \mathrm{RM}(m,r) \ : \ P(z) = 0 \text{ for all } z \in Z\}.$$

Related to the vanishing ideals is also the notion of the *degree-r closure* (similar to the Zariski closure in standard algebraic geometry but restricted to the setting of $\mathbb{F}_2^m$), which is the set of points on which every polynomial in the degree-r vanishing space vanishes.

▶ **Definition 3.2** (degree-r closure). *For a set $Z \subseteq \mathbb{F}_2^m$, we use $\mathrm{Closure}_r(Z)$ to denote the* degree-r closure *of Z defined as*

$$\mathrm{Closure}_r(Z) := \{u \in \mathbb{F}_2^m \ : \ P(u) = 0 \text{ for all } P \in \mathbb{I}_r(Z)\}.$$

The above notion can be equivalently defined as the set of all $u \in \mathbb{F}_2^m$ such that column of $E(m,r)$ indexed by u is in the span of columns of $E(m,r)$ indexed by $z \in Z$. That is,

$$\mathrm{Closure}_r(Z) = \left\{u \in \mathbb{F}_2^m \ : \ u^{(r)} \in \mathrm{span}\left\{z^{(r)} \ : \ z \in Z\right\}\right\},$$

where $z^{(r)}$ denotes the $\binom{m}{\leq r}$-dimension vector of evaluations of all m-variate degree at most r monomials at the point z.

For any set $Z \subseteq \mathbb{F}_2^m$ of size k, note that $\mathbb{I}_r(Z)$ is a vector space of dimension at least $\binom{m}{\leq r} - k$, as each constraint $P(z) = 0$ adds one homogeneous linear constraint on the ambient space $\mathrm{RM}(m,r)$. In fact, it is easy to see that $\mathbb{I}_r(Z)$ has rank $\binom{m}{\leq r} - k$ if and only if each point of Z is not in the closure of the previous points.

[3] [13, Theorem 1.3] only guarantees this for $\ell = o(m)$ but [13, Remark 1.1] after the theorem statement says that in this setting it holds for $\ell = \Omega(m)$.

▶ **Observation 3.3** (vanishing ideals of minimal rank). *Let $Z = \{z_1, \ldots, z_k\} \subseteq \mathbb{F}_2^m$. Then,*

$$\dim \mathbb{I}_r(Z) = \binom{m}{\leq r} - k \iff (\forall i = 1, \ldots, k-1 : \; z_i \notin \mathrm{Closure}_r(\{z_1, \ldots, z_{i-1}\})) \,. \qquad \blacktriangleleft$$

3.1 Dimension of the degree-r vanishing spaces of random sets

In this section, we will be interested in studying the vanishing spaces and closures of a random set Z obtained by picking $K = (1-\varepsilon)\binom{m}{\leq r}$ points from $\mathbb{F}_2^m$ (where $\varepsilon > 0$ is some constant). We restate Theorem 1.1 below.

▶ **Theorem 1.1** (dimension of degree-r vanishing space of random set). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let $K = (1-\varepsilon) \cdot \binom{m}{\leq r}$ and $r < \gamma_0 m$. Then,*

$$\Pr_{Z \subseteq \mathbb{F}_2^m} \left[\dim(\mathbb{I}_r(Z)) = \binom{m}{\leq r} - K \right] = 1 - o(1)$$

where $Z = \{z_1, \ldots, z_K\}$ is a uniformly random set of K distinct points in $\mathbb{F}_2^m$. In other words, the probability that the set of columns $\{z_1^{(r)}, z_2^{(r)}, \ldots, z_K^{(r)}\}$ is not linearly independent is $o(1)$, where $z^{(r)}$ denotes the vector consisting of evaluations of all multilinear m-variate degree $\leq r$ monomials at the point z.

We will use the following lemma to prove the above theorem.

▶ **Lemma 3.4** (size of degree r-vanishing space of random set). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let $K = (1-\varepsilon) \cdot \binom{m}{\leq r}$ and $r < \gamma_0 m$. Then,*

$$\mathbb{E}\left[\frac{|\mathbb{I}_r(Z)|}{\exp_2\left(\binom{m}{\leq r}\right)} \right] = 2^{-K}(1 + o(1))$$

where $Z = \{z_1, \ldots, z_K\}$ is a uniformly random set of K distinct points in $\mathbb{F}_2^m$.

We first use the above lemma to prove Theorem 1.1. Lemma 3.4 is proved in Subsection 3.2.

Proof of Theorem 1.1. Let p be the probability that $\dim(\mathbb{I}_r(Z)) = \binom{m}{\leq r} - K$. Note that $\dim(\mathbb{I}_r(Z))$ is always *at least* $\binom{m}{\leq r} - K$, as the space $\mathbb{I}_r(Z)$ is a subspace of $RM(m,r)$ defined by K linear equations. Thus,

$$\begin{aligned} \mathbb{E}\left[\frac{|\mathbb{I}_r(Z)|}{\exp_2\left(\binom{m}{\leq r}\right)} \right] &= \mathbb{E}\left[\exp_2\left(\dim(\mathbb{I}_r(Z)) - \binom{m}{\leq r} \right) \right] \\ &\geq p \cdot \exp_2(-K) + (1-p) \cdot 2 \cdot \exp_2(-K). \end{aligned}$$

Combining this with Lemma 3.4 we get that

$$p \cdot \exp_2(-K) + (1-p) \cdot 2 \cdot \exp_2(-K) \leq 2^{-K}(1 + o(1))$$

which gives $1 - p \leq o(1)$. Hence, we get that $p = 1 - o(1)$. $\blacktriangleleft$

3.2 Proof of Lemma 3.4

We start with reformulating the problem of bounding size of $\mathbb{I}_r(Z)$ to computing a certain weighted sum of the polynomials in $\mathrm{RM}(m,r)$.

Let Q be a uniformly random polynomial chosen from $\mathrm{RM}(m,r)$. Then,

$$\begin{aligned}
\mathbb{E}\left[\frac{|\mathbb{I}_r(Z)|}{\exp_2\left(\binom{m}{\leq r}\right)}\right] &= \mathop{\mathbb{E}}_{Z,Q}\left[\mathbb{1}[Q \in \mathbb{I}_r(Z)]\right] \\
&= \mathop{\mathbb{E}}_{Q}\left[\mathop{\mathbb{E}}_{Z}[\mathbb{1}[Q \in \mathbb{I}_r(Z)]]\right] \\
&= \sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\binom{m}{\leq r}\right) \cdot \frac{\binom{(1-\mathrm{wt}(P))2^m}{K}}{\binom{2^m}{K}} \\
&= \exp_2(-K) \cdot \left[\sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\varepsilon \cdot \binom{m}{\leq r}\right) \cdot \frac{\binom{(1-\mathrm{wt}(P))2^m}{K}}{\binom{2^m}{K}}\right].
\end{aligned}$$

To complete the proof of Lemma 3.4 we therefore need the following claim.

▷ Claim 3.5.

$$\sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\varepsilon \cdot \binom{m}{\leq r}\right) \cdot \frac{\binom{(1-\mathrm{wt}(P))2^m}{K}}{\binom{2^m}{K}} \leq 1 + o(1).$$

Given Claim 3.5, we see that

$$\begin{aligned}
\mathbb{E}\left[\frac{|\mathbb{I}_r(Z)|}{\exp_2\left(\binom{m}{\leq r}\right)}\right] &= \exp_2(-K) \cdot \left[\sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\varepsilon \cdot \binom{m}{\leq r}\right) \cdot \frac{\binom{(1-\mathrm{wt}(P))2^m}{K}}{\binom{2^m}{K}}\right] \\
&\leq \exp_2(-K) \cdot (1 + o(1)),
\end{aligned}$$

which concludes the proof of Lemma 3.4.

Next, we proceed to prove Claim 3.5.

Proof of Claim 3.5. Let $u = \binom{m}{\leq r}$ in the following. Recall that $K = (1-\varepsilon)u$. Also,

$$\sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\varepsilon u\right) \cdot \frac{\binom{(1-\mathrm{wt}(P))2^m}{K}}{\binom{2^m}{K}} \leq \sum_{P \in \mathrm{RM}(m,r)} \exp_2\left(-\varepsilon u\right) \cdot (1-\mathrm{wt}(P))^{(1-\varepsilon)u}.$$

Set $\delta = \left(\frac{1}{\binom{m}{\leq r}}\right)^2$ and let $\mathrm{RM}_\delta(m,r) = \{P \in \mathrm{RM}(m,r) \colon |\,\mathrm{wt}(P) - 1/2| \geq \delta/2\}$, i.e., the set of polynomials with bias at least δ.

Now,

$$\begin{aligned}
&\sum_{P \in \mathrm{RM}_\delta(m,r)} \exp_2\left(-\varepsilon u\right) \cdot (1-\mathrm{wt}(P))^{(1-\varepsilon)u} \\
&\quad \leq 2 \cdot \sum_{P\colon\ \mathrm{wt}(P) \leq 1/2 - \delta/2} \exp_2\left(-\varepsilon u\right) \cdot (1-\mathrm{wt}(P))^{(1-\varepsilon)u},
\end{aligned}$$

since the term corresponding to a polynomial P of weight greater than $1/2 + \delta/2$ can be upper bounded by the term corresponding to the polynomial $1 + P$ which has weight at most $1/2 - \delta/2$.

Since the RHS of the above equation involves polynomials whose weights are in the interval $[1/2^r, 1/2 - \delta]$, we will split this interval into sub-intervals and analyse the contribution from each.

$$\begin{aligned} \mathrm{Low}_i &:= [1/2^{i+1}, 1/2^i] && \text{for } i = 2, 3, \ldots, r-1, \\ \mathrm{Med}_i &:= [(1/2 - 1/2^i), (1/2 - 1/2^{i+1})] && \text{for } i = 2, 3, \ldots, t = \log \frac{1}{\delta}. \end{aligned}$$

Let us use the following quantities to denote the number of polynomials with weights in the above intervals:

$$\begin{aligned} L_i &:= |\{P \in \mathrm{RM}(m,r) \ : \ \mathrm{wt}(P) \in \mathrm{Low}_i\}| \\ M_i &:= |\{P \in \mathrm{RM}(m,r) \ : \ \mathrm{wt}(P) \in \mathrm{Med}_i\}| \,. \end{aligned}$$

Therefore,

$$\sum_{P \in \mathrm{RM}_\delta(m,r)} \exp_2(-\varepsilon u) \cdot (1 - \mathrm{wt}(P))^{(1-\varepsilon)u} \le 2 \cdot \sum_{P:\ \mathrm{wt}(P) \le 1/2 - \delta/2} \exp_2(-\varepsilon u) \cdot (1 - \mathrm{wt}(P))^{(1-\varepsilon)u}$$

$$\le 2 \left(\sum_{i=2}^{r-1} L_i \cdot (1 - 1/2^{i+1})^k + \sum_{i=2}^{t} M_i \cdot (1/2 + 1/2^i)^k \right) \cdot \frac{1}{\exp_2\left(\binom{m}{\le r} - K\right)}.$$

By Claim 3.6 proved below using the weight distribution bounds of Sberlo and Shpilka (Theorem 2.1 and Theorem 2.2), we have that

$$\left(\sum_{i=2}^{r-1} L_i \cdot (1 - 1/2^{i+1})^k + \sum_{i=2}^{t} M_i \cdot (1/2 + 1/2^i)^k \right) \cdot \frac{1}{\exp_2\left(\binom{m}{\le r} - K\right)} \le O(\delta^2).$$

Hence,

$$\begin{aligned} &\sum_{P \in \mathrm{RM}(m,r)} \exp_2(-\varepsilon u) \cdot (1 - \mathrm{wt}(P))^{(1-\varepsilon)u} \\ &\quad = \sum_{P \in \mathrm{RM}_\delta(m,r)} \exp_2(-\varepsilon u) \cdot (1 - \mathrm{wt}(P))^{(1-\varepsilon)u} \\ &\qquad + \sum_{P \notin \mathrm{RM}_\delta(m,r)} \exp_2(-\varepsilon u) \cdot (1 - \mathrm{wt}(P))^{(1-\varepsilon)u} \\ &\quad \le O(\delta^2) + \sum_{P \notin \mathrm{RM}_\delta(m,r)} \exp_2(-\varepsilon u) \cdot (1/2 + \delta/2)^{(1-\varepsilon)u} \\ &\quad \le O(\delta^2) + (1+\delta)^{(1-\varepsilon)u} \\ &\quad \le O(\delta^2) + \exp(\delta u) \\ &\quad \le 1 + O(1/u) = 1 + o(1). \end{aligned}$$

◁

It remains to prove the following technical claim.

▷ Claim 3.6. **1.** For all $i = 2, \ldots, r-1$, we have

$$\frac{L_i \cdot (1 - 1/2^{i+1})^k}{\exp_2\left(\binom{m}{\le r} - k\right)} \le \delta^3.$$

2. For all $i = 2, \ldots, t = \log \frac{1}{\delta}$, we have

$$\frac{M_i \cdot (1/2 + 1/2^i)^k}{\exp_2\left(\binom{m}{\leq r} - k\right)} \leq \delta^3.$$

Prooff of Claim 3.6(1). Note that $(1 - 1/2^{i+1})^k \leq \exp(-k/2^{i+1}) \leq \exp_2(-k/2^{i+1})$. Hence,

$$\frac{L_i \cdot (1 - 1/2^{i+1})^k}{\exp_2\left(\binom{m}{\leq r} - k\right)} \leq \frac{\mathrm{WtDist}_{m,r}(2^{-i}) \cdot \exp_2(-k/2^{i+1})}{\exp_2\left(\binom{m}{\leq r} - k\right)}.$$

Using Theorem 2.1 to bound $\mathrm{WtDist}_{m,r}(2^{-i})$, we get

$$\begin{aligned}\frac{L_i \cdot (1 - 1/2^{i+1})^k}{\exp_2\left(\binom{m}{\leq r} - k\right)} &\leq \exp_2\left(O(m^4) + O(i\gamma^{i-1}) \cdot \binom{m}{\leq r} - k/2^{i+1} - \binom{m}{\leq r} + k\right) \\ &= \exp_2\left(O(m^4) - \binom{m}{\leq r} \cdot (1 - O(i\gamma^{i-1})) + k \cdot (1 - 2^{-i-1})\right),\end{aligned}$$

where $\gamma = r/m$. If this γ is a small enough absolute constant, we can see that the $O(i\gamma^{i-1})$ term is always at most 2^{-i-1}. Using this and continuing the computation above, we get

$$\begin{aligned}\frac{L_i \cdot (1 - 1/2^{i+1})^k}{\exp_2\left(\binom{m}{\leq r} - k\right)} &\leq \exp_2\left(O(m^4) - \left(\binom{m}{\leq r} - k\right) \cdot (1 - 2^{-i-1})\right) \\ &\leq \exp_2\left(O(m^4) - \varepsilon\binom{m}{\leq r} \cdot (1 - 2^{-i-1})\right) \\ &\leq \exp_2\left(O(m^4) - \frac{\varepsilon}{2} \cdot \binom{m}{\leq r}\right) \leq \exp_2\left(-\Omega\left(\binom{m}{\leq r}\right)\right) \leq \delta^3\end{aligned}$$

where for the second inequality above, we used the fact that $k \leq K = (1 - \varepsilon) \cdot \binom{m}{\leq r}$. ◁

Proof of Claim 3.6(2). Note that

$$\begin{aligned}(1/2 + 1/2^i)^k &= 2^{-k} \cdot (1 + 1/2^{i-1})^k \\ &\leq 2^{-k} \cdot \exp(k/2^{i-1}) = 2^{-k} \cdot \exp_2(k/2^{i-1} \cdot \log_2 e) \\ &= \exp_2(-k \cdot (1 - 2\log_2 e \cdot 2^{-i})).\end{aligned}$$

Using the fact that $M_i \leq \mathrm{WtDist}_{m,r}(1/2 - 1/2^{i+1})$, we get

$$\begin{aligned}&\frac{\mathrm{WtDist}_{m,r}(1/2 - 1/2^{i+1}) \cdot \exp_2(-k \cdot (1 - 2\log_2 e \cdot 2^{-i}))}{\exp_2\left(\binom{m}{\leq r} - k\right)} \\ \leq &\frac{\exp_2\left(O(m^4) + (1 - 2^{-O(\max\{\gamma^2 i, \gamma\})}) \cdot \binom{m}{\leq r} - k \cdot (1 - 2\log_2 e \cdot 2^{-i})\right)}{\exp_2\left(\binom{m}{\leq r} - k\right)} \\ = &\exp_2\left(O(m^4) - \frac{1}{2^{O(\max\{\gamma^2 i, \gamma\})}} \cdot \binom{m}{\leq r} + k \cdot \frac{1}{2^{-(i-1-\log_2\log_2 e)}}\right)\end{aligned}$$

where $\gamma = r/m$, and we used Theorem 2.2 for the second inequality.

Note that as long as γ is a small enough absolute constant, the $O(\max\{\gamma^2 i, \gamma\})$ term above is at most $i - 1 - \log_2 \log_2 e$ for each $i \in \{2, \ldots, t\}$. Using this bound, we continue the above computation as follows.

$$\begin{aligned}
\frac{M_i \cdot (1/2 + 1/2^i)^k}{\exp_2\left(\binom{m}{\leq r} - k\right)} &\leq \exp_2\left(O(m^4) - 2^{-O(\max\{\gamma^2 i, \gamma\})} \cdot \left(\binom{m}{\leq r} - k\right)\right) \\
&\leq \exp_2\left(O(m^4) - 2^{-O(\max\{\gamma^2 i, \gamma\})} \cdot \left(\binom{m}{\leq r} - K\right)\right) \\
&= \exp_2\left(O(m^4) - 2^{-O(\max\{\gamma^2 i, \gamma\})} \cdot \varepsilon \cdot \binom{m}{\leq r}\right) \\
&\leq \exp_2\left(O(m^4) - \Omega\left(\sqrt{\binom{m}{\leq r}}\right)\right) \leq \delta^3,
\end{aligned}$$

where for the last inequality we used the fact that $\binom{m}{\leq r} \geq 2^{\Omega(\gamma \log(1/\gamma) m)}$ and hence for small enough absolute constant γ, we have

$$\begin{aligned}
2^{-O(\max\{\gamma^2 i, \gamma\})} \cdot \binom{m}{\leq r} \geq 2^{-O(\gamma^2 t)} \cdot \binom{m}{\leq r} &\geq \exp_2(\Omega(\gamma \log(1/\gamma) m) - O(\gamma^2 \log(1/\delta))) \\
&= \exp_2(\Omega(\gamma \log(1/\gamma) m) - O(\gamma^3 \log(1/\gamma) m)) = \exp_2(\Omega(m)). \quad \lhd
\end{aligned}$$

This completes the proof of Lemma 3.4.

4 Resilience of Reed-Muller codes under erasures and errors

For parameters m, r, recall that $E(m, r)$ is the $\binom{m}{\leq r} \times 2^m$-matrix where the columns are indexed by elements in $\mathbb{F}_2^m$, and the z-th column is the column vector consisting of *evaluations* of all multilinear m-variate degree $\leq r$ monomials on z. It is well-known that the matrix $E(m, r)$ is the generator matrix of the $\mathrm{RM}(m, r)$ code, and is also the parity-check matrix of the $\mathrm{RM}(m, m - r - 1)$ code.

Theorem 1.1 can be re-interpreted as a statement about random sub-matrices of $E(m, r)$ in the regime where $r \leq \gamma m$ for a small absolute constant $\gamma > 0$.

▶ **Corollary 4.1** (rank of random sub-matrices of $E(m, r)$)**.** *Let $\gamma > 0$ be an absolute constant and m, r growing parameters with $r \leq \gamma m$. Then, for any constant $\varepsilon > 0$, a random set of $K = (1 - \varepsilon)\binom{m}{\leq r}$ columns of $E(m, r)$ are linearly independent with probability $1 - o(1)$.*

Proof. A set of columns of $E(m, r)$ indexed by $Z = \{z_1, \ldots, z_K\}$ are linearly dependent if and only if $\mathbb{I}_r(Z)$ has dimension strictly larger than $\binom{m}{\leq r} - K$. Theorem 1.1 asserts that this happens with only $o(1)$ probability. ◀

4.1 Channels under consideration

The above corollary can be used to talk about the resilience of Reed-Muller codes under the *erasure* and *error* channels. We first define the precise definitions to be able to state the results accurately.

▶ **Definition 4.2** (binary erasure channel (BEC))**.** *The* binary erasure channel *with parameter α, denoted by BEC_α, is the channel with input alphabet $\{0, 1\}$ where the each binary symbol is "erased" (replaced by the '?' symbol) independently with probability α.*

A closely related model is where we have a fixed number of random erasures instead of each coordinate being erased with a fixed probability. We will refer to this as the BEC* model although this isn't a channel in the traditional sense of altering each coordinate independently.

▶ **Definition 4.3** (capped binary erasure channel (BEC*))**.** *The* capped binary erasure channel *with parameter K, denoted by* BEC^*_K*, refers to the channel with input alphabet* $\{0,1\}^n$ *that replaces a random subset of at most K of the coordinates by the '?' symbol.*

The Binary Symmetric Channel deals with *errors* or *corruptions* as opposed to erasures in the Binary Erasure Channel.

▶ **Definition 4.4** (binary symmetric channel (BSC))**.** *The* binary symmetric channel *with parameter α, is the channel where the each binary symbol is "flipped" (that is,* 0 *changed to* 1 *and vice-versa) independently with probability α.*

In similar spirit to Definition 4.3, we define the capped binary symmetric channel.

▶ **Definition 4.5** (capped binary symmetric channel (BSC*))**.** *The* capped binary symmetric channel *with parameter K, denoted by* BSC^*_K*, refers to the channel with input alphabet* $\{0,1\}^n$ *that "flips" a random subset of at most K of the coordinates.*

In most cases, resilience with respect to BEC_α (or BSC_α) is the same as resilience with respect to BEC^*_K (or BSC^*_K)for $K = \alpha n$, by standard concentration inequalities but this might require some subtlety when dealing with codes of rate very close to zero or close to one. For a concrete example, the code $\mathrm{RM}(m, m-1)$ has rate $R = 1 - 1/2^m$ and is *not* resilient under BEC_α for $\alpha = (1-R)/2 = 1/2^{m+1}$ (with constant probability we may have two coordinates erased, and this is not recoverable), but is vacuously resilient under BEC^*_K for $K = \alpha 2^m$. To avoid these nuances, we will *only* be dealing with the setting of capped channels although this does not make much difference with most of our range of parameters.

Notion of "capacity achieving" codes with respect to BEC_α

Shannon's seminar work [14] showed that, for any constant rate, the supremum of rates of random codes that is resilient to BEC_α is exactly $R = 1-\alpha$. That is, for any $\varepsilon > 0$, there are codes of rate $1 - \alpha - \varepsilon$ that can decode from BEC_α with decoding error $1 - o(1)$. However, when the rate is very close to zero or one, the situation becomes more nuanced due to the asymptotics involved.

Dealing specifically with Reed-Muller codes, let us consider the code $\mathrm{RM}(m, m-r-1)$ with $r \le m/2$. This has rate $R = 1 - \frac{\binom{m}{\le r}}{2^m}$ and can recover from a maximum of $\binom{m}{\le r}$ errors. Abbe, Shpilka and Wigderson [1] defined the notion of "*capacity achieving under* BEC" to mean that the code can decode (with $1 - o(1)$ probability) from $(1-\varepsilon)\binom{m}{\le r}$ erasures for any constant $\varepsilon > 0$. We refer the reader to [1, Section 1] and [13, Section 2.2] for a nuanced discussion on this.

4.2 Reed-Muller codes under BEC*

An immediate consequence of the above corollary is that $\mathrm{RM}(m, d)$ with $d \ge m(1-\gamma)$ for a small but absolute constant $\gamma > 0$ *achieves capacity* (as defined above) under the Binary Erasure Channel.

▶ **Theorem 1.2** (high-degree RM codes under erasures). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let m, d be growing parameters with $d > m(1-\gamma_0)$. Then, the code $\mathrm{RM}(m,d)$ can correct $K = (1-\varepsilon)\binom{m}{\leq m-d-1}$ random errors with probability $1 - o(1)$.*

Proof. The code $\mathrm{RM}(m,d)$ can recover from erasures on coordinates indexed by $Z \subseteq \mathbb{F}_2^m$ if and only if the corresponding columns of the parity check matrix are linearly independent. Since the parity-check matrix of $\mathrm{RM}(m,d)$ is $E(m,r)$ for $r = m-d-1$, we have from Corollary 4.1 that a random set of K columns are linearly independent with probability $1 - o(1)$. ◀

4.3 Reed-Muller codes under BSC*

The following theorem of Abbe, Shpilka and Wigderson [1] provides a method to derive the resilience of certain Reed-Muller codes under the BSC by using the resilience of appropriate Reed-Muller codes under the BEC. Subsequent work of Saptharishi, Shpilka and Volk also showed that these codes also have efficient decoding procedures to recover from random errors.

▶ **Theorem 4.6** ([1, 12]). *For any growing parameters $m, r > 0$, if the code $\mathrm{RM}(m, m-2r-2)$ can recover from a subset $Z \subseteq [2^m]$ of erasures, then the code $\mathrm{RM}(m, m-r)$ can efficiently recover from errors on the subset Z.*

In particular, if the code $\mathrm{RM}(m, m-2r-2)$ can recover K random erasures with probability $1 - o(1)$, then the code $\mathrm{RM}(m, m-r)$ can efficiently decode from K random errors.

Applying the above theorem to Theorem 1.2 yields the following corollary.

▶ **Corollary 1.3** (high-degree RM codes under errors). *There exists a constant $\gamma_0 > 0$ such that for all $\varepsilon > 0$ the following is true. Let m, r be growing parameters with $r < \gamma_0 m$. Then, the code $\mathrm{RM}(m, m-2r-2)$ can be efficiently decode from $K = (1-\varepsilon)\binom{m}{\leq r}$ random errors.*

▶ **Corollary 4.7.** *There exists a constant $\gamma_0 > 0$ such that for all $\gamma < \gamma_0$ and $\varepsilon > 0$ the following is true. Let m, r be growing parameters with $r = \gamma m$. Then, the code $\mathrm{RM}(m, m-2r-2)$ can be efficiently decode from $K = (1-\varepsilon)\binom{m}{\leq r}$ random errors.* ◀

It is worth noting that the minimum distance of the code $\mathrm{RM}(m, m-2r-2)$ is merely $2^{2r+2} \ll \binom{m}{\leq r}$ when $r = \gamma m$ for a small enough $\gamma > 0$. Thus, the above corollary shows that high-degree (or high-rate) Reed-Muller codes are resilient to random errors well beyond their minimum distance, and efficiently so.

References

1 Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed-Muller codes for random erasures and errors. *IEEE Trans. Inform. Theory*, 61(10):5229–5252, 2015. (Preliminary version in *47th STOC*, 2015). doi:10.1109/TIT.2015.2462817.

2 Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low-degree polynomials are hard to approximate. *Comput. Complexity*, 21(1):63–81, 2012. (Preliminary version in *13th RANDOM*, 2009). eccc:2008/TR08-080. doi:10.1007/s00037-011-0020-6.

3 Aidao Chen, Anindya De, and Aravindan Vijayaraghavan. Learning a mixture of two subspaces over finite fields. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Algorithmic Learning Theory (ALT)*, volume 132 of *Proceedings of Machine Learning Research*, pages 481–504, 2021. arXiv:2010.02841.

4 Tali Kaufman, Shachar Lovett, and Ely Porat. Weight distribution and list-decoding size of Reed-Muller codes. *IEEE Trans. Inform. Theory*, 58(5):2689–2696, 2012. (Preliminary version in *1st ICS*, 2010). doi:10.1109/TIT.2012.2184841.

5 Peter Keevash and Benny Sudakov. Set systems with restricted cross-intersections and the minimum rank of inclusion matrices. *SIAM J. Discrete Math.*, 18(4):713–727, 2005. doi:10.1137/S0895480103434634.

6 Shrinivas Kudekar, Santhosh Kumar, Marco Mondelli, Henry D. Pfister, Eren Eren Şaşoğlu, and Rüdiger L. Urbanke. Reed-Muller codes achieve capacity on erasure channels. *IEEE Trans. Inform. Theory*, 63(7):4298–4316, 2017. (Preliminary version in *48th STOC*, 2016). doi:10.1109/TIT.2017.2673829.

7 David E. Muller. Application of Boolean algebra to switching circuit design and to error detection. *Trans. IRE Prof. Group Electron. Comput.*, 3(3):6–12, 1954. doi:10.1109/IREPGELC.1954.6499441.

8 Zipei Nie and Anthony Y. Wang. Hilbert functions and the finite degree Zariski closure in finite field combinatorial geometry. *J. Comb. Theory, Ser. A*, 134:196–220, 2015. doi:10.1016/j.jcta.2015.03.011.

9 Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *Trans. IRE Prof. Group Inf. Theory*, 4:38–49, 1954. doi:10.1109/TIT.1954.1057465.

10 Galen Reeves and Henry D. Pfister. Reed-Muller codes achieve capacity on BMS channels. (manuscript), 2021. arXiv:2110.14631.

11 Alex Samorodnitsky. An upper bound on ℓ_q norms of noisy functions. *IEEE Trans. Inform. Theory*, 66(2):742–748, 2020. doi:10.1109/TIT.2019.2944698.

12 Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Efficiently decoding Reed-Muller codes from random errors. *IEEE Trans. Inform. Theory*, 63(4):1954–1960, 2017. (Preliminary version in *48th STOC*, 2016). doi:10.1109/TIT.2017.2671410.

13 Ori Sberlo and Amir Shpilka. On the performance of Reed-Muller codes with respect to random errors and erasures. In Shuchi Chawla, editor, *Proc. 31st Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1357–1376, 2020. doi:10.1137/1.9781611975994.82.

14 Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

15 Victor K.-W. Wei. Generalized Hamming weights for linear codes. *IEEE Trans. Inform. Theory*, 37(5):1412–1418, 1991. doi:10.1109/18.133259.

On the Partial Derivative Method Applied to Lopsided Set-Multilinear Polynomials

Nutan Limaye ✉
Computer Science Department, IT University of Copenhagen, Denmark

Srikanth Srinivasan ✉
Department of Computer Science, Aarhus University, Denmark
On leave from Department of Mathematics, IIT Bombay, India

Sébastien Tavenas ✉
Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, France

Abstract

We make progress on understanding a lower bound technique that was recently used by the authors to prove the first superpolynomial constant-depth circuit lower bounds against algebraic circuits.

More specifically, our previous work applied the well-known partial derivative method in a new setting, that of *lopsided* set-multilinear polynomials. A set-multilinear polynomial $P \in \mathbb{F}[X_1, \ldots, X_d]$ (for disjoint sets of variables $X_1, \ldots, X_d$) is a linear combination of monomials, each of which contains one variable from $X_1, \ldots, X_d$. A lopsided space of set-multilinear polynomials is one where the sets $X_1, \ldots, X_d$ are allowed to have different sizes (we use the adjective "lopsided" to stress this feature). By choosing a suitable lopsided space of polynomials, and using a suitable version of the partial-derivative method for proving lower bounds, we were able to prove constant-depth superpolynomial set-multilinear formula lower bounds even for very low-degree polynomials (as long as d is a growing function of the number of variables N). This in turn implied lower bounds against general formulas of constant-depth.

A priori, there is nothing stopping these techniques from giving us lower bounds against algebraic formulas of *any* depth. We investigate the extent to which this lower bound can extend to greater depths. We prove the following results.

1. We observe that our choice of the lopsided space and the kind of partial-derivative method used can be modeled as the choice of a multiset $W \subseteq [-1, 1]$ of size d. Our first result completely characterizes, for any product-depth Δ, the best lower bound we can prove for set-multilinear formulas of product-depth Δ in terms of some combinatorial properties of W, that we call the *depth-Δ tree bias* of W.
2. We show that the maximum depth-3 tree bias, over multisets W of size d, is $\Theta(d^{1/4})$. This shows a stronger formula lower bound of $N^{\Omega(d^{1/4})}$ for set-multilinear formulas of product-depth 3, and also puts a non-trivial constraint on the best lower bounds we can hope to prove at this depth in this framework (a priori, we could have hoped to prove a lower bound of $N^{\Omega(\Delta d^{1/\Delta})}$ at product-depth Δ).
3. Finally, we show that for small Δ, our proof technique cannot hope to prove lower bounds of the form $N^{\Omega(d^{1/\operatorname{poly}(\Delta)})}$. This seems to strongly hint that new ideas will be required to prove lower bounds for formulas of unbounded depth.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Partial Derivative Method, Barriers to Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.32

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/090/

Funding *Srikanth Srinivasan*: Supported by Startup grant from Aarhus University.

Acknowledgements We would like to thank the anonymous reviewers of the paper for their comments which helped us improve the presentation in the paper. We would also like to thank Niranjan Balachandran for his comments and an alternate proof of a combinatorial lemma in the paper, and Swastik Kopparty and Shachar Lovett for discussions.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 32; pp. 32:1–32:23

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

1 Introduction and Motivation

Basic background

This paper is motivated by questions arising in the area of *Algebraic Circuit complexity,* which studies the computational complexity of problems defined by families of multivariate polynomials. Given an infinite family of polynomials $(P_N(x_1, \ldots, x_N))_{N \geqslant 1}$ over a field $\mathbb{F}$, we consider the computational problem of evaluating P_N at input point $a \in \mathbb{F}^N$. Many natural and important computational problems can be stated in this language, including the problems of computing the determinant and the permanent, and that of multiplying matrices.

Algebraic circuits are succinct representations of multivariate polynomials that allow us to solve computational problems of the above form. More precisely, an algebraic circuit is a directed acyclic graph, where the sources are labelled by variables $x_1, \ldots, x_N$ or field elements and internal nodes (or *gates*) by algebraic operations $+$ and $\times$. Each internal node thus represents a polynomial in the variables $x_1, \ldots, x_N$ and a designated output gate represents the polynomial computed by the algebraic circuit. The *size* of the algebraic circuit is given by the number of gates. The *depth* and *product-depth* of an algebraic circuit denote the maximum number of gates and $\times$-gates respectively, on a directed path in the circuit.[1] Finally, we call an algebraic circuit an *Algebraic formula* if the underlying directed graph is a tree. (Equivalently, an Algebraic formula is just a nested algebraic expression made up of additions and multiplications, as one might write down on paper, represented in the form of a tree.)

An algebraic circuit for a polynomial P allows us to evaluate the polynomial P on a given input in time polynomially related to the size of the circuit. Thus, algebraic circuits are a restricted, but natural, model of computation for computational problems of this form. The study of this model of computation is one of the principal topics of study in Algebraic circuit complexity, and has received much attention over the past four decades (see e.g. [3, 26, 24] for nice introductions). Many central questions in Boolean circuit complexity have analogous and closely-related versions in the algebraic setting. For instance, the VP vs. VNP question [28], which is the problem of proving explicit lower bounds against algebraic circuits, is formally easier than the (non-uniform) P vs. NP question [2]. The problem of proving lower bounds against algebraic formulas is similarly closely related to the problem of proving lower bounds against the Boolean complexity class NC^1.

A recent result [18]

While circuit lower bounds in the algebraic setting are formally easier than the Boolean setting, they still have been hard to come by. For example, a famous line of research in the 1980s [1, 7, 13, 23, 27] showed exponential lower bounds against Boolean circuits of constant-depth, but did not yield such results for algebraic circuits.[2] This situation was somewhat rectified recently by the authors [18], building on some important earlier results in the area [20, 21]. In particular, we were able to prove superpolynomial lower bounds against constant-depth algebraic circuits over fields of characteristic zero.

---

[1] W.l.o.g., we may assume that the product-depth and depth of a circuit are related to each other by a multiplicative factor of 2. However, some results are easier to state in terms of product-depth.

[2] Note that algebraic circuit lower bounds are not necessarily easier than Boolean circuit lower bounds in the constant-depth setting. However, some of these ideas did translate in the setting of constant-sized fields. [8, 9]

This paper is motivated by the problem of extending this lower bound to stronger models of computation. At a high level, our results are as follows.

- We show that our previous result [18] can be formulated purely in terms of a combinatorial property of the space of polynomials under consideration.
- We characterize the best lower bound that can be achieved in this framework at product-depth 3. It is better than the analogous lower bound from [18], but not as good as one might hope at first sight (as explained below).
- We place limitations on how well the bound extends to higher depths.

To describe these results in detail, we first recall the outline of the proof of [18].

The proof of [18]

The proof of [18] proceeds in two steps. In the first step, we reduce the problem of proving lower bounds for general circuits of depth Δ to proving lower bounds for *product-depth-*$(\Delta-1)$ circuits that have a special structure. In the second step, we prove lower bounds for the structured circuits. We describe these steps in some more detail next.

Step 1: Set Multilinearization. We work throughout with a partition of the variable set $X = \{x_1, \ldots, x_N\}$ into $X_1 \cup X_2 \cup \cdots \cup X_d$. Given such a partition, a *set-multilinear* monomial w.r.t. this variable partition is a monomial of degree d that contains exactly one variable from each of $X_1, X_2, \ldots, X_d$. A set-multilinear polynomial P is a linear combination of set-multilinear monomials. We denote the space of set-multilinear polynomials w.r.t. $X_1, \ldots, X_d$ by $\mathbb{F}_{\text{sm}}[X_1, \ldots, X_d]$. A set-multilinear circuit or formula is one where each gate computes a set-multilinear polynomial w.r.t. a subset of $\{X_1, \ldots, X_d\}$. An important example of a set-multilinear polynomial is the *Iterated Matrix Multiplication* polynomial $\text{IMM}_{n,d}$, where $X_1, \ldots, X_d$ are square matrices of dimension $n \times n$ with distinct indeterminates, and the polynomial represents, say, the $(1,1)$th entry of the product of these matrices.

In the first step of the proof, we show that if a polynomial $P \in \mathbb{F}_{\text{sm}}[X_1, \ldots, X_d]$ has a circuit C of depth Δ and size s, then it also has a set-multilinear circuit C' of product-depth $\Delta - 1$ and size $s' = \text{poly}(s) \cdot d^{O(d)}$. Note that while the blow-up in size in going from C to C' is large as a function of d, it can be made small (say $\text{poly}(N)$) assuming that d is a slow-growing function of N (say, $d = O(\log N / \log\log N)$). So, to prove superpolynomial constant-depth circuit lower bounds, it suffices to prove superpolynomial lower bounds for constant-depth set-multilinear circuits in this *low-degree setting.*

Step 2: Set-multilinear lower bounds for low-degree polynomials. Lower bounds for constant-depth set-multilinear circuits have been known since the work of Nisan and Wigderson [20] from the 1990s. However, such lower bounds were typically of the form $\exp(d^{\Omega(1)}) \cdot \text{poly}(N)$, which are not good enough for our purposes in the low-degree setting. The main contribution of [18] was to prove a lower bound of the form $N^{\omega_d(1)}$, which yields a superpolynomial lower bound for any degree $d = d(N)$ which is a growing function of N.

Somewhat surprisingly, the proof of this latter lower bound used just the lower bound technique of Nisan and Wigderson [20], which goes by the name of the *partial derivative method.* The key observation was to apply this technique to a suitable space of set-multilinear polynomials. Specifically, it is crucial in the proof to allow for the sets $X_1, \ldots, X_d$ to have fairly different sizes. To stress this feature, we refer to such a space of set-multilinear polynomials as *lopsided.*

For such polynomials that have efficient small-depth set-multilinear formulas, we argue that certain matrices associated to these polynomials have low rank. This is the basic recipe suggested by the partial derivative method, and is described in more detail later.

To complete the argument, we need to find explicit polynomials for which the associated matrices have high (ideally maximal) rank. We do this by considering suitable restrictions of $\mathrm{IMM}_{n,d}$ where $n = \max_{i\in[d]} |X_i|$. Using this idea, we showed [18] a lower bound of $N^{d^{\exp(-O(\Delta))}}$ for set-multilinear circuits of product-depth Δ. In conjunction with Step 1, this implies a superpolynomial lower bound for constant-depth algebraic circuits, and in fact for circuits of depth $o(\log\log d)$.

The potential of this lower bound technique

Can the above proof strategy be used to prove lower bounds for stronger models of computation, such as algebraic formulas of unbounded depth or, optimistically, even algebraic circuits? It turns out that Step 1 of the strategy still works, as shown in previous work of Nisan and Wigderson [20] and Raz [22]. Consequently, proving superpolynomial set-multilinear lower bounds against these models in the low-degree setting imply general circuit or formula lower bounds.

However, a problem arises because of the technique used in Step 2. As $\mathrm{IMM}_{n,d}$ (or more precisely, its restrictions) is a polynomial of "maximal complexity" for the partial derivative method, we cannot use it to prove lower bounds for computational models that can compute this polynomial efficiently. In particular, this suggests a new idea is required to prove lower bounds for, say, set-multilinear circuits of depth $O(\log d)$, which can compute $\mathrm{IMM}_{n,d}$ efficiently.

Nevertheless, this does not seem to rule out lower bounds for circuits of depth $o(\log d)$, or for formulas (of any depth). A simple, folklore divide-and-conquer strategy shows that $\mathrm{IMM}_{n,d}$ has set-multilinear circuits of product-depth Δ and size $n^{O(d^{1/\Delta})}$, and also set-multilinear formulas of product-depth Δ and size $n^{O(\Delta d^{1/\Delta})}$. Given the fact that this basic bound has not been improved upon significantly[3] for a long time, it is tempting to conjecture that it is tight, at least in the set-multilinear setting. If so, it seems that we could hope to prove lower bounds for set-multilinear circuits of depth $o(\log d)$ and formulas of any depth. Doing this would yield at least lower bounds for general algebraic formulas, which would be a very interesting result. This brings us to our main motivating question.

▶ **Question 1.** *Can we hope to use the partial derivative method (as applied to lopsided spaces of set-multilinear polynomials) to prove set-multilinear lower bounds that match the standard divide and conquer algorithms for* $\mathrm{IMM}_{n,d}$*?*

Our results in this paper indicate that the answer to this question is probably "No", and that, alone, the proof technique from [18] is not powerfull enough to handle formulas of depth $(\log d)^{o(1)}$. In the process of proving these results, we also introduce what we believe is a clean framework for studying the power of this technique.

We start with a more formal description of the partial derivative method and then state our results.

---

[3] A famous result of Gupta, Kamath, Kayal and Saptharishi [12] does improve this bound, but gives up on set-multilinearity. Moreover, the basic form of the bound is still preserved. More precisely, their work implies circuits of product-depth Δ and size $n^{O(d^{1/2\Delta})}$.

The partial derivative method for lopsided set-multilinear polynomials

We prove lower bounds for set-multilinear polynomials $P(X_1, \ldots, X_d)$ where each $|X_i| = n^{\alpha_i}$ for some $\alpha_i \in (0, 1]$. Given such a polynomial P, we associate with it a matrix as follows. We partition $[d]$ into sets $\mathcal{P}$ and $\mathcal{N}$. The rows of the matrix are associated with set-multilinear monomials over the variable sets $\{X_i \ : \ i \in \mathcal{P}\}$, and the columns symmetrically with the set-multilinear monomials over $\{X_j \ : \ j \in \mathcal{N}\}$. Given a row labelled by monomial m_1 and a column labelled by monomial m_2, the corresponding entry in the matrix is the coefficient of the set-multilinear monomial $m_1 m_2$ in the polynomial P. We use the rank of this matrix (or, more precisely, how close it is to full-rank) to prove lower bounds on the algebraic circuit complexity of P.

We define this more precisely now. Note that the matrix is completely specified by the choice of the numbers $\alpha_1, \ldots, \alpha_d$ and the partition $[d] = \mathcal{P} \cup \mathcal{N}$. We can describe these together by the multiset $W \subseteq [-1, 1]$, defined by $W = \{\alpha_i \ : \ i \in \mathcal{P}\} \cup \{-\alpha_j \ : \ j \in \mathcal{N}\}$. Finally, we use $M_W(P)$ to denote the above matrix.

Note that $M_W(P)$ is a matrix with $R = n^{\sum_{\alpha \in W \cap (0,1]} \alpha}$ rows and $C = n^{\sum_{\alpha \in W \cap [-1,0)} |\alpha|}$ columns. In particular, the rank of the matrix $M_W(P)$ is bounded by the minimum of these quantities. We consider the *relative rank* of P, defined as follows.

$$\mathrm{relrk}_W(P) = \frac{\mathrm{rank}(M_W(P))}{\sqrt{RC}} = \frac{\mathrm{rank}(M_W(P))}{n^{\frac{1}{2}\sum_{\alpha \in W} |\alpha|}}. \tag{1}$$

Observe that the quantity in the denominator is the geometric mean of the number of rows and the number of columns of $M_W(P)$ and hence $\mathrm{relrk}_W(P) \in [0, 1]$. In fact, more generally, it is not hard to see that as $\mathrm{rank}(M_W(P)) \leqslant \min\{R, C\}$, we have $\mathrm{relrk}_W(P) \leqslant n^{-|\sum_{\alpha \in W} \alpha|/2}$.

Further, it was shown by the authors [18] that for any W, there is a polynomial P_0 such that $\mathrm{relrk}_W(P_0) = n^{-|\sum_{\alpha \in W} \alpha|/2}$ and P_0 can be obtained by starting with an instance of $\mathrm{IMM}_{\mathrm{poly}(n),d}$ and setting some variables to 0 and identifying variables within certain sub-matrices, i.e. by a *set-multilinear projection.*

High-level description of the results

Our results give a better understanding of what lower bounds the partial derivative method can hope to show in this setting.

- Our first main result is a transformation of our problem to a combinatorial problem about labelled trees. More precisely, we show that understanding the best lower bound our techniques can hope to prove in the low-degree setting is perfectly captured by the best-possible "tree-like decomposition" of the set W.[4]
  While this transformation is simple, it is conceptually clean, and simplifies the problem in multiple ways. Firstly, it eliminates the parameter n (which is roughly the number of variables in the underlying polynomial) and makes completely clear the dependence of the lower bound on properties of the multiset W. Secondly, this reformulation of the problem completely eliminates any mention of polynomials or algebra from the problem. It is now purely a problem about the "additive structure" of W.
- Our second result uses the above characterization of the problem to give a near-perfect understanding of the best lower bounds we can prove for set-multilinear formulas of product-depth 3 (i.e. $\Sigma\Pi\Sigma\Pi\Sigma\Pi\Sigma$ formulas). More precisely, we show that the best product-depth-3 lower bound we can prove via our proof technique is $n^{\Theta(d^{1/4})}$. This is interesting for the following two different reasons.

[4] This is not to be confused with standard tree decompositions of graphs, which have no connection with objects studied here.

For one, this is a stronger lower bound than known previously for set-multilinear formulas of product-depth 3 in the low-degree regime: Nisan and Wigderson [20] showed a lower bound of $\exp(\Omega(d^{1/3})) \cdot \mathrm{poly}(N)$ (which does not yield anything for $d = O(\log N)$), while in our earlier work [18], we showed lower bounds of $n^{\Omega(d^{1/7})}$.
On the other hand, the result also implies that this technique does not go as far as we would like. Recall from above that the (suspected) optimal lower bound for $\mathrm{IMM}_{n,d}$ at product-depth 3 is $n^{\Omega(d^{1/3})}$. So, our result implies that this technique cannot be used to obtain this bound at product-depth 3.

- The above results already indicate that we are not able to prove the best possible lower bound we could hope for product-depth-3 set-multilinear formulas. However, it is still conceivable that we can hope to prove a lower bound which stays "close" to the right expected bound for $\mathrm{IMM}_{n,d}$ (say a bound of the form $n^{\Delta d^{\Omega(1/\Delta)}}$), which could as yet lead to superpolynomial formula lower bounds.
  In our third result, we give strong indication that this is not the case, by showing that this technique cannot prove lower bounds of the form $n^{d^{1/\Gamma(\Delta)}}$ for a quasipolynomial function $\Gamma(\cdot)$, and small enough Δ.

1.1 Formal description of the results

To describe the results formally, we introduce a combinatorial measure of the complexity of the multiset $W \subseteq [-1, 1]$. In the low-degree setting, this will characterize the best lower bound we can prove via our lower bound technique.

Notation

Let $W \subseteq \mathbb{R}$ be a multiset. Throughout $|W|$ denotes the size of the multiset (i.e. counted with multiplicity) and $\mathrm{Sum}(W)$ denote the sum of its elements. Finally, $\|W\|_1$ denotes the L_1-norm of W (i.e. the sum of the absolute values of the elements of W).

▶ **Definition 2** (W-trees, path bias, tree bias)**.** *Let $W = \{\alpha_1, \ldots, \alpha_d\}$ be a multiset contained in $[-1, 1]$. A W-*tree *T, or equivalently* a tree T for W*, is a rooted, directed tree*[5] *with $d = |W|$ leaves which are labelled by distinct elements of the form (i, α_i) ($i \in [d]$).*[6] *Any vertex v of T thus corresponds to a subset W_v of W (corresponding to the leaves of the subtree induced by v) and we define* $\mathrm{Sum}(v)$ *to be* $\mathrm{Sum}(W_v)$*.*

An internal path *π in T is a path from the root to an internal (i.e. non-leaf) node. Given such an internal path π, we define the set of* Off-path nodes *of π, denoted* $\mathrm{Offpath}(\pi)$ *to be the set of nodes v of the tree T that are not on the path π, but have a parent on the path π. We define* the bias of the path π*, denoted* $\mathrm{bias}(\pi) = \left(\sum_{v \in \mathrm{Offpath}(\pi)} |\mathrm{Sum}(v)|\right) - |\mathrm{Sum}(r)|$ *where r is the root of T.*

(It is easy to check that if π is any internal path, then $W = W_r$ is the disjoint union of W_v ($v \in \mathrm{Offpath}(\pi)$). Hence, by the triangle inequality, we have $|\mathrm{Sum}(r)| \leqslant \sum_{v \in \mathrm{Offpath}(\pi)} |\mathrm{Sum}(v)|$. Thus, $\mathrm{bias}(\pi) \geqslant 0$ for any internal path π.)

Finally, we define the path bias of T w.r.t. W*, denoted* $\mathrm{Pathbias}_W(T)$*, to be the* maximum *bias of any internal path of T. If the tree T has depth 0 (i.e. it consists of just the root node), then we define the path bias of T w.r.t. W to be 0.*

---

[5] The edges are directed away from the root.

[6] We require the label to be a pair here as W is a multiset where elements may repeat. If the elements of W are all distinct, then we can think of the labels as simply elements of W.

With the above notation in place, we can define the combinatorial measure mentioned above. We define the depth-Δ tree bias of W *to be the* minimum *path bias of any depth-Δ W-tree T. We denote this quantity by* $\mathrm{Treebias}_\Delta(W)$.

Our first theorem relates the depth-Δ tree bias of $W = \{\alpha_1, \ldots, \alpha_d\} \subseteq [-1, 1]$ with the best lower bound we can prove using the complexity measure $\mathrm{relrk}_W(\cdot)$.

▶ **Theorem 3** (Connecting tree bias with relative rank). *Let n, d be positive integer parameters.*[7] *Let $\Delta \geqslant 1$ be any integer. Assume $W \subseteq [-1, 1]$ is a multiset of size d such that $\mathrm{Treebias}_\Delta(W) = t$. Then, for any set-multilinear formula F of product-depth at most Δ and size at most s, we have $\mathrm{relrk}_W(F) \leqslant (d^{3d} \cdot s \cdot n^{-t/2}) \cdot n^{-|\mathrm{Sum}(W)|/2}$. Conversely, for any n and d, there is a set-multilinear formula F with at most $3^d n^{t/2}$ leaves and of product-depth Δ such that $\mathrm{relrk}_W(F) \geqslant 2^{-d} \cdot n^{-|\mathrm{Sum}(W)|/2}$.*

This theorem is the consequence of Lemmas 13 and 14 and will be proved in Section 3. As already noted above, for any polynomial $P \in \mathbb{F}_{sm}[X_1, \ldots, X_d]$ (with $|X_i| = n^{|\alpha_i|}$ for each $i \in [d]$), we have $\mathrm{relrk}_W(P) \leqslant n^{-|\mathrm{Sum}(W)|/2}$. Theorem 3 shows that this maximum possible relative rank can be achieved by product-depth-Δ formulas of size $n^{O(t)}$, but not those of size $n^{o(t)}$, where $t = \mathrm{Treebias}_\Delta(T)$. This means that the best lower bound we can hope to prove via this technique is $n^{\Theta(t)}$.

The next couple of theorems give an understanding of the maximum possible tree bias for various depths Δ. The first result gives tight bounds on the maximum possible tree bias of a given multiset W for depth 3 (Section 4 will be dedicated to this result).

▶ **Theorem 4** (Tight bounds on tree bias for depth 3). *Let d be a growing integer parameter. Then, $\max_W \mathrm{Treebias}_3(W) = \Theta(d^{1/4})$, where W ranges over multisets from $[-1, 1]$ of size d in the expression above.*

The second result (proved in the long version of the paper) gives an asymptotic bound for larger depths (as long as Δ is bounded by a small function of d).

▶ **Theorem 5** (Bounds on tree bias for larger depths). *Let d, Δ be growing integer parameters with $\Delta = 2^{o(\sqrt{\log \log d})}$. Then, we have $\max_W \mathrm{Treebias}_\Delta(W) \leqslant d^{1/\Delta^{\Omega(\log \Delta)}}$, where W ranges over multisets from $[-1, 1]$ of size d.*

1.2 Proof Outline

Throughout this section, we work with a multiset $W = \{\alpha_1, \ldots, \alpha_d\} \subseteq [-1, 1]$ and a space of lopsided set-multilinear polynomials $\mathbb{F}_{\mathrm{sm}}[X_1, \ldots, X_d]$ where $|X_i| = n^{|\alpha_i|}$. Recall also that we are working in the low-degree setting, i.e. d is a slow-growing function of n. All formulas in this section should be assumed to be set-multilinear.

Motivation for tree bias

We start by motivating the notion of tree bias which, at first sight, might appear mysterious to the reader. In fact, this notion comes up quite naturally in the course of constructing small set-multilinear formulas that have large relative rank. These constructions, in turn, are motivated by the following basic properties of relative rank which are all slight modifications of standard facts used in the literature. In this form they can be found in our earlier work [19].[8]

[7] We think of d as a slow-growing function of n.

[8] The paper deals with a related notion of relative rank w.r.t. *ordered* W (or equivalently, W is replaced by a tuple $(\alpha_1, \ldots, \alpha_d)$). However, the proof works in the same way for multisets.

▶ **Lemma 6** (Properties of Relative Rank)**.**

1. *(Imbalance) Say* $P \in \mathbb{F}_{\mathrm{sm}}[X_1, \ldots, X_d]$*. Then,* $\mathrm{relrk}_W(P) \leqslant n^{-|\mathrm{Sum}(W)|/2}$.
2. *(Sub-additivity) Say* $P, Q \in \mathbb{F}_{\mathrm{sm}}[X_1, \ldots, X_d]$*. Then* $\mathrm{relrk}_W(P+Q) \leqslant \mathrm{relrk}_W(P) + \mathrm{relrk}_W(Q)$.
3. *(Multiplicativity) Say* $P = P_1 \cdot P_2 \cdot \ldots \cdot P_t$ *and assume that for each* $i \in [t]$*,* $P_i \in \mathbb{F}_{\mathrm{sm}}[X_j : j \in S_i]$*, where* $\{S_1, \ldots, S_t\}$ *is a partition of* $[d]$*. Then* $\mathrm{relrk}_W(P) = \mathrm{relrk}_W(P_1 \cdot P_2 \cdot \ldots \cdot P_t) = \prod_{i \in [t]} \mathrm{relrk}_{W_i}(P_i)$*, where* $W_i = \{\alpha_j \mid j \in S_i\}$.

With these properties in mind, we try to construct small set-multilinear formulas with optimally large relative rank. We do not lose much generality in assuming that $\mathrm{Sum}(W) \approx 0$, which we will do in the rest of this proof outline. So, the optimal relative rank is 1.

It is instructive to consider the example of W such that $\alpha_1 = \cdots = \alpha_{d/2} = 1$ and $\alpha_{d/2+1} = \cdots = \alpha_d = -1$. We start with a trivial formula F that consists of a single variable $x_1 \in X_1$, which has relative rank $n^{-1/2}$. Does it make sense to take linear combinations of such formulas? From the perspective of relative rank, the answer is No, because that increases the size without increasing the relative rank at all, by the Imbalance criterion in Lemma 6. So we can only multiply variables (from different sets, as we are dealing with set-multilinear formulas). Moreover, it makes sense to multiply variables such that the corresponding α_is have different signs, as multiplying variables from X_1 and X_2 (say) would only make the imbalance worse. So we multiply $x_1 \in X_1$ and $x_{d/2+1} \in X_{d/2+1}$. This creates a formula of relative rank $1/n$, by the property of Multiplicativity. By Sub-additivity, we need to sum at least n such formulas to get a formula of relative rank 1 (which is optimal). And indeed, this can be done, say, with an inner product between the variables of X_1 and $X_{d/2+1}$. Multiplying $d/2$ such formulas together (for a partition of $\alpha_1, \ldots, \alpha_d$ into positive and negative pairs) gives us a formula of product-depth 2, size $O_d(n)$, and relative rank 1.[9] One can see that the underlying multiplicative structure of the formula thus constructed naturally suggests a W-tree T of the form shown in Figure 1. This is a W-tree of depth-2 and bias 2 (which is the best possible for this W).

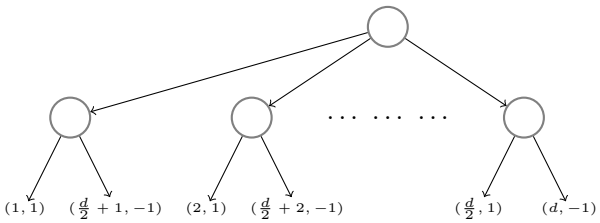


Figure 1 The W-tree of depth 2 and bias 2 arising from the formula construction above.

The above indicates a general technique for constructing formulas of large relative rank. Start by finding a $W' \subseteq W$ such that $|\mathrm{Sum}(W')|$ is small. Construct a formula of plausibly optimal relative rank (i.e. $n^{-|\mathrm{Sum}(W'))|/2}$) over the variable sets corresponding to W' by adding enough set-multilinear monomials so that sub-additivity no longer indicates that the rank of the formula is small. In doing this, we end up taking a sum of size n^b where

$$b := \frac{1}{2} \sum_{i \in W'} |\alpha_i| - \frac{|\mathrm{Sum}(W')|}{2}. \tag{2}$$

[9] This is an example of Nisan and Wigderson [20], aptly called the *Product of Inner Products* polynomial.

This indicates that it helps to take W' to be a small set, since otherwise this formula would be too large (if there were no such constraint, we could simply have taken $W' = W$). We partition W into small sets $W'_1, \ldots, W'_r$ this way, and construct formulas for each. Then, applying again the same principle to the multiset $\{\mathrm{Sum}(W'_1), \ldots, \mathrm{Sum}(W'_r)\}$, we get a high-rank set-multilinear formula over all of $X_1, \ldots, X_d$. As in the simple example above, this gives rise to a multiplicative structure that can be described by means of a W-tree T. The set W' constructed above corresponds to one of the nodes at height 1 in T and the quantity b in (2) is (almost) something we will define later to be the bias of the corresponding node, and $n^{b/2}$ lower bounds the size of the constructed formula. However, a careful analysis of the construction shows that the size of the formula is actually larger: at each node of the tree T, the formula uses a sum governed by the bias of the corresponding node. This naturally ends up yielding a formula whose size is governed by the path bias of T. Minimizing this over the choice of all trees yields the tree bias of W, as defined above.

Proof of Theorem 3

The above outline already indicates how to construct a set-multilinear formula of product-depth Δ and size $n^{O(\mathrm{Treebias}_\Delta(W))}$ that computes a polynomial of optimal relative rank. The only part that is unclear is how to ensure that the bounds on relative rank imposed by sub-additivity are actually tight. We do this by a careful inductive definition of the formulas. In a revision of our earlier paper [18], we showed how to do this for a specific W which contains only the two distinct elements -1 and $1/\sqrt{2}$. In this paper, we extend this construction to all W. This gives the second part of Theorem 3.

In the process, we note that the formulas we construct all have a special property: they have a *unique* multiplicative structure, i.e. they build up all their set-multilinear monomials in the same way, given by a single W-tree T. In principle, a general set-multilinear formula could contain many different kinds of trees (e.g. by summing formulas corresponding to different trees). These special formulas that we construct have been studied before: they are called *Pure* formulas [20] or *Unique Parse Tree* (UPT) formulas [16, 15]. We use the latter terminology.

For the first part of Theorem 3, we proceed as follows. We first show that UPT formulas of product-depth Δ have indeed the claimed upper bound on the relative rank, by using the basic properties of relative rank from Lemma 6 and a simple inductive argument. To argue about a general set-multilinear formula F, we show that any set-multilinear formula can be written as a sum of $O_d(1)$ many UPT formulas of the same size and product-depth. Using the sub-additivity of relative rank and the bound for UPT formulas, we see that F also has small relative rank.

We illustrate the power of the latter theorem with a short proof of one of the main results of [18]: an $n^{\Omega(\sqrt{d})}$ lower bound for set-multilinear formulas of product-depth 2.[10] By Theorem 3, it suffices to construct a multiset $W \subseteq [-1, 1]$ with $|\mathrm{Sum}(W)| = 0$ and tree bias $\Omega(\sqrt{d})$. Consider a W with $\Theta(d)$ copies each of (-1) and $\alpha := (1 - 1/\sqrt{d})$ so that $\mathrm{Sum}(W) = 0$. Given any depth-2 W-tree T, it can be checked that one of the following hold.

- There is a depth-1 vertex u with $t_u \geqslant \sqrt{d}/2$ children. In this case, any path through u has bias $\Omega(\sqrt{d})$.
- Every u at depth-1 has $t_u < \sqrt{d}/2$ children, in which case $|\mathrm{Sum}(u)| \geqslant t_u/(2\sqrt{d})$. This implies that any path in T has bias $\sum_u t_u/(2\sqrt{d}) = \sqrt{d}/2$.

[10] This is essentially the heart of the argument of [18], abstracting away the details about algebraic formulas, and keeping only the combinatorial core.

Proof of Theorem 4

In a similar way, we can also extend the results of [18] to show improved lower bounds for product-depth 3 (i.e. $\Sigma\Pi\Sigma\Pi\Sigma\Pi\Sigma$ formulas). More precisely, taking W as above but redefining $\alpha = 1-(1/d^{1/4})-(1/d^{3/4})$, we are able to prove a tree-bias lower bound of $\Omega(d^{1/4})$. This implies a formula lower bound of $n^{\Omega(d^{1/4})}$, which improves upon a lower bound of $n^{\Omega(d^{1/7})}$ from our previous work.

In the second part of the proof, we show that this is the best bound that this technique can prove, for *any* choice of W. Equivalently, we can show that every W has depth-3 tree bias $O(d^{1/4})$. We illustrate the idea with a sketch of the special case when W has two distinct elements (as in the two lower bounds above). In this case, it is not hard to argue that without loss of generality, the two distinct elements of W are (-1) and $\alpha \in (0,1]$.

First of all, we observe that any W has a tree of depth Δ and path bias $O(\Delta \|W\|_1^{1/\Delta})$, where $\|W\|_1$ denotes the sum of the absolute values of the elements of W. This is analogous to the fact that $\mathrm{IMM}_{n,d}$ has set-multilinear formulas of depth-Δ and size $n^{O(\Delta d^{1/\Delta})}$. Call this the "basic construction".

Now, given W as above, we proceed as follows. By a classical result of Dirichlet (see, e.g. [14, Theorem 4.9]), for any t, there exist integers $q \in [t]$ and $p \in \{0,\ldots,t\}$ such that $|q\alpha - p| \leqslant 1/t$. Note that this gives a multiset $W' \subseteq W$ of size $p+q$ such that $|\mathrm{Sum}(W')| \leqslant 1/t$. We apply this result with $t = \sqrt{d}$ and proceed in one of two ways depending on the value of $p+q$.

- If $p+q \geqslant d^{1/4}$, then we can partition W into at most $r \leqslant d^{3/4}$ sets $W_1,\ldots,W_r$ of size $p+q$, each of which has sum at most $1/\sqrt{d}$. As $p+q \leqslant 2\sqrt{d}$, using the basic construction of depth 2, we get a tree T_i of bias $O(d^{1/4})$ for each W_i. Attaching all these to a common root gives a tree of path-bias $O(d^{1/4})$ (the root adds at most $d^{3/4}\cdot(1/\sqrt{d}) = d^{1/4}$ to the bias of any path).
- If $p+q \leqslant d^{1/4}$, then by using $d^{1/4}/(p+q)$ many disjoint sets of sum $1/\sqrt{d}$ each, we get a set W' of size $d^{1/4}$ and sum at most $d^{1/4}/((p+q)\cdot\sqrt{d}) \leqslant 1/d^{1/4}$. We partition W into $r \leqslant d^{3/4}$ sets $W'_1,\ldots,W'_r$ of this form. We use a tree T_i of depth-1 for each W'_i (which has path bias at most $d^{1/4}$ trivially) and attach these to the leaves of a depth-2 tree for the set $\tilde{W} = \{\mathrm{Sum}(W_1),\ldots,\mathrm{Sum}(W_r)\}$. The latter tree is constructed using the basic construction of depth 2, and has bias $O(d^{1/4})$ as $\|\tilde{W}\|_1 \leqslant r/d^{1/4} \leqslant \sqrt{d}$.

This gives the argument in the case of W with only two distinct elements. For general W, we use a similar high-level argument. However, we need a suitable replacement for Dirichlet's theorem, which only works for the special W dealt with above. We prove a generalization of this theorem (see Lemma 9 below) to the setting of arbitrary multisets W. We think the statement is natural and interesting in its own right, but could not find mention of it in the literature.

In the special case that W contains d copies of $\alpha \in (0,1)$ and d copies of -1, the above implies the standard Dirichlet theorem used above. With the above generalized theorem in place, we can follow the structure of the argument for the special case, with technical modifications. This yields the depth-3 relative rank upper bound for any W.

Proof of Theorem 5 for depth Δ

While the proof of this theorem employs the same high-level argument as Theorem 22 described above, it is considerably more technical. We illustrate the idea again with the case when W contains only two distinct elements, which we can assume to be -1 and some $\alpha \in [0,1]$. Let $\mathrm{Bias}(\Delta, d)$ denote the largest possible bias of a depth-Δ W-tree. We give a constructive bound on this quantity by an inductive construction (based on Δ).

For $\Delta = 1$, we have the trivial bound $\mathrm{Bias}(1, d) \leqslant d$. For $\Delta > 1$, we use Dirichlet's theorem to find integers $p, q \leqslant d^{1-\varepsilon}$ such that $|q - p\alpha| \leqslant d^{-(1-\varepsilon)}$. This gives us a set $W' \subseteq W$ of size $p + q$ such that $|\mathrm{Sum}W'| \leqslant d^{-(1-\varepsilon)}$. There are again two cases to consider based on the magnitude of q.

- If $q \geqslant d^{\varepsilon}$, then this yields that $|W'| \geqslant d^{\varepsilon}$. Partitioning W into $t = d^{1-\varepsilon}$ subsets $W'_1, \ldots, W'_t$ of this form and using a recursive construction for each of $W'_1, \ldots, W'_t$, we get a W-tree of bias $\mathrm{Bias}(\Delta - 1, d^{\varepsilon}) + O(1)$. (Here, the last $O(1)$ term accounts for the bias accrued at the root, which is only a constant.)
- Conversely, if $q \leqslant d^{\varepsilon}$, we pick as many sets $W'_1, \ldots, W'_r$ as we can to form a set W'' of size (roughly) $d^{1-\varepsilon}$. Note that $|\mathrm{Sum}(W'')| \leqslant d^{1-\varepsilon}/d^{1-\varepsilon} \leqslant 1$. We partition W into $s \leqslant d^{\varepsilon}$ sets $W''_1, \ldots, W''_s$ of this form. We can construct a W-tree T of depth $\Delta = \Delta_1 + \Delta_2$ by
  - Constructing a W''_i-tree T_i of depth Δ_1 by constructing W'_j-tree $T_{i,j}$ of depth $\Delta_1 - 1$ for each $W'_j \subseteq W''_i$ and connecting these trees to a common root.
  - Constructing a depth-Δ_2 $\tilde{W}$-tree $\tilde{T}$, where $\tilde{W} = \{\mathrm{Sum}(W''_1), \ldots, \mathrm{Sum}(W''_s)\}$[11] and replacing the leaf labelled i with the tree T_i.

  As the sets $\tilde{W}$ and W'_i have size d^{ε} each, it makes sense to take $\Delta_1 = \Delta_2 = \Delta/2$. This leads to a bound on the bias of the tree T of $2 \cdot \mathrm{Bias}(\Delta/2, d^{\varepsilon}) + O(1)$.

We choose ε to balance the bias obtained from each of the above two strategies. It is clear that if $\varepsilon < 1/(2\Delta)$ (say), then the first strategy yields a bad bound of $d^{1/2}$ (or worse). This implies that we must take $\varepsilon \geqslant 1/2\Delta$, which can yield a best possible upper bound of $d^{1/\Delta^{O(\log \Delta)}}$ from the second strategy. We show that this upper bound is indeed achievable, by taking $\varepsilon = \Theta(\log^2 \Delta/\Delta)$.

1.3 Related Work

Barriers for lower bound techniques

The partial derivative method and its variants have been used to prove several lower bounds in algebraic complexity theory including the recent work of the authors. While these techniques have been quite useful, it is unclear whether they can be used to separate VP from VNP. In the last decade, there were many attempts at understanding the limitations of these lower bound techniques. This has led to a body of work about *barrier* results [25, 10, 6, 11, 5, 4] in algebraic complexity theory. These results typically consider a large family of lower bound techniques and argue that such techniques cannot be used to prove strong lower bounds. However, all such results are either conditional, or hold for relatively weak models of computation (such as set-multilinear formulas of product-depth 1). In contrast to these results, here we focus on a specific technique, namely the technique that gave the first super-polynomial lower bound for low-depth circuits. We show an unconditional limitation on this technique with respect to a reasonably strong model of computation. Hence, our work is incomparable to this literature.

[11] There is a small technical point here, which is that we will be left with a few more elements not covered by any of the W''_is. We ignore this here.

Our other recent work [17]

In a different recent paper, we prove algebraic formula lower bounds for formulas of larger depths. Specifically, we are able to prove superpolynomial set-multilinear formula lower bounds for $\mathrm{IMM}_{n,n}$ and *non-commutative* formula[12] lower bounds for formulas of depths up to $o(\sqrt{\log d})$. Note that the first of these results is a lower bound in the *high-degree* setting. This does not immediately imply a lower bound for general formulas, as we do not know of an efficient transformation to set-multilinear formulas when the degree is large. The second result does not imply any lower bounds in the commutative setting, as far as we know. The results of this paper are thus somewhat orthogonal, as they apply to set-multilinear (commutative) formulas in the low-degree setting.

Organization

We start with some preliminaries in Section 2. We then prove Thorems 3 and 4 in Sections 3 and 4 respectively. The proofs of Theorem 5 and many other statements are deferred to the full version for lack of space.

2 Basic Preliminaries and Results from Previous Work

Fix any multiset $W = \{\alpha_1, \ldots, \alpha_d\} \subseteq [-1, 1]$ and let $\mathbb{F}_{\mathrm{sm}}[X_1, \ldots, X_d]$ be a lopsided set-multilinear space of polynomials with $|X_i| = n^{\alpha_i}$.

The following is a consequence of earlier work of the authors.

▶ **Lemma 7** (Lower bounds from relative rank, Implicit in [18]). *Let d and n be integer parameters. Assume that $W = \{\alpha_1, \ldots, \alpha_d\} \subseteq [-1, 1]$ is an arbitrary multiset and consider the space $\mathbb{F}_{sm}[X_1, \ldots, X_d]$ where $|X_i| = n^{|\alpha_i|}$. Assume that we have shown the following: for any set-multilinear formula F (over variable sets $X_1, \ldots, X_d$) of size at most $s(n, d)$ and product-depth at most Δ, we have*

$$\mathrm{relrk}_W(F) \leqslant C_d \cdot \varepsilon_n \cdot n^{-|\mathrm{Sum}(W)|/2},$$

where C_d depends only on d and $\varepsilon_n \to 0$ as $n \to \infty$.

Then, for n large enough in comparison to d, any set-multilinear formula F of product-depth Δ computing $\mathrm{IMM}_{\mathrm{poly}(n),d}$ must have size at least $s(n, d)$. Further, any (possibly non-set-multilinear) formula of depth at most $\Delta + 1$ computing $\mathrm{IMM}_{\mathrm{poly}(n),d}$ must have size at least $s(n, d)/d^{O(\Delta d)}$.

The following simple proposition regarding path bias will be useful.

▶ **Proposition 8.** *Let $W \subseteq [-1, 1]$ be any finite multiset and let T be a W-tree with internal vertex u. If u has children $u_1, \ldots, u_r$, then*

$$\mathrm{Pathbias}_{W_u}(T_u) = \left(\max_{i \in [r]} \mathrm{Pathbias}_{W_{u_i}}(T_{u_i})\right) + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)|$$

where T_v denotes the subtree rooted at vertex v (which is, by definition, a W_v-tree in the natural way).

[12] This means that the operations of the formula are those of the non-commutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_N\rangle$ where variables do not commute.

Proof. Let p_v denote $\mathrm{Pathbias}_{W_v}(T_v)$ for any vertex v of T.

For any $i \in [r]$, let π_{u_i} denote the path of bias p_{u_i} in T_{u_i}. Let π_u denote the path in T_u obtained by adding the vertex u to π_{u_i}. Note that the off-path nodes of π_u are precisely the off-path nodes of π_{u_i} along with u_j $(j \neq i)$. Thus, the bias of π_u can be written as

$$\begin{aligned}\mathrm{bias}(\pi_u) &= \left(\sum_{v \in \mathrm{Offpath}(\pi_u)} |\mathrm{Sum}(v)|\right) - |\mathrm{Sum}(u)| \\ &= \mathrm{bias}(\pi_{u_i}) + |\mathrm{Sum}(u_i)| + \sum_{j \in [r]\setminus\{i\}} |\mathrm{Sum}(u_j)| - |\mathrm{Sum}(u)| \\ &= p_{u_i} + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)|.\end{aligned}$$

As this holds for each $i \in [r]$, we have shown that

$$p_u \geqslant \left(\max_{i \in [r]} p_{u_i}\right) + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)|. \tag{3}$$

For the reverse inequality, we proceed in the same way. Let π_u be a path in T_u of bias p_u. If π_u has length 0, then we have

$$p_u = \mathrm{bias}(\pi_u) = \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)| \leqslant \left(\max_{i \in [r]} p_{u_i}\right) + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)|$$

and hence we are trivially done. Otherwise, the path π_u passes through some child u_i of u. Let π_{u_i} be the path in T_{u_i} obtained by removing u from π_u. Then, through the same sequence of equalities proved above, we get

$$p_u = p_{u_i} + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)| \leqslant \left(\max_{i \in [r]} p_{u_i}\right) + \left(\sum_{j=1}^{r} |\mathrm{Sum}(u_j)|\right) - |\mathrm{Sum}(u)|.$$

Hence, we have proved the reverse inequality to (3) and we are done. ◀

2.1 A Generalized form of Dirichlet's theorem

Here we prove a generalized form of the standard Dirichlet Principle (see, e.g. [14, Theorem 4.9]), which we will use in Sections 4 and in the proof of Theorem 5 in the full version.

▶ **Lemma 9** (A Generalized Form of the Dirichlet Principle). *Assume $d \geqslant 2$. Let $W \subseteq [-1, 1]$ be any multiset with at least d non-negative and d non-positive elements. Then, for each positive integer $t \leqslant 2d$, there is a multiset $T \subseteq W$ of size at most t such that $|\mathrm{Sum}(T)| \leqslant 4/(t-1)$.*

Proof. The proof is via the Pigeonhole principle. Fix a t as above and let $\ell = \lfloor t/2 \rfloor$. If W contains an element x such that $|x| \leqslant 2/\ell$, then we are done trivially, so we assume that this is not the case.

Let $\{x_1, \ldots, x_\ell\}$ and $\{-y_1, \ldots, -y_\ell\}$ be any ℓ positive and negative elements of W respectively (here, $x_i, y_i \in (2/\ell, 1]$ for each i).

For $i \in \{0, \ldots, \ell\}$, define $u_i = \sum_{j=1}^{i} x_j$ and $v_i = \sum_{j=1}^{i} y_j$. For $i, j \in \{0, \ldots, \ell\}$, let $w_{i,j} = u_i + v_j$. Note that as $x_i, y_i \in [0, 1]$ for each $i \in [\ell]$, we have $u_i, v_j \in [0, \ell]$ and $w_{i,j} \in [0, 2\ell]$ for each $i, j \in \{0, \ldots, \ell\}$. Also note that $u_0, \ldots, u_\ell$ and $v_0, \ldots, v_\ell$ are increasing sequences in which the difference between any pair of elements is strictly more than $2/\ell$.

Divide the interval $[0, 2\ell]$ into ℓ^2 sub-intervals of length $2/\ell$ each. By the pigeonhole principle, there exist distinct (i, j) and (i', j') from $\{0, \ldots, \ell\} \times \{0, \ldots, \ell\}$ such that $w_{i,j}$ and $w_{i',j'}$ lie in the same interval. In particular, we have

$$|w_{i,j} - w_{i',j'}| = |(u_i - u_{i'}) - (v_{j'} - v_j)| \leqslant \frac{2}{\ell}. \tag{4}$$

Fix such (i, j) and (i', j'). Since these pairs are distinct, they must differ in some coordinate. We assume that they differ in the first coordinate (the other case is similar).

Without loss of generality, assume that $i > i'$. We note that it cannot be the case that $j \geqslant j'$. This is because we would then have

$$|w_{i,j} - w_{i',j'}| = (u_i + v_j) - (u_{i'} + v_{j'}) \geqslant u_i - u_{i'} > \frac{2}{\ell}$$

where for the inequalities we use the fact that $u_0, \ldots, u_\ell$ and $v_0, \ldots, v_\ell$ are increasing sequences in which the difference between any pair of elements is strictly more than $2/\ell$. This contradicts (4) above. In particular, this implies that $j < j'$. By (4), this yields

$$|(u_i - u_{i'}) - (v_{j'} - v_j)| = \left| \sum_{p=i'+1}^{i} x_k - \sum_{q=j+1}^{j'} y_j \right| \leqslant \frac{2}{\ell}.$$

This implies that to get a set T satisfying the requirements of the lemma, it is sufficient to take $T = \{x_{i'+1}, \ldots, x_i, -y_{j+1}, \ldots, -y_{j'}\}$. Note that $|T| \leqslant 2\ell \leqslant t$, and by the above computation $|\mathrm{Sum}(T)| \leqslant 2/\ell \leqslant 4/(t-1)$. ◀

3 The Lower Bound technique and Tree bias

In this section we will show that tight bounds on the tree bias yield the best possible bound on the relative-rank of set-multilinear low-depth formulas. Specifically, we prove Theorem 3.

3.1 Set-multilinear formulas and Unique Parse Trees

First, it will be helpful to make some structural changes to the formula. We will write a set-multilinear formula as a *small* sum of set-multilinear formulas such that each formula has a unique *parse tree*. In order to describe this we introduce some definitions.

▶ **Definition 10** (Parse Formula). *Let F be a set-multilinear formula. A parse formula F' is obtained from F as follows.*

- *The root + gate is added to F'.*
- *For every + gate added to F', one of its children is added to F'.*
- *For every × gate added to F', all its children are added to F'.*

Note that, such a parse formula computes a set-multilinear monomial. The polynomial computed by F is the sum of monomials computed by its parse formulas.

Parse trees and W-trees

Let F' be a parse formula from a set-multilinear formula F. We define the parse trees of F as follows. Let g be a + gate with the parent u and child v. We draw a direct edge between u and v and remove the + gate from F'. We do this *short-circuiting* step for each + gate of the parse formula. Similarly, we remove the + root of F'. Let $\mathfrak{T}$ be the tree thus obtained. We call this the *shape* of F'.

Let ℓ be a leaf of $\mathfrak{T}$. It corresponds to a gate g in F which is either a $+$ gate in F' or a leaf in F'. The polynomial computed by g is a linear polynomial on variable set X_i for some $i \in [d]$. We label ℓ with (i, α_i). This way, we label each leaf of $\mathfrak{T}$ with elements of W. We call the W-tree T thus obtained a *parse tree* of F. Note that the depth of T is the same as the product-depth of F.

▶ **Definition 11** (UPT formula). *We say that a set-multilinear formula F is a Unique Parse Tree formula (or UPT) if all the parse trees of F are identical.*

▶ **Lemma 12.** *Let F be a set-multilinear formula of size s and depth Δ. Then F can be written as a sum of at most d^{3d} many set-multilinear UPT formulas such that each formula has size at most s and depth Δ.*

We defer the proof of this lemma to the full version due to lack of space.

3.2 Tree bias lower bounds imply formula lower bounds

In this section, we show how lower bounds on $\text{Treebias}_\Delta(W)$ imply set-multilinear formula lower bounds in the low-degree setting. By Lemma 7, this implies lower bounds for general formulas as well.

We first show this connection for a UPT formula and then use the lemma from the previous section to conclude the same for general set-multilinear formulas. Specifically, we prove the following statement.

▶ **Lemma 13.** *Let n, d be positive integers. Let $\Delta \geqslant 1$. Let W be a multiset of $[-1, 1]$ of size d. Let F be a set-multilinear UPT formula of size s, product-depth Δ, and parse tree T. Assume, moreover, that* $\text{Pathbias}_W(T) = p$. *Then,*

$$\text{relrk}_W(F) \leqslant (s \cdot n^{-p/2}) \cdot n^{-|\text{Sum}(W)|/2}.$$

We first use this lemma to prove part (1) of Theorem 3.

Proof of Part (1) of Theorem 3. Let W and t be as in the statement of Theorem 3. Let F be a set-multilinear formula of product depth Δ and size at most s. From Lemma 12 we know that F can be written as a sum of UPT formulas, say $\Psi_1, \Psi_2, \ldots, \Psi_r$, where $r \leqslant d^{3d}$. We also know that the size of each Ψ_i is at most s and their depth is Δ. Let $\Gamma_1, \ldots, \Gamma_r$ be the parse trees of these formulas and let $p_i = \text{Pathbias}_W(\Gamma_i)$ for $i \in [r]$.

By Lemma 13, for each $i \in [r]$, $\text{relrk}_W(\Psi_i) \leqslant (s \cdot n^{-p_i/2}) \cdot n^{-|\text{Sum}(W)|/2}$. As $\text{Treebias}_\Delta(W) = t$, we have $p_i \geqslant t$ for each $i \in [r]$. Therefore, we get

$$i \in [r], \text{relrk}_W(\Psi_i) \leqslant (s \cdot n^{-t/2}) \cdot n^{-|\text{Sum}(W)|/2}.$$

As $F = \sum_{i=1}^{r} \Psi_i$, $r \leqslant d^{3d}$ and by sub-additivity of relrk, we get the claimed bound on the relrk of F, i.e.

$$\text{relrk}_W(F) \leqslant (d^{3d} \cdot s \cdot n^{-t/2}) \cdot n^{-|\text{Sum}(W)|/2}. \qquad \blacktriangleleft$$

We now prove Lemma 13.

Proof of Lemma 13. We prove the statement by induction on the depth of T (which is also the product depth of F).

Base case. Let $F = \sum_i \prod_j F_{i,j}$ be a set-multilinear UPT formula of product-depth $\Delta = 1$. Let T be the W-tree corresponding to F. Let u_0 be the root of T and let $u_1, \ldots, u_d$ be the children of u_0 with labels $(1, \alpha_1), \ldots, (d, \alpha_d)$, respectively.

By sub-additivity and sub-multiplicativity (Lemma 6, Items 2 and 3) of relrk, we can say that $\mathrm{relrk}_W(F) \leqslant \sum_i \prod_j \mathrm{relrk}_{\{\alpha_j\}}(F_{i,j})$. By using the Imbalance bound (Lemma 6 Item 1) on the relative rank of each $F_{i,j}$ we get that

$$\mathrm{relrk}_W(F) \leqslant \sum_i n^{-\sum_j |\alpha_j|/2} = s \cdot n^{-\sum_j |\alpha_j|/2} = s n^{-p/2} n^{-|\mathrm{Sum}(W)|/2}$$

where the last equality follows from Proposition 8. We get the desired bound.

Induction step. Let $F = \sum_i \prod_j F_{i,j}$ be a set-multilinear UPT formula of depth $\Delta > 1$. Let T be the W-tree corresponding to F. Let u_0 be the root of T and let $u_1, \ldots, u_k$ be the children of u_0. Let $T_1, \ldots, T_k$ be the trees rooted at $u_1, \ldots, u_k$ respectively.

As F is a UPT formula, we have that for each $i \neq i'$ and for any $j \in [k]$, the parse tree of $F_{i,j}$ is the same as the parse tree of $F_{i',j}$. Without loss of generality let us say the parse tree of $F_{i,j}$ is T_j for every i.

Also, for T, let us assume without loss of generality that the path bias of T is realised by a path π, where $\pi = u_0 \cdot u_1 \cdot \pi'$, i.e. specifically it passes through u_1. Let p_1 denote $\mathrm{Pathbias}_{W_{u_1}}(T_1)$.

Finally, let $s_{i,j}$ denote the size of the subformula $F_{i,j}$. Note that $\sum_{i,j} s_{i,j} \leqslant s$.

$$\begin{aligned}
\mathrm{relrk}_W(F) &\leqslant \sum_i \mathrm{relrk}_{W_{u_1}}(F_{i,1}) \cdot \prod_{j \geqslant 2} \mathrm{relrk}_{W_{u_j}}(F_{i,j}) && \text{Properties of relrk} \\
&\leqslant \sum_i \mathrm{relrk}_{W_{u_1}}(F_{i,1}) \cdot \prod_{j \geqslant 2} n^{-|\mathrm{Sum}(u_j)|/2} && \text{Trivial bound on relrk} \\
&\leqslant \sum_i \left((s_{i,1} \cdot n^{-p_1/2}) \cdot n^{-|\mathrm{Sum}(u_1)|/2} \right) \cdot \prod_{j \geqslant 2} n^{-|\mathrm{Sum}(u_j)|/2} && \text{Induction Hypothesis} \\
&\leqslant \sum_i s_{i,1} \cdot n^{-\left(p_1 + \sum_{j=1}^{k} \mathrm{Sum}(u_j)\right)/2} \\
&= \sum_i s_{i,1} \cdot n^{-(p + |\mathrm{Sum}(W)|)/2} && \text{Proposition 8} \\
&\leqslant s \cdot n^{-p/2} \cdot n^{-|\mathrm{Sum}(W)|/2}. && \blacktriangleleft
\end{aligned}$$

3.3 Tree bias upper bounds imply formula upper bounds

We now sketch the proof of the second part of Theorem 3. The main idea is an abstraction of a proof from our earlier result [18][13] where we constructed polynomials to show that our lower bound technique was "tight" for certain concrete spaces of lopsided set-multilinear polynomials. In the second part of Theorem 3, we essentially show that the lower bound proved via tree-bias is tight for *all* lopsided spaces.

The main technical result (which generalizes [18, Lemma 26]) is the following, which handles the case where each $|X_i|$ is a power of 2.

[13] More specifically, this result appeared in a later version of the paper that can be found on ECCC.

▶ **Lemma 14.** *Let n, d be growing parameters and Δ any positive integer. Let $W = \{\alpha_1, \ldots, \alpha_d\} \subseteq [-1, 1]$ be a multiset. Assume that $\mathbb{F}_{\mathrm{sm}}[X_1, \ldots, X_d]$ be a lopsided space of set-multilinear polynomials with $|X_i| = n^{|\alpha_i|} = 2^{k_i}$ for non-negative integers $k_1, \ldots, k_d$.*

Let T be any W-tree of depth Δ with $\mathrm{Pathbias}_W(T) = p$. Then, there is a UPT formula F of parse tree T (and hence product-depth Δ) with at most $d \cdot n^{p/2}$ leaves such that $\mathrm{rank}(M_W(F))$ is as large as possible (i.e. equal to either the number of its rows or columns).

We defer the proof of this lemma, as well as its generalization which yields the second part of Theorem 3, to the full version of the paper.

4 Optimal bounds for depth 3 via our technique

This section is devoted to the proof of Theorem 4, which characterizes (up to constant factors) the maximum possible tree bias of a tree of depth 3.

In proving this theorem, it will be useful to consider a variant on the notion of tree bias defined above, that we will call *node bias*. The node bias of W (at any given depth Δ) is equal to the tree bias of W up to a factor of $O(\Delta)$. By Theorem 3, for constant-depths Δ, the node bias also captures the best lower bound that we can hope to prove via our technique.

▶ **Definition 15** (Node bias). *Fix a W-tree T. For an internal node v of T, we define the bias of v, denoted $\mathrm{bias}(v)$, to be $\sum_u |\mathrm{Sum}(u)|$ where the sum runs over the children u of v. The node bias of T, denoted $\mathrm{Nodebias}_W(T)$, is the largest bias of any internal node v of T. Further, the depth-Δ node bias of W, is the minimum node bias of any depth-Δ, W-tree T. This quantity is denoted $\mathrm{Nodebias}_\Delta(W)$.*

The following basic proposition (proof omitted) relates the node bias of W and the tree bias of W.

▶ **Proposition 16.** *For any depth-Δ W-tree T, we have*

$$\mathrm{Nodebias}_W(T) \leqslant \mathrm{Pathbias}_W(T) + |\mathrm{Sum}(W)| \leqslant \Delta \cdot \mathrm{Nodebias}_W(T).$$

In particular, for any multiset $W \subseteq [-1, 1]$ and any depth Δ, we have $\mathrm{Nodebias}_\Delta(W) \leqslant \mathrm{Treebias}_\Delta(W) + |\mathrm{Sum}(W)| \leqslant \Delta \cdot \mathrm{Nodebias}_\Delta(W)$.

4.1 Some simple claims

This section presents several statements about W-trees. As the proofs are simple, we defer them to the full version of the paper.

Given a partition[14] P of the elements of W, we define the *grouping* W' of W to be the multiset obtained by taking the sums of elements of P. Formally,

$$W' = \{\mathrm{Sum}(A) \mid A \in P\}.$$

The following basic lemma shows how to construct a W-tree from trees of its groupings and subsets.

[14] We follow the usual convention that $\varnothing \notin P$.

▶ **Lemma 17.** *Assume that $P = \{W_1, \dots, W_t\}$ is a partition of W and let W' be the corresponding grouping. Say we have a W'-tree T' of node bias b' and depth Δ' and for each $i \in [t]$, a W_i-tree T_i of depth Δ_i and node bias at most b_i. Then, there is a W-tree T of node bias at most $\max\{b', b_i\}$ and depth at most $\Delta' + \max_{i\in[t]} \Delta_i$.*

Moreover, if each W_i is sign-monochromatic *(i.e., all elements of W_i have the same sign)*[15]*, then there is a W-tree T of depth Δ' and bias b'.*

▶ **Lemma 18** (Preprocessing Lemma)**.** *Let $W \subseteq [-1, 1]$ be any multiset. Then, there is a partition $P = \{W_1, \dots, W_t\}$ of W such that each W_i is sign-monochromatic (as in Lemma 17), $\mathrm{Sum}(W_i) \in [-1, 1]$ for all $i \in [t]$ and $\mathrm{Sum}(W_i) \in [-1, -1/2] \cup [1/2, 1]$ for each $i \in \{3, \dots, t\}$.*

In particular, there is a grouping $W' \subseteq [-1, 1]$ of W such that $|W' \setminus ([-1, -1/2] \cup [1/2, 1])| \leqslant 2$ and for each W'-tree T', there is a W-tree T of same depth and node bias.

The next lemma shows, in particular, how to construct W-trees of depth Δ and node bias $O(d^{1/\Delta})$ for any multiset $W \subseteq [-1, 1]$ of size d and of sum at most 1.

▶ **Lemma 19.** *Let $W \subseteq [-1, 1]$ such that $\|W\|_1 \leqslant L$ and $|\mathrm{Sum}(W)| \leqslant 1$. Then, for any $\Delta \geqslant 1$, there is a W-tree of depth at most Δ and node bias at most $5L^{1/\Delta}$.*

We also have the following simple "pasting" lemma.

▶ **Lemma 20.** *Let $P = \{W_1, \dots, W_r\}$ be a partition of W and assume that for all i there is a W_i-tree T_i of depth at most Δ, node bias at most b_i and such that the root node of each T_i has bias at most b'_i. Then, there is a W-tree T of depth at most Δ and of node bias at most $\max\{b_1, \dots, b_r, \sum_i b'_i\}$.*

Finally, the following claim will allow us to balance a given subset of W so that removing this subset results in two sets of absolute sum at most 1.

▶ **Lemma 21** (Balancing lemma)**.** *Say $W \subseteq [-1, 1]$ is such that $|\mathrm{Sum}(W)| \leqslant 1$. Let $W' \subseteq W$ be arbitrary. Then there exists $W'' \subseteq W$ such that $W'' \supseteq W'$ and $\|W'' \setminus W'\|_1 \leqslant |\mathrm{Sum}(W')| + 1$ and $|\mathrm{Sum}(W'')|, |\mathrm{Sum}(W \setminus W'')| \leqslant 1$.*

4.2 Depth-3 trees of small bias

The main theorem of this section is the following.

▶ **Theorem 22.** *Let $W \subseteq [-1, 1]$ be any multiset such that $|W| \leqslant d$ and $|\mathrm{Sum}(W)| \leqslant 1$. Then, there is a W-tree T of depth 3 and node bias $O(d^{1/4})$.*

The rest of the section is devoted to the proof of the above theorem. To construct the required W-tree T, we use the following procedure.

1. Preprocessing: By the Preprocessing lemma (Lemma 18), it suffices to consider multisets W such that $|W \setminus ([-1, -1/2] \cup [1/2, 1])| \leqslant 2$.
2. We apply the following procedure to our multiset W.

[15] Here, we think of 0 as having the same sign as any other number.

Algorithm 1 $\mathcal{A}(W)$.

Assignment $d := |W|$.
Initialization If $d \leqslant 25$ then return the trivial depth-1 W-tree of node bias at most 25.
Phase 1: As long as it is possible, pick pairwise disjoint sets A such that $|A| \leqslant d^{1/4}$ and

$$\frac{|\mathrm{Sum}(A)|}{|A|} \leqslant \frac{12}{d^{1/2}}.$$

When this is no longer possible, let $A_1, \ldots, A_{e_1}$ be the sequence of sets picked and let $W_1' = A_1 \cup A_2 \cdots \cup A_{e_1}$. Using the Balancing lemma, let $W_1 = W_1' \cup \{a_1, \ldots, a_{f_1}\} \subseteq W$ be such that $\sum_{i \leqslant f_1} |a_i| \leqslant |\mathrm{Sum}(W_1')| + 1$ and $|\mathrm{Sum}(W_1)|, |\mathrm{Sum}(W \backslash W_1)| \leqslant 1$.
Construct a W_1-tree T_1 in the following way. Fix the grouping $\tilde{W}_1$ corresponding to the partition $P_1 = \{A_1, \ldots, A_{e_1}, \{a_1\}, \ldots, \{a_{f_1}\}\}$ of W_1. For each element of P_1, construct a trivial tree of depth-1 and for the grouping $\tilde{W}_1$, construct a depth-2 tree $\tilde{T}_1$ of node bias at most $5\sqrt{\|\tilde{W}_1\|_1}$ (using Lemma 19). Combine these using Lemma 17 to get a tree T_1 of depth 3 for W_1.
Set $W' = W \backslash W_1$ and continue.
Phase 2: As long as it is possible, pick pairwise disjoint sets $B \subseteq W'$ such that $|B| \leqslant d^{1/2}$ and

$$\frac{|\mathrm{Sum}(B)|}{|B|} \leqslant \frac{12}{d^{3/4}}.$$

When this is no longer possible, let $B_1, \ldots, B_{e_2}$ be the sequence of sets picked and let $W_2' = B_1 \cup B_2 \cdots \cup B_{e_2}$. Using the balancing lemma, let $W_2 = W_2' \cup \{b_1, \ldots, b_{f_2}\} \subseteq W'$ be such that $\sum_{i \leqslant f_1} |b_i| \leqslant |\mathrm{Sum}(W_2')| + 1$ and $|\mathrm{Sum}(W_2)|, |\mathrm{Sum}(W' \backslash W_2)| \leqslant 1$.
Construct a W_2-tree T_2 in the following way. Fix the grouping $\tilde{W}_2$ corresponding to the partition $P_2 = \{B_1, \ldots, B_{e_2}, \{b_1\}, \ldots, \{b_{f_2}\}\}$ of W_2. Construct a trivial depth-1 $\tilde{W}_2$-tree $\tilde{T}_2$ of node bias $\|\tilde{W}_2\|_1$. For each element B of P_2, construct a depth-2 tree of node bias at most $5\sqrt{\|B\|_1}$ (using Lemma 19). Combine these using Lemma 17 to get a tree T_2 of depth 3 for W_2.
Set $W'' = W' \backslash W_2$ and continue.
Recursive call Compute $T_3 = \mathcal{A}(W'')$.
Return The W-tree T of node bias at most $b_1 + b_2 + b_3$, where $b_i = \mathrm{Nodebias}(T_i)$ (for $i \in [3]$) given by T_1, T_2, T_3 and Lemma 20.

We now analyze the above construction. We first state a technical lemma.

▶ **Lemma 23.** *If $d > 25$, then after Phases 1 and 2, we have $|W''| \leqslant d/2$.*

Let us assume the above lemma for now and prove the theorem.

Let $b_i(d)$ denote the node bias of the tree T_i ($i \in [3]$) assuming that the word W has size at most d. Then, the node bias of the tree is $b_1(d) + b_2(d) + b_3(d)$. By Lemma 23, we can bound $b_3(d)$ by $b_1(d/2) + b_2(d/2) + b_3(d/2)$. Continuing recursively in this way (until d becomes smaller than 25) we have

$$\mathrm{Nodebias}(T) \leqslant \left(\sum_{i \geqslant 0} b_1(d/2^i) + b_2(d/2^i) \right) + 25.$$

So to prove Theorem 22, it suffices to show that $b_1(d), b_2(d) \leqslant O(d^{1/4})$. From now on, we fix d and let $b_i = b_i(d)$ for $i \in [2]$.

We first bound b_1. By construction each element of the partition P_1 is a set A of size at most $d^{1/4}$ and hence has a depth-1 tree of node bias at most $d^{1/4}$. Moreover, we have

$$\begin{aligned}\|\tilde{W}_1\|_1 &= \sum_{i \leqslant e_1} |\mathrm{Sum}(A_i)| + \sum_{j \leqslant f_1} |a_j| \\ &\leqslant \sum_{i \leqslant e_1} |\mathrm{Sum}(A_i)| + |\mathrm{Sum}(W_1')| + 1 \leqslant 2 \sum_{i \leqslant e_1} |\mathrm{Sum}(A_i)| + 1 \\ &\leqslant 2 \sum_{i \leqslant e_1} \frac{12|A_i|}{d^{1/2}} + 1 \leqslant O(d^{1/2}).\end{aligned}$$

Hence, the tree $\tilde{T}_1$ has node bias $\tilde{b}_1 = O(d^{1/4})$. Hence, by Lemma 17, we see that $b_1 \leqslant O(d^{1/4})$.

We can bound b_2 similarly. By construction, each element of P_2 is a set B of size at most $d^{1/2}$ and hence by Lemma 19 has a depth-2 tree of node bias at most $5d^{1/4}$. Moreover, we have

$$\begin{aligned}\|\tilde{W}_2\|_1 &= \sum_{i \leqslant e_2} |\mathrm{Sum}(B_i)| + \sum_{j \leqslant f_1} |b_j| \\ &\leqslant \sum_{i \leqslant e_2} |\mathrm{Sum}(B_i)| + |\mathrm{Sum}(W_2')| + 1 \leqslant 2 \sum_{i \leqslant e_2} |\mathrm{Sum}(B_i)| + 1 \\ &\leqslant 2 \sum_{i \leqslant e_1} \frac{12|B_i|}{d^{3/4}} + 1 \leqslant O(d^{1/4}).\end{aligned}$$

In particular, this implies that the tree $\tilde{T}_2$ has node bias $\tilde{b}_2 = O(d^{1/4})$. In particular, by Lemma 17, we see that $b_2 \leqslant O(d^{1/4})$.

Thus, we have shown that $b_1, b_2 = O(d^{1/4})$ and we are done.

It remains only to prove Lemma 23, which we do now.

Proof of Lemma 23. Let $d'' = |W''|$. Assuming that $d > 25$ and $d'' > d/2$, we will show that Phases 1 and 2 of the algorithm could not have concluded, and hence derive a contradiction.

Let W''_+ and W''_- denote the positive and negative elements of W'' respectively. Recall that $W'' \subseteq W$ and the latter set contains at most two elements of absolute value less than $1/2$ (by the preprocessing in Step 1). Further using the fact that $|\mathrm{Sum}(W'')| \leqslant 1$, it is easy to see that $|W''_+|, |W''_-| \geqslant d'''$ where $d''' = (d'' - 4)/3 > 2$.

By Lemma 9, it follows that there is a non-empty set $T \subseteq W''$ of size $t \leqslant \sqrt{d'''} + 1$ such that $|\mathrm{Sum}T| \leqslant 4/\sqrt{d'''}$. Since $d \geqslant 24$, this set T has size at most $\sqrt{d}$ and satisfies $|\mathrm{Sum}T| \leqslant 12/\sqrt{d}$.

Now, we do a short case analysis. Assume $|T| \leqslant d^{1/4}$. Then, T is the kind of set that the algorithm tries to find in Phase 1. Hence, the existence of such a T tells us that Phase 1 could not have concluded.

Otherwise, we have $|T| > d^{1/4}$. In this case, we have $|\mathrm{Sum}T|/|T| \leqslant 12d^{-3/4}$ and is hence the kind of set that the algorithm tries to find in Phase 2. Hence, the existence of such a T tells us that Phase 2 could not have concluded.

In either case, we are done. ◀

4.3 Optimality of the quartic bound

We will show here that the bound of Theorem 22 is optimal.

▶ **Proposition 24.** *Let d be a growing integer parameter. There exists a multiset $W \subseteq [-1, 1]$ such that $|W| \leqslant d$, $|\mathrm{Sum}W| \leqslant 1$, and for all W-tree T of depth 3, T has node bias at least $\Omega(d^{1/4})$.*

Proof. If $d < 16$ the result follows immediately (just adapt the constant in the $\Omega()$ to deal with these cases). So let us assume that $d \geqslant 16$. Let d' be the largest integer such that $d' \leqslant d$ and d' is a fourth power of an integer. So $d'^{1/4} \geqslant 2$ and $d' \geqslant d/16$.

Let q be the closest integer to $\frac{d'}{2-1/d'^{1/4}+1/(2d'^{3/4})}$. So $\left|q - \frac{d'}{2-1/d'^{1/4}+1/(2d'^{3/4})}\right| \leqslant \frac{1}{2}$. Let us construct W with q copies of $1 - 1/d'^{1/4} + 1/(2d'^{3/4})$ and $p = d' - q$ copies of -1. So $|W| \leqslant d' \leqslant d$ and

$$\begin{aligned}
|\mathrm{Sum}(W)| &= \left|-p + q(1 - 1/d'^{1/4} + 1/(2d'^{3/4}))\right| \\
&= \left|-d' + q(2 - 1/d'^{1/4} + 1/(2d'^{3/4}))\right| \\
&\leqslant \left|(-d' + d')\right| + \frac{1}{2}\left|(2 - 1/d'^{1/4} + 1/(2d'^{3/4}))\right| \leqslant 1.
\end{aligned}$$

It is sufficient to prove that any W-tree has large enough node bias.

Let T be any W-tree. Let us assume that $\mathrm{Nodebias}_W(T) < d'^{1/4}/4$.

Since every internal node α at distance two of the roots with k children (in particular the children of α are leaves of T) has bias at least $\mathrm{bias}(\alpha) \geqslant k \min_{v \in W} |v| \geqslant k/2$, it implies that $k < d'^{1/4}/2$.

Assume then that there is an internal node α at distance one of the root such that the subtree rooted in α has at least $d'^{1/2}$ leaves. Notice that for any children β of α with p_β negative children and q_β positive ones we have

$$\begin{aligned}
|\mathrm{Sum}\beta| &= \left|-p_\beta + q_\beta(1 - 1/d'^{1/4} + 1/(2d'^{3/4}))\right| \\
&= \left|(-p_\beta + q_\beta) - q_\beta(1 - 1/2d'^{1/2})/d'^{1/4}\right|.
\end{aligned}$$

Since $q_\beta \leqslant p_\beta + q_\beta < d'^{1/4}/2$ and p_β, q_β are integers, it implies that the fractional part of $|\mathrm{Sum}\beta|$ is at least $q_\beta(1 - 1/2d'^{1/2})/d'^{1/4}$. Moreover, if $|\mathrm{Sum}(\beta)| < 1$, it means that $q_\beta \geqslant p_\beta$, i.e., $q_\beta \geqslant (p_\beta + q_\beta)/2$. Hence in all cases,

$$|\mathrm{Sum}(\beta)| \geqslant \frac{q_\beta + p_\beta}{2} \cdot \frac{1}{2} \cdot \frac{1}{d'^{1/4}}.$$

Consequently, $\mathrm{bias}(\alpha) = \sum_{\beta \text{ child of } \alpha} |\mathrm{Sum}(\beta)| \geqslant \frac{d'^{1/2}}{4d'^{1/4}} = \frac{d'^{1/4}}{4}$, which contradicts the hypothesis. So any node at depth 1 of the tree has less than $d'^{1/2}$ leaves in its subtree.

Let us show that finally the root ρ of T has large bias. Let β one of its children. Say that in the tree rooted in β, there are p_β negative leaves and q_β positive ones. So,

$$\begin{aligned}
|\mathrm{Sum}\beta| &= \left|-p_\beta + q_\beta(1 - 1/d'^{1/4} + 1/(2d'^{3/4}))\right| \\
&= \left|(-p_\beta d'^{1/4} + q_\beta d'^{1/4} - q_\beta)\frac{1}{d'^{1/4}} + q_\beta \frac{1}{2d'^{3/4}}\right|.
\end{aligned}$$

Since $q_\beta/(2d'^{3/4}) < 1/(2d'^{1/4})$, it implies that the distance of $|\mathrm{Sum}(\beta)|$ to the set $\mathbb{N}/d'^{1/4}$ is at lest $q_\beta/(2d'^{3/4})$. Again, if $|\mathrm{Sum}(\beta)| < 1$, it ensures that $q_\beta \geqslant (p_\beta + q_\beta)/2$. So in all cases,

$$|\mathrm{Sum}(\beta)| \geqslant \frac{p_\beta + q_\beta}{2} \cdot \frac{1}{2d'^{3/4}}.$$

Consequently,

$$\mathrm{bias}(\rho) = \sum_{\beta \text{ child of } \rho} |\mathrm{Sum}(\beta)| \geqslant \frac{1}{4d'^{3/4}} \sum_{\beta \text{ child of } \rho} p_\beta + q_\beta = \frac{d'^{1/4}}{4}$$

which again contradicts the hypothesis.

In conclusion, we have that for any W-tree T and $\mathrm{Nodebias}_W(T) \geqslant \frac{d'^{1/4}}{4} \geqslant \frac{d^{1/4}}{8}$. ◀

▶ Remark 25. We can generalize the previous proof to larger depths by defining q to be the closest integer to $d'/(2+\sum_{i=1}^{\Delta-1}(-1)^i/d'^{(2^i-1)/2^{\Delta-1}})$. It implies that for all Δ, there exists a multiset W such that any W-tree of depth Δ has node bias at least $\Omega(d^{1/2^{\Delta-1}})$. It improves the constant in the exponent slightly in the lower bound from [18].

References

1 Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.

2 Peter Bürgisser. Cook's versus valiant's hypothesis. *Theoretical Computer Science*, 235(1):71–88, 2000.

3 Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.

4 Klim Efremenko, Ankit Garg, Rafael Mendes de Oliveira, and Avi Wigderson. Barriers for rank methods in arithmetic complexity. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 94 of *LIPIcs*, pages 1–19, 2018. doi:10.4230/LIPIcs.ITCS.2018.1.

5 Klim Efremenko, J. M. Landsberg, Hal Schenck, and Jerzy Weyman. The method of shifted partial derivatives cannot separate the permanent from the determinant. *Mathematics of Computation*, 87(312):2037–2045, 2018. doi:10.1090/mcom/3284.

6 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.

7 Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

8 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 (STOC)*, pages 577–582, 1998. doi:10.1145/276698.276872.

9 Dima Grigoriev and Alexander A. Razborov. Exponential lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Applicable Algebra in Engineering, Communication and Computing*, 10(6):465–487, 2000. doi:10.1007/s002009900021.

10 Joshua A. Grochow. Unifying known lower bounds via geometric complexity theory. *Computational Complexity*, 24(2):393–475, 2015. doi:10.1007/s00037-015-0103-x.

11 Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. arXiv:1701.01717.

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

13 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 6–20, 1986. doi:10.1145/12130.12132.

14 Stasys Jukna. *Extremal Combinatorics - With Applications in Computer Science*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2011. doi:10.1007/978-3-642-17364-6.

15 Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower bounds and PIT for non-commutative arithmetic circuits with restricted parse trees. *Computational Complexity*, 28(3):471–542, 2019. doi:10.1007/s00037-018-0171-9.

16 Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. Non-commutative computations: lower bounds and polynomial identity testing. *Chicago Journal of Theoretical Computer Science*, 2019:2, 2019. URL: http://cjtcs.cs.uchicago.edu/articles/2019/2/contents.html.

17 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Set-multilinear and non-commutative formula lower bounds for iterated matrix multiplication. *To appear in STOC 2022. Electron. Colloquium Comput. Complex.*, page 94, 2021. URL: `https://eccc.weizmann.ac.il/report/2021/094`.

18 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 804–814. IEEE, 2021. `doi:10.1109/FOCS52979.2021.00083`.

19 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. *Electron. Colloquium Comput. Complex.*, page 81, 2021. URL: `https://eccc.weizmann.ac.il/report/2021/081`.

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

21 Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *Journal of the ACM*, 56(2):8:1–8:17, 2009. `doi:10.1145/1502793.1502797`.

22 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *Journal of the ACM*, 60(6):40:1–40:15, 2013. `doi:10.1145/2535928`.

23 Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(2):598–607, 1986.

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

25 Andrew Drucker Scott Aaronson. Arithmetic natural proofs theory is sought. *Blog post*, 24(1):1–48, 2008.

26 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, 2010. `doi:10.1561/0400000039`.

27 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82, 1987. `doi:10.1145/28395.28404`.

28 Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing (STOC)*, pages 249–261. ACM, 1979. `doi:10.1145/800135.804419`.

Further Collapses in TFNP

Mika Göös
EPFL, Lausanne, Switzerland

Siddhartha Jain
EPFL, Lausanne, Switzerland

William Pires
McGill University, Montreal, Canada

Ran Tao
McGill University, Montreal, Canada

Alexandros Hollender
University of Oxford, UK

Gilbert Maystre
EPFL, Lausanne, Switzerland

Robert Robere
McGill University, Montreal, Canada

Abstract

We show EOPL = PLS ∩ PPAD. Here the class EOPL consists of all total search problems that reduce to the End-of-Potential-Line problem, which was introduced in the works by Hubáček and Yogev (SICOMP 2020) and Fearnley et al. (JCSS 2020). In particular, our result yields a new simpler proof of the breakthrough collapse CLS = PLS ∩ PPAD by Fearnley et al. (STOC 2021). We also prove a companion result SOPL = PLS ∩ PPADS, where SOPL is the class associated with the Sink-of-Potential-Line problem.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness

Keywords and phrases TFNP, PPAD, PLS, EOPL

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.33

Acknowledgements We thank Aviad Rubinstein for his many questions during e-mail correspondence, and the anonymous reviewers for their suggestions that helped improve the presentation of the paper.

1 Introduction

Our main results are two collapses of total NP search problem (TFNP) classes.

▶ **Theorem 1.** EOPL = PLS ∩ PPAD.

▶ **Theorem 2.** SOPL = PLS ∩ PPADS.

Let us explain what these collapses mean and how they fit into the diverse complexity zoo of search problem classes, as summarised in Figure 1. The classes PLS, PPAD, PPADS are classical. They were all introduced in the original pioneering works [17, 16, 20] that founded the theory of TFNP. To define these classes, it is most convenient to describe a canonical complete problem for each class. (See Section 2 for more formal definitions).

PLS: Sink-of-DAG (SoD). We are given *implicit access* to a directed graph $G = (V, E)$ that is acyclic, has out-degree at most 1, and has exponentially many nodes, $|V| = 2^n$. The graph is described by a poly(n)-sized circuit: for any node $v \in V$, we can compute its unique *successor* (out-neighbour) u, if any, and also an integer *potential*, which is guaranteed to increase along the direction of the edge (v, u). The goal is to find a *sink* node (in-degree ≥ 1, out-degree 0).

PPAD: End-of-Line (EoL). We are given access to a directed graph $G = (V, E)$ that has in/out-degree at most 1, and has $|V| = 2^n$ nodes. The graph is described by a poly(n)-sized circuit: for any $v \in V$, we can compute its *successor* u and *predecessor* u',

if any. We are guaranteed that if v's successor is u, then u's predecessor is v, and vice versa. In addition, we are given the name of a *distinguished source* node v^* (in-degree 0, out-degree 1). The goal is to find any source or sink other than v^*.

PPADS: Sink-of-Line (SoL). Same as EoL except the goal is to find a sink.

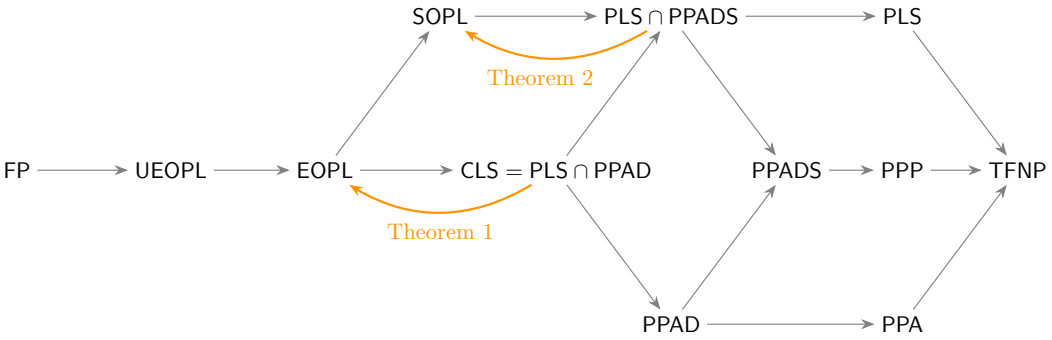


Figure 1 Class diagram for TFNP with new inclusions highlighted. An arrow A → B denotes A ⊆ B.

Modern classes

Research in the past decade has studied several relatively weak classes of search problems that lie below PLS and PPAD. The intersection class PLS∩PPAD is, of course, one immediate such example. This class, however, feels quite artificial at first glance. It does not seem to admit any "natural" complete problem. Motivated by this, Daskalakis and Papadimitriou [5] introduced the *continuous local search* class CLS ⊆ PLS∩PPAD, which, by its very definition, admits natural complete problems related to the local optimisation of continuous functions over the real numbers (computed by arithmetic circuits). The class CLS is exceptional in that it captures the complexity of *real continuous* optimisation problems, while most classical search problem classes are designed to capture *combinatorial principles*, often phrased in terms of directed graphs.

In order to understand CLS from a more combinatorial perspective, Hubáček and Yogev [14] and Fearnley, Gordon, Mehta, and Savani [9] introduced the class EOPL ⊆ CLS, whose complete problem is the namesake End-of-Potential-Line (EoPL) problem, defined below. (The paper [14] initially defined a more restricted "metered" version of this problem, but we use the formulation from [9], which they prove is equivalent to the one from [14].) It is also natural to define a *sink-only* version of EOPL as suggested by [12].

EOPL: End-of-Potential-Line (EoPL). We are given access to a directed graph $G = (V, E)$ that is acyclic, has in/out-degree at most 1, and has $|V| = 2^n$ nodes; that is, G is a disjoint union of directed paths. The graph is described by a poly(n)-sized circuit: for any node we can compute its successor and predecessor, if any, and also an integer potential, which is guaranteed to increase along the directed edges. In addition, we are given the name of a distinguished source v^*. The goal is to find any source or sink other than v^*.

SOPL: Sink-of-Potential-Line (SoPL). Same as EoPL except the goal is to find a sink.

It is comforting to know that the definition of EOPL is robust: Ishizuka [15] showed that a version of EoPL that guarantees poly(n) many distinguished sources is still equivalent (via polynomial-time reductions) to the above standard version with a single source.

Fearnley et al. [9] also defined a more restricted subclass UEOPL $\subseteq$ EOPL where the complete problem is Unique-EoPL, a version of EoPL with a *unique* directed path. They showed that this class contains many important search problems with unique witnesses, such as unique sink orientations, linear complementary problems, Arrival [6, 10]. Other problems known to lie in UEOPL are a restricted version of the Ham-Sandwich problem [3] and a pizza cutting problem [21]. Fearnley et al. [9] conjecture that UEOPL $\neq$ EOPL.

A surprising collapse

In a breakthrough, Fearnley, Goldberg, Hollender, and Savani [8] showed that, despite appearances to the contrary, CLS $=$ PLS $\cap$ PPAD. This goes against the conjecture of Daskalakis and Papadimitriou [5] that the classes are distinct, a belief which underlied much of their original motivation for introducing CLS. The nontrivial direction of the collapse is a reduction from a canonical complete problem SoD $\curlywedge$ EoL $\in$ PLS $\cap$ PPAD (defined below) to a problem KKT $\in$ CLS, which involves computing a Karush–Kuhn–Tucker point of a smooth function. We may summarise the main technical result of Fearnley et al. [8] as

$$\text{SoD} \curlywedge \text{EoL} \;\leq\; \text{KKT} \qquad \text{which implies } \mathsf{PLS} \cap \mathsf{PPAD} \subseteq \mathsf{CLS}. \tag{1}$$

Here we use $\leq$ to denote a polynomial-time reduction between search problems. The operator $\curlywedge$ produces the *meet* of two search problems: the input to problem A $\curlywedge$ B is a pair (x, y) where x is an instance of A and y is an instance of B and the goal is to output either a solution to x or to y. Then SoD $\curlywedge$ EoL is the canonical (albeit "unnatural") complete problem for PLS $\cap$ PPAD [5].

Our new collapses

Our main results, Theorems 1 and 2, follow from two new reductions, the first one of which strengthens the reduction (1) from [8]:

$$\text{SoD} \curlywedge \text{EoL} \;\leq\; \text{EoPL} \qquad \text{which implies } \mathsf{PLS} \cap \mathsf{PPAD} \subseteq \mathsf{EOPL}, \tag{2}$$

$$\text{SoD} \curlywedge \text{SoL} \;\leq\; \text{SoPL} \qquad \text{which implies } \mathsf{PLS} \cap \mathsf{PPADS} \subseteq \mathsf{SOPL}. \tag{3}$$

These reductions are between purely combinatorially defined search problems. In the case of (2), this bypasses the continuous middle-man of CLS and makes our reduction relatively simple to describe. In particular, we get a new simpler proof of the breakthrough collapse of [8] by combining (2) with the inclusion EOPL $\subseteq$ CLS proved by [14]. Furthermore, the new collapse implies that problems related to Tarski's fixpoint theorem [7] and to a colourful version of Carathéodory's theorem [18] lie in EOPL.

A further surprise?

Given that the collapse CLS $=$ PLS $\cap$ PPAD was considered extremely surprising by most people, how surprised should we be by the further collapse

$$\mathsf{EOPL} \;=\; \mathsf{CLS} \;=\; \mathsf{PLS} \cap \mathsf{PPAD} \;?$$

Fearnley et al. [9] wrote regarding EOPL vs. CLS that "we actually think it could go either way." In the wake of their breakthrough, the paper [8] explicitly conjectured EOPL $\neq$ CLS.

For the authors of the present paper, the new collapse did come as an utter shock. When we began work on this project, our intuitions convinced us that, again, EOPL $\neq$ CLS, a conjecture which had just found its way to the second author's PhD thesis [13, Section 7.5].

In our convictions, we set out to prove this separation in the *black-box model* where, instead of circuits, the directed graphs are described by black-box oracles. We tried in vain for nine months. The upshot is that Theorems 1 and 2 now crush this possibility, as they hold even in the black-box model.

2 A Unified View: The Grid Problem

In this section we formally define all the problems of interest. We take the unusual approach of defining a single problem (which we call the GRID problem) with various parameters which can be tweaked to obtain all of the problems we study in this paper. This mainly serves two purposes. First of all, it is particularly convenient for presenting our reductions, since it allows us to combine instances from different problems more easily. The second reason is that we believe that this unified view of seemingly very different problems is of independent interest.

The Grid problem

For $n \in \mathbb{N}$, let $[n] \coloneqq \{1, 2, \ldots, n\}$. We define a general problem on a grid $[N] \times [M]$, where N and M should be thought of as being (potentially) exponentially large. The problem involves A paths starting from column 1 ($[N] \times \{1\}$) and moving from column i ($[N] \times \{i\}$) to column $i+1$ ($[N] \times \{i+1\}$). On the last column ($[N] \times \{M\}$) there are at most B valid ends of paths. If paths are not allowed to merge, then by the Pigeonhole Principle $A > B$ ensures the existence of a solution, i.e., a path that does not end at a valid position on the last column. If paths are allowed to merge, then a solution is guaranteed to exist as long as $B = 0$. To make things more precise, the paths start from nodes 1 to A in the first column (i.e., $[A] \times \{1\}$), and the valid termination points are nodes 1 to B in the last column (i.e., $[B] \times \{M\}$).

In more detail, we are given a boolean circuit $S\colon [N] \times [M] \to [N] \cup \{\mathsf{null}\}$, the *successor circuit*, which allows us to efficiently compute the outgoing edge at a node. If $S(x, y) = \mathsf{null}$, then (x, y) does not have an outgoing edge. Otherwise, there is an outgoing edge from (x, y) to $(S(x, y), y + 1)$. The problem also has two parameters which are used to tweak the definition: r (*reversible*) and b (*bijective*). Intuitively, when $r = 1$, we change the representation of paths to make them *reversible*. Namely, in addition to the successor circuit S, we are also given access to a *predecessor circuit* $P\colon [N] \times [M] \to [N] \cup \{\mathsf{null}\}$, which, analogously to S, allows us to efficiently compute the incoming edge at a node. In particular, when $r = 1$, every node can have at most one incoming edge, i.e., two paths cannot merge. When $r = 1$, the other parameter b is used to introduce additional solutions. Namely, when $b = 1$, then we do not allow any new paths apart from the original A paths, and we also require that all B valid ends of paths are actually reached by a path. The combination $r = 0, b = 1$ is not allowed.

We use the term *sink* to refer to a node with at least one incoming edge but no outgoing edge. Similarly, a *source* is a node with an outgoing edge but no incoming edge. The formal definition of the problem is as follows.

▶ **Definition 3.** *In the* GRID *problem, given* N, M, A, B *with* $N \geq A > B \geq 0$ *and* $M \geq 2$, *boolean circuits* $S, P\colon [N] \times [M] \to [N] \cup \{\mathsf{null}\}$, *and bits* $r, b \in \{0, 1\}$, *output any of the following:*

1. $x \in [A]$ *such that* $S(x, 1) = \mathsf{null}$, (missing pigeon/source)
2. $x \in [N]$ *such that* $S(x, M-1) > B$, (invalid hole/sink)
3. $x \in [N]$ *and* $y \in [M-2]$ *such that* $S(x, y) \neq \mathsf{null}$ *and* $S(S(x, y), y + 1) = \mathsf{null}$, (pigeon interception/sink)

4. *If* $r = 1$ *and* $b = 1$*:*
   a. $(x, y) \in ([N] \times [M-1]) \setminus ([A] \times \{1\})$ *such that* $S(x, y) \neq \mathsf{null}$ *and* $P(x, y) = \mathsf{null}$*, or* (pigeon genesis/source)
   b. $x \in [B]$ *such that* $P(x, M) = \mathsf{null}$. (empty hole/sink)

We also enforce the following two conditions syntactically:

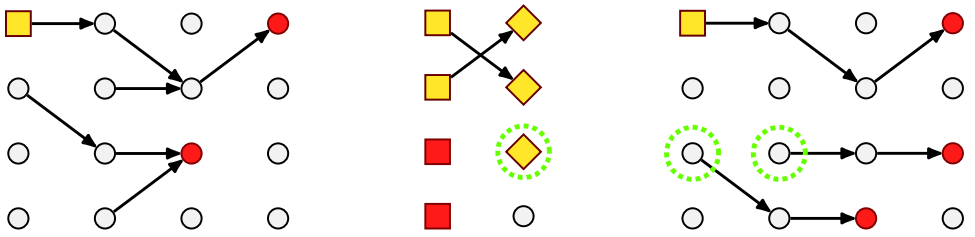
- *If* $r = 0$*, then* $b = 0$ *and* $B = 0$.
- *If* $r = 1$*, then the successor and predecessor circuits are consistent, which can be enforced as follows. The circuit* S *is replaced by the circuit* $\overline{S}$*, which on input* (x, y) *computes* $x' := S(x, y)$ *and outputs* x'*, unless* $(x', y+1) \notin [N] \times [M]$ *or* $P(x', y+1) \neq x$*, in which case it outputs* null*. Similarly, the circuit* P *is replaced by the circuit* $\overline{P}$*, which on input* (x, y) *computes* $x' := P(x, y)$ *and outputs* x'*, unless* $(x', y-1) \notin [N] \times [M]$ *or* $S(x', y-1) \neq x$*, in which case it outputs* null*.*

Canonical complete problems as special cases of Grid

As defined above, the inputs N, M, A, B of the Grid problem are completely unrestricted, apart from the natural restrictions $N \geq A > B \geq 0$ and $M \geq 2$. By imposing various additional restrictions on these inputs, we obtain the following canonical complete problems; see Figure 2. (Here inj-PHP/bij-PHP stand for Injective/Bijective Pigeonhole Principle.)

- SoD: $r = 0, b = 0, A = 1, B = 0$. (PLS-complete)
- SoPL: $r = 1, b = 0, A = 1, B = 0$. (SOPL-complete)
- EoPL: $r = 1, b = 1, A = 1, B = 0$. (EOPL-complete)
- inj-PHP: $r = 1, b = 0, M = 2, N = A = B + 1$. (PPADS-complete)
- bij-PHP: $r = 1, b = 1, M = 2, N = A = B + 1$. (PPAD-complete)

Note that beyond those restrictions, the inputs are left unrestricted. For example, in SoD, the input M can be very large, which is indeed needed for the problem to be PLS-complete.



(a) Sink-of-DAG (SoD). **(b)** inj-PHP. **(c)** Sink-of-Potential-Line (SoPL).

Figure 2 Examples of Grid problems. Square nodes are valid starts of paths (top-most A nodes in the first column) and diamonds are valid ends of paths (top-most B nodes in the last column). Solutions are drawn in red. However, for visual clarity we highlight the actual sinks rather than the *sink predecessors* as in Definition 3. Nodes with a null successor are drawn without an outgoing pointer. (2a) has parameters $(r = 0, b = 0, A = 1, B = 0)$ and defines an SoD instance. Only the successor circuit is drawn, as the predecessor circuit is not used by SoD. In particular, directed paths can *merge*, such as for node $(2, 3)$. (2b) has parameters $(r = 1, b = 0, A = N, B = N - 1)$ and defines an inj-PHP instance. The diamond with a green circle would be a solution of bij-PHP (with $b = 1$) but is not a solution of inj-PHP. (2c) has parameters $(r = 1, b = 0, A = 1, B = 0)$ and defines an SoPL instance. Sources with green circles would be solutions of an EoPL instance (with $b = 1$).

▶ Remark 4. Here we have slightly abused notation by calling these problems SoD, SoPL and EoPL even though their original definitions (in [16, 12, 9], respectively) do not use a grid structure, and instead come with an additional circuit computing the potential of any node. It is not too hard to see that these grid-versions of the problems are indeed polynomial-time equivalent to the original versions. The main idea is that the grid implicitly provides a potential value for every node (x, y), namely its column number y. Thus, given such a problem on a grid, it is easy to define a potential circuit by simply assigning the potential value y to any node (x, y) of the grid.

The other direction is slightly more involved. Consider an instance of one of the original problems with vertex set $V = [N]$ and potential values lying in $P = [M]$. Without loss of generality, we can assume that along any edge the potential increases by exactly one. Indeed, this was proved explicitly by [9] when they reduced EoPL to EoML, and the same idea applies to SoD and SoPL as well. The reduction to the grid-version of the problem is then obtained by identifying a vertex $x \in V$ that has potential $p \in P$ with the node (x, p) on the $[N] \times [M]$ grid.

The following is essentially folklore (see, e.g., [2]), so we only provide a brief proof sketch.

▶ **Lemma 5.** INJ-PHP *and* BIJ-PHP *are respectively* PPADS- *and* PPAD-*complete.*

Proof Sketch. To see that BIJ-PHP lies in PPAD, we can reduce to EoL (see, e.g., [4] for a formal definition) with vertex set $V = [N] \times [2]$ as follows: add a directed edge from node $(x, 2)$ to node $(x, 1)$ for all $x \leq A - 1$. By taking (x, A) as the distinguished source node, this yields an EoL instance with the same solutions as the original BIJ-PHP instance. On the other hand, given an instance of EoL with vertex set $V = [N]$ and distinguished source node N (without loss of generality), we construct an instance of BIJ-PHP on $[N] \times [2]$ as follows: for any isolated vertex $x \in [N]$, create an edge from $(x, 1)$ to $(x, 2)$; for any edge from x to y, create an edge from $(x, 1)$ to $(y, 2)$. This simple reduction proves the PPAD-hardness of BIJ-PHP. The exact same constructions can be used to prove that INJ-PHP is PPADS-complete, by reducing to and from the SoL problem (formally defined by Beame et al. [1], who call it Sink). ◀

In Section 6, we briefly explain how an extended version of the Grid problem can be used to also capture PPP, the class defined by Papadimitriou [20] to capture a version of the Pigeonhole Principle where edges can only be computed efficiently in the forward direction. We do not currently see any natural way of extending the definition of the Grid problem so that it also captures the class PPA.

3 Path-Pigeonhole Problems

In this section we use the Grid problem to define some interesting extensions of the two pigeonhole problems. Namely, we consider the case where, instead of just two columns, there are many columns. In a certain sense, this corresponds to allowing the pigeons to travel for a long time before reaching a hole. In particular, we can no longer efficiently tell in which hole a given pigeon will land. This allows us to show that the problems remain hard even when there are significantly more pigeons than holes. This fact, stated in Lemma 6 below, will be crucial to obtain our main result later.

Let $f\colon \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function with $f(t) > t$. In this section, we consider the following restrictions of Grid:

- Path-inj-PHP$_f$: $r = 1, b = 0, A = f(B)$.
- Path-bij-PHP$_f$: $r = 1, b = 1, A = f(B)$.

The following lemma is an important ingredient for the proof of our main result.

▶ **Lemma 6.** *Let $f(t) > t$ be polynomial-time computable. There exists a reduction* INJ-PHP $\leq$ PATH-INJ-PHP$_f$ *that maps an instance with parameters $(A, B) = (T + 1, T)$ to an instance with parameters $(A, B) = (f(T), T)$ and $(N, M) = (f(T), f(T) - T + 1)$.*

Proof. The idea behind this reduction is very simple. Intuitively, we have the ability to "merge" $T + 1$ paths into T paths by using the INJ-PHP instance. Namely, we can go from having $T + 1$ paths on some column i to having only T paths on the next column $i + 1$, and such that finding a "mistake", i.e., a path that stops between the two columns, requires solving the INJ-PHP instance. In particular, if we start with some N paths, where $N \geq T+1$, then we can "merge" those paths into $N - 1$ paths by "merging" the first $T + 1$ paths into T paths, and leaving the remaining $N-(T+1)$ paths unchanged. Applying this idea repeatedly, we can "merge" $f(T)$ paths into just T paths in $f(T) - T$ steps. This results in an instance of PATH-INJ-PHP$_f$ with $f(T) - T + 1$ columns, where every solution yields a solution to the INJ-PHP instance; see Figure 3. More formally, let (S, P) denote an instance of INJ-PHP with parameters $A = T + 1$ and $B = T$. Without loss of generality, we can assume that no pigeon goes to the invalid hole, i.e., $S(x, 1) \neq T + 1$ for all $x \in [T + 1]$. Indeed, if there is such an edge, we can just remove it and this does not change the set of solutions; the node pointing to the invalid hole was a solution before, and now it is still a solution, because it has no successor. We construct an instance $(\widehat{S}, \widehat{P})$ of PATH-INJ-PHP$_f$ on the $[N] \times [M]$ grid, where $N = f(T)$, $M = f(T) - T + 1$, $A = f(T)$ and $B = T$. The successor circuit $\widehat{S}$ is defined as follows:

$$\widehat{S}(x, y) \coloneqq \begin{cases} S(x, 1) & \text{if } x \in [T + 1] \text{ and } y \in [M - 1], \\ x - 1 & \text{if } T + 2 \leq x \leq f(T) - y + 1 \text{ and } y \in [M - 1], \\ \mathsf{null} & \text{otherwise,} \end{cases}$$

and the predecessor circuit $\widehat{P}$ is then defined accordingly to be consistent with $\widehat{S}$. Both circuits can be constructed in polynomial time, given S and P, and given that f can be computed in polynomial time. It is straightforward to check that any solution of the constructed instance yields a solution to the original INJ-PHP instance. ◀

The same proof idea also yields that BIJ-PHP $\leq$ PATH-BIJ-PHP$_f$.

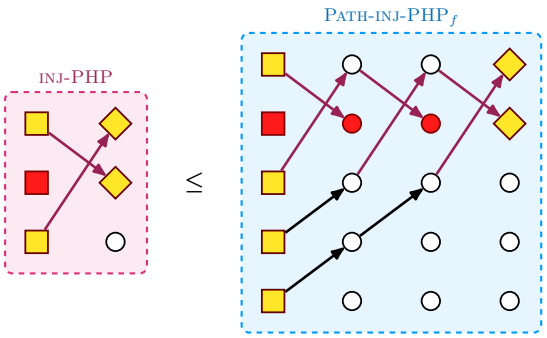


Figure 3 The reduction INJ-PHP $\leq$ PATH-INJ-PHP$_f$ in Lemma 6. We build a PATH-INJ-PHP$_f$ instance by chaining several identical INJ-PHP instances side-by-side. Note that a solution to the PATH-INJ-PHP$_f$ instance can be directly mapped to one for the original INJ-PHP instance.

4 SOPL = PLS ∩ PPADS

In this section, we prove Theorem 2, namely SOPL = PLS∩PPADS. To prove this we provide a reduction from a PLS∩PPADS-complete problem to the canonical SOPL-complete problem.

▶ **Lemma 7.** SoD ⅄ INJ-PHP ≤ SoPL.

Proof Sketch. There are two obstacles to a direct reduction from SoD to SoPL: (i) we can only compute edges in the forward direction (i.e., we only have access to a successor circuit), and (ii) multiple edges can point to the same node.

To resolve the first issue, we modify the original $[N] \times [M]$ grid of the SoD instance by taking N copies of each node. This ensures that there is a separate copy of each node v for each potential predecessor on the previous column. As a result, edges from different predecessor nodes will point to different copies of v. This means that predecessor nodes can now also be computed efficiently. Namely, in order to compute the predecessors of the ith copy of node v, it suffices to check whether in the original SoD instance the ith node on the previous column points to v. If that is the case, then all copies of this ith node are predecessors in the modified instance. Otherwise, there are no predecessors. However, the second issue remains: since we have made N copies of each node, there are also N copies of each predecessor node, and thus N edges pointing to the corresponding copy of v. This is not acceptable, since SoPL allows at most one incoming edge.

To overcome the second obstacle, we make use of the following high-level idea: use the INJ-PHP instance (which maps $K+1$ pigeons to K holes) to "hide" the fact that multiple edges can point to a single node. Unfortunately, we cannot use the INJ-PHP instance to hide the fact that N paths merge into a single node. But, if we take KN copies of each original node, instead of just N, then we have KN paths and K target nodes. By Lemma 6, the INJ-PHP instance can be turned into a PATH-INJ-PHP$_f$ instance that hides the fact that KN paths merge into K paths. Thus, we replace each original node v of the SoD instance by a gadget that has KN nodes in the left-most column and K nodes in the right-most column, and such that finding the sink of a path inside the gadget requires solving the INJ-PHP instance. Importantly, we only construct the paths inside this gadget when the original node v has a successor in the SoD instance. This ensures that when v is an isolated node, the corresponding gadget does not contain any edges. Figure 4 illustrates the construction for $N = 3$, $M = 4$ and $K = 2$.

Note that although we might have added many new sources to the graph (which are irrelevant for SoPL), it remains the case that from any sink of the new graph, we can extract either a solution to SoD or to INJ-PHP.

In the final construction, edges can indeed be computed in both directions efficiently. Namely, given any node, we can determine in polynomial time if it has an incoming and/or outgoing edge, as well as the identity of the potential predecessor and successor nodes. Here, we crucially use the fact that edges can be computed efficiently in both directions in the INJ-PHP instance. ◀

Proof. Let S be an instance of SoD on the grid $[N] \times [M]$. We are also given an instance of INJ-PHP with parameters $(A, B) = (K+1, K)$. Without loss of generality we can assume that $K = N$, because we can easily pad the SoD or INJ-PHP instance with additional rows without changing the set of solutions. By Lemma 6, we can reduce this INJ-PHP instance to a PATH-INJ-PHP$_{t^2}$ instance on the grid $[N^2] \times [M']$ with parameters $(A, B) = (N^2, N)$. Without loss of generality, we can assume that $M' = M$, because we can pad the SoD or PATH-INJ-PHP$_{t^2}$ instance with additional columns, if needed. This is not important for the reduction, but will be convenient.

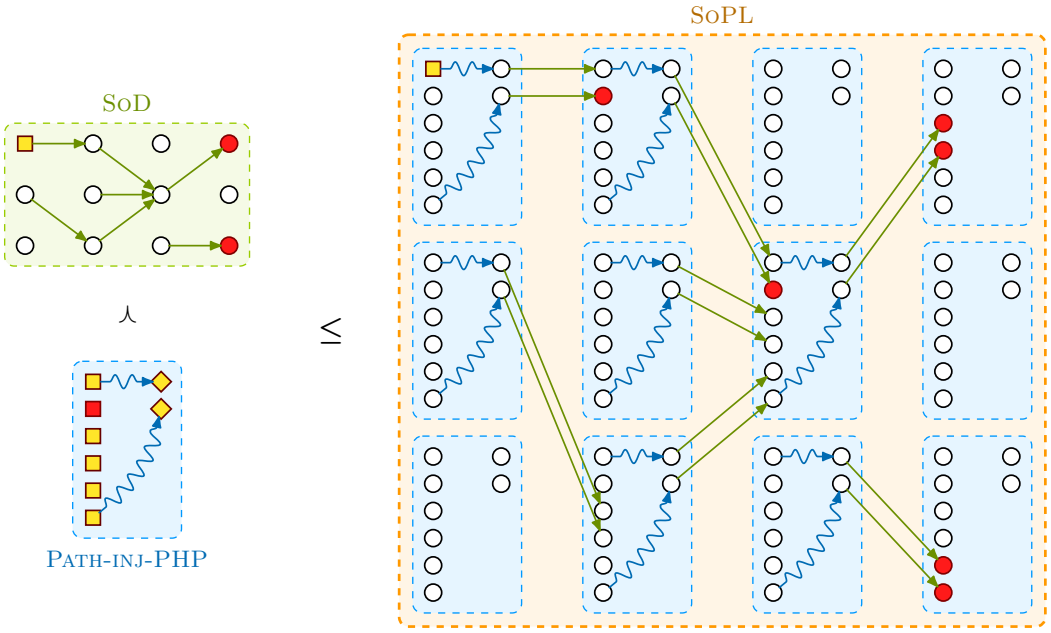


Figure 4 The reduction SoD ⋔ Path-inj-PHP ≤ SoPL in the proof of Lemma 7. Given instances of SoD and Path-inj-PHP, we build an SoPL instance whose solutions can be traced back to solutions of SoD ⋔ Path-inj-PHP. To overcome the issue of merging paths in SoD, the nodes of the SoD instance are replaced with a copy of a Path-inj-PHP gadget (in blue). Those gadgets are ultimately built out of the initial inj-PHP instance (not shown) using Lemma 6.

We will take N^2 copies of each node in the original SoD instance, and make M copies of each column. As a result, our SoPL instance will be defined on the $[N^3] \times [M^2]$ grid. It will be convenient to use some special notation to refer to points in this grid. For $\alpha \in [N^2]$ and $x \in [N]$, we use the notation (α, x) to denote the row $\alpha + (x-1) \cdot N^2 \in [N^3]$. This corresponds to indexing the αth copy of row x of the original instance. We also introduce some additional notation to index these $[N^2]$ copies: for $i, j \in [N]$, we let $[i, j] \coloneqq i + (j-1) \cdot N \in [N^2]$. Thus, $([i, j], x)$ denotes the $[i, j]$th copy of row x. The "$[i, j]$" notation essentially subdivides $[N^2]$ into N blocks containing N values each, which will be useful for routing incoming edges to the correct copy of a node. Using the analogous subdivision also on the columns, the notation $(\alpha, x; k, y) \in [N^2] \times [N] \times [M] \times [M]$ denotes the node $(\alpha + (x-1) \cdot N^2, k + (y-1) \cdot M) \in [N^3] \times [M^2]$. In particular, the notation $([i, j], x; k, y)$ is well-defined.

The circuits $\widehat{S}, \widehat{P}$ of the SoPL instance on $[N^3] \times [M^2]$ are defined as follows:

$$\widehat{S}([i,j],x;k,y) \coloneqq \begin{cases} ([i,x], S(x,y)) & \text{if } k = M \text{ and } j = 1, \\ (S'([i,j],k), x) & \text{if } k < M \text{ and } S(x,y) \neq \mathsf{null}, \\ \mathsf{null} & \text{otherwise} \end{cases}$$

$$\widehat{P}([i,j],x;k,y) \coloneqq \begin{cases} ([i,1], j) & \text{if } k = 1 \text{ and } y > 1 \text{ and } S(j, y-1) = x, \\ (P'([i,j],k), x) & \text{if } k > 1 \text{ and } S(x,y) \neq \mathsf{null}, \\ \mathsf{null} & \text{otherwise} \end{cases}$$

where $(\alpha, z) \in [N^2] \times [N]$ is interpreted as an element in $[N^3]$ as above, and where we use the convention $(*, \mathsf{null}) = (\mathsf{null}, *) = \mathsf{null}$. Using the fact that S' and P' are consistent, it can be checked that $\widehat{S}$ and $\widehat{P}$ are also consistent.

In order to argue about the correctness of the reduction, consider any sink $([i,j],x;k,y)$ of the SoPL instance. If $2 \leq k \leq M-1$, then it must be that $([i,j],k)$ is a sink of the Path-inj-PHP$_{t^2}$ instance (S',P'). If $k=M$ and $j=1$, then $([i,j],x;k,y)$ cannot be a sink, since $\widehat{P}([i,j],x;k,y) \neq \mathsf{null}$ implies that $S(x,y) \neq \mathsf{null}$, and thus $\widehat{S}([i,j],x;k,y) \neq \mathsf{null}$. If $k=M$ and $j>1$, then $([i,j],k)$ is an invalid sink on the last column of the Path-inj-PHP$_{t^2}$ instance, and so in particular a solution. If $k=1$ and $S(x,y) \neq \mathsf{null}$, then $([i,j],k)$ is a missing source on the first column of the Path-inj-PHP$_{t^2}$ instance, and so again a solution. Finally, if $k=1$ and $S(x,y)=\mathsf{null}$, then it must be that $S(j,y-1)=x$ and thus (x,y) is a sink of the original SoD instance, and this is witnessed by the node $(j,y-1)$. ◀

5 EOPL = PLS ∩ PPAD

In this section, we prove Theorem 1, namely EOPL = PLS ∩ PPAD. The equality SOPL = PLS ∩ PPADS (Theorem 2) proved in the previous section, together with the fact that PPAD ⊆ PPADS, immediately imply that

$$\mathsf{SOPL} \cap \mathsf{PPAD} = \mathsf{PLS} \cap \mathsf{PPAD}.$$

As a result, in order to prove Theorem 1, it suffices to give a reduction from an SOPL∩PPAD-complete problem to an EOPL-complete problem:

▶ **Lemma 8.** SoPL ⅄ bij-PHP ≤ EoPL.

Proof Sketch. A very natural attempt at a reduction from SoPL to EoPL is to try to remove all undistinguished sources, i.e., all sources except the trivial one. Then, clearly, any EoPL-solution would have to be a sink, and thus also a solution to SoPL.

There is a simple trick that *almost* achieves this. First, make a *reversed* copy of the SoPL instance, i.e., reverse the direction of all edges, and the ordering of the potential. Note that sources of the original instance have now become sinks in the reversed copy, and vice versa. Then, for each source node v of the original graph, add an edge pointing from its copy $\overline{v}$ (which is a sink) to v.

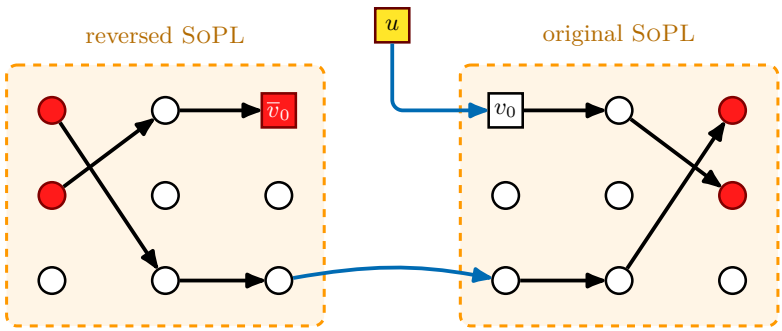


Figure 5 A naive attempt at a reduction SoPL ≤ EoPL. Although most solutions arise from the sinks of the original SoPL instance, a spurious solution is introduced at $\overline{v}_0$, which does not correspond to any sink of the original instance.

The only problem with this reduction is that we have eliminated *all* sources of the original graph, including the distinguished one. In particular, the distinguished source v_0 of the original instance is no longer a source, since there is an edge from its copy $\overline{v}_0$ to v_0. As a result, the reduction fails, because the instance of EoPL we have constructed does not have a distinguished source. Furthermore, we cannot hope to turn one of the new sources into a distinguished source, since any such source yields a solution to the original instance (where it is a sink).

In order to address this issue, we add a new node u and select it as our new distinguished source. Clearly, u is a solution of the instance, since it is a distinguished source that is not actually a source, but just an isolated node. Now, imagine that we remove the edge $(\overline{v}_0, v_0)$ and instead introduce an edge (u, v_0); see Figure 5. Then, u is no longer a solution, but $\overline{v}_0$ becomes a sink, and thus a solution, instead. In other words, the reduction can pick whether it wants u or $\overline{v}_0$ to be a solution by changing this edge. Of course, in both cases, the resulting instance is very easy to solve, but this minor observation already provides the idea for the next step.

Take k copies of the instance we have constructed (before adding u). There are now k copies $v_0^{(1)}, \dots, v_0^{(k)}$ of the original distinguished source, and k copies of the reverse copy $\overline{v}_0^{(1)}, \dots, \overline{v}_0^{(k)}$. Remove the edges $(\overline{v}_0^{(i)}, v_0^{(i)})$ for $i = 1, \dots, k$. If we now introduce the new distinguished source u, we have $k+1$ nodes that "need" an outgoing edge in order to not be solutions (namely, $u, \overline{v}_0^{(1)}, \dots, \overline{v}_0^{(k)}$) and k nodes that "need" an incoming edge (namely, $v_0^{(1)}, \dots, v_0^{(k)}$). Clearly, no matter how we introduce edges here, one of $u, \overline{v}_0^{(1)}, \dots, \overline{v}_0^{(k)}$ will not have an outgoing edge and will be a solution. However, we can use a BIJ-PHP instance to make it hard to find such a solution. Let K denote the parameter of the BIJ-PHP instance, i.e., $K+1$ points are mapped to K points. Then, we let $k := K$ and add edges between $u, \overline{v}_0^{(1)}, \dots, \overline{v}_0^{(k)}$ and $v_0^{(1)}, \dots, v_0^{(k)}$ according to the BIJ-PHP instance. An example of the construction is depicted in Figure 6.

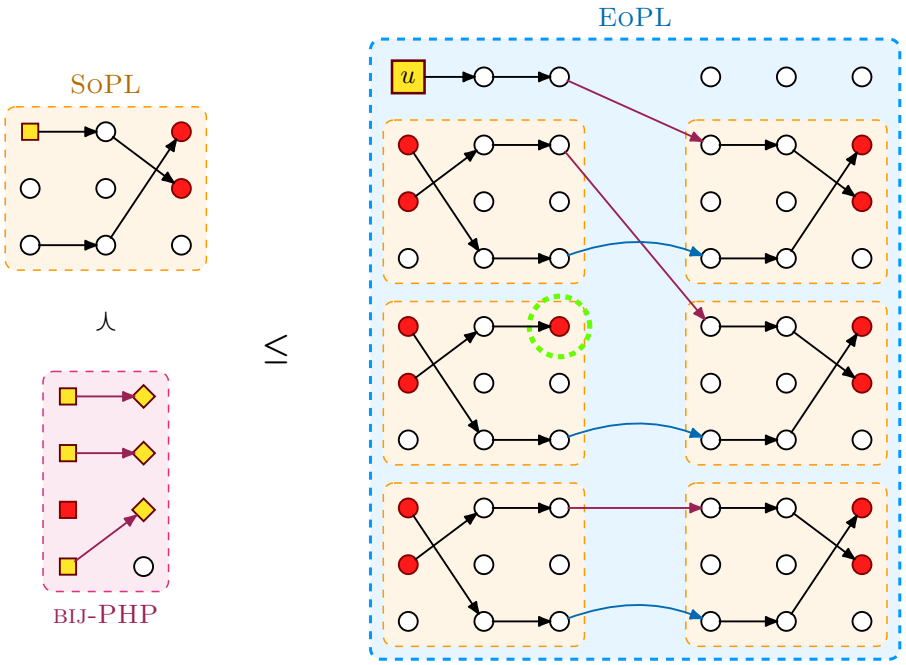


Figure 6 The reduction SoPL 人 BIJ-PHP $\leq$ EoPL in Lemma 8. The BIJ-PHP instance (in pink) connects the newly introduced source u together with the distinguished sources and sinks of the copied SoPL instances. Non-distinguished sources and sinks are connected with blue edges. The node circled in green corresponds to a solution of the BIJ-PHP instance.

Now, it is easy to check that any undistinguished source or any sink of the resulting graph must yield a solution to the BIJ-PHP instance or a solution of the SoPL instance. In particular, if u is not a source, then this yields a solution to BIJ-PHP. ◀

Proof. Let (S, P) be an instance of SoPL on the grid $[N] \times [M]$. Without loss of generality, we can assume that all sources occur on the first column, i.e., for any source $(x, y) \in [N] \times [M]$ it holds that $y = 1$. Indeed, by appropriately increasing N, for each source (x, y) we can add a path that starts on the first column and ends at (x, y), thus effectively "transferring" the source to the first column. Let (S', P') be an instance of BIJ-PHP on the grid $[K + 1] \times [2]$ that maps $K + 1$ pigeons to K holes.

We take K copies of the SoPL instance and K copies of the reversed SoPL instance, all together in a single grid. This grid will be of the form $[KN] \times [2M]$. For clarity, we will use the notation $(i, x; y) \in [K] \times [N] \times [2M]$ to denote the element $(x + (i-1) \cdot N, y) \in [KN] \times [2M]$. The ith copy of the instance will be embedded in $\{i\} \times [N] \times ([2M] \setminus [M])$, while the ith reversed copy will be in $\{i\} \times [N] \times [M]$. Formally, we define new successor and predecessor circuits $\widehat{S}, \widehat{P}$ on $[KN] \times [2M]$ as follows:

$$\widehat{S}(i, x; y) \coloneqq \begin{cases} (i, P(x, M - y + 1)) & \text{if } y \leq M, \\ (i, S(x, y - M)) & \text{if } y \geq M + 1 \end{cases}$$

$$\widehat{P}(i, x; y) \coloneqq \begin{cases} (i, S(x, M - y + 1)) & \text{if } y \leq M, \\ (i, P(x, y - M)) & \text{if } y \geq M + 1 \end{cases}$$

where $(i, z) \in [K] \times [N]$ represents the element $z + (i-1) \cdot N \in [KN]$, and where we use the convention $(i, \mathsf{null}) = \mathsf{null}$.

Since S and P are consistent, $\widehat{S}$ and $\widehat{P}$ are also consistent. Note that there are currently no edges between column M and column $M + 1$. In the second step of the reduction we add edges between these two columns as follows. For every $i \in [K]$ and $x \in [N] \setminus \{1\}$, if $(i, x; M+1)$ is a source, then we add an edge from $(i, x; M)$ to $(i, x; M+1)$. Note that in that case $(i, x; M)$ was a sink. The case where $x = 1$ is handled separately, because it corresponds to nodes that are copies of the distinguished source of the original SoPL instance. For any $i \in [K]$ and for $x = 1$, if $S'(i, 1) = j \neq \mathsf{null}$, we add an edge from $(i, 1; M)$ to $(j, 1; M + 1)$. Note that here we also use P' (which is assumed to be consistent with S') to implement this edge in $(\widehat{S}, \widehat{P})$.

Finally, we introduce a new special node u on column M which will act as our new distinguished source. If $S'(K + 1, 1) = j \neq \mathsf{null}$, then we add an edge from u to $(j, 1; M + 1)$. By extending the grid to be $[KN + 1] \times [2M]$, by renaming nodes and by "transferring" the source u to the first column as before, we can ensure that the distinguished source is $(1, 1)$.

It is easy to check that the new circuits $\widehat{S}, \widehat{P}$ can be constructed in polynomial time. For the correctness of the reduction, note that any source or sink that occurs on columns $[2M] \setminus \{M, M + 1\}$ must correspond to a sink of the original SoPL instance. On the other hand, any source or sink that occurs on column M or $M + 1$ must correspond to a solution of the BIJ-PHP instance (namely, a pigeon without a hole, or a hole without a pigeon). This completes the reduction. ◀

6 Discussion

As mentioned in the introduction, it remains open whether $\mathsf{UEOPL} \stackrel{?}{=} \mathsf{EOPL}$. Separating the two classes in the black-box model would be an important first step towards pinning down the complexity of the various natural problems contained in UEOPL, since it would provide strong evidence that these problems are unlikely to be complete for $\mathsf{PLS} \cap \mathsf{PPAD}$.

The techniques developed in this paper do not seem to yield any other major class collapse. Indeed, our reductions are all black-box, and the main classes are known to be distinct in that model [1, 19, 2]. A notable exception is the question of whether PLS is a subset of PPP, or even of PPADS. This remains open even in the black-box model.

In the remainder of this section we briefly present some observations about the path pigeonhole problems, as well as a further consequence of our reduction techniques: a version of SoD where paths are not allowed to merge turns out to be PLS ∩ PPP-complete.

Path-Pigeonhole problems

Lemma 6 in particular establishes that Path-inj-PHP$_f$ is PPADS-hard. Membership in PPADS can be shown by reducing to inj-PHP using a construction similar to the reduction from EoL to bij-PHP in the proof of Lemma 5.

The statement of Lemma 6 also holds for bij-PHP $\leq$ Path-bij-PHP$_f$, and the proof is essentially the same. This shows that Path-bij-PHP$_f$ is PPAD-hard. However, it is unclear whether Path-bij-PHP$_f$ lies in PPAD. Indeed, using the same idea as for Path-inj-PHP$_f \leq$ inj-PHP yields an instance with $A \gg B$, and we cannot increase B artificially here (whereas this is possible in inj-PHP). Another way to state this is to say that we can reduce Path-bij-PHP$_f$ to an instance of EoL that has many distinguished source nodes, instead of just one. It is known that EoL with a polynomial number of distinguished sources remains PPAD-complete [11], but in general we will obtain an exponential number of such sources here.

Extending the Grid problem to capture PPP

The canonical PPP-complete problem is Pigeon-Circuit [20]: given a circuit mapping N pigeons to $N-1$ holes, find a *collision*, i.e., two pigeons that are mapped to the same hole. Importantly, unlike in inj-PHP or bij-PHP, we are not given a circuit to compute the mapping in the other direction, i.e., from holes to pigeons. In order to capture this problem, we extend the definition of Grid by introducing an additional parameter bit $c \in \{0,1\}$, which stands for *collision*. We also introduce a new solution type:

5. If $r = 0$ and $c = 1$: $x_1, x_2 \in [N]$ and $y \in [M-1]$ such that
$x_1 \neq x_2$ and $S(x_1, y) = S(x_2, y) \neq \mathsf{null}$, *(pigeon collision/merging)*

Furthermore, the syntactic condition "If $r = 0$, then $b = 0$ and $B = 0$" is replaced by the condition:

- If $r = 0$, then $b = 0$. If $r = 0$ and $c = 0$, then $B = 0$.

The PPP-complete problem Pigeon-Circuit is then obtained by setting $r = 0, c = 1, M = 2, N = A = B + 1$. In fact, Grid remains in PPP even if we just set $r = 0, c = 1$ and leave the other parameters unfixed. This can be shown by using a construction similar to the reduction from EoL to bij-PHP in the proof of Lemma 5.

SoD without merging

What is the complexity of SoD if paths are not allowed to merge? In other words, what is the complexity of the Grid problem with parameters $r = 0, c = 1, A = 1, B = 0$? Clearly, this restricted version still lies in PLS, and by the previous paragraph it also lies in PPP. Using the ideas developed in this paper, it can be shown that the problem is in fact PLS ∩ PPP-complete. To see this, note that using the simple construction in the proof of Lemma 6 we can reduce Pigeon-Circuit to a path-version of the problem where $f(T)$ pigeons are mapped to T holes. Then, the construction in the proof of Lemma 7 can be used to reduce SoD $\curlywedge$ Pigeon-Circuit to SoD without merging.

References

1 Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

2 Joshua Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC)*, pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

3 Man-Kwun Chiu, Aruni Choudhary, and Wolfgang Mulzer. Computational Complexity of the α-Ham-Sandwich Problem. In *47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 31:1–31:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.31.

4 Constantinos Daskalakis, Paul Goldberg, and Christos Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi:10.1137/070699652.

5 Constantinos Daskalakis and Christos Papadimitriou. Continuous local search. In *Proceedings of the 22nd Symposium on Discrete Algorithms (SODA)*, pages 790–804, January 2011. doi:10.1137/1.9781611973082.62.

6 Jérôme Dohrau, Bernd Gärtner, Manuel Kohler, Jiří Matoušek, and Emo Welzl. ARRIVAL: A zero-player graph game in NP ∩ coNP. In *A Journey Through Discrete Mathematics*, pages 367–374. Springer, 2017. doi:10.1007/978-3-319-44479-6_14.

7 Kousha Etessami, Christos Papadimitriou, Aviad Rubinstein, and Mihalis Yannakakis. Tarski's theorem, supermodular games, and the complexity of equilibria. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 18:1–18:19, 2020. doi:10.4230/LIPIcs.ITCS.2020.18.

8 John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. In *Proceedings of the 53rd Symposium on Theory of Computing (STOC)*, pages 46–59, 2021. doi:10.1145/3406325.3451052.

9 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. doi:10.1016/j.jcss.2020.05.007.

10 Bernd Gärtner, Thomas Dueholm Hansen, Pavel Hubácek, Karel Král, Hagar Mosaad, and Veronika Slívová. ARRIVAL: Next stop in CLS. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107, pages 60:1–60:13. Schloss Dagstuhl, 2018. doi:10.4230/LIPICS.ICALP.2018.60.

11 Paul Goldberg and Alexandros Hollender. The Hairy Ball problem is PPAD-complete. *Journal of Computer and System Sciences*, 122:34–62, 2021. doi:10.1016/j.jcss.2021.05.004.

12 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124, pages 38:1–38:19, 2018. doi:10.4230/LIPIcs.ITCS.2019.38.

13 Alexandros Hollender. *Structural results for total search complexity classes with applications to game theory and optimisation*. PhD thesis, University of Oxford, 2021. URL: https://ora.ox.ac.uk/objects/uuid:67e2d80b-76bf-4b49-9b7d-8bbd91633dd7.

14 Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM Journal on Computing*, 49(6):1128–1172, 2020. doi:10.1137/17m1118014.

15 Takashi Ishizuka. The complexity of the parity argument with potential. *Journal of Computer and System Sciences*, 120:14–41, September 2021. doi:10.1016/j.jcss.2021.03.004.

16 David Johnson, Christos Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, August 1988. doi:10.1016/0022-0000(88)90046-3.

17 Nimrod Megiddo and Christos Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, April 1991. doi:10.1016/0304-3975(91)90200-L.

18 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrabezolles, and Yannik Stein. The rainbow at the end of the line—a PPAD formulation of the colorful Carathéodory theorem with applications. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1342–1351, 2017. doi:10.1137/1.9781611974782.87.

19 Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. Master's thesis, University of Toronto, 2001. URL: https://www.collectionscanada.ca/obj/s4/f2/dsk3/ftp04/MQ58775.pdf.

20 Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, June 1994. doi:10.1016/s0022-0000(05)80063-7.

21 Patrick Schnider. The Complexity of Sharing a Pizza. In *32nd International Symposium on Algorithms and Computation (ISAAC)*, pages 13:1–13:15, 2021. doi:10.4230/LIPIcs.ISAAC.2021.13.

Improved Pseudorandom Generators for AC^0 Circuits

Xin Lyu
Department of EECS, University of California at Berkeley, CA, USA

Abstract

We give PRG for depth-d, size-m AC^0 circuits with seed length $O(\log^{d-1}(m)\log(m/\varepsilon)\log\log(m))$. Our PRG improves on previous work [27, 25, 18] from various aspects. It has optimal dependence on $\frac{1}{\varepsilon}$ and is only one "$\log\log(m)$" away from the lower bound barrier. For the case of $d = 2$, the seed length tightly matches the best-known PRG for CNFs [5, 26].

There are two technical ingredients behind our new result; both of them might be of independent interest. First, we use a partitioning-based approach to construct PRGs based on restriction lemmas for AC^0. Previous works [27, 25, 18] usually built PRGs on the Ajtai-Wigderson framework [1]. Compared with them, the partitioning approach avoids the extra "$\log(n)$" factor that usually arises from the Ajtai-Wigderson framework, allowing us to get the almost-tight seed length. The partitioning approach is quite general, and we believe it can help design PRGs for classes beyond constant-depth circuits.

Second, improving and extending [27, 25, 18], we prove a full derandomization of the powerful multi-switching lemma [13]. We show that one can use a short random seed to sample a restriction, such that a family of DNFs simultaneously simplifies under the restriction with high probability. This answers an open question in [18]. Previous derandomizations were either partial (that is, they pseudorandomly choose variables to restrict, and then fix those variables to truly-random bits) or had sub-optimal seed length. In our application, having a fully-derandomized switching lemma is crucial, and the randomness-efficiency of our derandomization allows us to get an almost-tight seed length.

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

Keywords and phrases pseudorandom generators, derandomization, switching Lemmas, AC^0

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.34

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/021/

Acknowledgements I would like to thank my advisor, Avishay Tal, for numerous insightful discussions during the project. I am grateful to Lijie Chen and Avishay Tal for helpful comments on an early draft, which helped me improve the presentation significantly. Finally, thank anonymous CCC reviewers for useful comments.

1 Introduction

Let $\mathcal{F}$ be a class of functions. A pseudorandom generator (PRG) for $\mathcal{F}$ is an algorithm $G : \{0,1\}^s \to \{0,1\}^n$ that maps a short random seed x into a longer string $\mathcal{G}(x)$ that appears random to every distinguisher in $\mathcal{F}$. More specifically, we say that $\mathcal{G}$ ε-fools $\mathcal{F}$, if for every $f \in \mathcal{F}$, it holds

$$|\mathbb{E}_{x\sim\mathcal{U}_s}[f(G(x))] - \mathbb{E}_{x\sim\mathcal{U}_n}[f(x)]| \leq \varepsilon.$$

In this work, we consider the class of bounded-depth Boolean circuits (aka. AC^0 circuits), a circuit class that has been studied extensively over the past few decades. Constructing PRGs for AC^0 circuits is a central problem that has been studied extensively [1, 14, 24, 3, 26, 12, 27, 25, 18].

Quantitatively, let $\mathsf{AC}^0_d[m(n)]$ be the class of depth-d, size-m circuits. By the probabilistic method, one can show that there exists an ε-PRG for $\mathsf{AC}^0_d[m(n)]$ with seed length $O(\log(m/\varepsilon))$. However, finding an explicit PRG with the same seed length seems beyond current technique. In particular, any PRG for $\mathsf{AC}^0_d[m(n)]$ with seed length $\log^{o(d)}(m)$ would give a non-trivial PRG for NC^1 and imply that $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ (see, e.g., [8, Appendix A]). Also, it was observed in [27, 25, 18] that, if there is an explicit pseudorandom generator with seed length $o(\log^d(m/\varepsilon))$ for $\mathsf{AC}^0_d[m(n)]$, then there is an explicit function that requires AC^0_d-circuits of size $2^{\omega(n^{1/(d-1)})}$ to compute, improving Håstad's lower bound [14] that has resisted attack for more than 30 years!

There is an extensive line of work [1, 14, 24, 3, 26, 12, 27, 25, 18] aiming to construct better and better PRGs for AC^0. The seminal paper by Ajtai and Wigderson [1] gave the first non-trivial pseudorandom generator for AC^0. Their PRG has seed length $n^{o(1)}$ for polynomial-size AC^0 circuits. Later, Nisan constructed PRG for AC^0 circuits by applying Håstad's correlation bound [14] to the Nisan-Wigderson "hardness-to-randomness" framework [24]. Nisan's PRG has seed length $\log^{2d+O(1)}(m/\varepsilon)$ when ε-fooling $\mathsf{AC}^0_d[m(n)]$ circuits. A breakthrough result by Braverman [3] showed that any $\log^{O(d^2)}(m/\varepsilon)$-wise independent distribution ε-fools $\mathsf{AC}^0_d[m(n)]$. Combined with the standard construction of k-wise independent distribution, this gave a PRG with seed length $\log^{O(d^2)}(m/\varepsilon)$. Braverman's analysis was further sharpened by Tal [26] and by Harsha and Srinivasan [12], bringing the seed length down to $\log^{3d+O(1)}(m)\log(1/\varepsilon)$. The Ajtai-Wigderson technique was revisited by Trevisan and Xue [27], who constructed a PRG with seed length $\log^{d+O(1)}(m/\varepsilon)$. Recently there were two incomparable improvements over the Trevisan-Xue result, one by Servedio and Tan [25] with seed length $\log^{d+O(1)}(m)\cdot\log(1/\varepsilon)$ (i.e., it had optimal dependence on $\frac{1}{\varepsilon}$), and the other by Kelley [18], who got seed length $\widetilde{O}(\log^d(m/\varepsilon)\log n)$.

1.1 Our Result

The main result of this work is a new PRG for $\mathsf{AC}^0_d[m]$ with improved seed length $O(\log^{d-1}(m)\cdot\log(m/\varepsilon)\cdot\log\log m)$.

▶ **Theorem 1.** *For every $d\in\mathbb{N}$ the following is true. For every $m,n\in\mathbb{N}$ such that $m\geq n$ and every $\varepsilon>0$, there is an ε-PRG for $\mathsf{AC}^0_d[m]$ circuits with seed length $O(\log^{d-1}(m)\cdot\log(m/\varepsilon)\cdot\log\log(m))$.*

Our PRG construction improves two incomparable results by Servedio, Tan [25] and Kelley [18]. Its seed length has optimal dependence on $\frac{1}{\varepsilon}$, and is only one "$\log\log m$" away from the barrier of Håstad's lower bounds [14, 13]. For the case of $d=2$, the seed length becomes $O(\log(m)\log(m/\varepsilon)\log\log m)$, tightly matching the best-known PRG for CNFs [5, 26]. Furthermore, if the $\log\log(m)$ term in the PRG for CNF can be shaved, then our construction directly implies PRG for depth-d circuits with seed length $O(\log^{d-1}(m)\log(m/\varepsilon))$, tightly matching current hardness bounds for AC^0 circuits. Interpreted from the "hardness-to-randomness" perspective [24], our result has converted *almost* all the "hardness" against AC^0 into pseudorandomness for AC^0.

2 Techniques

Our PRG crucially depends on two new technical ingredients. Both of them might be of independent interest. First, we show a framework to construct PRGs based on switching lemmas[1]. Our framework shares some similarities with the seminal Ajtai-Wigderson framework [1] but

[1] More generally, just like the Ajtai-Wigderson framework, our framework can apply to any kind of "simplify-under-restriction" lemmas (e.g., the shrinkage lemma for De-Morgan formulae).

achieves shorter seed length. Second, improving and extending results from [27, 25, 18], we show a fully-derandomized multi-switching lemma for small-width DNFs. That is to say, we give an algorithm that samples a pseudorandom restriction from a short random seed, such that a family of DNFs *simultaneously* simplifies under the restriction with high probability. Applying our framework with the new derandomization gives PRGs for AC^0 circuits with the claimed seed length.

Notation

We define some useful pieces of notation first. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function. Let $\Lambda \subseteq [n]$ be a set and $x \in \{0,1\}^n$ be a string. A set-string pair (Λ, x) gives a restriction to f. The restricted function is denoted by $f|_{\Lambda[x,\star]}$ and is defined as $f|_{\Lambda[x,\star]}(y) = f(\Lambda[x,y])$, where

$$\Lambda[x,y]_i = \begin{cases} x_i & i \notin \Lambda \\ y_i & i \in \Lambda \end{cases}.$$

Intuitively, this means that all but the Λ part of the input is fixed to the corresponding bits in x, and $f|_{\Lambda[x,\star]}$ is now only a function of those Λ bits.

Let $\mathbf{\Lambda} \subseteq [n]$ be a random variable. We say that $\mathbf{\Lambda}$ has marginal p, if for each $i \in [n]$ it holds that $\Pr[i \in \mathbf{\Lambda}] = p$. We say that $\mathbf{\Lambda}[\mathbf{U}, \star]$ is a (truly) p-random restriction, if each $i \in [n]$ is included in $\mathbf{\Lambda}$ independently with probability p, and $\mathbf{U}$ is a uniformly random n-bit string. We always use $\mathbf{U}$ to denote the uniform distribution over $\{0,1\}^n$.

2.1 A Partitioning-Based PRG

Our almost-tight (with respect to the lower bounds barrier) seed length crucially depends on a new approach to construct PRGs, which we call a partitioning-based approach. Previous best PRGs for AC^0 [25, 18] were built on the iterative restriction framework, developed by Ajtai and Wigderson in their seminal work [1]. Using a partitioning strategy, we get simpler proof for the correctness of our PRG with even improved seed length. We believe the partitioning-based approach could also apply to function classes beyond bounded-depth circuits.

The PRG Template

We briefly describe our construction. Suppose we want to design a PRG for a circuit class $\mathcal{C}_{goal}$. What we have is a pseudorandom distribution $\mathbf{X} \in \{0,1\}^n$ for another related circuit class $\mathcal{C}_{simple}$. We further assume a derandomized "simplify-under-restriction" lemma: there is an integer $k \geq 1$, a real $p > 0$ and a pseudorandom distribution $\mathbf{Y} \in \{0,1\}^n$ satisfying the following:

- Let $\mathbf{\Lambda} \subseteq [n]$ be a k-wise independent set with marginal p. Then for every $\mathcal{C}_{goal}$ circuit C, with high probability over $\mathbf{\Lambda}, \mathbf{Y}$, the restricted circuit

  $$C|_{\mathbf{\Lambda}[\mathbf{Y},\star]}(x) := C(\mathbf{\Lambda}[\mathbf{Y}, x])$$

  is in $\mathcal{C}_{simple}$.

Then, we choose $w = \frac{1}{p}$ and let $\mathbf{H} : [n] \to [w]$ be a k-wise independent hash function. Consider the following distribution

$$\mathbf{B} = \mathbf{Y} \oplus (\mathbf{X}^{(1)} \wedge \mathbf{H}_1) \oplus (\mathbf{X}^{(2)} \wedge \mathbf{H}_2) \oplus \cdots \oplus (\mathbf{X}^{(w)} \wedge \mathbf{H}_w).$$

Here, $\oplus, \wedge$ denote bitwise $\mathsf{XOR}, \mathsf{AND}$ respectively. $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(w)}$ are w independent copies of $\mathbf{X}$.

The idea is, for every $i \in [w]$, if we zoom in and check the set $\mathbf{H}^{-1}(i)$, we see that $\mathbf{H}^{-1}(i)$ is a k-wise independent set with marginal $\frac{1}{w}$. Let $C \in \mathcal{C}_{goal}$ be a circuit that we wish to fool. Imagine that we sample all but the $\mathbf{X}^{(i)}$ part of $\mathbf{B}$ first. We then calculate $\mathbf{Z}_i := \mathbf{Y} \oplus \bigoplus_{j \neq i}(\mathbf{X}^{(j)} \wedge \mathbf{H}_j)$ and consider the restricted function $C|_{\mathbf{\Lambda}[\mathbf{Z}_i,\star]}$. We hope that with high probability over $\mathbf{H}$ and $\mathbf{Z}_i$, the restricted function is in $\mathcal{C}_{simple}$, which can be fooled by $\mathbf{X}^{(i)}$. As we will show, this is indeed the case with one minor technicality[2]. Therefore, we conclude that C cannot distinguish $\mathbf{B}$ from another distribution where we replace the $\mathbf{X}^{(i)}$ part in $\mathbf{B}$ with a uniform random string $\mathbf{U}^{(i)}$. Applying a hybrid argument allows us to show that C fails to distinguish between $\mathbf{B}$ and

$$\mathbf{B}' = \mathbf{Y} \oplus (\mathbf{U}^{(1)} \wedge \mathbf{H}_1) \oplus (\mathbf{U}^{(2)} \wedge \mathbf{H}_2) \oplus \cdots \oplus (\mathbf{U}^{(w)} \wedge \mathbf{H}_w) \equiv \mathbf{U},$$

implying that $\mathbf{B}$ fools $\mathcal{C}_{goal}$ circuits.

Assume the seed length to sample $\mathbf{Y}, \mathbf{H}$ is short enough so that it would not be the bottleneck to sample $\mathbf{B}$. Then, the seed length for sampling $\mathbf{B}$ is larger than that for $\mathbf{X}$ by a factor of $w = O(1/p)$.

Application to AC^0 circuits

In the next section, we will show a derandomized simplify-under-restriction lemma (i.e., the derandomized Håstad's multi-switching lemma) for AC^0 circuits. For now let us assume the lemma, which works with parameter $\frac{1}{p} = O(\log m)$ and simplifies depth-d circuits to depth-$(d-1)$ circuits with high probability. Plugging the lemma in our framework gives a PRG for AC^0_d circuits with seed length longer than AC$^0_{d-1}$-PRG by $O(\log m)$. Currently, the best PRG for depth-2 circuits (i.e., CNF/DNFs) has seed length $O(\log(m)\log(m/\varepsilon)\log\log m)$ ([5, 26]). Using it as our starting point, for every $d \geq 3$ we construct a PRG for depth-d circuits with seed length $O(\log^{d-1}(m)\log(m/\varepsilon)\log\log(m))$, as claimed.

Comparison with the Ajtai-Wigderson framework

Ajtai and Wigderson were the first to apply restriction lemmas to construct PRGs [1]. They developed the so-called "iterative restrictions" framework and gave the first non-trivial PRG for AC^0 circuits. Compared with the Nisan-Wigderson "hardness-to-randomness" framework [24], the Ajtai-Wigderson framework can "open up" the black box of lower bounds proof, which enables us to construct short PRGs for some delicate circuit classes (e.g., read-once AC^0 formulae [10]). For these reasons, the Ajtai-Wigderson framework has been increasingly popular in recent years, and its applications went far beyond AC^0 [10, 27, 11, 19, 7, 22, 6].

It would be instructive to compare our new approach with their framework. In the following, we briefly review their framework first. Let $t = \Theta(\log n/p)$ be a parameter. Let $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t)}$ be t independent copies of $\mathbf{X}$ (the pseudorandom distribution for $\mathcal{C}_{simple}$). We sample a list of random sets $\mathbf{\Lambda}^1, \ldots, \mathbf{\Lambda}^t$ as follows.

- First, sample $\mathbf{\Lambda}^1 \subseteq [n]$ being a k-wise independent p-marginal subset of $[n]$.
- Having observed $\mathbf{\Lambda}^1$, we sample $\mathbf{\Lambda}^2 \subseteq [n] \setminus \mathbf{\Lambda}^1$ in a k-wise independent and p-marginal way.

[2] Specifically, note that there might be a correlation between $\mathbf{H}$ and $\mathbf{Z}_i$, because $\mathbf{H}$ determines which part of each $\mathbf{X}^{(j)}$ gets added to $\mathbf{Z}_i$. We show how to handle this issue in Section 2.2. See also Section 4 for the formal proof of the construction.

- For every $i \geq 3$. We first observe $\mathbf{\Lambda}^1 \cup \cdots \cup \mathbf{\Lambda}^{i-1}$ and then sample $\mathbf{\Lambda}^i \subseteq [n] \setminus (\bigcup_{j=1}^{i-1} \mathbf{\Lambda}^j)$, also in a k-wise independent and p-marginal way.

In the real implementation, we can first sample $\mathbf{\Lambda}^t \subseteq [n]$ and then subtract $\bigcup_{j=1}^{i-1} \mathbf{\Lambda}^j$ from it. Given these primitives, the Ajtai-Wigderson PRG outputs

$$\mathbf{D} = (\mathbf{X}^{(1)} \wedge \mathbf{\Lambda}^1) \oplus (\mathbf{X}^{(2)} \wedge \mathbf{\Lambda}^2) \oplus \cdots \oplus (\mathbf{X}^{(t)} \wedge \mathbf{\Lambda}^t).$$

We observe that with high probability, $\mathbf{\Lambda}^1 \sqcup \cdots \sqcup \mathbf{\Lambda}^t$ forms a partition of $[n]$. The proof of correctness is also by a hybrid argument. We observe two major differences between the Ajtai-Wigderson framework and ours.

1. As an advantage, the Ajtai-Wigderson framework does not need to sample $\mathbf{Y}$, and a partial derandomization of the "simplify-under-restriction" lemma suffices for applying Ajtai-Wigderson. That is, it only requires that the circuit $C|_{\mathbf{\Lambda}[\mathbf{U},\star]}$ simplifies with high probability over a partially-pseudorandom restriction $(\mathbf{\Lambda}, \mathbf{U})$, where the restriction set $\mathbf{\Lambda}$ is pseudorandom and the string $\mathbf{U}$ is truly random.
   To see why this is true, we look into the analysis of the hybrid argument. For example, consider comparing the hybrid distribution

   $$\mathbf{D}^{(0)} = (\mathbf{U}^{(1)} \wedge \mathbf{\Lambda}^1) \oplus (\mathbf{U}^{(2)} \wedge \mathbf{\Lambda}^2) \oplus \cdots \oplus (\mathbf{U}^{(t)} \wedge \mathbf{\Lambda}^t),$$

   with

   $$\mathbf{D}^{(1)} = (\mathbf{X}^{(1)} \wedge \mathbf{\Lambda}^1) \oplus (\mathbf{U}^{(2)} \wedge \mathbf{\Lambda}^2) \oplus \cdots \oplus (\mathbf{U}^{(t)} \wedge \mathbf{\Lambda}^t).$$

   For simplicity, let us assume that $\mathbf{\Lambda}^1 \sqcup \cdots \sqcup \mathbf{\Lambda}^t$ always covers $[n]$. Then we have $\mathbf{D}^{(0)} = \mathbf{\Lambda}^1[\mathbf{U}, \mathbf{U}^{(1)}]$ and $\mathbf{D}^{(1)} = \mathbf{\Lambda}^1[\mathbf{U}, \mathbf{X}^{(1)}]$, which enable us to apply the partially-derandomized restriction lemma.
2. However, when there is a fully-derandomized restriction lemma, using our approach results in a PRG of shorter seed length. Note that the Ajtai-Wigderson framework partitions the $[n]$ coordinates into $t = \Theta(\log n/p)$ blocks and fills in each block with independent pseudorandom strings. Since the set $\mathbf{\Lambda}^i$ covers (roughly) p-fraction of *currently uncovered* coordinates at each time $i \in [t]$, it is crucial to set $t = \Omega(\log n/p)$ so that $\bigcup_{j=1}^{t} \mathbf{\Lambda}^j$ covers $[n]$ with high probability. Our construction, on the other hand, samples a k-wise independent hash function $\mathbf{H}\colon [n] \to [w]$, which naturally induces a partition $\mathbf{H}^{-1}(1) \sqcup \cdots \sqcup \mathbf{H}^{-1}(w)$. Then we only need $w = O(1/p)$ independent samples of $\mathbf{X}$ to complete the construction. In other words, we save the $\log(n)$ overhead by exploiting the symmetry between blocks in our design.

Concluding remarks

The partitioning-based approach is quite general: for every scenario that Ajtai-Wigderson applies, if we can prove a fully-derandomized "simplify-under-restriction" lemma, then we may hope to use the new framework to shave the $\log(n)$ overhead in seed length. Two more concrete examples are sparse $\mathbb{F}_2$-polynomials [25] and small-size De-Morgan formulae [17, 15]. However, the $\log(n)$ overhead from the Ajtai-Wigderson framework is minor in those applications. For example, the PRG for S-sparse $\mathbb{F}_2$-polynomials has seed length $2^{O(\sqrt{S})}$ [25]. Improving a $\log(n)$ factor here is not as significant as for AC^0 circuits.

In the case of (arbitrary-order) read-once branching programs, Forbes and Kelley [7] have shown PRGs with seed length $O(\log^3 n)$ and $\widetilde{O}(\log^2 n)$ for general and constant-width ROBPs, respectively. Their PRGs are based on the Ajtai-Wigderson framework, and both

of them have an extra $\log(n)$ overhead in seed length due to this reason. It would be exciting to see if one can use the partitioning-based approach to give improved PRG for these models. If this turns out to be true, then we may hope for a simpler construction of nearly-logarithmic seed PRG for ROBPs of all constant width, which would be a major advance in the derandomization of small-space computation. Currently, we only know a nearly-logarithmic seed PRG for width-3 ROBP [22], whose proof seems hard to generalize to larger widths.

Partitioning (or called "bucketing" in some literature) is not a completely new strategy in the pseudorandomness literature. There are works [21, 9] using partitioning to design approximate counting algorithms for DNF. We also note that Meka and Zuckerman [23] have used a similar strategy to construct PRGs for low-degree polynomial threshold functions (PTFs). Their PRG also partitions $[n]$ coordinates into small blocks by a bounded-independent hash function and fills in each block with independent but pseudorandom bits. However, their analysis was completely different from ours, nor did they need to add a noise string "$\mathbf{Y}$" to fool any restriction lemma. As far as we are aware, our work is novel in using partitioning to construct PRGs based on restriction lemmas.

2.2 Derandomized Multi-Switching Lemma

The second technical ingredient behind our result is a fully-derandomized multi-switching lemma for small-width DNFs.

Håstad's switching lemma

Switching lemmas are perhaps the most powerful and versatile tools in analyzing low-depth Boolean circuits, with applications ranging from proving lower bounds [14, 13, 29], constructing pseudorandom generators [1, 27, 25, 18], learning of AC^0 functions [20], designing circuit-analysis algorithms for AC^0 [2, 16], to proving Fourier-analytic properties of AC^0 [20, 26], to name a few.

The standard switching lemma, originally proved by Håstad, says that for a width-w DNF[3] F, if we apply a $\frac{1}{20w}$-random restriction $(\mathbf{\Lambda}, \mathbf{x})$, then with probability $1-\varepsilon$, $F|_{\mathbf{\Lambda}[\mathbf{x},\star]}$ collapse to a decision tree of depth $O(\log(1/\varepsilon))$. We can also prove a switching lemma for small-size DNFs: suppose F is a size-m unbounded-width DNF. Then, applying a $\frac{1}{\log(m/\varepsilon)}$-random restriction collapses F to a depth-$O(\log(m/\varepsilon))$ decision tree with probability $1-\varepsilon$.

In Section 5, we show a derandomization for the standard switching lemma. That is, we prove

▶ **Lemma 2** (Derandomized Switching Lemma, slightly-simplified). *Let $k, m \geq 1$ be integers, and $\varepsilon, p > 0$ be reals. Let $F = \bigvee_{i=1}^{m} C_i$ be a size-m width-k DNF over inputs $\{0,1\}^n$. For all $t \geq 1$, let $(\mathbf{\Lambda}, \mathbf{x})$ be any joint random variable such that:*

- *$\mathbf{\Lambda}$ is a $(t+k)$-wise p-marginal subset of $[n]$.*
- *Conditioning on $\mathbf{\Lambda}$, $\mathbf{x}$ is a random string that ε-fools CNF of size at most m.*

Consider the random restriction $F|_{\mathbf{\Lambda}[\mathbf{x},\star]}$, we have:

$$\Pr_{\mathbf{\Lambda},\mathbf{x}}[\mathsf{DT}(F|_{\mathbf{\Lambda}[\mathbf{x},\star]}) > t] \leq (10kp)^t + (4m)^{t+k}\varepsilon.$$

See Section 5 for the stronger statement and the formal proof. Also note that we allow correlation between $\mathbf{\Lambda}$ and $\mathbf{x}$, which is crucial to apply the lemma in our PRG framework.

[3] The width of a DNF F is defined as the maximum number of variables in any term of F.

One can already construct a $\frac{1}{m}$-error PRG for $\mathsf{AC}^0_d[m]$ with seed length $O(\log^d(m)\log\log(m))$ based on Lemma 2. Let $C \in \mathsf{AC}^0_d[m]$ be a size-m depth-d circuit. We assume that each bottom-layer gate of C has fan-in bounded by $O(\log(m))$ for simplicity[4]. We apply Lemma 2 with $p = \frac{1}{c\log m}, t = c\log(m)$, and $\varepsilon = 2^{-c\log^2(m)}$ for some large constant $c > 1$. We also take $\mathbf{x}$ as a pseudorandom string that ε-fools CNF. Then with probability at least $1-\frac{1}{m^2}$ over the pseudorandom restriction $(\Lambda, \mathbf{x})$, every depth-2 sub-circuit of C simplifies to a depth-t decision tree, which means that we can express $C|_{\mathbf{\Lambda}[\mathbf{x},\star]}$ as a depth-$(d-1)$ circuit of size $\mathrm{poly}(m)$. Hence, assuming we can fool depth-$(d-1)$ circuit with seed length $O(\log^{d-1}(m)\log\log(m))$, then we can fool depth-d circuit with seed length $O(\log^d(m)\log\log(m))$ by applying the partitioning-based PRG. Here we have omitted the seed length to sample $\mathbf{Y}$ and $\mathbf{H}$. It turns out they will not be the bottleneck: see Section 4 for the details.

Multi-switching lemma

For the case that $\varepsilon < m^{-\omega(1)}$, using Lemma 2 may result in a longer seed length. In fact, to simplify the circuit with probability at least $1-\varepsilon$, one must take the "t" parameter in Lemma 2 as $\Theta(\log(m/\varepsilon))$. Then, applying Lemma 2 once simplifies C to a depth-$(d-1)$ circuit with bottom fan-in bounded by $t = \Theta(\log(m/\varepsilon))$. To further apply the lemma, one has to set p as $\frac{1}{\Omega(\log(m/\varepsilon))}$ to make the probability bound in Lemma 2 non-trivial. Therefore, the depth-d PRG would have seed length longer than the depth-$(d-1)$ PRG by $\frac{1}{p} = \Omega(\log(m/\varepsilon))$, bringing the total seed length to $\Omega(\log^d(m/\varepsilon)\log\log(m))$.

If we insist on using a $\frac{1}{\log(m)}$-random restriction and want to have the same $(1-\varepsilon)$ probability guarantee, we can use the multi-switching lemma. We give its statement first.

▶ **Lemma 3.** *Let $t, w, k, n, m \geq 1$ be integers. Let $p, \delta > 0$ be reals. Let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a list of size-m width-k DNFs on inputs $\{0,1\}^n$. Let $(\mathbf{\Lambda}, \mathbf{x})$ be a joint random variable satisfying the following:*

- *$\mathbf{\Lambda}$ is a $(t+k)$-wise p-marginal subset of $[n]$,*
- *Conditioning on $\mathbf{\Lambda}$, $\mathbf{x}$ is an n-bit random string that δ-fools size-(m^2) CNF.*

Then with probability at least $1 - \left(4m^{t/w}(24pk)^t + (24m)^{t+k}\cdot\delta\right)$ over $(\mathbf{\Lambda}, \mathbf{x})$, there exists a common w-partial depth-t decision tree[5] for $\mathcal{F}|_{\mathbf{\Lambda}[\mathbf{x},\star]}$. That is, we can construct a list of decision trees $T_1, \ldots, T_m$ computing $F_1|_{\mathbf{\Lambda}[\mathbf{x},\star]}, \ldots, F_m|_{\mathbf{\Lambda}[\mathbf{x},\star]}$. Each T_i is of depth at most $(t+w)$, and all of T_i's share the same query strategy in the first t queries.

Let $C \in \mathsf{AC}^0_d[m]$ be a circuit with bottom fan-in bounded by $k = \log(m)$. Let $\mathcal{F}$ denote the family of depth-2 sub-circuits of C. Apply Lemma 3 on $\mathcal{F}$ with $\frac{1}{p} = O(\log(m))$, $t = O(\log(m/\varepsilon)), w = O(\log(m))$ and $\delta = \frac{\varepsilon}{m^{O(t)}}$. We know that $\mathcal{F}|_{\mathbf{\Lambda}[\mathbf{x},\star]}$ fails to simplify with probability at most

$$\left(4m^{t/w}(24pk)^t + (24m)^{t+k}\cdot\delta\right) \leq \varepsilon.$$

When $\mathcal{F}$ does simplify, we can compute $C|_{\mathbf{\Lambda}[\mathbf{x},\star]}$ by a hybrid model: a depth-t decision tree with $\mathsf{AC}^0_{d-1}[m\cdot 2^w]$-circuits on leaves. The decision tree part performs t adaptive queries according to the common partial decision tree of $\mathcal{F}$. After that, functions in $\mathcal{F}$ can be expressed as depth-w decision trees, which means that C can be computed by an $\mathsf{AC}^0_{d-1}[m\cdot 2^w]$ circuit.

[4] This assumption can be met by first applying a $\frac{1}{2}$-(pseudo)random restriction, because with high probability every bottom-layer gate with large fan-in is killed under such a restriction.

[5] See Section 3.2 for the formal definition of "partial decision tree".

If we can fool every depth-$(d-1)$ circuit on the leaves with error $\frac{\varepsilon}{2^t}$, then we can fool this hybrid model with error ε. Assuming a $O(\log^{d-2}(m)\log(m/\varepsilon)\log\log(m))$ PRG for depth-$(d-1)$ circuits, fooling this hybrid model requires seed length

$$O\left(\log^{d-2}(m\cdot 2^w)\log(m2^t/\varepsilon)\log\log(m)\right) = O(\log^{d-2}(m)\log(m/\varepsilon)\log\log(m)).$$

This allows us to construct a PRG for depth-d circuits with seed length $O(\log^{d-1}(m)\cdot\log(m/\varepsilon)\cdot\log\log(m))$, as claimed.

Proof intuition

When the restriction string is truly random, Kelley's technique [18] shows a clear picture about what makes a random restriction $\mathbf{\Lambda}[\mathbf{U},\star]$ bad. Fix an $\mathsf{AC}^0_d[m]$ circuit C. Given a restriction $\mathbf{\Lambda}[\mathbf{U},\star]$, we want to know whether C simplifies under $\mathbf{\Lambda}[\mathbf{U},\star]$. Roughly speaking, Kelley shows that one can first observe the restriction string $\mathbf{U}$ and come up with a list of $t=\log(m)^{\log(m)}$ sets $S_1,\ldots,S_t$, each of size $c\log(m)$ for a large constant $c\geq 1$. Then, we look at the set $\mathbf{\Lambda}$: C fails to simplify under $\mathbf{\Lambda}[\mathbf{U},\star]$, only when $\mathbf{\Lambda}$ contains at least one set S_i from the list. Since $\mathbf{\Lambda}$ is $O(\log(m))$-wise $\frac{1}{c\log(m)}$-marginal, this happens with probability at most $2^{-c\log(m)}$ by a simple union bound.

We apply Kelley's technique to derandomize the multi-switching lemma [13]. The proof of the multi-switching lemma involves many tricks and technicalities. Here we try to give some (over-simplified) intuition. At a very high level, the multi-switching lemma is proved by combining the standard switching lemma with a union bound. If $\mathcal{F}|_\rho$ fails a have w-partial depth-t decision tree, then there is a subset of *at most* $\frac{t}{w}$ formulae in $\mathcal{F}_\rho$, such that the summation of their decision tree complexities exceeds t. This is because every formula with decision tree complexity no larger than w can be handled "for free". Therefore, each bad formula contributes at least w to the summation. There are at most $m^{t/w}=2^{O(t)}$ such subsets. For each of them, we bound the probability that the summation of their DT complexities exceeds t by $O(kp)^t$. This step is rather similar (in spirit) to the case of standard switching lemma, and Kelley's technique applies.

The final piece in our analysis is the full derandomization. By Kelley's technique, we know that the partially-pseudorandom restriction $\mathbf{\Lambda}[\mathbf{U},\star]$ is as good as a truly random one. We further derandomize the random string by using the techniques by Trevisan, Xue, and by Servedio, Tan [27, 25]. Specifically, they constructed bounded-depth circuits (called "testers") that take a restriction as input and decide whether the restriction is good or not (for simplifying the target circuit). Now, consider sampling a string $\mathbf{x}$ from a distribution that fools bounded-depth circuits, Given the tester, we can show that $\mathbf{\Lambda}[\mathbf{x},\star]$ is as good as $\mathbf{\Lambda}[\mathbf{U},\star]$. Since fooling higher-depth circuits requires longer random bits, to control the final seed length of our PRG, we need the tester to be implementable in AC^0_2. For the standard switching lemma, the Trevisan-Xue tester [27] does have depth 2. For the multi-switching lemma, things become a bit trickier: the Servedio-Tan tester [25] was designed as a depth-3 circuit and did the test faithfully. We (implicitly) implemented a "upper-side approximator" of their tester. Our one-sided tester can be expressed as a CNF, and is equally useful when upper-bounding the probability of picking bad restrictions.

See our derandomized switching lemmas for the standard and multi-switching versions in Section 5 and Section 6, respectively. Instead of playing with decision trees and tracing down query paths, we strive to present the proof based on the "canonical query algorithm". Our proof is more operational and, in our opinion, easier to follow.

Comparison with previous works

For the task of designing PRGs for AC^0 circuits, before our result, there were two incomparable results on the frontier, one by Servedio and Tan [25] and the other by Kelley [18]. Both of them were built on derandomization results for switching lemmas.

The Servedio-Tan PRG is based on a derandomization of Håstad's multi-switching lemma [13], and has seed length $\log^{d+O(1)}(m)\log(1/\varepsilon)$ when ε-fooling $\mathsf{AC}^0_d[m(n)]$. Due to the usage of multi-switching lemma, their PRG has optimal dependence on the error parameter ε. However, their derandomization of the switching lemma is weaker in seed length. The two factors together determined the final seed length of their PRG.

Kelley's PRG, on the other hand, is based on a stronger derandomization of the standard switching lemma [14]. It has seed length $\widetilde{O}(\log^d(m/\varepsilon)\log(n))$. The exponent on $\log(m)$ matches the lower bound barrier, credit to the fact that their stronger derandomization allows one to sample a restriction using a much shorter seed. However, Kelley only showed a partial derandomization, which is not applicable in our framework. Also, the dependence on $\frac{1}{\varepsilon}$ is inferior due to the somewhat coarse analysis in the standard switching lemma. It was left as an open question in [18] whether one can get the same high-equality derandomization of the multi-switching lemma and optimize the dependency on $\frac{1}{\varepsilon}$.

We answer this question in the affirmative by showing a fully-derandomized multi-switching lemma that improves both works. Combined with the partitioning-based PRG framework, our lemma gives a PRG for AC^0 with an almost tight seed length. We hope our derandomization of the switching lemmas could find applications in other contexts.

Finally, we remark that the "decision-tree-followed-by-circuit" type hybrid model also appears in many previous works. The applications include proving correlation bounds and Fourier spectrum bounds [13, 26], constructing PRGs [25, 15], designing circuit-analysis algorithms [4], etc.

3 Preliminaries

In this section, we set up necessary pieces of notation, and review some well-known and useful facts from the literature of pseudorandmoness and complexity theory.

3.1 Restrictions, Partial-Assignments and Strings

We use the term "restriction" and "partial assignment" interchangeably. Both of them refer to a string of the form $\rho \in \{0,1,\star\}$. Here, if $\rho_i = 0/1$, it means the i-th bit of ρ is fixed to that value. Otherwise, the i-th bit of ρ is unfixed.

For two partial assignments $\rho, \sigma \in \{0,1,\star\}$, define their composition $\rho \circ \sigma$ as:

$$(\rho \circ \sigma)_i = \begin{cases} \rho_i & \rho_i \neq \star \\ \sigma_i & o.w. \end{cases}.$$

Note that the left partial assignment always has a higher priority than the right one.

Let $\Lambda \subseteq [n]$ be a set. For two partial assignments $\rho, \sigma \in \{0,1,\star\}^n$, let $\Lambda[\rho,\sigma]$ be the assignment defined as

$$\Lambda[\rho,\sigma]_i = \begin{cases} \rho_i, & i \notin \Lambda \\ \sigma_i, & i \in \Lambda \end{cases}.$$

Let $f : \{0,1\}^n \to \{0,1\}$ be a function and $\rho \in \{0,1,\star\}^n$ be a partial assignment. We use $f|_\rho \colon \{0,1\}^n \to \{0,1\}$ to denote the restriction of f on ρ. That is, $f|_\rho(y) := f(\rho \circ y)$.

3.2 Computational Models

$\mathsf{AC}^0_d[s(n)]$ denotes the family of Boolean circuits of size at most $s(n)$ and depth at most d. Such circuits can have $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ gates. Here, NOT gates do not count in the depth, and $\mathsf{AND}, \mathsf{OR}$ gates can have unbounded fan-in. We measure the size of a circuit by the total number of wires (including input wires) in it. For the case of $d = 2$, we also use the terms DNF and CNF to refer to $\mathsf{OR} \circ \mathsf{AND}$ and $\mathsf{AND} \circ \mathsf{OR}$ circuits respectively. The width of a DNF or CNF is defined as the maximum of its bottom fan-in. We also use k-DNF (resp. k-CNF) to denote DNF (resp. CNF) of width at most k.

A decision tree T is a binary tree. Each inner node of T is labelled with an index $i \in [n]$, and has exactly two children, which are labelled with 0 and 1. Each leaf of T is labelled with a Boolean value $b \in \{0,1\}$. A decision tree T computes a function in the following manner: on an input $x \in \{0,1\}^n$, we start from the root of T. In each turn we observe the index i of current node, query x_i and move to the left/right child depending on the bit x_i we received. Once we reach a leaf with label b, we output $T(x) = b$. The depth of a decision tree is the length of the longest path from root to any leaf.

We also consider a special decision tree model for *a list of functions*. See the definition below.

▶ **Definition 4.** *Let $t, w, n, m \geq 1$ be integers. Let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a list of functions mapping from $\{0,1\}^n$ to $\{0,1\}$. A w-partial depth-t decision tree for $\mathcal{F}$ is a depth-t decision tree T satisfying the following. For every $F_i \in \mathcal{F}$ and every leaf ℓ of T, let $\alpha \in \{0,1,\star\}^n$ be the partial assignment that corresponds to ℓ (That is, α_i equals $\star$ if T does not query x_i before reaching ℓ, otherwise α_i equals to the value that leads T to move towards ℓ). Then it holds that $\mathsf{DT}(F_i|_\alpha) \leq w$.*

3.3 Pseudorandomness

Recall the definition of bounded independence.

▶ **Definition 5.** *Let n, m be two integers. Let $\mathcal{H}$ be a distribution over hash functions mapping $\{0,1\}^n$ into $\{0,1\}^m$. We say that $\mathcal{H}$ is k-wise independent, if for any k input-output pairs $(x_1, y_1), \ldots, (x_k, y_k) \in \{0,1\}^n \times \{0,1\}^m$ where $x_1, \ldots, x_t$ are distinct, it holds that*

$$\Pr_{h \sim \mathcal{H}}[\forall i \in [k], h(x_i) = y_i] = 2^{-km}.$$

We have the following standard construction of bounded independence hash functions (check e.g., [28, Chapter 3.5.5]).

▶ **Lemma 6.** *For every $n, m, k \geq 1$, there is an explicit k-wise independent hash functions $\mathcal{H}$ that maps $\{0,1\}^n$ into $\{0,1\}^m$. One can sample a function in $\mathcal{H}$ using $O(k(n+m))$ random bits.*

We also consider a weaker notion of pseudorandomness called "k-wise p-boundedness", first defined and studied by [18].

▶ **Definition 7.** *Suppose $\mathbf{\Lambda}$ is a random subset of $[n]$. We say that Λ is k-wise p-bounded if for any set $B \subseteq [n]$ of size at most k, it holds that $\Pr_{\mathbf{\Lambda}}[B \subseteq \mathbf{\Lambda}] \leq p^{|B|}$.*

For intuition, if we sample $\mathbf{\Lambda}$ by independently including each i in $\mathbf{\Lambda}$ with probability at most p, then $\mathbf{\Lambda}$ is n-wise p-bounded.

4 Improved PRG for Constant-Depth Circuits

In this section, we aim to prove the main theorem, re-stated below.

▶ **Reminder of Theorem 1.** *For every $d \geq 2$ the following is true. For every $m, n \in \mathbb{N}$ such that $m \geq n$ and every $\varepsilon > 0$, there is an ε-PRG for $\mathsf{AC}^0_d[m]$ circuits with seed length $O(\log^{d-1}(m) \cdot \log(m/\varepsilon) \cdot \log\log(m))$.*

We start with the following fact, which is crucial in our construction.

▶ **Theorem 8** ([5, 26])**.** *For every $m, n \in \mathbb{N}$ such that $m \geq n$ and every $\varepsilon > 0$, there is an ε-PRG for $\mathsf{AC}^0_2[m]$ circuits (namely, CNF/DNF formulae) with seed length $O(\log(m) \cdot \log(m/\varepsilon) \cdot \log\log(m))$.*

We give the formal statement of the derandomized multi-switching lemma below. This is the full version of Lemma 3 in the introduction.

▶ **Lemma 9.** *Let $t, w, k, n, m \geq 1$ be integers. Let $p, \varepsilon > 0$ be reals. Let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a list of size-m k-DNFs on inputs $\{0,1\}^n$. Let $(\mathbf{\Lambda}, \mathbf{x})$ be a joint random variable satisfying the following:*

- *$\mathbf{\Lambda}$ is a $(t+k)$-wise p-bounded subset of $[n]$,*
- *Conditioning on $\mathbf{\Lambda}$, $\mathbf{x}$ is an n-bit random string that ε-fools size-(m^2) CNF.*

Then we have

$$\Pr_{T \sim \mathbf{\Lambda}, x \sim \mathbf{x}}[\mathcal{F}|_{\Lambda[x,\star]} \text{ does not have } w\text{-partial depth-}t \text{ } DT] \leq 4m^{t/w}(24pk)^t + (24m)^{t+k} \cdot \varepsilon.$$

We defer the proof of Lemma 9 to Section 6. Assuming Lemma 9, we prove Theorem 1. We prove a slightly stronger form of Theorem 1, stated below.

▶ **Theorem 10.** *For every $d \geq 2$ the following is true. For every $m, n \in \mathbb{N}$ such that $m \geq n$, $k \in \mathbb{N}$ and every $\varepsilon > 0$, there is an ε-error, $O((\log^2(m) + k \cdot \log^{d-2}(m)) \cdot \log(m/\varepsilon) \cdot \log\log(m))$-seed PRG for $\mathsf{AC}^0_d[m]$ circuits with bottom fan-in bounded by k.*

Theorem 10 shows that, when the bottom fan-in of the AC^0_d circuits is smaller than $o(\log(m))$, we can hope for shorter seed length through our construction. Note that we can always interpret a depth-d AC^0 circuit with unbounded bottom fan-in as a depth-$(d+1)$ AC^0 circuit with bottom fan-in being 1. Then, Theorem 1 follows from Theorem 10 easily.

Proof. We use induction on the depth d. The case for $d = 2$ follows from Theorem 8. Assuming this is true for depth $d - 1 \geq 2$, we prove it for the case of d. Let $w = 40k$ and $t = 80\log(m/\varepsilon)$. We prepare the following pseudorandom primitives.

- First, let $\mathbf{H} : [n] \to [w]$ be a $2t$-wise independent hash function, samplable using $O(\log(n)\log(m/\varepsilon))$ bits. In the following, we will use $\mathbf{H}_i$ to denote $\mathbf{H}^{-1}(i)$. We remark that $\mathbf{H}_i$ can be equivalently expressed as an n-bit string. Namely, $(\mathbf{H}_i)_j = 1$ if and only if $j \in \mathbf{H}_i$.
- Second, let $\varepsilon' = \varepsilon/(w \cdot 2^{t+1})$. Sample $\mathbf{X}_1, \ldots, \mathbf{X}_w \in \{0,1\}^n$, each being an independent string that ε'-fools AC^0_{d-1}-circuits of size $2m^2$ and bottom-width $\log m$. When $d-1 \geq 3$, $\mathbf{X}_i$ is samplable using $O(\log^{d-2}(m) \cdot \log(m/\varepsilon) \cdot \log\log(m))$ bits by the induction hypothesis. For the case $d - 1 = 2$, $\mathbf{X}_i$ is samplable using $O(\log(m)\log(m/\varepsilon)\log\log(m))$ bits by Theorem 8.
- Lastly, let $\mathbf{Y} \in \{0,1\}^n$ be a random string that $\varepsilon/(24m)^{2t}$-fools size-m CNF, samplable using $O(\log^2 m \cdot \log(m/\varepsilon) \cdot \log\log m)$ bits (by Theorem 8).

The seed for our generator is the concatenation of the seeds used to sample all the primitives above. We compute the output of our generator as

$$\mathbf{Y} \oplus (\mathbf{X}_1 \wedge \mathbf{H}_1) \oplus (\mathbf{X}_2 \wedge \mathbf{H}_2) \oplus \cdots \oplus (\mathbf{X}_w \wedge \mathbf{H}_w). \tag{1}$$

Here, $\wedge$ and $\oplus$ denote bit-wise AND and XOR respectively. The seed length is bounded by

$$\begin{aligned} & O\left(\log(n)\log(m/\varepsilon) + w \cdot \log^{d-2}(m)\log(m \cdot 2^t/\varepsilon)\log\log(m) + \log^2 m \log(m/\varepsilon)\log\log m\right) \\ \leq\ & O((\log^2(m) + k \cdot \log^{d-2}(m)) \cdot \log(m/\varepsilon) \cdot \log\log(m)). \end{aligned}$$

We argue the correctness by a hybrid argument. Fix C to be an AC^0_d-circuit that we wish to fool. Let $\mathbf{U}_1, \ldots, \mathbf{U}_w$ denote w independent uniformly random strings from $\{0,1\}^n$. For every $i \in \{0, 1, \ldots, w\}$, we define the i-th hybrid distribution as

$$\mathcal{D}_i := \mathbf{Y} \oplus (\mathbf{U}_1 \wedge \mathbf{H}_1) \oplus \cdots \oplus (\mathbf{U}_i \wedge \mathbf{H}_i) \oplus (\mathbf{X}_{i+1} \wedge \mathbf{H}_{i+1}) \oplus \cdots \oplus (\mathbf{X}_w \wedge \mathbf{H}_w). \tag{2}$$

We observe that $\mathcal{D}_0$ is the output distribution of our PRG, while $\mathcal{D}_w$ is a uniformly random string from $\{0,1\}^n$. Hence, it suffices to show that

$$|\mathbb{E}_{x\sim\mathcal{D}_0}[C(x)] - \mathbb{E}_{x\sim\mathcal{D}_w}[C(x)]| \leq \varepsilon. \tag{3}$$

To show (3), it suffices to show for every $i \in \{1, \ldots, w\}$ that

$$|\mathbb{E}_{x\sim\mathcal{D}_{i-1}}[C(x)] - \mathbb{E}_{x\sim\mathcal{D}_i}[C(x)]| \leq \varepsilon/w. \tag{4}$$

In the following, we prove (4). We observe that $\mathbf{H}_i$ is $2t$-wise $\frac{1}{w}$-bounded. Conditioning on an instantiation of $\mathbf{H}$, we have that $\mathbf{Z}_i := \mathbf{Y} \oplus \sum_{j<i}(\mathbf{U}_j \wedge \mathbf{H}_j) \oplus \sum_{j>i}(\mathbf{X}_j \wedge \mathbf{H}_j)$ is an $\varepsilon/(24m)^{2t}$-pseudorandom string for size-(m^2) CNF, because $\mathbf{Y}$ is. Let $\mathcal{F}$ be the family of all next-to-bottom layer sub-circuits of C. Denote by $\mathcal{E}$ the event

"$\mathcal{F}|_{\mathbf{H}_i[\mathbf{Z}_i,\star]}$ does not have $\log(m)$-partial depth-t DT."

Then it follows from Lemma 9 that

$$\Pr_{\mathbf{H},\mathbf{Y},\mathbf{U}_1,\ldots,\mathbf{U}_{i-1},\mathbf{X}_{i+1},\ldots,\mathbf{X}_w}[\mathcal{E}] \leq 4m^{t/\log(m)}\left(24\frac{k}{w}\right)^t + \frac{\varepsilon \cdot (24m)^{t+\log(m)}}{(24m)^{2t}} \leq \frac{\varepsilon}{2w}.$$

Conditioning on $\neg\mathcal{E}, \mathbf{H}, \mathbf{Y}, \mathbf{U}_1, \ldots, \mathbf{U}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_w$ and calculating $\mathbf{Z}_i$ as defined above, one can then write $C|_{\mathbf{H}_i[\mathbf{Z}_i,\star]}$ as a depth-t decision tree T where each leaf of T is labelled by an AC^0_{d-1}-circuit of size m^2 and bottom fan-in $\log(m)$. Let $\{\ell_1, \ldots, \ell_{2^t}\}$ enumerate the leaves of the decision tree. Each ℓ_j is associated with a size-m^2 depth-$(d-1)$ circuit, which is also denoted by $\ell_j : \{0,1\}^n \to \{0,1\}$ for brevity. Then one can write $C|_{(H_i)[Z_i,\star]}$ as

$$C|_{\mathbf{H}_i[Z_i,\star]}(y) = \sum_{j=1}^{2^t} \ell_j(y) \cdot \mathbb{1}\{T(y) \text{ reaches leaf } \ell_j\}.$$

Let's fix an index $j \in [2^t]$ for now. Note that $\ell_j(y) \cdot \mathbb{1}\{\text{reach } \ell_j \text{ on } y\}$ is itself a depth-$(d-1)$ circuit of size at most $2m^2$. By the construction of X_i we know that

$$\begin{aligned} \Big|&\mathbb{E}_{\mathbf{X}_i}[\ell_j(\mathbf{X}_i + Z_i) \cdot \mathbb{1}\{T \text{ reaches } \ell_j \text{ on } \mathbf{X}_i + Z_i\}] - \\ &\mathbb{E}_{\mathbf{U}_i}[\ell_j(\mathbf{U}_i + Z_i) \cdot \mathbb{1}\{T \text{ reaches } \ell_j \text{ on } \mathbf{U}_i + Z_i\}]\Big| \leq \frac{\varepsilon}{2^{t+1}w}. \end{aligned}$$

Taking a summation over all leaves j, one gets

$$\left|\mathbb{E}_{\mathbf{X}_i}[C|_{\mathbf{H}_i[Z_i,\star]}(\mathbf{X}_i+Z_i)]-\mathbb{E}_{\mathbf{U}_i}[C|_{\mathbf{H}_i[Z_i,\star]}(\mathbf{U}_i+Z_i)]]\right|\leq\frac{\varepsilon}{2w}.$$

Finally, one has

$$|\mathbb{E}_{x\sim\mathcal{D}_{i-1}}[C(x)]-\mathbb{E}_{x\sim\mathcal{D}_i}[C(x)]|\leq\Pr[\neg\mathcal{E}]\cdot\frac{\varepsilon}{2w}+\Pr[\mathcal{E}]\leq\frac{\varepsilon}{w},$$

proving (4). ◀

Given Theorem 10, we prove Theorem 1 by tuning parameters.

Proof of Theorem 1. For every $d\geq 3$ and every $\mathsf{AC}^0_d[m]$ circuit with unbounded bottom fan-in, we can interpret it as a depth-$(d+1)$ circuit with bottom fan-in being 1. Applying Theorem 10 in this case gives a PRG with seed length $O(\log^{d-1}(m)\log(m/\varepsilon)\log\log(m))$, as desired. For the case of $d=2$, we use Theorem 8 directly. This completes the proof. ◀

As a final remark, suppose we could have $O(\log(m)\log(m/\varepsilon))$-seed PRG for CNFs (namely, if we can shave the $\log\log m$ factor in Theorem 8). Then our construction implies PRG for AC^0_3 with seed length $O(\log^2(m)\log(m/\varepsilon))$, and further implies PRG for AC^0_d with seed length $O(\log^{d-1}(m)\log(m/\varepsilon))$, matching the lower bound barrier [14, 13].

5 Fully-Derandomized Switching Lemma

Before we show the proof of Lemma 9, we state and prove the simpler version of the classical switching lemma in this section. The following statement is the full version of Lemma 2 in Introduction. Lemma 2 follows from Lemma 11 trivially.

▶ **Lemma 11.** *Let $k,m\geq 1$ be integers, and $\varepsilon,p>0$ be reals. Let $F=\bigvee_{i=1}^m C_i$ be a size-m k-DNF over inputs $\{0,1\}^n$. For all $t\geq 1$, let $(\mathbf{\Lambda},\mathbf{x})$ be a joint random variable such that:*

- *$\mathbf{\Lambda}$ is a $(t+k)$-wise p-bounded subset of $[n]$.*
- *Conditioning on $\mathbf{\Lambda}$, $\mathbf{x}$ is a random string that ε-fools CNF of size at most m.*

Consider the random restriction $F|_{\mathbf{\Lambda}[\mathbf{x},\star]}$, we have:

$$\Pr_{\mathbf{\Lambda},\mathbf{x}}[\mathsf{DT}(F|_{\mathbf{\Lambda}[\mathbf{x},\star]})>t]\leq(10kp)^t+(4m)^{t+k}\cdot\varepsilon.$$

Understanding the proof of Lemma 11 is necessary to read the proof of Lemma 9. On the other hand, once Lemma 11 is established, we can prove Lemma 9 using a rather similar strategy. The rest of the section is devoted to the proof of Lemma 11.

We will first introduce the important concept of "canonical decision tree" ([14, 27, 25, 26]) in Section 5.1. We prove for the case that $\mathbf{x}$ is truly random (and only $\mathbf{\Lambda}$ is pseudorandom) in Section 5.2, and then argue how to prove for pseudorandom $\mathbf{x}$ in Section 5.3.

Throughout the whole section, we always use $F=\bigvee_{i=1}^m C_i$ to denote the k-DNF that we are analyzing. For every $i\in[m]$ let $V_i\subseteq[n]$ be the variables involved in the term C_i.

5.1 Canonical decision tree

For a DNF $F=\bigvee_{i=1}^m C_i$, denote by T_F the *canonical decision tree* of F, whose construction is shown in Algorithm 1. We use $\mathsf{CDT}(F)$ to denote the depth of the canonical decision tree for F. Since canonical decision tree is one particular decision tree of F, its depth must be no less than $\mathsf{DT}(F)$. To prove Lemma 11, we will analyze canonical decision trees, and show that with high probability over the random restriction ρ, we have $\mathsf{CDT}(F|_\rho)<t$.

Algorithm 1 Canonical Decision Tree.

Input: A DNF $F = \bigvee_{i=1}^{m} C_i$, black-box access to a string $\alpha \in \{0,1\}^n$.
initialize:
  $j^* \leftarrow 0$.
  $x \leftarrow (\star)^n$.
while $j^* < m$ **do**
  Find the first $j > j^*$ such that $C_j(x) \not\equiv 0$. If no such j exists, exit the loop.
  $B_j \leftarrow$ the set of unknown variables in C_j.
  Query α_{B_j}.
  Set $x_{B_j} \leftarrow \alpha_{B_j}$.
  if $C_j(x) = 1$ **then**
    return 1.
  $j^* \leftarrow j$.
return 0

5.2 Proof when x is truly random

Defining witness

Let $\rho = \Lambda[x, \star]$ being a bad restriction, under which $\mathsf{CDT}(F|_\rho) \geq t$. Consider simulating the canonical decision tree $T_{F|_\rho}$. We know that on some inputs $\alpha \in \{0,1\}^n$, $T_{F|_\rho}$ fails to output the decision after making $(t-1)$ queries. We choose one such α and simulate $T_{F|_\rho}$ until it makes at least t queries. The "running transcript" of $T_{F|_\rho}$ on α is naturally a witness to the fact that $\mathsf{CDT}(F_\rho) \geq t$. We formalize this idea in the following definition.

▶ **Definition 12.** *Let $F = \bigvee_{i=1}^{m} C_i$ be the k-DNF and $\rho \in \{0,1,\star\}^n$ be a restriction. Let $t \geq 1$. Consider a tuple $(r, \ell_i, s_i, B_i, \alpha_i)$, where:*

1. *$r \in [1,t]$ is an integer.*
2. *$(\ell_1, \ldots, \ell_r) \in [m]^r$ is a list of increasing indices.*
3. *$(s_1, \ldots, s_r)$ is a list of positive integers such that $s := \sum_{i=1}^{r} s_i \in [t, t+k-1]$.*
4. *$(B_1, \ldots, B_r)$ is a list of subsets of $[k]$. Moreover, for every $i \in [r]$, $|B_i| = s_i$.*
5. *$(\alpha_1, \ldots, \alpha_r)$ is a list of binary strings. For every $i \in [r]$, $|\alpha_i| = s_i$.*

We call $(r, \ell_i, s_i, B_i, \alpha_i)$ a t-witness for ρ, if there exists $\alpha \in \{0,1\}^n$ such that:

- *When we run $T_{F|_\rho}$ on α, for every $i \in [r]$, C_{ℓ_i} is the i-th term queried by $T_{F|_\rho}$.*
- *By the time $T_{F|_\rho}$ issues the i-th set of query, exactly s_i variables in C_{ℓ_i} are not known, and their "relative positions'' in V_{ℓ_i} are specified by B_i.*
- *The response to the i-th set of query is α_i.*

We define the size *of the witness $(r, \ell_i, s_i, B_i, \alpha_i)$ as $s := \sum_{i=1}^{r} s_i$.*

Notation convention

When B_i and α_i are associated with a term C_{ℓ_i}, sometimes we will slightly abuse notation by using B_i to refer to the set of variables it corresponds to in V_{ℓ_i} (Recall V_{ℓ_i} is the set of variables that appear in C_{ℓ_i}), and use α_i to denote a partial assignment $\alpha_i \in \{0,1,\star\}^n$ in which we only assign the B_i part of α_i and leave other coordinates unfixed. In this section, we will mostly use the term "witness" to denote t-witness, since we are always analyzing for a fixed t.

It is easy to see that if $\mathsf{DT}(F|_\rho) \geq t$, there must be a witness for $\mathsf{CDT}(F|_\rho) \geq t$[6]. By now, a natural idea to bound the probability of picking a bad restriction would be (1) enumerating every possible witness, (2) calculating the probability of a random ρ having such a witness, and (3) union-bounding over them. Unfortunately this is too expensive for us: we have at least $\binom{m}{t}$ choices of the list (ℓ_i). In order for this approach to be meaningful, we have to bound the probability that a random restriction has a particular witness by m^{-t}, which seems very hard, if not impossible.

It turns out we can avoid the enumeration of (ℓ_i) part in the witness. To succinctly describe the idea, let us define partial witnesses first.

▶ **Definition 13.** *Let $F = \bigvee_{i=1}^{m} C_i$ be the k-DNF and $\rho \in \{0,1,\star\}^n$ be a restriction. Let $t \geq 1$. Consider a tuple (r, s_i, B_i, α_i). We call (r, s_i, B_i, α_i) a partial t-witness for ρ, if there exists $(\ell_1, \ldots, \ell_r)$ such that $(r, \ell_i, s_i, B_i, \alpha_i)$ is a t-witness for ρ.*

▶ Remark 14. Here we make an important observation: if (r, s_i, B_i, α_i) is a partial witness for ρ, then there is only one valid list (ℓ_i) which makes $(r, \ell_i, s_i, B_i, \alpha_i)$ a witness for ρ (To see this, first note that ℓ_1 is fixed. After querying ℓ_1 and getting the response α_1, use induction). This observation will prove useful in Section 5.3 and Section 6.

Then, the proof goes by enumerating r, s_i, B_i, α_i and bounding the following:

$$\Pr_\rho[\mathsf{DT}(F|_\rho) \geq t] \leq \sum_{(r,s_i,B_i,\alpha_i)} \Pr_\rho[(r, s_i, B_i, \alpha_i) \text{ is a partial witness for } \rho]. \tag{5}$$

Fixing an $s \in [t, t+k-1]$, there are at most $(4k)^s$ possible partial witnesses (r, s_i, B_i, α_i) of size s. For each of them, if we can bound the probability by, say, $(2p)^s$, then the lemma is proved.

The witness searcher

Now, given a restriction ρ and a candidate partial witness (r, s_i, B_i, α_i), how can we decide if there is a list $(\ell_i)_i$ such that $(r, \ell_i, s_i, B_i, \alpha_i)$ constitutes a witness for ρ? We will do so by designing and using a simple procedure, which we call the witness searcher. More specifically, our searcher receives as input a restriction ρ, a partial witness (r, s_i, B_i, α_i) and an advice string $y \in \{0,1\}^n$. It outputs either a complete witness $(r, \ell_i, s_i, B_i, \alpha_i)$, or a special ERROR symbol. Its description in shown in Algorithm 2.

In the following, we use $\mathcal{S}$ to denote Algorithm 2.

▶ **Lemma 15.** *If (r, s_i, b_i, α_i) is a partial witness for ρ, then there exists an advice y that makes $\mathcal{S}$ find $(\ell_i)_{i=1}^r$. More importantly, on a uniformly random $\mathbf{y} \sim \mathcal{U}_n$, $\mathcal{S}$ finds $(\ell_1, \ldots, \ell_r)$ with probability exactly 2^{-s}.*

Proof. If $(r, s_i, \ell_i, b_i, \alpha_i)$ is a witness for ρ, it records a running transcript of $T_{F|_\rho}$ on some input $\alpha \in \{0,1\}^n$. This mean that C_{ℓ_1} is the first term queried by $T_{F|_\rho}$, which implies:

1. Every term before C_{ℓ_1} is falsified by ρ (because $T_{F|_\rho}$ skipped them).
2. C_{ℓ_1} is consistent with ρ (because $T_{F|_\rho}$ was uncertain of the value of $C_{\ell_i}(\alpha)$.

Now, consider running $\mathcal{S}$ on (r, s_i, b_i, α_i), $\mathcal{S}$ will also skip all the terms before C_{ℓ_1}. At C_{ℓ_1}, since ρ is consistent with C_{ℓ_1}, the random string $\rho \circ \mathbf{y}$ satisfies C_{ℓ_1} with probability $2^{-|B_1|}$. Conditioning on this happened, in the later execution, we modify z and replace the part corresponding to B_1 with α_1.

[6] However, we note that the converse may not be true, since our witness can only refute the existence of shallow *canonical* decision trees.

Algorithm 2 Witness Searcher.

Input: A DNF $F = \bigvee_{i=1}^{m} C_i$, a restriction $\rho \in \{0,1,\star\}^n$, a partial witness (r, s_i, B_i, α_i), an advice $y \in \{0,1\}^n$.

initialize:

- $z \leftarrow \rho \circ y$.
- $j^* \leftarrow 0$.
- $c \leftarrow 1$.

while $c \leq r$ **do**

- Find the first $j > j^*$ such that C_j is satisfied by z. If no such i exists, return ERROR.
- Set $\ell_c \leftarrow j$, and associate B_c, α_c with C_{ℓ_c}.
- Replace the B_c part of z with α_c. That is, $z \leftarrow \rho \circ \alpha_1 \circ \cdots \circ \alpha_c \circ y$.
- $c \leftarrow c + 1$.
- $j^* \leftarrow j$.

return $(r, \ell_i, s_i, B_i, \alpha_i)$

At this point, we go back and inspect the execution of $T_{F|_\rho}$ on α. Since $(r, s_i, \ell_i, b_i, \alpha_i)$ is the running transcript of $T_{F|_\rho}$ on α, we know that α_1 was the response that $T_{F|_\rho}$ received from querying alive variables in C_{ℓ_1}. Since $T_{F|_\rho}$ issued its second bunch of queries to alive variables in C_{ℓ_2}, it implies that terms from C_{ℓ_1+1} to C_{ℓ_2-1} are all falsified by $\rho \circ \alpha_1$, and C_{ℓ_2} is consistent with $\rho \circ \alpha_1$. Then, it follows that $\rho \circ \alpha_1 \circ \mathbf{y}$ satisfies C_{ℓ_2} with probability $2^{-|B_2|}$. Conditioning on this happened, the searcher will proceed with string $\rho \circ \alpha_1 \circ \alpha_2 \circ y$ and we can again consider the execution of $T_{F|_\rho}$ after querying C_{ℓ_2}. We do this argument so on and so forth, until we have identified all of r indices $\ell_1, \ldots, \ell_r$ and exit the procedure. In summary, we have shown there exists y which makes $\mathcal{S}$ find the list $(\ell_i)_{i=1}^r$, and the probability of sampling such a y is

$$2^{-\sum_{i=1}^{c} |B_i|} = 2^{-s}$$

as desired. ◀

Decoupling

If we inspect the execution of $\mathcal{S}$ carefully, we can notice that it only needs to know the string $z = \rho \circ y$ to work. In particular, it *does not* need to know which part of z is fixed in the restriction ρ. Therefore, we can revise $\mathcal{S}$ to get a searcher $\mathcal{S}'$: the input to $\mathcal{S}'$ is now a string z and (r, s_i, B_i, α_i). Otherwise it runs identically the same way as $\mathcal{S}$. We denote the output of $\mathcal{S}'$ as $\mathcal{S}'(z, (r, s_i, B_i, \alpha_i))$. By Lemma 15, we have

$$\begin{aligned} &\Pr_{\rho \sim \mathcal{R}}[(r, s_i, B_i, \alpha_i) \text{ is a partial witness for } \rho] \\ \leq\ & 2^s \cdot \Pr_{\rho \sim \mathcal{R}, y \sim \mathcal{U}_n}[\mathcal{S}'(\rho \circ y, (r, s_i, B_i, \alpha_i)) \text{ is a witness for } \rho]. \end{aligned}$$

Denote $\rho = \Lambda[x, \star]$. We observe that

$$\mathbb{1}\{\ (r, \ell_i, s_i, B_i, \alpha_i) \text{ is a witness for } \rho\} \leq \mathbb{1}\left\{ \left(\bigcup_{j=1}^{r} B_j \right) \subseteq \Lambda \right\}. \tag{6}$$

For a $(t+k)$-wise p-bounded set $\mathbf{\Lambda}$, the event on the right hand side holds with probability $p^{\sum_j |B_j|}$. Then, we have

$$\begin{aligned}
&\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},0],y\sim\mathcal{U}_n}[\ \mathcal{S}'(\rho\circ y,(r,s_i,B_i,\alpha_i))\text{ is a witness for }\rho]\\
&=\mathbb{E}_{\mathbf{\Lambda}}\mathbb{E}_{x,y\sim\mathcal{U}_n}[\mathbb{1}\{\mathcal{S}'(\mathbf{\Lambda}[x,y],(r,s_i,B_i,\alpha_i))\text{ is a witness for }\rho\}]\\
&=\mathbb{E}_{z\sim\mathcal{U}_n}\left[\Pr_{\mathbf{\Lambda}}[\mathcal{S}'(z,(r,s_i,B_i,\alpha_i))\text{ is a witness for }\rho]\right]\\
&\le p^s,
\end{aligned}$$

where the third equality is due to that $\Lambda[x,y]$ is distributed as $\mathcal{U}_n$ when $x,y\sim\mathcal{U}_n$, and the last inequality holds by (6) and the $(t+k)$-wise p-bounded property of $\mathbf{\Lambda}$.

Wrapping-up

We finish the proof by enumerating all (r,s_i,B_i,α_i) and taking a summation.

$$\begin{aligned}
\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[\mathsf{DT}(F|_\rho)\ge t]&\le\sum_{(r,s_i,B_i,\alpha_i)}\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[(r,s_i,B_i,\alpha_i)\text{ is a partial witness for }\rho]\\
&\le\sum_{(r,s_i,B_i,\alpha_i)}2^s\cdot\mathbb{E}_{z\in\mathcal{U}_n}\left[\Pr_{\mathbf{\Lambda}}[\mathcal{S}'(z,(r,s_i,B_i,\alpha_i))\text{ is a witness for }\mathbf{\Lambda}[\mathbf{x},\star]]\right]\\
&\le\sum_{s=t}^{t+k}2^s(4k)^sp^s\\
&\le(10kp)^t.
\end{aligned}\tag{7}$$

5.3 Proof when x is pseudorandom

Now we consider the case that $\mathbf{x}$ is not truly random. There could even be correlation between $\mathbf{x}$ and $\mathbf{\Lambda}$. Our only requirement for $\mathbf{x}$ is that, for every fixed $\mathbf{\Lambda}$, $\mathbf{x}$ is a pseudorandom string that ε-fools CNFs of size at most m. First of all, by Remark 14, the following equation is established.

$$\mathbb{1}\{(r,s_i,B_i,\alpha_i)\text{ is a partial witness for }\rho\}=\sum_{\ell_i}\mathbb{1}\{(r,\ell_i,s_i,B_i,\alpha_i)\text{ is a witness for }\rho\}.\tag{8}$$

If we inspect the deduction in (7), it actually shows the following:

$$\sum_{(r,s_i,B_i,\alpha_i)}\sum_{\ell_i}\Pr_{\rho\sim\mathbf{\Lambda}[\mathcal{U}_n,\star]}[(r,\ell_i,s_i,B_i,\alpha_i)\text{ is a witness for }\rho]\le(10kp)^t.\tag{9}$$

Next, fixing the set Λ and a tuple of $(r,\ell_i,s_i,B_i,\alpha_i)$, we consider the predicate

$$h_\Lambda^{(r,\ell_i,s_i,B_i,\alpha_i)}(x):=\mathbb{1}\{(r,\ell_i,s_i,B_i,\alpha_i)\text{ is a witness for }\Lambda[x,\star]\}.$$

We omit the superscript and subscript of h when they are clear from context. We claim that h can be implemented by a CNF of size at most m. To see this, note that $h(x)=1$ if all of the following hold.

- For every $j<\ell_1$, C_j is falsified by $\Lambda[x,\star]$. For this to be true, there is at least one variable in C_j that is not chosen by Λ and is assigned to the opposite value to its appearance in C_j. Note that we can use an OR over relevant variables or their negations to verify if C_j is falsified.

- For $j = \ell_1$, C_{ℓ_1} cannot be falsified by $\Lambda[x, \star]$, which means all variables in C_{ℓ_1} that are not chosen by Λ are assigned so that it does not contradict C_{ℓ_1}. We can use an AND over those variables to verify this.
- For every $\ell_1 < j < \ell_2$, C_j is falsified by $\Lambda[x, \alpha_1]$, which means that either C_j is falsified by α_1, or there is at least one variable in C_j that is not chosen by Λ and is assigned to the opposite value. If C_j is falsified by α_1, there is nothing to verify. Otherwise, we can verify it by an OR over relevant variables.
- The same reasoning applies to C_j for every $j \geq \ell_2$. Finally, after we finish the verification for C_{ℓ_r}, we are done.

In summary, we can implement h by a CNF (AND of ORs). It is obvious that the size of the CNF is no larger than the size of F, which is m. Therefore, we conclude that $\mathbf{x}$ ε-fools h. That is,

$$\left| \Pr_{x \sim \mathcal{U}_n} [h_\Lambda^{(r,\ell_i,s_i,B_i,\alpha_i)}(x)] - \Pr_{x \sim \mathbf{x}} [h_\Lambda^{(r,\ell_i,s_i,B_i,\alpha_i)}(x)] \right| \leq \varepsilon. \tag{10}$$

Finally, we have

$$\begin{aligned}
\Pr_{\rho \sim \mathbf{\Lambda}[\mathbf{x},\star]} [\mathsf{DT}(F|_\rho) \geq t] &\leq \sum_{(r,s_i,B_i,\alpha_i)} \Pr_{\rho \sim \mathbf{\Lambda}[\mathbf{x},\star]} [(r, s_i, B_i, \alpha_i) \text{ is a partial witness for } \rho] \\
&\leq \sum_{(r,s_i,B_i,\alpha_i)} \sum_{\ell_i} \Pr_{\rho \sim \mathbf{\Lambda}[\mathbf{x},\star]} [(r, \ell_i, s_i, B_i, \alpha_i) \text{ is a witness for } \rho] \\
&\leq \sum_{(r,s_i,B_i,\alpha_i)} \sum_{\ell_i} \varepsilon + \Pr_{\rho \sim \mathbf{\Lambda}[\mathcal{U}_n,\star]} [(r, \ell_i, s_i, B_i, \alpha_i) \text{ is a witness for } \rho] \\
&\leq (10kp)^t + (4m)^{t+k} \cdot \varepsilon.
\end{aligned}$$

Here, the first line is due to the argument in Section 5.2. The second line holds by (8). The third line is due to (10). The last line holds because (9) and there are at most $(4m)^{k+t}$ possible tuples $(r, \ell_i, s_i, B_i, \alpha_i)$.

6 Derandomizing the Multi-Switching Lemma

In this section, we prove Lemma 9. Recall the statement.

▶ **Reminder of Lemma 9.** *Let $t, w, k, n, m \geq 1$ be integers. Let $p, \varepsilon > 0$ be reals. Let $\mathcal{F} = \{F_1, \ldots, F_m\}$ be a list of size-m k-DNFs on inputs $\{0,1\}^n$. Let $(\mathbf{\Lambda}, \mathbf{x})$ be a joint random variable satisfying the following:*

- *$\mathbf{\Lambda}$ is a $(t+k)$-wise p-bounded subset of $[n]$,*
- *Conditioning on $\mathbf{\Lambda}$, $\mathbf{x}$ is a random string that ε-fools size-(m^2) CNF.*

Then we have

$$\Pr_{T \sim \mathbf{\Lambda}, x \sim \mathbf{x}} [\mathcal{F}|_{\Lambda[x,\star]} \text{ does not have } w\text{-partial depth-}t \text{ } DT] \leq 4m^{t/w}(24pk)^t + (24m)^{t+k} \cdot \varepsilon.$$

The rest of the section is devoted to the proof of Lemma 9. Section 6.1 states some preliminary tools and includes a proof overview. Section 6.2 considers the case that $\mathbf{x}$ is truly random. It extends the idea in Section 5.2 and has a rather similar structure. Finally in Section 6.3, we consider the case that $\mathbf{x}$ is pseudorandom, and finish the proof.

6.1 Preliminaries

The canonical partial decision tree

Let ρ be a random restriction under which $\mathcal{F}|_\rho$ fails to have a w-partial depth-t decision tree. Let $\beta \in \{0,1\}^n$ be an unknown string that our decision tree shall query. We consider the following attempt (see Algorithm 3) to construct a partial decision tree for $\mathcal{F}|_\rho$. Here $z \in \{0,1\}^n$ is an auxiliary string to be chosen. Every choice of $z \in \{0,1\}^n$ yields a different partial decision tree for $\mathcal{F}$. In [25], the construction was called "canonical partial decision tree".

Algorithm 3 Canonical Partial Decision Tree.

Input: A list of DNFs $\mathcal{F} = \{F_1, \ldots, F_m\}$, black-box access to a string $\beta \in \{0,1\}^n$, and an auxiliary string $z \in \{0,1\}^n$.

initialize:
- $x \leftarrow (\star)^n$.
- $j \leftarrow 1$.
- counter $\leftarrow 0$.

while counter $< t$ **do**
- Find the smallest $i \geq j$ such that $\mathsf{DT}(F_i|_x) > w$. If no such i exists, exit the loop.
- $y \leftarrow (\star)^n$.
- $I \leftarrow \emptyset$.
- **while** $F_i|_{x \circ y}(\star)$ *is not constant and* counter $< t$ **do**
  - $C_{i,q} \leftarrow$ the term that $T_{F_i|_{x\circ y}}$ will query.
  - $B_{i,q} \leftarrow$ the set of unknown variables in $C_{i,q}|_{x\circ y}$.
  - $y_{B_{i,q}} \leftarrow z_{B_{i,q}}$.
  - $I \leftarrow I \cup B_{i,q}$.
  - counter $\leftarrow$ counter $+ |B_{i,q}|$.
- Query β_I, and set $x_I \leftarrow \beta_I$.
- $j \leftarrow i$.

return x

Note that in Algorithm 3, it is possible that for a formula F_i to get picked by the outer "while" loop more than once.

Proof overview

Before we dive into the formal proof, we try to give some (over-simplified) intuition here. If $\mathcal{F}$ does not have w-partial DT of depth t, then Algorithm 3 fails to construct a partial decision tree of depth t on every auxiliary string z. However, we will only consider some "adversarially chosen" z's. That is, we only consider those z's that trick Algorithm 3 to make at least w queries on each chosen formula. By doing so, we are guaranteed that Algorithm 3 only chooses $\frac{t}{w}$ formulae in total.

Then, we can pay a factor of $m^{\frac{t}{w}}$ to enumerate a subset of $\frac{t}{w}$ formulae in $\mathcal{F}$, and calculate the probability that Algorithm 3 "gets stuck" on those formulae. Having fixed those $\frac{t}{w}$ formulae, we can use ideas similar to Section 5.2, and bound the probability by $O(pk)^t$. In summary, we get

$$\Pr[\text{bad restriction}] \leq m^{\frac{t}{w}} \cdot O(pk)^t.$$

Here we only considered the case that the restriction string $\mathbf{x}$ is purely random. For the case that both $\mathbf{\Lambda}$ and $\mathbf{x}$ are pseudorandom, we use tricks similar to Section 5.3 and pay another additive factor.

6.2 Proof when $\mathbf{x}$ is truly random

We start the proof by considering the case that $\mathbf{x}$ is a truly random string. Fix ρ to be a bad restriction, under which $\mathcal{F}|_\rho$ fails to have a w-partial depth-t decision tree. It implies that on every $z \in \{0,1\}^n$, Algorithm 3 fails to construct a partial decision tree for $\mathcal{F}$. However, for ease of our analysis, we will only consider a special class of string z. We give the following definition.

▶ **Definition 16.** *Let ρ be a bad restriction for $\mathcal{F}$. We call $z \in \{0,1\}^n$ a* powerful (w,t)*-refutation for ρ, if there exists $\beta \in \{0,1\}^n$ satisfying the following: Algorithm 3 on input $(\mathcal{F}|_\rho, \beta, z)$ makes at least t queries. Moreover, at each time we query β_I, it is guaranteed that $|I| \geq w$.*

Powerful refutations always exist for bad restrictions, as shown in the following.

▶ **Lemma 17.** *If $\mathcal{F}|_\rho$ does not have w-partial decision tree of depth t, then there exists a powerful (w,t)-refutation for ρ.*

Proof. If $\mathcal{F}_\rho$ does not have w-partial decision tree of depth t, then it means Algorithm 3 fails to construct a w-partial DT of depth t for every $z \in \{0,1\}^n$. Then we construct a powerful refutation z, together with its associated adversarial input β, in the following way.

- Initially, we set $z = \beta = (\star)^n$. Then we monitor the execution of Algorithm 3 on $(\mathcal{F}, \beta, z)$, and gradually fill in z, β when they are accessed by Algorithm 3.
- Whenever Algorithm 3 chooses one formula $F_i|_{\rho \circ x}$ in the outer while-loop, we do the following:
  - When running inside the inner while-loop, Algorithm 3 needs to consult the auxiliary string z. We can fill in relevant variables of z in such a way that Algorithm 3 consults at least w bits from z in the loop. Since $\mathsf{DT}(F_i|_{\rho \circ x}) \geq w$, this is always possible.
  - After finishing the inner loop, Algorithm 3 will make a query to β_I. At this point, we set β_I in such a way that $\mathcal{F}|_{\rho \circ x \circ \beta}$ does not have w-partial DT of depth $(t - \text{counter})$. This is always possible as long as $\mathcal{F}_{\rho \circ x}$ does not have w-partial DT of depth $(t - \text{counter} + |I|)$, which holds by induction.
- After Algorithm 3 returns, we fill in the remaining bits of β, z with zeros. ◀

Having established Lemma 17, we come up with the following definition of "global witness" naturally.

▶ **Definition 18.** *Let t, w be two integers. Consider a list of k-DNFs $\mathcal{F} = \{F_1, \ldots, F_m\}$. Suppose $\rho \in \{0,1,\star\}^n$ is a restriction. Let $(R, L_i, S_i, W_i, \beta_i)$ be a tuple, where*

1. *$1 \leq R \leq \frac{t}{w}$ is an integer;*
2. *$1 \leq L_1 \leq L_2 \leq \cdots \leq L_R \leq m$ is a list of R non-decreasing indices;*
3. *$S_1 \ldots, S_R$ is a list of R integers such that $\sum_{i=1}^R S_i \in [t, t+k]$;*
4. *$W_1, \ldots, W_R$ is a list of witnesses (as per Definition 12). For every $i \in [R]$, W_i has size S_i;*
5. *$\beta_1, \ldots, \beta_R$ are R strings where $|\beta_i| = S_i$ for every $i \in [R]$.*

We call the tuple a (w,t)-global witness for ρ, if it satisfies the following.

1. *Set $\rho_1 = \rho$. W_1 is a S_1-witness for $F_{L_1}|_{\rho_1}$.*
2. *For every $i \geq 2$, let $I_{i-1} \subseteq [n]$ be the set of variables involved in W_{i-1}. Note that $|I_{i-1}| = S_{i-1}$ since the size of W_{i-1} is S_{i-1}. Identify β_{i-1} as a partial assignment in $\{0,1,\star\}^n$ where only the part $\beta_{i-1,I_{i-1}}$ is set and other coordinates are filled in with $\star$. Construct $\rho_i = \rho_{i-1} \circ \beta_{i-1}$. Then W_i is a S_i-witness for $F_{L_i}|_{\rho_i}$.*

The size of the global witness is defined as $\sum_{i=1}^{R} S_i$.

As a corollary of Lemma 17, we have:

▶ **Corollary 19.** *Consider a list of k-DNFs $\mathcal{F} = \{F_1, \ldots, F_m\}$. Suppose $\rho \in \{0,1,\star\}^n$ is a restriction such that $\mathcal{F}|_\rho$ does not have w-partial depth-t decision tree. Then there exists a (w,t)-global witness for ρ.*

Proof. By Lemma 17, there exists a powerful (w,t)-refutation for $\mathcal{F}|_\rho$. Take one such refutation z with its adversarial input β. Inspect the execution of Algorithm 3 on $(\mathcal{F}|_\rho, \beta, z)$ and record the transcript. The transcript contains the desired tuple. ◀

Similar to what we have done in Section 5.2, we define the global partial witness.

▶ **Definition 20.** *Let t, w be two integers. Consider a list of k-DNFs $\mathcal{F} = \{F_1, \ldots, F_m\}$. Suppose $\rho \in \{0,1,\star\}^n$ is a restriction. Let $(R, L_i, S_i, P_i, \beta_i)$ be a tuple, where*

1. *$1 \leq R \leq \frac{t}{w}$ is an integer;*
2. *$1 \leq L_1 \leq L_2 \leq \cdots \leq L_R \leq m$ is a list of R non-decreasing indices;*
3. *$S_1 \ldots, S_R$ is a list of R integers such that $\sum_{i=1}^{R} S_i \in [t, t+k]$;*
4. *$P_1, \ldots, P_R$ is a list of partial witnesses. For every $i \in [R]$, P_i has size S_i.*
5. *$\beta_1, \ldots, \beta_R$ are R strings where $|\beta_i| = S_i$ for every $i \in [R]$.*

We call $(R, L_i, S_i, P_i, \beta_i)$ a (w,t)-global partial witness for ρ, if we can complete P_i to get a witness W_i for every $i \in [R]$, such that $(R, L_i, S_i, W_i, \beta_i)$ is a global witness for ρ.

▶ Remark 21. Again, it is easy to see by induction that, for every global partial witness, there is exactly one way to complete it and get a global witness.

By a simple counting, it turns out for every $S \in [t, t+k]$, there are at most $2m^{t/w}(12k)^S$ possible global partial witnesses of size S:

1. First, $2m^{t/w}$ ways to choose at most $\frac{t}{w}$ formulae in $\mathcal{F}$.
2. Then, 3^S ways to partition the s units of "query budget" into R partial witnesses as $S = \sum_{i=1}^{R} S_i$, and further partition the budget for each partial witness as $S_i = \sum_{j=1}^{r_i} s_{i,j}$.
3. Then, at most k^S ways to construct sets $B_{i,j}$ for each partial witness.
4. Finally, there are 4^S ways to choose β_i's, as well as $\alpha_{i,j}$'s in each partial witness.

Fixing a global partial witness $(R, L_i, S_i, P_i, \beta_i)$, we try to bound the probability that a random ρ has this witness by $(2p)^k$. Using the witness searcher (Algorithm 2) as a subroutine, we design a global witness searcher first. Again, the global witness searcher needs access to an advice string $y \in \{0,1\}^n$. See Algorithm 4.

In the following, we use $\mathcal{S}$ to denote Algorithm 4 for brevity. Based on Lemma 15, we prove:

▶ **Lemma 22.** *If $(R, L_i, S_i, P_i, \beta_i)$ is indeed a global partial witness for ρ, then there exists an advice y that makes $\mathcal{S}$ find $(R, L_i, S_i, P_i, \beta_i)$. More importantly, on a uniformly random $\mathbf{y} \sim \mathcal{U}_n$, $\mathcal{S}$ finds $(\ell_1, \ldots, \ell_r)$ with probability exactly 2^{-S}, where $S := \sum_{i=1}^{R} S_i$ is the size of the global witness.*

Algorithm 4 Global Witness Searcher.

Input: A list of DNFs $\mathcal{F} = \{F_1, \ldots, F_m\}$, a restriction $\rho \in \{0,1,\star\}^n$, a global partial witness $(R, L_i, S_i, P_i, \beta_i)$, and an advice $y \in \{0,1\}^n$.

initialize:

- $z \leftarrow \rho \circ y$.
- $c \leftarrow 1$.
- $\rho^{(1)} \leftarrow \rho$.

while $c \le R$ **do**

- Run Algorithm 2 on $(F_{L_c}, \rho^{(c)}, P_c, y)$. If it reports ERROR, report ERROR and terminate the procedure. Otherwise let W_c be the witness returned.
- $I_c \leftarrow$ the set of variables involved in W_c.
- Identify β_c as a partial assignment, where only β_{I_c} is fixed.
- $\rho^{(c+1)} \leftarrow \rho^{(c)} \circ \beta_c$.
- $c \leftarrow c + 1$.

return x

Proof. We examine the execution of $\mathcal{S}$. It first runs Algorithm 2 as a subroutine, trying to find W_1 for P_1. With probability 2^{-S_1} over $\mathbf{y}$, Algorithm 2 succeeds in finding W_1. Conditioning on this happened, we have only committed the assignment of $\mathbf{y}_{I_1}$, and the rest part of $\mathbf{y}$ is still uniformly random. Also note that we set $\rho^{(2)}_{I_1}$ to β_1, which will hide $\mathbf{y}_{I_1}$ in the later execution. This enables us to do an induction and finish the proof. ◀

Decoupling

Now, we can observe that $\mathcal{S}$ only needs to know $z = \rho \circ y$ to work. In particular, it does not need to know which part of z is fixed in ρ. Therefore, we can revise $\mathcal{S}$ to get a searcher $\mathcal{S}'$: the input to $\mathcal{S}'$ is now a string z and a global partial witness $(R, L_i, S_i, P_i, \beta_i)$. Otherwise it runs identically the same way as $\mathcal{S}$. We denote the output of $\mathcal{S}'$ as $\mathcal{S}'(z, (R, L_i, S_i, P_i, \beta_i))$. By Lemma 22, we have

$$\begin{aligned}
&\Pr_{\rho\sim\mathcal{R}}[(R, L_i, S_i, P_i, \beta_i) \text{ is a global partial witness for } \rho] \\
&\le 2^S \cdot \Pr_{\rho\sim\mathcal{R}, y\sim\mathcal{U}_n}[\mathcal{S}'(\rho \circ y, (R, L_i, S_i, P_i, \beta_i)) \text{ is a global witness for } \rho]
\end{aligned}$$

We observe that

$$\mathbb{1}\{\ (R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \rho\} \le \mathbb{1}\left\{\left(\bigcup_{j=1}^{R} I_j\right) \subseteq \Lambda\right\}. \tag{11}$$

For a $(t+k)$-wise p-bounded $\mathbf{\Lambda}$, the event on the right hand side holds with probability $p^{\sum_j |I_j|}$. Then, we have

$$\begin{aligned}
&\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},0], y\sim\mathcal{U}_n}[\ \mathcal{S}'(\rho \circ y, (R, L_i, S_i, P_i, \beta_i)) \text{ is a global witness for } \rho] \\
&= \mathbb{E}_{\mathbf{\Lambda}}\mathbb{E}_{x,y\sim\mathcal{U}_n}[\mathbb{1}\{\mathcal{S}'(\mathbf{\Lambda}[x,y], (R, L_i, S_i, P_i, \beta_i)) \text{ is a global witness for } \rho\}] \\
&= \mathbb{E}_{z\sim\mathcal{U}_n}\left[\Pr_{\mathbf{\Lambda}}[\mathcal{S}'(z, (R, L_i, S_i, P_i, \beta_i)) \text{ is a global witness for } \rho]\right] \\
&\le p^S,
\end{aligned}$$

where the third line is due to that $\Lambda[x,y]$ is distributed as $\mathcal{U}_n$ when $x, y \sim \mathcal{U}_n$, and the last inequality holds by (11) and the $(t+k)$-wise p-bounded property of Λ.

Wrapping-up

We finish the proof by enumerating all $(R, L_i, S_i, P_i, \beta_i)$ and taking a summation.

$$\begin{aligned}
&\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[\mathcal{F}|_\rho \text{ does not have } w\text{-partial depth-}t \text{ decision tree}] \\
\leq &\sum_{(R,L_i,S_i,P_i,\beta_i)} \Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[(R, L_i, S_i, P_i, \beta_i) \text{ is a global partial witness for } \rho] \\
\leq &\sum_{(R,L_i,S_i,P_i,\beta_i)} 2^S \cdot \mathbb{E}_{z\in\mathcal{U}_n}\left[\Pr_{\mathbf{\Lambda}}[\mathcal{S}'(z,(R, L_i, S_i, P_i, \beta_i)) \text{ is a global witness for } \mathbf{\Lambda}[\mathbf{x},\star]]\right] \\
\leq &\sum_{S=t}^{t+k} 2m^{t/w}2^S(12k)^S p^S \\
\leq &\, 4m^{t/w}(24kp)^t. \qquad (12)
\end{aligned}$$

6.3 Proof when x is pseudorandom

Now we consider the case that $\mathbf{x}$ is not truly random. Our only requirement for $\mathbf{x}$ is that, for every fixed $\mathbf{\Lambda}$, $\mathbf{x}$ is a pseudorandom string that ε-fools CNFs of size at most m. First of all, by Remark 21, the following equation is established.

$$\begin{aligned}
&\mathbb{1}\{(R, L_i, S_i, P_i, \beta_i) \text{ is a global partial witness for } \rho\} \\
= &\sum_{W_i:\text{completion of } P_i} \mathbb{1}\{(R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \rho\}. \qquad (13)
\end{aligned}$$

Then, the deduction in (12) implies that

$$\sum_{(R,L_i,S_i,W_i,\beta_i)} \Pr_{\rho\sim\mathbf{\Lambda}[\mathcal{U}_n,\star]}[(R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \rho] \leq 4m^{t/w}(24kp)^t. \qquad (14)$$

Next, fixing a tuple $(R, L_i, S_i, W_i, \beta_i)$ and an instantiation of Λ, we consider the predicate

$$h_\Lambda^{(R,L_i,S_i,W_i,\beta_i)}(x) := \mathbb{1}\{(R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \Lambda[x,\star]\}.$$

We omit the superscript and subscript of h when they are clear from context. We claim that h can be implemented by a CNF of size at most $\sum_{i=1}^m \mathsf{size}(F_i) \leq m^2$. To see this, note that $h(x) = 1$ if for every $i \in [R]$, W_i is a witness for $\rho \circ \beta_1 \circ \cdots \circ \beta_{i-1}$. By argument in Section 6.3, this can be verified using a CNF of size $\mathsf{size}(F_{L_i})$. Therefore, we can evaluate h using an AND over R CNFs, which is itself a larger CNF. Therefore, we conclude that $\mathbf{x}$ ε-fools h. That is,

$$\left|\Pr_{x\sim\mathcal{U}_n}[h_\Lambda^{(R,L_i,S_i,W_i,\beta_i)}(x)] - \Pr_{x\sim\mathbf{x}}[h_\Lambda^{(R,L_i,S_i,W_i,\beta_i)}(x)]\right| \leq \varepsilon. \qquad (15)$$

Finally, we have

$$\begin{aligned}
&\Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[\mathcal{F}|_\rho \text{ does not have } w\text{-partial depth-}t \text{ decision tree}] \\
\leq &\sum_{(R,L_i,S_i,W_i,\beta_i)} \Pr_{\rho\sim\mathbf{\Lambda}[\mathbf{x},\star]}[(R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \rho] \\
\leq &\sum_{(R,L_i,S_i,W_i,\beta_i)} \varepsilon + \Pr_{\rho\sim\mathbf{\Lambda}[\mathcal{U}_n,\star]}[(R, L_i, S_i, W_i, \beta_i) \text{ is a global witness for } \rho] \\
\leq &\, 4m^{t/w}(24kp)^t + (24m)^{t+k}\cdot\varepsilon.
\end{aligned}$$

Here, the first line is due to the argument in Section 6.2 and observation (13). The second line is due to (15). The last line holds because (14) and there are at most $(24m)^{t+k}$ possible tuples $(R, L_i, S_i, W_i, \beta_i)$.

References

1 Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Adv. Comput. Res.*, 5:199–222, 1989.

2 Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating acˆ0 by small height decision trees and a deterministic algorithm for #acˆ0sat. In *Computational Complexity Conference*, pages 117–125. IEEE Computer Society, 2012.

3 Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5):28:1–28:10, 2010. doi:10.1145/1754399.1754401.

4 Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. *Theory Comput.*, 14(1):1–55, 2018. doi:10.4086/toc.2018.v014a009.

5 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517. Springer, 2010. doi:10.1007/978-3-642-15369-3_38.

6 Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil P. Vadhan. Monotone branching programs: Pseudorandomness and circuit complexity. *Electron. Colloquium Comput. Complex.*, page 18, 2021. URL: https://eccc.weizmann.ac.il/report/2021/018.

7 Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 946–955. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00093.

8 Oded Goldreich and Avi Wigderson. On the size of depth-three boolean circuits for computing multilinear functions. *Electron. Colloquium Comput. Complex.*, page 43, 2013. URL: https://eccc.weizmann.ac.il/report/2013/043.

9 Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Comput. Complex.*, 22(2):275–310, 2013. doi:10.1007/s00037-013-0068-6.

10 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 120–129. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.77.

11 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018. doi:10.1137/17M1129088.

12 Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to AC. *Random Struct. Algorithms*, 54(2):289–303, 2019. doi:10.1002/rsa.20786.

13 Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014. doi:10.1137/120897432.

14 John Hastad. Almost optimal lower bounds for small depth circuits. *Adv. Comput. Res.*, 5:143–170, 1989.

15 Pooya Hatami, William Hoza, Avishay Tal, and Roei Tell. Fooling constant-depth threshold circuits. *Electron. Colloquium Comput. Complex.*, page 2, 2021. URL: https://eccc.weizmann.ac.il/report/2021/002.

16 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for ac^0. In *SODA*, pages 961–972. SIAM, 2012.

17 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 111–119. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.78.

18 Zander Kelley. An improved derandomization of the switching lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 272–282. ACM, 2021. doi:10.1145/3406325.3451054.

19 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. *Theory Comput.*, 16:1–50, 2020. doi:10.4086/toc.2020.v016a007.

20 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

21 Michael Luby and Boban Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4/5):415–433, 1996. doi:10.1007/BF01940873.

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 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM J. Comput.*, 42(3):1275–1301, 2013. doi:10.1137/100811623.

24 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.

25 Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 45:1–45:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.45.

26 Avishay Tal. Tight bounds on the fourier spectrum of AC0. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CCC.2017.15.

27 Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 242–247. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.32.

28 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

29 Emanuele Viola. AC0 unpredictability. *ACM Trans. Comput. Theory*, 13(1):5:1–5:8, 2021. doi:10.1145/3442362.

Characterizing Derandomization Through Hardness of Levin-Kolmogorov Complexity

Yanyi Liu
Cornell Tech, New York, NY, USA

Rafael Pass
Cornell Tech, New York, NY, USA
Tel-Aviv University, Israel

Abstract

A central open problem in complexity theory concerns the question of whether all efficient randomized algorithms can be simulated by efficient deterministic algorithms. We consider this problem in the context of promise problems (i.e,. the prBPP v.s. prP problem) and show that for all sufficiently large constants c, the following are *equivalent*:

- prBPP = prP.
- For every $\mathsf{BPTIME}(n^c)$ algorithm M, and every sufficiently long $z \in \{0,1\}^n$, there exists some $x \in \{0,1\}^n$ such that M fails to decide whether $Kt(x \mid z)$ is "very large" ($\geq n-1$) or "very small" ($\leq O(\log n)$).

where $Kt(x \mid z)$ denotes the Levin-Kolmogorov complexity of x conditioned on z. As far as we are aware, this yields the first full *characterization* of when prBPP = prP through the hardness of some class of problems. Previous hardness assumptions used for derandomization only provide a one-sided implication.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases Derandomization, Kolmogorov Complexity, Hitting Set Generators

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.35

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2022/084/

Funding *Rafael Pass*: Worked done while being on a sabbatical at Tel-Aviv University. Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, and a JP Morgan Faculty Award. This material is based upon work supported by DARPA under Agreement No. HR00110C0086. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

Acknowledgements We thank the anonymous reviewers for many helpful comments, and most notably for making us aware of [11].

1 Introduction

Randomness is an ubiquitous tool in algorithm design. A central open problem in complexity theory concerns the question of whether all randomized algorithms can be derandomized; that is, can every randomized polynomial-time algorithm be simulated by a deterministic polynomial-time one? In this work, we consider this question with respect to promise problems; as usual, we refer to prBPP as the class of promise problems (as opposed to languages) that can be solved in probabilistic polynomial time (with 2-sided error), and prP to the class of promise problems than can be solved in deterministic polynomial time, and we here consider the question of whether prBPP = prP.

A long sequence of works originating with the works of Blum-Micali [5], Yao [29], Nisan [21], Nisan-Widgerson [22], Babai-Fortnow-Nisan-Wigderson [4], Impagliazzo-Wigderson [14] have presented beautiful connections between this problem and the problem of proving computational-complexity lower bounds – the so-called *hardness v.s. randomness* paradigm.

For instance, the results of [22, 14] show that $\mathsf{prBPP} = \mathsf{prP}$ under the assumption that $\mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ contains a language that requires Boolean circuits of size $2^{\Omega(n)}$ for almost all input lengths (i.e., E is not contained in $\mathsf{ioSIZE}(2^{\Omega(n)})$). Additionally, results by Impagliazzo, Kabanets and Wigderson [13] show a *partial* converse: if $\mathsf{prBPP} = \mathsf{prP}$, then some non-trivial circuit lower bound also must hold. In more detail, if $\mathsf{prBPP} = \mathsf{prP}$ (or even just $\mathsf{MA} = \mathsf{NP}$), then $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$; very recent works [26, 20] managed to strengthen the conclusion to e.g., $\mathsf{NTIME}[n^{\mathsf{poly}\log n}] \not\subseteq \mathsf{P/poly}$.

But despite over 40 years of research on the topic of derandomization, there is still a large "gap" between the hardness assumptions required for derandomizing prBPP, and the ones that are known to be necessary for derandomization, leaving open the following question:

Does there exist some hardness assumption that is equivalent to $\mathsf{prBPP} = \mathsf{prP}$*?*

Most notably, known derandomization results for prBPP require complexity lower-bounds on functions in EXP, whereas it is only known that derandomization of prBPP implies complexity lower bounds for functions in non-deterministic classes.

Circumventing this problem, an elegant result by Goldreich [10] – which will be instrumental in the current work – provides a characterization of $\mathsf{prBPP} = \mathsf{prP}$ through the existence of a generalized form of a pseudo-random generator (PRG). In more detail, Goldreich [10] shows that $\mathsf{prBPP} = \mathsf{prP}$ if and only if a certain type of a *targeted PRG* exists; roughly speaking, this is a PRG g that gets an additional target z as input, and indistinguishability holds with respect to uniform algorithms that also get the target z as input. In other words, g is just like a normal PRG, but with the exception that both the PRG and the distinguisher get access to the auxiliary "target" string z, and we require security to hold for all strings z. Since we consider PRGs in the context of derandomization, we allow the running-time of the PRG to be (polynomially) larger than the running-time of the distinguisher. As noted by Goldreich, such targeted PRGs suffice to derandomize prBPP, where the instance to be decided can be used as the "target"; Goldreich next provides a construction of such a targeted PRG assuming that $\mathsf{prBPP} = \mathsf{prP}$. As pointed out by Goldreich, however, the existence of a PRG is not a "hardness" assumption, and thus his work does not provide a characterization of derandomization in terms of some hardness assumption.

As far as we are aware, the only work that that shows an *equivalence* between $\mathsf{prBPP} = \mathsf{prP}$ and some hardness assumption does so under a conjecture (a weaker version of the non-deterministic exponential-time hypothesis) [8]. Very recently, however, two intriguing works make progress on closing the gap between the necessary and sufficient assumptions for derandomization:

- Chen and Tell [9], relying on the work by Goldreich [10], present a new uniform hardness assumption – roughly speaking, that there exists a multi-output function f computable by polynomial size logspace-uniform circuits with depth bounded by n^2 that cannot be computed in some (a-priori bounded) probabilistic polynomial time on *any* sufficiently large inputs – which implies that $\mathsf{prBPP} = \mathsf{prP}$. They also show a partial converse: That a relaxed version of this conjecture, where the depth requirement is dropped, also is necessary.
- A work by Hirahara [11] presents an equivalence between *hitting set generators* (HSG) [1, 2] with respect to some circuit class and the hardness of approximating a Kolmogorov complexity problem (in more detail, the Levin-Kolmogorov complexity problem) with the same circuit class. While his result is stated with respect to low-level complexity classes (such as $\mathsf{AC}^0 \circ \mathsf{XOR}$), his proof actually extends also to classes such as $\mathsf{P/poly}$. While HSG w.r.t., $\mathsf{P/poly}$ are known to imply that $\mathsf{prBPP} = \mathsf{prP}$, and they are a central tool toward establishing this in known results, they are not known to be implied by derandomization.

1.1 Characterizing prBPP = prP through the Hardness of Conditional Kt-complexity

In this work, we present a hardness assumption that is both *necessary* and *sufficient* for derandomizing prBPP. In more detail, we present an (in our eyes) natural class of promise problems such that prBPP = prP if and only if *every problem* in this class is (almost-everywhere) worst-case hard. The class of problems is related to Kolmogorov complexity.

Conditional Time-bounded Kolmogorov Complexity. What makes the string 121212 12121212121 less random than 604848506683403574 92? The notion of *Kolmogorov complexity* (K-complexity), introduced by Solomonoff [24], Kolmogorov [16] and Chaitin [7], provides a method for measuring the amount of "randomness" in individual strings: The K-complexity of a string is the length of the shortest program Π (to be run on some fixed universal Turing machine U) that outputs the string x. K-complexity, however, disregards the running time of the program Π. Levin-Kolmogorov Complexity [18] is an elegant way for incorporating the running time of the program Π into the complexity: $Kt(x)$ is defined as the minimal *cost* of a program Π that outputs x, where the cost of Π is defined as the sum of the length of Π and the *logarithm* of its running time.

We can also consider a conditional version of Kt-complexity: The *Conditional Levin-Kolmogorov Complexity* [30, 18, 27, 19] of a string x conditioned on a string z – denoted $Kt(x \mid z)$ – is the minimal "cost" of a program that, given the "auxiliary input" z (for free), outputs the string x.

We will here be interested in a *promise version* of the decisional conditional Kt-complexity problem, parametrized by thresholds T_{YES}, T_{NO} that specify on what Kt-complexities a decider needs to work. In more detail, the promise problem $\mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$ is defined as follows:

- YES instances: (x, z) such that $|x| = |z|$, and $Kt(x \mid z) \leq T_{\mathsf{YES}}(|x|)$.
- NO instances: (x, z) such that $|x| = |z|$, and $Kt(x \mid z) \geq T_{\mathsf{NO}}(|x|)$.

Note that this is a "gap" problem as we are not considering strings that have "intermediary" conditional Kt-complexity.

The Main Theorem. We are now ready to state our main theorem.

▶ **Theorem 1.** *There exists a constant c such that the following are equivalent:*

- $\mathsf{prBPP} = \mathsf{prP}$*;*
- *For all* $\mathsf{BPTIME}(n^c)$ *algorithm M, all sufficiently large $z \in \{0,1\}^n$, there exists some $x \in \{0,1\}^n$ such that M fails to decide whether $(x, z) \in \mathsf{GapMcKtP}[O(\log n), n-1]$.*

In other words, $\mathsf{prBPP} = \mathsf{prP}$ iff $\mathsf{GapMcKtP}[O(\log n), n-1]$ is hard to decide *for all* (sufficiently large) auxiliary inputs z (w.r.t., n^c time probabilistic algorithms).

Comparison with [11]. Let us start by comparing Theorem 1 with the results established by Hirahara [11]. As mentioned above, Hirahara characterizes hitting sets generators as opposed to derandomization, but the problem he considered is very related to the one we consider. More specifically, Hirahara consider the exact same promise problem, but without any conditioning/auxiliary inputs, and shows that hardness with respect to *circuits* (as opposed to uniform probabilistic algorithms as we do) implies HSGs that are hard with respect to the same class of circuits. He also used this result to characterize some non-trivial derandomization (i.e., $\mathsf{RTIME}[2^{\tilde{O}(\sqrt{n})}]$ into $\mathsf{DTIME}[2^{n-\tilde{\omega}(\sqrt{n})}]$ on feasibly generated inputs).

On a high level, Hirahara constructs a HSG by using the first $O(\log n)$ bits of the seed to select a program M, lets the output of the program determine a truthtable of a function f, and then relies on the Impagliazzo-Wigderon (IW) PRG [14] (applied to the second part of the seed) using f as the "hard function".[1] Hirahara shows that any attacker for such a HSG can be used to distinguish whether $Kt(x)$ is smaller than $O(\log n)$ or at least $n-1$.

Our construction of a targeted PRG relies on similar principles. Similarly to [11], we use the first $O(\log n)$ bits of the seed to select a program M, but instead of letting M operate not just on the empty input (as in [11]), we also let the program M access the target string z; in other words, we can think of this approach as using the target/instance to get a hard function, and next applying IW to this function. As we shall see, when doing this, we can show that any distinguisher for the targeted HSG that works given a target z can be used, together with the IW reconstruction procedure, to distinguish for *any* $x \in \{0,1\}^{|z|}$ whether $Kt(x \mid z)$ is small or large.

Comparison with [9]. It is also worthwhile to compare Theorem 1 with the result of Chen and Tell [9]. Most importantly, Theorem 1 present a full characterization of when $\mathsf{prBPP} = \mathsf{prP}$, whereas the result in [9] has a gap between the sufficient and necessary assumptions. Nevertheless, there are also similarities: The condition where we require hardness for *all* auxiliary inputs is closely related to the hardness condition in [9] which requires hardness for (almost) all inputs. Indeed, on a technical level, the reason these requirements arise are quite similar in both works, and our condition is inspired by [9]. On the other hand, our condition is also more complex than the one in [9] in that the input to our problem consists of two parts – the auxiliary input z and the instance x – and whereas we require hardness w.r.t. all sufficiently large z (just like [9]), we only require the algorithm to fail on *some* instance x (similarly to standard notions of worst-case hardness).

An alternative way of looking at our result is as presenting an *explicit* multi-output function F where the ith component of the output of F on input z is $Kt(i,z)$ such that almost-all-input hardness of $n - O(\log n)$ *approximating* F is equivalent to $\mathsf{prBPP} = \mathsf{prP}$. This condition differs from the one in [9] in that (1) we are considering an explicit function, rather that just any function, (2) the output length of the function is exponential (similar to [12]) whereas it is polynomial in [9], and (3) we require hardness of approximating the function, as opposed to computing it exactly.

Finally, we note that we actually prove an even stronger result: we do not actually require hardness of $\mathsf{GapMcKtP}$ w.r.t (almost) all auxiliary input strings to deduce that $\mathsf{prBPP} = \mathsf{prP}$. In fact, we show that for every γ, there exists a *universal* and *uniformly* computable sequence $\mathcal{Z}^\gamma = \{z_1, z_2, \ldots\}$ such that n^c-time hardness of $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ w.r.t. the specific sequence $\mathcal{Z}^\gamma$ implies that $\mathsf{prBPP} = \mathsf{prP}$.

We also mention that we can characterize quasi-polynomial time derandomization of prBPP using the same problems, by changing the YES-threshold to $\mathsf{poly}\log n$. For technical reasons, our approach does not extend to subexponential-time derandomization.[2]

---

[1] Hirahara presented his construction in a somewhat different way. In more detail, his construction of the HSG is simply a "universal HSG" obtained by interpreting the seed as a program M that is executed. He then uses the [14] construction in the analysis of the HSG. For our purposes, the above alternative presentation where incorporating the IW PRG directly into the construction will be helpful.

[2] More precisely, our characterization only works for complexity classes $\mathcal{C}$ of running times T such that if $T \in \mathcal{C}$ then $T(T(\cdot)) \in \mathcal{C}$ as well. This follows from our use of [23, 17, 6].

1.2 Proof Overview

To prove Theorem 1, we prove each direction of the equivalence separately.

Hardness of GapMcKtP from prBPP = prP. The first direction involves showing the hardness of GapMcKtP assuming that prBPP = prP. This direction follows mostly leveraging Goldreich's [10] earlier result showing the existence of a targeted PRG assuming prBPP = prP. We here consider a notion of a targeted PRG that is essentially identical to the notion of a "targeted canonical derandomizer" defined by Goldreich, but generalizes/strengthens his notion in several ways; most notably, we consider randomized distinguishers that may have superlinear running time, whereas Goldreich restricts attention to *deterministic* distinguishers with *linear* running time. Nevertheless, we observe that the PRG constructed by Goldreich (with minor modifications) actually satisfies the notion of a targeted PRG that we consider. Additionally, we observe that this (slightly new) notion of a targeted PRG suffices for demonstrating hardness of GapMcKtP for all sufficiently large auxiliary inputs z – roughly speaking, any solver for GapMcKtP can break the PRG as random strings (most often) are NO-instances, and strings in the range of the PRG are YES-instances; the target of the PRG will here correspond to the auxiliary input z used for the GapMcKtP problem.

prBPP = prP from the Hardness of GapMcKtP. To prove the second direction, we first observe that by the result of Buhrman and Fortnow [6] (building on [23, 17]), it suffices to show that prRP = prP to deduce that prBPP = prP. (We remark that for this result to hold, it is crucial that we are considering promise problems and not languages). Thus, it will suffice to derandomize prRP. Next, we consider the notion of a targeted HSG (discussed above) and demonstrate how to construct such a targeted HSG assuming that GapMcKtP is hard for all sufficiently large auxiliary inputs z. As mentioned above, the construction relies on ideas similar to those employed by Hirahara [11] except that, similarly to [9], we are using the target/instance z to obtain a "hard function" that we can plug into the IW generator.

Finally, to weaken the assumption to require hardness of GapMcKtP with respect to a specific universal and uniformly computable sequence of auxiliary inputs, we observe more generally that for any candidate construction of a HSG, there exists some universal sequence of targets such that security of the HSG w.r.t. this target sequence implies security w.r.t. all target sequences; and furthermore, this target sequence can be computed (in exponential time).

2 Preliminaries and Definitions

We assume familiarity with basic concepts such as Turing machines, polynomial-time algorithms, and probabilistic algorithms and computational classes such as prBPP, prRP, and prP. We let $\mathcal{U}_n$ the uniform distribution over $\{0,1\}^n$. Given a string $x \in \{0,1\}^n$ and an index $j \in [n]$, we let $[x]_j$ denote the length-j prefix of x.

Let $S \subseteq \{0,1\}^*$ be a set. We say that S is *decidable* if there exists a Turing machine M such that for all $x \in \{0,1\}^*$, $x \in S$ iff $M(x) = 1$. Let $\mathcal{Z} = \{z_n\}_{n\in\mathbb{N}}$ be a sequence. We say that $\mathcal{Z}$ is *uniform* if there exists a Turing machine M such that for all $n \in \mathbb{N}$, $z_n = M(1^n)$. We say that a function $f : \mathbb{N} \to \mathbb{N}$ is *time-constructible* if f is increasing and for all $n \in \mathbb{N}$, $f(n)$ can be computed by a Turing machine in time $\mathsf{poly}(f(n))$.

2.1 Levin's Conditional Kolmogorov Complexity

We recall the notion of Levin-Kolmogorov complexity. Roughly speaking, the *Levin's Kolmogorov complexity* [16, 23, 27, 15, 18], $Kt(x \mid z)$, of a string $x \in \{0,1\}^*$ conditioned on a "auxiliary input" string $z \in \{0,1\}^*$ is the cost of the most "efficient" program Π such that $\Pi(z)$ outputs x in t steps, where the efficiency of Π is defined to be the sum of the length of Π and the logarithm of t. We proceed to the formal definition. Let U be some fixed Universal Turing machine that can emulate any Turing machine Π with polynomial overhead. Let $U(\Pi(z), 1^t)$ denote the output of $\Pi(z)$ when emulated on U for t steps.

▶ **Definition 2.** *For all $x \in \{0,1\}^*$ and $z \in \{0,1\}^*$, define*

$$Kt(x \mid z) = \min_{\Pi \in \{0,1\}^*, t \in \mathbb{N}} \{|\Pi| + \lceil \log t \rceil : U(\Pi(z), 1^t) = x\}$$

We will here focus on a promise version of the decisional minimum conditional Levin-Kolmogorov complexity problem, parametrized by thresholds $T_{\mathsf{YES}}, T_{\mathsf{NO}}$.

▶ **Definition 3** (GapMcKtP)**.** *Let $T_{\mathsf{YES}}, T_{\mathsf{NO}}$ be two threshold functions. The promise problem* $\mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$ *is defined as follows.*

- *YES instances: (x, z) such that $|x| = |z|$, and $Kt(x \mid z) \leq T_{\mathsf{YES}}(|x|)$.*
- *NO instances: (x, z) such that $|x| = |z|$, and $Kt(x \mid z) \geq T_{\mathsf{NO}}(|x|)$.*

▶ **Definition 4** (Deciding GapMcKtP)**.** *We say that a probabilistic machine M* **fails to decides whether** $(x, z) \in \mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$ *if either $Kt(x \mid z) \leq T_{\mathsf{YES}}(|x|)$ but $\Pr[M(x, z) = 0] > 1/3$ or $Kt(x \mid z) \geq T_{\mathsf{NO}}(|x|)$ but $\Pr[M(x, z) = 1] > 1/3$.*

We will consider two notions of hardness of deciding GapMcKtP: either when the auxiliary input is fixed to some particular sequence (one for each input length), or hardness with respect to almost all auxiliary inputs.

▶ **Definition 5** (Hardness of GapMcKtP)**.** *We say that* $\mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$ *is:*

- **hard for probabilistic T-time algorithms given the auxiliary input sequence $\mathcal{Z} = \{z_1, z_2, \ldots\}$** *if for all probabilistic T-time algorithms M, all sufficiently large n, there exists a string $x \in \{0,1\}^n$ such that M fails to decide whether $(x, z_n) \in \mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$.*
- **hard for probabilistic T-time algorithms on almost all auxiliary inputs** *if for all sufficiently large z, there exists some $x \in \{0,1\}^{|z|}$ such that M fails to decide whether $(x, z) \in \mathsf{GapMcKtP}[T_{\mathsf{YES}}, T_{\mathsf{NO}}]$.*

2.2 Targeted Pseudorandom Generator

We consider the notion of a *targeted pseudorandom generator (targeted PRG)*, which is a generalization of the notion of a *targeted derandomizer* due to Goldreich [10]. Roughly speaking, a targeted pseudorandom generator g takes a seed along with a "target" string z as input, and we require that its output is indistinguishable from uniform by (computationally-bounded) distinguishers that additionally get the target z as input. In other words, g is just like a normal PRG, but with the exception that both the PRG and the distinguisher get access to the auxiliary "target" string z, and we require security to hold for all strings z. Since we consider PRGs in the context of derandomization, we allow the running-time of the PRG to be (polynomially) larger than the running-time of the distinguisher. We highlight that our notion slightly generalizes the notion of Goldreich by allowing the length of the target string to be different than the length of the output of the PRG, and additionally, we require the PRG to be defined over all output lengths. Furthermore, it strengthens Goldreich's notion by considering randomized distringuishers that may have superlinear running time (whereas Goldreich restricts attention to deterministic distinguishers with linear running time).

▶ **Definition 6** (Targeted pseudorandom generator (generalizing [10])). *Let $g : 1^n \times \{0,1\}^{\ell(n)} \times \{0,1\}^{m(n)} \to \{0,1\}^n$ be an efficiently computable function. We say that g is an $T(n)$-secure $(\ell(n), m(n))$-targeted pseudorandom generator (T-secure $(\ell(n), m(n))$-targeted PRG) if for all probabilistic attackers D that run in $T(n)$ time (where n is the length of its first input), for all sufficiently large $n \in \mathbb{N}$ and all strings $z \in \{0,1\}^{\ell(n)}$, it holds that*

$$|\Pr[s \leftarrow \{0,1\}^{m(n)} : D(1^n, z, g(1^n, z, s)) = 1] - \Pr[x \leftarrow \{0,1\}^n : D(1^n, z, x) = 1]| < \frac{1}{6}.$$

2.3 Targeted Hitting Set Generator

We turn to introducing the notion of *hitting set generator (HSG)* [1, 2] that we rely on. Recall that a standard hitting set generator requires its image set to have an overlap with any dense set that can be accepted by a small circuit. However, we here restrict our attention to uniform deterministic machines and we will consider the targeted variant of HSGs [10] (see also [9]).

▶ **Definition 7** (Targeted hitting set generator). *Let $g : 1^n \times \{0,1\}^{\ell(n)} \times \{0,1\}^{m(n)} \to \{0,1\}^n$ be an efficiently computable function. We say that g is an $T(n)$-secure $(\ell(n), m(n))$-targeted hitting set generator (T-secure (ℓ, m)-targeted HSG) secure w.r.t. deterministic attackers if for all deterministic attackers D that run in $T(n)$ time (where n is the length of its first input), for all sufficiently large $n \in \mathbb{N}$ and all strings $z \in \{0,1\}^{\ell(n)}$, it holds that if*

$$\Pr[x \leftarrow \{0,1\}^n : D(1^n, z, x) = 1] \geq \frac{1}{6}$$

then

$$\Pr[s \leftarrow \{0,1\}^{m(n)} : D(1^n, z, g(1^n, z, s)) = 1] > 0$$

For any targeted HSG g, we say that g is $O(T(n))$-secure if for all constant $c > 0$, g is $(cT(n))$-secure.

In addition, we will also consider a weaker notion of the targeted hitting set generator, where security is only guaranteed given some particular sequence of target inputs (rather than for all target inputs). Formally, let $\ell(n)$ be a function and let $\mathcal{Z} = \{z_n\}_{n\in\mathbb{N}}$ such that $|z_n| = \ell(n)$. Let $g : 1^n \times \{0,1\}^{\ell(n)} \times \{0,1\}^{m(n)} \to \{0,1\}^n$ be an efficiently computable function. We say that g is a *$T(n)$-secure $(\mathcal{Z}, m(n))$-targeted hitting set generator (T-secure $(\mathcal{Z}, m)$-targeted HSG)* if its security requirement holds for all sufficiently large $n \in \mathbb{N}$ and $z = z_n$.

It is well-known that a (non-uniformly) secure HSG can derandomize prRP. We next show that when considering a targeted uniformly-secure HSG, the same derandomization result still holds. This in essence follow by the standard proof (that non-uniformly secure HSG derandomize RP), but with an additional padding argument to deal with the "target"/auxiliary input.

▶ **Lemma 8.** *Assume that there exist constants $c \geq 1, \theta \geq 1$ and a $O(n^\theta)$-secure $(n^\theta, c \log n)$-targeted HSG $g : 1^n \times \{0,1\}^{n^\theta} \times \{0,1\}^{c \log n} \to \{0,1\}^n$ secure w.r.t. deterministic attackers. Then,* prRP = prP.

Proof. To show that prRP = prP, it suffices to prove that for any polynomial-time randomized algorithm A, there exists a polynomial-time deterministic algorithm B such that for all sufficiently long $x \in \{0,1\}^*$, if $\Pr_r[A(x;r) = 1] \geq \frac{1}{2}$, then $B(x) = 1$; and if $\Pr_r[A(x;r) = 0] = 1$, then $B(x) = 0$ (where r denotes the random coins that A uses).

Consider any poly-time randomized algorithm A. We can without loss of generality assume that A runs in linear time and A uses as many random coins as its input length.[3] If $\theta > 1$, the following padding argument is needed. For any string $x \in \{0,1\}^*$, let x' be the string $x10^{|x|^\theta - |x| - 1}$; that is, we pad as many '0' at the end of $x||1$ until it becomes of length $|x|^\theta$. Let A' be an algorithm such that $A'(x';r) = A(x;r)$ for any x, r.[4] (If $\theta = 1$, we let $x' = x$ and $A' = A$.)

We proceed to constructing a poly-time deterministic algorithm B that deterministically emulates A. On input an instance $x \in \{0,1\}^n$, $B(x)$ tries all possible seeds $v \in \{0,1\}^{c \log n}$ and $B(x)$ outputs 1 if and only if there exists a seed v such that $A'(x', g(1^n, x', v)) = 1$; otherwise $B(x)$ outputs 0.

Observe that if $A(x,r)$ outputs 0 with probability 1 (over the random choice of r), $A'(x',r)$ will output 0 with probability 1 and thus $B(x)$ will also output 0. Also note that B runs in polynomial time.

We show that, for all sufficiently long x, $B(x)$ will output 1 if $A(x,r)$ outputs 1 with probability $\geq \frac{1}{2}$ Since g is a $O(n^\theta)$-secure $(n^\theta, c \log n)$-targeted HSG and A' runs in deterministic $O(n^\theta)$ time with respect to $n = |r| = |x|$, it follows that for all sufficiently large $n \in \mathbb{N}$, $x' \in \{0,1\}^{n^\theta}$, it holds that if

$$\Pr[r \leftarrow \{0,1\}^n : A'(x', r) = 1] \geq \frac{1}{6} \tag{1}$$

then

$$\Pr[v \leftarrow \{0,1\}^{c \log n} : A'(x'; g(1^m, x', v)) = 1] > 0. \tag{2}$$

Consider some string x such that g is secure on auxiliary input x' (with respect to A') and it holds that $\Pr_r[A(x;r) = 1] \geq \frac{1}{2}$. It follows that $\Pr_r[A'(x';r) = 1] \geq \frac{1}{2}$ which implies that Equation 1 holds. Therefore, by the hitting property of g, Equation 2 also holds and there exists a seed $v \in \{0,1\}^{c \log n}$ such that $A'(x', g(1^n, x', v)) = 1$. Thus, $B(x)$ will output 1. Finally note that g will be secure on all sufficiently long x', so B can always find a seed v such that $A'(x', g(1^n, x', v)) = 1$ if $\Pr_r[A(x,r) = 1] \geq \frac{1}{2}$ for all sufficiently long x. ◀

3 Universal Target Strings for Targeted HSG or PRG

In this section, we show a useful statement about targeted PRGs/HSGs: For every candidate PRG/HSG, there exists a *universal* sequence of targets (one for each input lenght) such that if the PRG/HSG is secure with respect to this sequence, then it will be secure with respect to any target. Furthermore, this universal sequence is computable (in exponential time). Looking ahead, this will later be useful to us to show that hardness of GapMcKtPwith respect to a particular, computable, sequence of auxiliary inputs, suffices to characterize derandomization.

▶ **Lemma 9.** *Let $g : 1^n \times \{0,1\}^{\ell(n)} \times \{0,1\}^{m(n)} \to \{0,1\}^n$ be an efficiently computable function such that ℓ, m are polynomially bounded, and let $T(n) \leq 2^n$ be a function. There exists an exponential time uniform sequence $\mathcal{Z} = \{z_n \in \{0,1\}^{\ell(n)}\}_{n \in \mathbb{N}}$ such that if g is a T-secure $(\mathcal{Z}, m(n))$-target HSG (resp PRG) secure on $\mathcal{Z}$, then g is a T-secure $(\ell(n), m(n))$-target HSG (resp PRG) secure on all target inputs.*

[3] If A does not run in linear time (or uses more random coins than its input length), we can pad the input of A so that the padded version (of A) now runs in linear time (or uses equally many random coins).

[4] More formally, A' is an algorithm proceeding as the following. A' takes input (x', r) and then removes as many '0' in the end of x' as it can. A' further remove a single bit '1' and denote the result by x. A' returns $A(x, r)$.

Proof. Let g, T be as in the lemma statement. We consider the Turing machine M that proceeds as follows. On input 1^n, $M(1^n)$ enumerates all TMs D of (description) length $\leq \log n$ in lexicographic order. $M(1^n)$ verifies whether the following two conditions are satisfied.

- $D(y_1, y_2, y_3)$ terminates within $T(|y_1|)$ steps for all strings y_1, y_2, y_3 satisfying $|y_1| \leq n, |y_2| = \ell(|y_1|), |y_3| = |y_1|$.
- There exists a string $x \in \{0,1\}^{\ell(n)}$ such that D breaks g on the target input x. Specifically, if g is a HSG candidate, it requires that

$$\Pr[s \leftarrow \{0,1\}^{m(n)} : D(1^n, x, g(1^n, x, s)) = 0] = 1$$

and

$$\Pr[r \leftarrow \{0,1\}^n : D(1^n, x, r) = 1] > \frac{1}{6}.$$

Let D' be the first machine M finds such that the above two checks are passed. Let x' be the lexicographically smallest string such that the second condition above is satisfied (with respect to D'). $M(1^n)$ will simply output x'. Finally, let $\mathcal{Z} = \{z_n\}$ be a sequence such that for all $n \in \mathbb{N}$, $z_n = M(1^n)$.

We next argue that if g is a T-secure $(\mathcal{Z}, m(n))$-target HSG (resp PRG) secure on $\mathcal{Z}$, then g is a T-secure $(\ell(n), m(n))$-target HSG (resp PRG) secure on all target inputs, which concludes our proof (since M also runs in exponential time). Assume for contradiction that there exists a T-time distinguisher D such that D breaks g on infinitely many target inputs. Let D^* be such a distinguisher with the lexicographically smallest description. Consider all TMs D' that are lexicographically smaller than D^*, and the following two observations will show that $M(1^n)$ will never accept any TM $D' <_{\mathsf{lex}} D^*$ when n is sufficiently large.

- If D' is not a T-time machine. Then there exists an input of the form y_1, y_2, y_3 to D' such that $D'(y_1, y_2, y_3)$ runs more than $T(|y_1|)$ steps before it halts. $M(1^n)$ will not accept D' if $n > |y_1|$.
- If D' is a T-time machine, since $D' <_{\mathsf{lex}} D^*$, D' will only break g on finitely many target inputs (if any). $M(1^n)$ will not accept D' if n is sufficient large (so that D' never breaks g on target inputs of length larger than n).

On the other hand, note that $M(1^n)$ will accept D^* if D^* breaks g on some target input of length n (which happens infinitely often). Whenever $M(1^n)$ accepts D^*, D^* will break g on target z_n since $z_n = M(1^n)$. Therefore, we conclude that D^* breaks the security of g on infinitely many z's $\in \mathcal{Z}$. ◀

4 Main Theorem

We are now ready to formally state out main theorem.

▶ **Theorem 10.** *There exist a constant $c \geq 1$ and a Turing machine M such that the following are equivalent.*

1. $\mathsf{prBPP} = \mathsf{prP}$.
2. *The existence of a constant γ_0 such that for all $\gamma \geq \gamma_0$, $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is hard for probabilistic n^c-time algorithms on almost all auxiliary inputs.*
3. *The existence of a constant γ_0 such that for all $\gamma \geq \gamma_0$, all uniform auxiliary sequence $\mathcal{Z}$, it holds that $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is hard for probabilistic n^c-time algorithms given the sequence $\mathcal{Z}$.*

4. *The existence of a constant* γ *such that* GapMcKtP$[\gamma \log n, n-1]$ *is hard for probabilistic* n^c*-time algorithms given the auxiliary input sequence* $\mathcal{Z} = \{z_1, z_2, \ldots\}$ *where* $z_i = M(\gamma, 1^i)$.
5. *The existence of constants* $\sigma \geq 1, \theta \geq 1$ *and an* $O(n^\theta)$*-secure* $(n^\theta, \sigma \log n)$*-targeted HSG.*
6. prRP = prP

Proof. The proof of this theorem relies on many results which will be stated and proved later. Let c be the constant as in Lemma 21. Although it is stated in Lemma 21 that for each constant γ, there exists a uniform auxiliary input sequence $\mathcal{Z}_0 = \{z_{0,n}\}_{n\in\mathbb{N}}$ (such that some desired propertise are satisfied), its proof actually proves a stronger statement: There exists a machine M such that M with γ as input will generate the auxiliary sequence $\mathcal{Z}_0$ in the sense that $z_{0,n} = M(\gamma, 1^n)$ for all $n \in \mathbb{N}$. Given the existence of such c and M, our proof proceeds as the following.

(1) ⇒ (2): it follows from Theorem 11.
(2) ⇒ (3) and (3) ⇒ (4): These two implications trivially hold.
(4) ⇒ (5): This implication follows from Lemma 21 due to our choice of M and the way we pick $\mathcal{Z}$.
(5) ⇒ (6): The proof of this implication relies on the fact that a targeted HSG allows us to emulate prRP computation in deterministic time, and a detailed proof can be found in Lemma 8.
(6) ⇒ (1): This implication can be proved using standard reductions in [23, 17] (and see also [6]). ◀

5 GapMcKtP Hardness from prBPP = prP

In this section, we show that the assumption prBPP = prP will imply the hardness of GapMcKtP with the desired parameters.

▶ **Theorem 11.** *Assume that* prBPP = prP*. Then for all constants* $c \geq 1$*, there exists a constant* $\gamma_0 > 0$ *such that for all constants* $\gamma \geq \gamma_0$*, it holds that* GapMcKtP$[\gamma \log n, n-1]$ *is hard for probabilistic* n^c*-time algorithms on almost all auxiliary inputs.*

Recall that Goldreich [10] showed that if prBPP = prP, then there exists a so-called "targeted derandomizer". Since our notion of a targeted PRG is very similar to his notion, his proof extends with just minor modifications of the parameters also to our notion.

▶ **Theorem 12** (essentially implicit in [10])**.** *Assume that* prBPP = prP*. Then for all constants* $\gamma > 0, c \geq 1$*, there exists a* n^c*-secure* $(n, \gamma \log n)$*-targeted PRG* g*.*

We defer the proof of Theorem 12 (which closely follows [10]) to the full version. [10] further shows that a targeted PRG can be used to derandomize prBPP (and thus shows equivalence of derandomization of prBPP and targeted PRGs). We here instead show that the existence of targeted PRGs implies hardness of GapMcKtP. Roughly speaking, this follows from the observation that any solver for GapMcKtP can break the PRG as random strings (most often) are NO-instances, and strings in the range of the PRG are YES-instances; the target of the PRG will here correspond to the auxiliary input z used for the GapMcKtP problem.

▶ **Lemma 13.** *Assume that for all constants* $\gamma > 0, c \geq 1$*, there exists a* n^c*-secure* $(n, \gamma \log n)$*-targeted PRG. Then, for all constants* $c \geq 1$*, there exists a constant* $\gamma_0 > 0$ *such that for all constants* $\gamma \geq \gamma_0$*,* GapMcKtP$[\gamma \log n, n-1]$ *is hard for probabilistic* n^c*-time algorithms on almost all auxiliary inputs.*

Proof. Consider any constant $c \geq 1$. By our assumption, it follows that there exists a n^c-secure $(n, \log n)$-targeted PRG $g : 1^n \times \{0,1\}^n \times \{0,1\}^{\log n} \to \{0,1\}^n$. Let γ_0 be a constant such that computing the PRG $g(1^n, x, v)$, $x \in \{0,1\}^n, v \in \{0,1\}^{\log n}$ can be done in time $n^{\gamma_0 - 2}$. Let γ be any constant such that $\gamma \geq \gamma_0$. Assume for contradiction that $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is easy for probabilistic n^c-time algorithms on almost all auxiliary inputs. Then, there exist a n^c-time probabilistic machine M such that for infinitely many $n \in \mathbb{N}$, there exists $z_n \in \{0,1\}^n$ such that for all $x \in \{0,1\}^n$, $\Pr[M(1^n, x, z_n) = 1] \geq 0.9$ if $Kt(x \mid z_n) \leq \gamma \log |x|$ and $\Pr[M(1^n, x, z_n) = 1] \leq 0.1$ if $Kt(x \mid z_n) \geq |x| - 1$. We will show that $M(1^n, z_n, \cdot)$ distinguishes between $g(1^n, z_n, \mathcal{U}_{\log n})$ and $\mathcal{U}_n$ on all z_n on which M succeeds, which contradicts the security of g. Towards this, let us fix some $n \in \mathbb{N}$, $z = z_n$ on which M succeeds.

We first prove that on input $(1^n, z, g(1^n, z, v))$ where $v \in \{0,1\}^{\log n}$, $M(1^n, z, g(1^n, z, v))$ will output 1 with probability ≥ 0.9. Observe that

$$Kt(g(1^n, z, v) \mid z) \leq \log n + \log(n^{\gamma_0 - 2}) + O(1) \leq \gamma \log n$$

when n is sufficiently large since $g(1^n, z, v)$ can be computed by hardwiring the seed v (of length $\log n$) and the code of g (of length $O(1)$) in time $n^{\gamma_0 - 2}$ when having access to the string z. Therefore, $\Pr[M(1^n, z, g(1^n, z, v)) = 1] \geq 0.9$ for every $v \in \{0,1\}^{\log n}$.

We next show that on input $(1^n, z, \mathcal{U}_n)$, M will output 1 with probability ≤ 0.6. Observe that there are at most 2^{n-1} strings x of length n that have conditional Kt-complexity $\leq n-2$ since there are at most 2^{n-1} machines of description length $\leq n-2$. It follows that $\Pr[Kt(\mathcal{U}_n \mid z) \geq n-1] \geq 1 - \frac{2^{n-1}}{2^n} \geq \frac{1}{2}$. Conditioned on this event, we know that the probability that M outputs 1 is at most 0.1. Thus, by a union bound, $\Pr[M(1^n, z, \mathcal{U}_n) = 1] \leq \frac{1}{2} + \frac{1}{2} \times 0.1 \leq 0.6$. ◀

We can now conclude the proof of Theorem 11.

of Theorem 11. Theorem 11 follows directly from Theorem 12 and Lemma 13. ◀

6 Derandomization from Hardness of GapMcKtP

We proceed to proving that hardness of GapMcKtP implies that prBPP = prP. Note that by standard techniques in [23, 17], it suffices to derive prRP = prP. Towards this, we will present how to construct a targeted HSG from the assumption, which is known to enable us to derandomize prRP (see also Lemma 8).

6.1 Targeted HSG from Hardness of GapMcKtP

We here show how to obtain an targeted HSG assuming hardness of GapMcKtP. The following result is the crux of our proof.

▶ **Lemma 14.** *There exists a constant $c \geq 1$ such that the following holds. For each constant $\gamma > 0$, there exist constants $\sigma \geq 1, \theta \geq 1$, and an efficiently computable function $g : 1^m \times \{0,1\}^{m^\theta} \times \{0,1\}^{\sigma \log m} \to \{0,1\}^m$ such that for any target input sequence $\mathcal{Z}_1 = \{z_{1,m}\}_{m \in \mathbb{N}}$, if g is not an $O(m^\theta)$-secure $(\mathcal{Z}_1, \sigma \log m)$-targeted HSG, then $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is not hard for probabilistic n^c-time algorithms given an auxiliary input sequence $\mathcal{Z}_0 = \{z_{0,n}\}_{n \in \mathbb{N}}$ where for all $n \in \mathbb{N}$, $z_{0,n} = z_{1,m(n)}$ and $m(n) = \lfloor n^{\frac{1}{\theta}} \rfloor$.*

Tool 1: List-decodable ECCs. We start by recalling the notion of a list-decodable error correcting code that we will be relying on.

▶ **Definition 15** (List-decodable error correcting code (see e.g. [28])). *For any $n, n', L \in \mathbb{N}$ and $\delta > 0$, a function $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{n'}$ is said to be a $(L, \frac{1}{2} - \delta)$*-list-decodable error correcting code *if there exists a function $\mathsf{Dec} : \{0,1\}^{n'} \to (\{0,1\}^n)^L$ such that for any $x \in \{0,1\}^n$ and $x' \in \{0,1\}^{n'}$ satisfying $\Pr_{i \in [n']}[\mathsf{Enc}(x)_i \neq x'_i] \leq \frac{1}{2} - \delta$ it holds that $x \in \mathsf{Dec}(x')$. We refer to Dec as a decoder of Enc.*

The following construction of a list-decodable error correcting code will be useful for us.

▶ **Theorem 16** ([25]; see also [28, Problem 5.2]). *There exist two deterministic polynomial time algorithms $\mathsf{Enc}, \mathsf{Dec}$ such that for all $n \in \mathbb{N}$, $\delta > 0$, the function $\mathsf{Enc}_{n,\delta} : \{0,1\}^n \to \{0,1\}^{2^r}$ where $r = O(\log(n/\delta))$ is a $(\mathsf{poly}(1/\delta), \frac{1}{2} - \delta)$-list-decodable error correcting code with $\mathsf{Dec}_{n,\delta}$ being its decoder, and both $\mathsf{Enc}_{n,\delta}$ and $\mathsf{Dec}_{n,\delta}$ run in time $\mathsf{poly}(n, 1/\delta)$.*

Tool 2: The NW PRG. We turn to recalling the construction of the Nisan-Wigderson (NW) PRG [22]. For any string $y \in \{0,1\}^d$ and subset $I \subseteq [d]$, we let y_I denote the $|I|$-bit string consisting of the the projection of y to the coordinates $\in I$.

▶ **Definition 17** (NW generator). *Let $\mathcal{I} = (I_1, \ldots, I_m)$ be a family of m subsets of $[d]$ with each $|I_j| = r$ and let $f : \{0,1\}^r \to \{0,1\}$ be a function. The $(\mathcal{I}, f)$-NW generator is the function $\mathsf{NW}^f_{\mathcal{I}} : \{0,1\}^d \to \{0,1\}^m$ that takes any string $y \in \{0,1\}^d$ as a seed and outputs*

$$\mathsf{NW}^f_{\mathcal{I}}(y) = f(y_{I_1}) \ldots f(y_{I_m})$$

The core ingredient of the Nisan-Wigderson construction is a combinatorial design which will be used as the family of subsets in a NW generator.

▶ **Definition 18** (Combinatorial designs). *For any integers $d, r, s \in \mathbb{N}$ such that $d > r > s$, a family $\mathcal{I} = \{I_1, \ldots, I_m\}$ of subsets of $[d]$ is said to be a (d, r, s)-*design *if for every $j \in [m]$, $|I_j| = r$, and for every $k \in [m], k \neq j$, $|I_j \cap I_k| \leq s$.*

Recall that combinatorial designs can be efficiently constructed.

▶ **Lemma 19** ([22]; see also [3, Lemma 16.18]). *There exists a deterministic algorithm $\mathsf{GenDesign}$ such that on input $d, r, s \in \mathbb{N}$ where $r > s$ and $d > 10r^2/s$, runs in $\mathsf{poly}(2^d)$ steps and outputs a (d, r, s)-design $\mathcal{I}$ containing $2^{s/10}$ subsets of $[d]$.*

The following version of the reconstruction theorem will be useful for us.

▶ **Lemma 20** (Implicit in [22, 14]). *There exists a PPT algorithm $\mathsf{NWRecon}$ such that the following conditions hold.*

- *Input: the truthtable of a function $f : \{0,1\}^r \to \{0,1\}$, a (d, r, s)-design $\mathcal{I} = \{I_1, \ldots, I_m\}$.*
- *Given oracle access to an oracle $D \subseteq \{0,1\}^m$ such that*

$$\left| \Pr[y \leftarrow \{0,1\}^d : D(\mathsf{NW}^f_{\mathcal{I}}(y)) = 1] - \Pr[w \leftarrow \{0,1\}^m : D(w) = 1] \right| \geq \frac{1}{6}. \tag{3}$$

- *Output: a (deterministic) program M of description length $\leq m \cdot 2^s + m + d + O(\log drsm)$ such that $M^D(\mathcal{I})$ will output a string $x' \in \{0,1\}^{2^r}$ in $\mathsf{poly}(2^r)$ steps and x' satisfies that*

$$\Pr[p \leftarrow [2^r] : x'_p \neq f(p)] \leq \frac{1}{2} - \frac{1}{12m}.$$

For the sake of completeness, we refer the read to the full version for a proof of Lemma 20.

Returning to the proof of Lemma 14. We are finally ready to prove Lemma 14 by relying on the above two tools/results.

Proof of Lemma 14. Before presenting a formal proof, it may be helpful to first discuss the choice of parameters in our construction.

Notations. Let m denote the output length of the targeted HSG g that we hope to construct. Let n denote the length of GapMcKtP instances and let $\theta \in \mathbb{N}$ be a constant such that $\frac{1}{\theta}$ is sufficiently small. In this proof, we usually assume that $n = \mathsf{poly}(m)$ and it holds that $n = m^\theta$. In some cases depending on the context, m is defined w.r.t. n and it holds that $m = \lfloor n^{\frac{1}{\theta}} \rfloor$ (and we can think of m as being sublinear in n).

Let c be a sufficiently large constant (which will be fixed later). Consider any constant $\gamma > 0$ and a YES-threshold of GapMcKtP $T_{\mathsf{YES}} = \gamma \log n$.

Constructing the HSG. Our HSG will take as input a unary string 1^m, a target string z of length $m^\theta = n$, along with a seed. Let $\delta = O(\frac{1}{m})$ and we will need a list-decodable ECC that corrects a $\frac{1}{2} - \delta$ fraction of errors. By Theorem 16, there exists a function $L = \mathsf{poly}(1/\delta)$, a function $r = O(\log n)$, a $(L, \frac{1}{2} - \delta)$-list-decodable ECC $\mathsf{Enc}_{n,\delta}$ that produces codewords of length 2^r, and a decoding algorithm $\mathsf{Dec}_{n,\delta}$ that outputs a candidate message set of size L (if Dec succeeds). We will also need a NW generator that takes functions with truthtable length 2^r (matching the output length of the ECC) and outputs m bits (matching the output length of our HSG). To achieve this, we require a (d, r, s)-design $\mathcal{I}$ that contains m subsets of $[d]$. By Lemma 19, we can pick $s = O(\log m)$ to ensure that $\mathcal{I}$ contains m subsets, and pick $d = \Omega(\log n)$ to satisfy that $d > 10r^2/s$. (Such designs can be efficiently generated by GenDesign.) For our HSG to be secure, it is crucial in the NW security proof that 2^s is small enough, say, $< \sqrt{n}$ (which will also guarantee that $s < r$). This can be achieved by picking θ to be sufficiently large. (For a concrete choice of parameters, consider picking $s = 10 \log m$ and $\theta = 20$.)

We turn to describing our HSG formally. We will consider a function $g : 1^m \times \{0,1\}^n \times \{0,1\}^{\log n + T_{\mathsf{YES}} + d} \to \{0,1\}^m$ defined as follows. On input $(1^m, z, (j, \Pi', y))$ where $z \in \{0,1\}^n, j \in \{0,1\}^{\log n}, \Pi' \in \{0,1\}^{T_{\mathsf{YES}}}, y \in \{0,1\}^d$. Let k' be an integer that $k' = j$ when k' is represented in a binary string and let $k = \min\{k', T_{\mathsf{YES}}\}$. Let $\Pi = [\Pi']_k$ be the length-k prefix of Π'. Let $t = 2^{T_{\mathsf{YES}}}$. The algorithm g proceeds in the following steps.

1. g first interprets the string $\Pi \in \{0,1\}^k$ as a program and computes the output $x_\Pi = U(\Pi(z), 1^t)$ of $\Pi(z)$ after t steps.
2. Then g lets $f : \{0,1\}^r \to \{0,1\}$ be the function $f = \mathsf{fn}(\mathsf{Enc}_{n,\delta}(x_\Pi))$ that is associated with the truthtable $\mathsf{Enc}_{n,\delta}(x_\Pi) \in \{0,1\}^{2^r}$. ($g$ simply aborts if $|x_\Pi| \neq n$.)
3. Next, g invokes the design generating algorithm $\mathsf{GenDesign}(d, r, s)$ to obtain a (d, r, s)-design $\mathcal{I} = \{I_1, \ldots, I_m\}$.
4. Finally, g outputs

$$g(1^m, z, (j, \Pi', y)) = \mathsf{NW}^f_{\mathcal{I}}(y) = f(y_{I_1}) \ldots f(y_{I_m})$$

where the function NW is defined in Definition 17.

(Note that the seed length of g is $\log n + T_{\mathsf{YES}} + d = O(\log n) = O(\log m)$. And we can let σ be the constant such that the seed length of g is $\sigma \log m$. Notice that g is a function of the form $1^m \times \{0,1\}^{m^\theta} \times \{0,1\}^{\sigma \log m} \to \{0,1\}^m$.)

Deciding GapMcKtP. Suppose that g is not an $O(m^\theta)$-secure $(\mathcal{Z}_1, \sigma \log m)$-targeted HSG w.r.t. deterministic algorithms and some target string sequence $\mathcal{Z}_1 = \{z_{1,m} \in \{0,1\}^{m^\theta}\}_{m\in\mathbb{N}}$; then, there exists a $O(m^\theta)$-time deterministic distinguisher D such that for infinitely many $m \in \mathbb{N}$,

$$\Pr[v \leftarrow \{0,1\}^{\sigma \log m} : D(1^m, z_{1,m}, g(1^m, z_{1,m}, v)) = 0] = 1 \tag{4}$$

and

$$\Pr[w \leftarrow \{0,1\}^m : D(1^m, z_{1,m}, w) = 0] < 1 - \frac{1}{6} \tag{5}$$

We will prove that there exists a probabilistic n^c-time algorithm that decides $\mathsf{GapMcKtP}[T_{\mathsf{YES}}, n-1]$ infinitely often given the auxiliary input sequence $\mathcal{Z}_0$, where $\mathcal{Z}_0$ is a sequence of auxiliary input strings such that $z_{0,n} = z_{1,m}$ for all $n \in \mathbb{N}$. (We can think of Z_0 as being a padded version of $\mathcal{Z}_1$ to ensure that $z_{0,n} = z_{1,m}$.)

We will construct an algorithm A that runs in a-priori bounded polynomial time such that for any sufficiently large $m \in \mathbb{N}$, $n = m^\theta$, $z = z_{1,m} = z_{0,n}$, if Equation 4 and Equation 5 hold w.r.t. m, then for any $x \in \{0,1\}^n$, the following is true. If $Kt(x \mid z) \leq T_{\mathsf{YES}}$, then $A(x,z)$ outputs a program Π such that $|\Pi| \leq |x|^{2/3}$ and $\Pi(z)$ produces x within (a-priori bounded) poly time with high probability. Let us fix c to be some sufficiently large constant such that the running time of A (together with the time needed to check whether the output of A is correct) is bounded by n^c. Note that the existence of algorithm A will imply $\mathsf{GapMcKtP}[T_{\mathsf{YES}}, n-1]$ can be decided by a n^c-time algorithm on auxiliary input sequence $\mathcal{Z}_0$ since it suffices to first run $A(x,z)$ and check whether the program Π output by A indeed produces x on input z within some fixed polynomially amount of time. If $Kt(x \mid z) \geq |x| - 1$, it follows that there exists no such machine Π and A will never find it.

We proceed to describing the algorithm A. On input strings $x, z \in \{0,1\}^n$ (and let $m = \lfloor n^{\frac{1}{\theta}} \rfloor$), the algorithm A acts as the follows.

1. $A(x,z)$ lets $f : \{0,1\}^r \to \{0,1\}$ be the function $f = \mathsf{fn}(\mathsf{Enc}_{n,\delta}(x))$ that is associated with the truthtable $\mathsf{Enc}_{n,\delta}(x) \in \{0,1\}^{2^r}$.
2. $A(x,z)$ runs the design generating algorithm $\mathsf{GenDesign}(d,r,s)$ to obtain a (d,r,s)-design $\mathcal{I} = \{I_1, \ldots, I_m\}$.
3. $A(x,z)$ executes the NW reconstruction algorithm $\mathsf{NWRecon}^{D(1^m,z,\cdot)}(f, \mathcal{I})$ and let M denotes the program it outputs.
4. $A(x,z)$ evaluates $M^{D(1^m,z,\cdot)}(\mathcal{I})$ and denotes the output string by x'.
5. $A(x,z)$ computes a list $\vec{x}$ of size L by letting $\vec{x} = \mathsf{Dec}_{n,\delta}(x')$ (which is a list of candidate strings for x output by the list decoding algorithm), and let pos denote a coordinate of $\vec{x}$ such that $\vec{x}_{\mathsf{pos}} = x$. (If x does not appear in $\vec{x}$, $A(x,z)$ simply aborts.)
6. Finally, A outputs a program Π with the values $n, m, d, r, s, \delta^{-1}, \mathsf{pos}$, the code of M, D, Dec, $\mathsf{GenDesign}$ hardwired in it. In addition, on input z, $\Pi(z)$ proceeds in the following steps.
   a. $\Pi(z)$ first invokes $\mathsf{GenDesign}(d,r,s)$ to get a (d,r,s)-design $\mathcal{I} = \{I_1, \ldots, I_m\}$.
   b. $\Pi(z)$ computes $x' = M^{D(1^m,z,\cdot)}(\mathcal{I})$.
   c. $\Pi(z)$ runs the list decoding algorithm $\mathsf{Dec}_{n,\delta}(x')$ and obtains a list $\vec{x}$.
   d. $\Pi(z)$ outputs the string $\vec{x}_{\mathsf{pos}}$ and halts.

Analyzing the reduction. We turn to proving that the program Π will indeed output x on input z within polynomial time if n is of the form $n = m^\theta$ for some m such that Equation 4 and Equation 5 hold w.r.t. m, $z = z_{1,m}$, and $Kt(x \mid z) \leq T_{\mathsf{YES}}$. Fix some such x, z that are sufficiently long. We first show that program Π will indeed output x on input z. Since

$Kt(x \mid z) \leq T_{\mathsf{YES}}$, there exists a machine Π_x with $|\Pi_x| \leq T_{\mathsf{YES}}$ such that $\Pi_x(z)$ will output the string x within $2^{T_{\mathsf{YES}}} = t$ steps. Let $j = |\Pi_x|$ (in its binary representation) and let Π'_x be a string $\in \{0,1\}^{T_{\mathsf{YES}}}$ such that $[\Pi'_x]_j = \Pi_x$. Observe that for all $y \in \{0,1\}^d$, the string $\mathsf{NW}^f_{\mathcal{I}}(y)$ equals $g(1^m, z, (j, \Pi'_x, y))$ and thus $\mathsf{NW}^f_{\mathcal{I}}(y)$ is in the range of the HSG g. Note that $D(1^m, z, \cdot)$ is a HSG distinguisher and will always output 0 in the range of $g(1^m, z, \cdot)$. By Equation 4, it follows that

$$\Pr[y \leftarrow \{0,1\}^d : D(1^m, z, \mathsf{NW}^f_{\mathcal{I}}(y)) = 0] = 1$$

Combining this with Equation 5, it holds that $D(1^m, z, \cdot)$ distinguishes the output of $\mathsf{NW}^f_{\mathcal{I}}$ from uniform with advantage $\frac{1}{6}$ and thus it breaks the NW generator $\mathsf{NW}^f_{\mathcal{I}}$. Then by Lemma 20, the NW reconstruction algorithm $\mathsf{NWRecon}^{D(1^m,z,\cdot)}(f, \mathcal{I})$ will output a good approximation for f; that is, it holds that

$$\Pr[p \leftarrow [2^r] : \mathsf{Enc}_{n,\delta}(x)_p \neq x'_p] = \Pr[p \leftarrow [2^r] : f(p) \neq x'_p] \leq \frac{1}{2} - \frac{1}{12m} \leq \frac{1}{2} - \delta$$

So x' is relatively close to $\mathsf{Enc}_{n,\delta}(x)$. Since Enc is a good error correcting code, by Theorem 16, $\mathsf{Dec}_{n,\delta}(x')$ will return a list contain x; i.e., $x \in \vec{x} = \mathsf{Dec}_{n,\delta}(x')$. $A(x,z)$ will then find the coordinate pos such that $\vec{x}_{\mathsf{pos}} = x$. Note that $\Pi(z)$ will finally output $\vec{x}_{\mathsf{pos}}$ and we conclude that $\Pi(z)$ will indeed produce x.

We next argue that Π has a short description. Π spends $O(\log n)$ bits to store the values $n, m, d, r, s, \delta^{-1}$, the code of $D, \mathsf{Dec}, \mathsf{GenDesign}$. In addition, Π uses

$$m \cdot 2^s + m + d + \log(drsm) \leq n^{\frac{1}{\theta}} \cdot \sqrt{n} + n^{\frac{1}{\theta}} + O(\log n)$$

bits to save the code of M. Π takes $O(\log n)$ bits to hardwire pos since the list $\vec{x}$ is of size L, which is polynomial in n. Thus, the description length of Π is at most $O(n^{\frac{1}{\theta}} \cdot \sqrt{n}) < n^{2/3} = |x|^{2/3}$ (if n is sufficiently large).

We then prove that the running time of $\Pi(z)$ is a priori-bounded and polynomial in n. It takes $\mathsf{poly}(2^d)$ time to compute $\mathsf{GenDesign}(d, r, s)$, executing the program $M^{D(z,\cdot)}(\mathcal{I})$ takes $\mathsf{poly}(2^r) \cdot O(m^\theta)$ time (since the distinguisher D runs in $O(m^\theta)$ time). The list decoding algorithm runs in $\mathsf{poly}(n, 1/\delta)$, and finally to find $\vec{x}_{\mathsf{pos}}$ and output takes $O(nL)$. So the total running time of $\Pi(z)$ is at most an a-priori bounded polynomial in n. Combining this and the proofs given above, we reach the conclusion that Π is of length at most $|x|^{2/3}$ and $\Pi(z)$ indeed outputs x within a fixed number of steps.

It remains to show that the algorithm $A(x,z)$ runs in poly time. Note that A takes $\mathsf{poly}(n, 1/\delta)$ time for running $\mathsf{Enc}_{n,\delta}(x)$, $\mathsf{poly}(2^d)$ time for $\mathsf{GenDesign}(d, r, s)$, $\mathsf{poly}(2^r) \cdot O(m^\theta)$ time for $\mathsf{NWRecon}^{D(z,\cdot)}(f, \mathcal{I})$, $\mathsf{poly}(2^r) \cdot O(m^\theta)$ time for emulating $M^{D(z,\cdot)}(\mathcal{I})$, $\mathsf{poly}(n, 1/\delta)$ for $\mathsf{Dec}_{n,\delta}(x')$, and finally $O(nL)$ to locate x in $\vec{x}$. To sum up, A runs in polynomial time. ◀

As a summary, Lemma 14 shows that if $\mathsf{GapMcKtP}$ is hard given some particular auxiliary input sequence, then we can obtain a "partial" targeted HSG which is only secure on some sequence of targeted strings. We next show how to make use of this result to obtain a full-fledged targeted HSG.

▶ **Lemma 21.** *There exists a constant $c \geq 1$, and for each constant $\gamma > 0$, there exists an exponential time uniform auxiliary input sequence $\mathcal{Z}_0$ such that the following holds. If $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is hard for probabilistic n^c-time algorithms given auxiliary input $\mathcal{Z}_0$, then there exist constants $\sigma \geq 1, \theta \geq 1$ and a $O(m^\theta)$-secure $(m^\theta, \sigma \log m)$-target HSG.*

Proof. Let c be the constant guaranteed to exist by Lemma 14. Consider any constant $\gamma > 0$, and let σ, θ be the constants and $g : 1^m \times \{0,1\}^{m^\theta} \times \{0,1\}^{\sigma \log m} \to \{0,1\}^m$ be the efficiently computable function where σ, θ, g are guaranteed to exist by Lemma 14. Given g and the security bound of g (which is taken to be $O(m^\theta)$), let $\mathcal{Z}_1 = \{z_{1,n}\}_{n\in\mathbb{N}}$ be the exponential time uniform (universal) sequence of target strings we pick in Lemma 9. Let $\mathcal{Z}_0 = \{z_{0,n}\}_{n\in\mathbb{N}}$ be an auxiliary input sequence such that $z_{0,n} = z_{1,n^{1/\theta}}$ (which is a simply padded version of $\mathcal{Z}_1$). Notice that $\mathcal{Z}_0$ can also be produced by an exponential machine.

Assume that $\mathsf{GapMcKtP}[\gamma \log n, n-1]$ is hard for probabilistic n^c-time algorithms given auxiliary input $\mathcal{Z}_0$. Then, by (the contrapositive of) Lemma 14, g is a $O(m^\theta)$-secure $(\mathcal{Z}_1, \sigma \log m)$-target HSG with respect to the target string sequence $\mathcal{Z}_1$. Finally, by Lemma 9, g is a $O(m^\theta)$-secure $(m^\theta, \sigma \log m)$-target HSG secure on all target strings. ◀

References

1 Alexander E Andreev, Andrea EF Clementi, and Jose DP Rolim. A new general derandomization method. *Journal of the ACM (JACM)*, 45(1):179–213, 1998.

2 Alexander E Andreev, Andrea EF Clementi, José DP Rolim, and Luca Trevisan. Weak random sources, hitting sets, and bpp simulations. *SIAM Journal on Computing*, 28(6):2103–2116, 1999.

3 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

4 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.

5 Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

6 Harry Buhrman and Lance Fortnow. One-sided versus two-sided error in probabilistic computation. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 100–109. Springer, 1999.

7 Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969.

8 Lijie Chen, Ron D Rothblum, Roei Tell, and Eylon Yogev. On exponential-time hypotheses, derandomization, and circuit lower bounds. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 13–23. IEEE, 2020.

9 Lijie Chen and Roei Tell. Hardness vs randomness, revised: Uniform, non-black-box, and instance-wise. *Electronic Colloquium on Computational Complexity*, 2021. URL: `https://eccc.weizmann.ac.il/report/2021/080/l`.

10 Oded Goldreich. In a world of p= bpp. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 191–232. Springer, 2011.

11 Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

12 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference, CCC 2020*, pages 22:1–22:36, 2020.

13 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

14 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if e requires exponential circuits: Derandomizing the xor lemma. In *STOC '97*, pages 220–229, 1997.

15 Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. `doi:10.1016/0304-3975(86)90081-2`.

16 A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

17 Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

18 Leonid A. Levin. Universal search problems (russian), translated into English by BA Trakhtenbrot in [27]. *Problems of Information Transmission*, 9(3):265–266, 1973.

19 Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. In Wen-Lian Hsu and Richard C. T. Lee, editors, *ISA '91 Algorithms, 2nd International Symposium on Algorithms, Taipei, Republic of China, December 16-18, 1991, Proceedings*, volume 557 of *Lecture Notes in Computer Science*, pages 308–315. Springer, 1991. doi:10.1007/3-540-54945-5_75.

20 Cody Murray and 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*, pages 890–901, 2018.

21 Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

22 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

23 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983.

24 R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964. doi:10.1016/S0019-9958(64)90223-2.

25 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

26 Roei Tell. Proving that prbpp= prp is as hard as proving that "almost np" is not contained in p/poly. *Information Processing Letters*, 152:105841, 2019.

27 Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

28 Salil P Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

29 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

30 A. K. Zvonkin and L. A. Levin. the Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms. *Russian Mathematical Surveys*, 25(6):83–124, December 1970. doi:10.1070/RM1970v025n06ABEH001269.

On One-Way Functions from NP-Complete Problems

Yanyi Liu ✉
Cornell Tech, New York, NY, USA

Rafael Pass ✉
Cornell Tech, New York, NY, USA
Tel-Aviv University, Israel

Abstract

We present the first natural NP-complete problem whose average-case hardness w.r.t. the uniform distribution over instances is *equivalent* to the existence of one-way functions (OWFs). The problem, which originated in the 1960s, is the *Conditional Time-Bounded Kolmogorov Complexity Problem*: let $K^t(x \mid z)$ be the length of the shortest "program" that, given the "auxiliary input" z, outputs the string x within time $t(|x|)$, and let $\mathsf{McK}^t\mathsf{P}[\zeta]$ be the set of strings (x, z, k) where $|z| = \zeta(|x|)$, $|k| = \log |x|$ and $K^t(x \mid z) < k$, where, for our purposes, a "program" is defined as a RAM machine.

Our main result shows that for every polynomial $t(n) \geq n^2$, there exists some polynomial ζ such that $\mathsf{McK}^t\mathsf{P}[\zeta]$ is NP-complete. We additionally extend the result of Liu-Pass (FOCS'20) to show that for every polynomial $t(n) \geq 1.1n$, and every polynomial $\zeta(\cdot)$, mild average-case hardness of $\mathsf{McK}^t\mathsf{P}[\zeta]$ is equivalent to the existence of OWFs. Taken together, these results provide the following crisp characterization of what is required to base OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$:

> *There exists concrete polynomials t, ζ such that "Basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$" is equivalent to providing a "worst-case to (mild) average-case reduction for $\mathsf{McK}^t\mathsf{P}[\zeta]$".*

In other words, the "holy-grail" of Cryptography (i.e., basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$) is equivalent to a basic question in algorithmic information theory.

As an independent contribution, we show that our NP-completeness result can be used to shed new light on the feasibility of the *polynomial-time bounded symmetry of information* assertion (Kolmogorov'68).

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

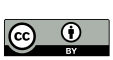
Keywords and phrases One-way Functions, NP-Completeness, Kolmogorov Complexity

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.36

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2021/059/

Funding *Rafael Pass*: Work partially done while being on a sabbatical at Tel-Aviv University. Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, and a JP Morgan Faculty Award. This material is based upon work supported by DARPA under Agreement No. HR00110C0086. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

Acknowledgements We are very grateful to Vinod Vaikuntanathan and Rahul Ilango for helpful comments on an earlier version of this paper. We thank the anonymous reviewers for their helpful comments.



37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 36; pp. 36:1–36:24

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

1 Introduction

A one-way function (OWF) [15] is a function f that can be efficiently computed (in polynomial time), yet no probabilistic polynomial-time (PPT) algorithm can invert f with inverse polynomial probability for infinitely many input lengths n. Whether OWFs exist is unequivocally the most important open problem in Cryptography: OWFs are both necessary [34] and sufficient for many of the most central cryptographic primitives and protocols (e.g., pseudorandom generators [10, 24], pseudorandom functions [19], private-key encryption [20], digital signatures [56], commitment schemes [51], identification protocols [16], coin-flipping protocols [9], and more). These primitives and protocols are often referred to as *private-key primitives*, or "Minicrypt" primitives [33] as they exclude the notable task of public-key encryption [15, 55].

While many candidate constructions of OWFs are known – most notably based on factoring [55], the discrete logarithm problem [15], or the hardness of lattice problems [1] – the question of whether OWFs can be based on some "standard" complexity-theoretic assumption is mostly wide open. Indeed, a central open problem – often referred to as the "holygrail of cryptography" – originating in the seminal work of Diffie and Hellman [15] is whether the existence of OWFs can be based on the assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$.[1] So far, however, most results in the literature have been negative. Notably, starting with the work by Brassard [13] in 1983, a long sequence of works have shown various types of black-box separations between *restricted* types of OWFs (e.g., one-way permutations) and NP-hardness (see e.g., [13, 12, 3, 53, 21, 47, 22, 11]). We emphasize, however, that these results only show limited separations: they either consider restricted types of one-way functions, or restricted classes of black-box reductions. Thus, even w.r.t. black-box reductions, the question of whether OWFs can be based on the assumption that $\mathsf{NP} \not\subseteq \mathsf{BPP}$, is wide open. In this work, our focus is on providing a complexity-theoretic characterization of *exactly* what is required for basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$:

> *Is there a simple complexity-theoretic characterization of what is required for basing OWFs on the assumption that* $\mathsf{NP} \not\subseteq \mathsf{BPP}$*?*

We believe that having a crisp complexity-theoretic characterization will be useful both for obtaining more meaningful separation results, and towards the goal of eventually getting a construction of OWFs based on $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

Towards Characterizing the Possibility of Basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$. A first step towards answering the above question is implied by a recent work by Liu and Pass [43]; they demonstrated the first natural NP-language whose average-case hardness characterizes the existence of OWFs. In more detail, they demonstrated that for any polynomial $t(n) \geq 1.1n$, OWFs exist if and only the *t-time-bounded Kolmogorov complexity problem*, $\mathsf{MK}^t\mathsf{P}$, is mildly hard-on-average, where a language L is said to be *mildly hard-on-average* if there exists some polynomial $p(\cdot)$ such that no PPT heuristic $\mathcal{H}$ can decide L with probability $1 - 1/p(n)$ over random n-bit instances for infinitely many input lengths n. (We provide more details on the definition of $\mathsf{MK}^t\mathsf{P}$ below.) $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ is contained in NP, but it is unknown whether this problem (which has been studied since the 1960s) is NP-complete. Indeed, this is one of the

[1] Or more precisely, whether OWFs can be based on the assumption that $\mathsf{NP} \not\subseteq \mathsf{ioBPP}$ since the definition of OWFs requires "almost everywhere" hardness. For convenience, in the introduction we are ignoring this issue.

long-standing open problems in algorithmic information-theory [38]. A simple corollary of the result from [43] (as far as we know, this has not been previously observed) is that basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$ is *equivalent* to (1) proving that $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ is NP-complete (perhaps with a non-constructive reduction), and (2) providing a worst-case to average-case reduction for $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$.[2] To see why this is the case, note that if (1) and (2) are satisfied, then by the result of [43], we have directly based OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$. For the converse direction, note that if OWFs can be based on $\mathsf{NP} \not\subseteq \mathsf{BPP}$, then $\mathsf{NP} \not\subseteq \mathsf{BPP}$ implies the existence of OWF, which by [43] implies that $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ is average-case hard (and thus also worst-case hard); thus we have that (1) must hold. To see that (2) also holds, note that since $\mathsf{MK}^{\mathsf{poly}}\mathsf{P} \in \mathsf{NP}$, it follows that $\mathsf{MK}^{\mathsf{poly}}\mathsf{P} \not\subseteq \mathsf{BPP}$ implies that $\mathsf{NP} \not\subseteq \mathsf{BPP}$, which in turn implies OWF (by assumption) which in turn by [43] implies that $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ is average-case hard, and thus (2) follows.

The above discussion, however, leaves open the question of whether a crisper characterization can be obtained. In particular, if one can come up with a *natural* NP-complete language L whose average-case hardness is equivalent to the existence of OWFs, then the question of whether OWFs can be based on $\mathsf{NP} \not\subseteq \mathsf{BPP}$ would be equivalent to the question of whether there exists a worst-case to average-case reduction for this particular problem. This thus begs the question whether there exists some natural NP-*complete* language that characterizes the existence of OWFs:

> *Does there exist some "natural"* NP*-complete language L such that OWFs exist iff L is hard-on-average?*

This question was recently raised (but not solved) in a paper by Allender et al [5] (and without the above motivation). We note that "naturality" of the language L is *key* for this question to make sense: It is easy to modify $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ into a new "artificial" language L' which is both NP-complete, yet (mild) average-case hardness of L' is equivalent to mild average-case hardness of $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ (and thus equivalent to the existence of OWFs).[3] But such an artificial problem would have no relevance to the central question that concerns us (i.e., providing a crisp characterization of when OWF can be based on $\mathsf{NP} \not\subseteq \mathsf{BPP}$).

There is a long history of work on trying to base OWFs on average-case hardness of NP-complete problems, starting with the work of Merkle and Hellman [50]. While the original attempts failed to produce secure schemes (see [52] for a survey), more recent approaches pioneered by Impaglizzo and Naor [35], Ajtai [1] and Ajtai and Dwork [2] produced not just OWFs but also more advanced cryptographic primitives (such as collision-resistant hash functions and public-key encryption) based on well-founded average-case hardness assumptions on the subset sum problem (which is NP-complete). However, it is not known whether the existence of OWFs implies average-case hardness of the subset sum problem (i.e., they only have a one-sided implication).

In this work, we identify the first natural NP-complete language L – *time-bounded Conditional Kolmogorov-complexity* [62, 42, 59, 48] – such that mild average-case hardness of L (with respect to the uniform distribution on instances) is equivalent to the existence

[2] We emphasize that we need a worst-case to *2-sided* error average-case reduction. Hirahara's elegant work [25] makes partial progress on this question by presenting a worst-case (approximate) to *errorless* average-case reduction; errorless average-case hardness does not suffice for [43].

[3] Simply consider the language L' of $2n$-bit instances $x||y$ where $x, y \in \{0,1\}^n$, and either (a) $x = 0^n$ and $y \in \mathsf{SAT}$, or (b) $x \neq 0^n$ and $y \in \mathsf{MK}^{\mathsf{poly}}\mathsf{P}$. In other words, L' is a combination of SAT and $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$, so clearly this language is NP-complete, but when considering uniform statements, we only hit SAT instances with negligible probability, and thus this language behaves essentially just like $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$ on average.

of OWFs. As a consequence, we get that basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$ is equivalent to providing a worst-case to average case reduction for this particular problem, yielding a simple complexity-theoretic characterization of exactly what it takes to base OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

1.1 Our Results

Before describing our results in detail, let us first briefly recall the notion of Time-bounded Kolmogorov Complexity and the result of [43] that we will be relying on.

Time-bounded Kolmogorov Complexity and OWFs. What makes the string 1212121212 1212121 less random than 60484850668340357492? The notion of *Kolmogorov complexity* (K-complexity) [58, 40, 14] from the field of algorithmic information theory provides an elegant method for measuring the amount of "randomness" in individual strings: The K-complexity of a string is the length of the shortest program (to be run on some fixed universal Turing machine U) that outputs the string x. The notion of *$t(\cdot)$-time-bounded Kolmogorov Complexity (K^t-complexity)* is a computationally-restricted version of K-complexity: $K^t(x)$ is defined as the length of the shortest program that outputs the string x within time $t(|x|)$. As surveyed by Trakhtenbrot [59], the problem of efficiently determining the K^t-complexity for $t(n) = \mathsf{poly}(n)$ predates the theory of NP-completeness and was studied in the Soviet Union since the 60s as a candidate for a problem that requires "brute-force search". The modern complexity-theoretic study of this problem goes back to Sipser [57], Ko [37] and Hartmanis [23]. Let $\mathsf{MK}^{t(\cdot)}\mathsf{P}$ denote the decisional $t(n)$-time bounded Kolmogorov complexity problem; namely, the language of pairs (x, k) where $|k| = \lceil \log |x| \rceil$ and $K^t(x) \leq k$.

As mentioned above, Liu and Pass [43] demonstrated that for every polynomial $t(n) \geq 1.1n$, mild average-case hardness of $\mathsf{MK}^t\mathsf{P}$ is equivalent to the existence of OWFs. But as mentioned, it is not known whether $\mathsf{MK}^t\mathsf{P}$ is NP-complete (for any polynomial t). Towards getting a characterization of OWFs based on average-case hardness of an NP-complete problem, we will consider a generalization of $\mathsf{MK}^t\mathsf{P}$ based on *conditional* Kolmogorov complexity.

Conditional Time-bounded Kolmogorov Complexity. The *$t(\cdot)$-time-bounded Conditional Kolmogorov Complexity* [62, 42, 59, 48] of a string x conditioned on the string z – denoted $K^t(x \mid z)$ – is the length of the shortest program that, given the "auxiliary input" z, outputs the string x within time $t(|x|)$. More formally,

$$K^t(x \mid z) = \min_{\Pi \in \{0,1\}^*} \{|\Pi| : U(\Pi(z), 1^{t(|x|)}) = x\},$$

where U is a universal Turing machine, and we let $U(\Pi(z), 1^t)$ denote the output of the program Π on input z after t steps. Whereas the notion of a "program" typically is taken to be a Turing machine, in this work we focus on the setting where a program is taken to be a RAM-machine – namely Π is now allowed to be a RAM-machine that can make Random Access queries into the auxiliary string z. Let $\mathsf{McK}^{t(\cdot)}\mathsf{P}[\zeta(\cdot)]$ denote the decisional $t(\cdot)$-time-bounded $\zeta(\cdot)$-conditional Kolmogorov complexity problem; namely, the language of triples (x, z, k) where $|z| = \zeta(|x|)$, $|k| = \lceil \log |x| \rceil$ and $K^t(x \mid z) \leq k$. Whereas conditional (time-bounded) Kolmogorov complexity has been studied for decades (see e.g., [48]), it has also remained an open question to determine whether this problem is NP-complete.[4]

[4] We remark, however, that as far as we know, we are the first to consider this problem w.r.t. RAM programs as opposed to Turing machines. In our view, this RAM version of the problem is as natural (if not more) than the "standard" TM version.

We observe that the result of [43] extends, with only relatively minor modifications in the proof, also to conditional Kolmogorov complexity: We show that for every polynomial $t(\cdot) \geq 1.1n$, and every polynomial $\zeta(\cdot)$, mild average-case hardness of $\mathsf{McK}^t\mathsf{P}[\zeta]$ is equivalent to the existence of OWFs.

▶ **Theorem 1.1** (closely following [43]). *For every polynomial $t(n) \geq 1.1n$, every polynomial $\zeta(\cdot)$, mild average-case hardness of $\mathsf{McK}^t\mathsf{P}[\zeta]$ is equivalent to the existence of OWFs.*

So, if we could show that $\mathsf{McK}^t\mathsf{P}[\zeta]$ is NP-complete for some polynomials t, ζ, we would be done. Our main theorem does exactly this.

▶ **Theorem 1.2** (Main Theorem). *For every polynomial $t(n) \geq n^2$, there exists some polynomial $\zeta(\cdot)$, such that $\mathsf{McK}^t\mathsf{P}[\zeta]$ is NP-complete (under randomized polynomial-time reductions).*

Let us emphasize that the combination of Theorem 1.1 and Theorem 1.2, for instance, yields the following crisp characterization of the "holygrail" of Cryptography:

> *There exists (concrete) polynomials t, ζ such that "Basing OWFs on $\mathsf{NP} \not\subseteq \mathsf{BPP}$" is equivalent to the existene of a worst-case to (mild) average-case reduction for $\mathsf{McK}^t\mathsf{P}[\zeta]$.*

In other words, the "holy grail" of Cryptography is equivalent to a basic question in algorithmic information theory. Furthermore, let us point out that for the unconditional time-bounded Kolmogorov complexity problem $\mathsf{MK}^{\mathsf{poly}}\mathsf{P}$, some partial[5] worst-case to average-case reductions are known [25], so this gives us hope that a full worst-case to average-case reduction may be possible also for $\mathsf{McK}^t\mathsf{P}$.

As we shall discuss shortly, Theorem 1.2 is also interesting in its own right and has other direct applications: we show how to shed new light on a long-standing open problem regarding *symmetry of information* [62] for the setting of time-bounded Kolmogorov complexity.

Let us emphasize that for the NP-completeness result to hold, it is imperative that our notion of conditional Kolmogorov complexity views programs as *RAM-machines* (as opposed to *Turing machines*). We leave it as an intriguing open problem to determine whether the "standard" conditional time-bounded Kolmogorov complexity (where interpreting a program as a Turing machine) is also NP-complete.

We proceed to providing a proof overview of the main theorem (i.e. Theorem 1.2).

1.2 Proof Overview

We first note that it directly follows that for all polynomials t, ζ, $\mathsf{McK}^t\mathsf{P}[\zeta] \in \mathsf{NP}$ – the witness for an instance (x, z, k) is simply a RAM program Π such that $|\Pi| \leq k$ and $\Pi(z)$ generates x within $t(|x|)$ steps. We turn to discussing how to prove that there exist polynomials t, ζ, such that $\mathsf{McK}^t\mathsf{P}[\zeta]$ is NP-hard. On a high-level, our approach will start off by using the recent breakthrough approach by Ilango [29, 30] showing NP-hardness of an *oracle*-variant of the *circuit minimization problem (MCSP)* [36] – that is, the problem of, given the truth table of a boolean function, determining the size of the smallest circuit that computes the function – and next extend it to deal with the conditional Kolmogorov complexity problem by appropriately embedding the "oracles" used in the construction of [29] in the auxiliary input.

[5] The reduction only shows so-called *errorless*, as opposed to 2-sided error, average-case hardness.

In more detail, following [8, 28, 29, 30, 32], we will embed an (approximate) Bounded Set Cover instance into an $\mathsf{McK}^t\mathsf{P}[\zeta]$ instance; the approximate Bounded Set Cover problem is known to be NP-complete [60]. Recall that in the Bounded Set Cover problem, we are given a collection of sets $S_1, S_2, \ldots S_r$, each of which is a *constant*-size subset of the universe $U = [n]$ and the goal is to find a minimal set of indexes, s, such that $\cup_{i \in s} S_i = [n]$ (i.e., finding the minimal collection $\mathcal{S}$ of sets S_i that cover $[n]$). We start off by generalizing an idea from [29, 30, 32] and replace the universe $U = [n]$ with n random strings $A_i \in \{0,1\}^m$, where $m(n)$ is some sufficiently large polynomial (in the formal proof $m(n) = n^3$). Roughly speaking, the rationale for doing this is that a set cover, intuitively, should give a succinct (proportional to the size of the set cover) way to generate the random string $A = A_1||A_2|| \ldots ||A_n$ if we have *oracle access* to the sets S_i – we simply need to specify the sets in the set cover and can then reconstruct the union of these sets. This construction was used in [29] to prove NP-hardness of the oracle-version of the MCSP problem – the sets S_i were simply placed into the oracle. ([30] provides a more elaborate construction that also shows NP-hardness of a conditional variant of the MCSP problem; we will, however, not rely on that extension.)

To convert the above set-cover instance into a conditional Kolmogorov complexity problem, our new idea will be to place the description of the sets S_i (each of which consists of some set of strings A_i) at *random locations* in the auxiliary string z and to make sure z is very long (yet still only of polynomial length), and consider the conditional Kolmogorov complexity problem of computing $K^t(A \mid z)$ where $A = A_1||A_2|| \ldots ||A_n$. Conceptually, one can view this approach as a way to *obfuscate* the oracle used in [29] and placing the obfuscation in z. Intuitively, since we are placing the descriptions of the sets S_i at random locations in z, a time-bounded algorithm can only access S_i if it "knows" the random location where it has been put, and thus, intuitively, we can view z as an information-theoretic obfuscation of gates that compute these descriptions.[6]

If there exists a set cover s of size ℓ, $K^t(A \mid z)$ should be no more than $\ell O(\log n) + O(1)$, by considering the program that simply hardcodes the location in z of the descriptions of the sets S_i for $i \in s$. The harder part is showing that if $K^t(A \mid z) \leq \ell O(\log n)$ then there exists a set-cover of size $O(\ell)$. Relying on the intuition that z acts as an obfuscation of the description of the sets $\{S_i\}$, the intuition for why this holds is that if z is sufficiently longer than $t(|x|)$, and the descriptions of the sets are put into random positions of z, any program with running-time $t(|x|)$ that reconstructs the string $A = A_1||A_2|| \ldots ||A_n$ (which with overwhelming probability has high Kolmogorov complexity) must "know" the location in the auxiliary string z of sets $\{S_i\}_{i \in s}$ in some set cover s, in the sense that by running this program, those positions can be "reconstructed". In a bit more detail, by running this program and looking at the memory access queries made by the program into z, we must be hitting the locations where the sets have been put. But since these locations are random (by construction of z), the program needs to basically "hard-code" them, or else we would be able to compress the indexes of these locations, but these indexes have high Kolmogorov complexity as they were picked at random, which is a contradiction.

The reader may note that, perhaps curiously, we are using an argument based on Kolmogorov complexity to formalize the statement that $K^t(A \mid z) \approx \ell O(\log n)$. In more detail, we are relying on a Kolmogorov-complexity style compression argument to formalize

[6] This intuition is somewhat misleading: since z is only of polynomial length, the "obfuscation" only works with respect to a-priori time-bounded attackers (that can only explore a small fraction of z) and can only have inverse polynomial security. But in our context, such a relaxed notion of security suffices.

that z acts as a good "obfuscation" of the description of the sets $\{S_i\}$. This proof technique bears similarities to the proof technique pioneered by Gennero and Trevisan [17] in the context of proving that a random permutation is one-way w.r.t. polynomial-size circuits.

Let us end by noting that the above proof outline oversimplifies and misses several crucial details that make the actual proof quite a bit more complicated.

1.3 Applications to Polynomial-time Bounded Symmetry of Information

The celebrated *symmetry of information* theorem by Kolmogorov and Levin from 1967 [62] states that for all strings $x, y \in \{0,1\}^*$:

$$K(xy) = K(y) + K(x \mid y) \pm O(\log |xy|)$$

where xy denotes the concatenation of the strings x, y. The proof of this theorem, however, involves a computationally expensive exhaustive search through all strings of lengths $|x|, |y|$. The question of whether a *polynomial-time bounded* version of information symmetry holds, where K-complexity is replaced by K^t-complexity for polynomials t, has remained an open problem. We refer to the assertion that there exists some constant $0 < \epsilon \leq 1$ such that for all sufficiently large polynomials t, all $x, y \in \{0,1\}^*$ (of polynomially-related length), it holds that

$$K^{t^\epsilon}(xy) \geq K^t(y) + K^t(x \mid y) - O(\log |xy|)$$

$$K^{t^{1/\epsilon}}(xy) \leq K^t(y) + K^t(x \mid y) + O(\log |xy|)$$

as the *polynomial-time symmetry of information assertion (*$\mathsf{polySOI}$*)*. The question of whether a polynomial-time symmetry of information assertion holds goes back to a work by Kolmogorov from 1968 [39]; as retold by Levin [41]:

> *Kolmogorov suggested at the time [39] that this information symmetry theorem may be a good test case to prove that for some tasks exhaustive search cannot be avoided (in today's terms,* $\mathsf{P} \neq \mathsf{NP}$*)*

The first formal complexity-theoretic investigation of this question goes back to works by Longpré and Mocas [48] and Longpré and Watanabe [49]. They consider a *length-restricted* version of the $\mathsf{polySOI}$ where we additionally add the requirement that $|x| = |y|$ and show that (1) the length-restricted $\mathsf{polySOI}$ assertion holds if $\mathsf{NP} = \mathsf{P}$, and (2) the length-restricted $\mathsf{polySOI}$ assertion is false if one-way functions exist. We here focus our attention on the above "length-unrestricted" version of the $\mathsf{polySOI}$ assertion, where x and y can be of arbitrary polynomially-related lengths. We demonstrate, as a corollary of the techniques behind the proof of Theorem 1.2, that the (length unrestricted) $\mathsf{polySOI}$ assertion is *unconditionally* false.

▶ **Theorem 1.3.** *The (length-unrestricted) polynomial-time symmetry of information assertion is false.*

Let us provide a brief overview of how this theorem is proven, and the instrumental role that the proof of Theorem 1.2 plays in this proof. The proof, roughly speaking, proceeds in the following steps:

- **Step 1: Polynomial increase in running-time can only decrease $K^t(x)$ by $O(\log |x|)$.** We first show that $\mathsf{polySOI}$ implies that if we polynomially increase the running-time bound, then this cannot have a significant effect on K^t; more precisely, for

large enough t, it holds that for all constants c, $K^t(x)$ cannot be more than $O(\log |x|)$ more than $K^{t^c}(x)$. Intuitively, this follows from polySOI by letting y be a sufficiently long all "dummy" string, 0^m. We note that this step inherently relies on us considering a length *unrestricted* polySOI.

- **Step 2: Showing that a "strong" polySOI assertion holds.** We next observe that combining Step 1 with polySOI yields a strong form of the polySOI assertion where $\epsilon = 1$; i.e., it holds that

$$K^t(xy) = K^t(y) + K^t(x \mid y) \pm O(\log |xy|)$$

- **Step 3: Polynomial increase in running-time can only decrease *conditional* $K^t(x|z)$ by $O(\log |xz|)$.** We then combine the "strong" polySOI assertion from Step 2 with Step 1 to show that an analog of Step 1 holds also with respect to conditional time-bounded Kolmogorov complexity. More precisely, for large enough t, it holds that for all constants c, $K^t(x \mid z)$ cannot be more than $O(\log |xz|)$ larger than $K^{t^c}(x \mid z)$.
- **Step 4: Contradicting the construction in Theorem 1.2.** We finally observe that the statement of Step 3 contradicts the construction in the proof of Theorem 1.2. In particular, in the proof of Theorem 1.2, we showed strings x, z where the condition time-bounded Kolmogorov complexity $K^t(x \mid z)$ is large (roughly the size of the set-cover). However, this relied on t being sufficiently small so that the program cannot read all of z. If t is large, so that the program can read all of z, then $K^t(x \mid z)$ is tiny (x can be trivially reconstructed from z). This contradicts the statement of Step 3.

Let us emphasize that Theorem 1.3 is incomparable to the result of [49]: [49] consider a weaker "length-restricted" polynomial-time symmetry of information assertion (where $|x| = |y|$), whereas we are considering a "length-unrestricted" version. While [49] show that the length-restricted polySOI indeed holds if $\mathsf{NP} = \mathsf{P}$, we show that the length-unrestricted version is unconditionally false. We do, however, note that the "standard" (non time-bounded) symmetry of information theorem of [62] consider the length unrestricted case, in analogy with what we do.

Furthermore, for our result to hold (and in contrast to [49]), it is crucial that we consider RAM-programs as the model of computation, as opposed to the (more standard in the context of time-bounded Kolmogorov complexity) notion of TM-programs; nevertheless, in our eyes, considering RAM programs is equally motivated (if not more) than TM programs.

1.4 Related Works

As mentioned above, there has been a recent sequence of surprising works proving NP-hardness results for variants of the MCSP problem [8, 28, 29, 30, 32]; in particular, as mentioned, Ilango [30] proves that a conditional version of the MCSP problem is NP-hard. As observed already in [59], and further explored in [4], the MCSP problem is closely related to the time-bounded Kolmogorov complexity problem – intuitively, the two problems capture the same concept, but using a different model of computation – but a formal reduction between these problems is not known so these results do not directly extend to the setting we consider. (However, as mentioned above, the starting point for our approach is the result of [29] showing NP-hardness for an oracle version of the MCSP problem.)

A recent result by Hirahara [26] directly addresses conditional time-bounded Kolmogorov complexity and shows NP-hardness for a variant of this problem, $\mathsf{McK}^{\mathsf{poly}}\mathsf{P}^{\mathsf{SAT}}$, where the program has access to a SAT-oracle. (The $\mathsf{McK}^{\mathsf{poly}}\mathsf{P}^{\mathsf{SAT}}$ problem, however, is not known to be in NP, but is in $\mathsf{NP}^{\mathsf{NP}}$, so NP-completeness is not shown).

An intriguing recent paper by Allender et al [5] presented a natural NP-complete problem L – a sparse variant of the MCSP problem – such that average-case hardness of L was claimed to imply the existence of OWFs; the authors also claimed a "weak" converse of this implication – that the existence of OWFs implies a very weak, so-called "non-trivial", notion of average-case hardness of the language[7]; unfortunately, an error was found in the paper. Concurrently and independently from the current work, the authors of [5] show how to repair the issues in their proof and present a different NP-complete language whose average-case hardness implies the existence of OWFs, and for which the same weak converse holds.[8] While their original posting [5] – which inspired the current work – attempted to base OWFs on the average-case hardness of a sparse version of the MCSP problem, their new paper [6] instead bases OWFs on average-case hardness of a conditional Kolmogorov complexity style problem, just as in the current work. Their conditional Kolmogorov complexity problem differs from ours in several aspects: (1) whereas we consider conditional Kolmogorov complexity w.r.t. RAM programs, [6] considers it w.r.t. Turing machines with "oracle-access" to the auxiliary input z; and (2) instead of considering a time-bounded version of conditional Kolmogorov complexity (as we do), [6] instead *charge* for running-time in their notion of Kolmogorov complexity, following the KT notion of [4]. Due to these differences, NP-completeness of their problem follows essentially directly from the NP-completeness results of [29] (whereas we have to work a lot harder, as explained above). However, due to these differences, they only manage to show a one-directional implication between average-case hardness of their problem and OWFs (and only a weak converse in the other direction), whereas we establish an *equivalence* between average-case hardness of $\mathsf{McK}^t\mathsf{P}[\zeta]$ (for any polynomials $t(n) > 1.1n, \zeta(\cdot)$) and OWFs.

Subsequent to the initial posting [44] of this paper,[9] [7] have shown, based on the results in [54] that (mild) average-case hardness of the NP-complete problem considered in [6] is equivalent to the existence of OWFs computable in log space; their work thus provides an elegant characterization of what it means to base OWFs computable in logspace on $\mathsf{NP} \not\subseteq \mathsf{BPP}$.

After the initial posting of this paper, we were informed by Rahul Ilango [31] that he had independently also shown NP-completeness of some conditional time-bounded Kolmogorov complexity problem, but without writing down the results. Indeed, as far as we can tell, our paper is the first to present any type of NP-completeness results for Kolmogorov complexity problems.

Resource bounded notions of conditional Kolmogorov complexity are useful also in other (related) contexts. In a companion paper to the current work [46], we rely on a notion of *space-bounded* conditional Kolmogorov complexity (defined similarly to the time-bounded notion of conditional Kolmogorov complexity used in the current paper) to characterize OWFs in NC^0; alternative characterizations without relying on condition Kolmogorov complexity were provided in [54].

In [46], we also identify a problem whose (infinitely-often) average-case hardness w.r.t. error-less heuristics is equivalent to $\mathsf{EXP} \neq \mathsf{BPP}$ (i.e., the problem is EXP-average-case complete w.r.t. errorless heuristics), yet (two-sided error) average-case hardness of this problem is equivalent to the existence of OWFs; related results were also obtained in [54].

---

[7] Roughly speaking, that average-case hardness holds for an inverse *exponential*, as opposed to inverse polynomial, fraction of inputs.

[8] The papers were submitted to on ECCC/Eprint within one day of each other.

[9] The initial posting [44] did not contain the results on polynomial-time symmetry of information

Taken together, the current work and [46, 54], demonstrate that the existence of OWFs can be characterized through the average-case hardness of both NP-complete (this work) and EXP-complete ([46, 54]) languages.

It has been recently shown in [27, 18] that some other variant of symmetry of information holds under the assumption that NP is easy on average. We mention that our result (proved in Theorem 1.3) combined with the aforementioned result in [27, 18] does *not* prove that NP is hard on average unconditionally (due to the difference in formulating symmetry of information). In the form used in [27, 18], they require that symmetry of information holds with respect to individual running time bounds $t \in \mathbb{N}$ that are not polynomially-related to $|x|$ (or $|y|$), whereas we consider polynomially-related running time bounds and allow a (polynomially) larger running time bound when the string is longer. Our proof does not work in the setting of [27, 18] for technical reasons.

1.5 Outline

We will provide the formalizations and proofs of Theorem 1.2 in Section 3. We refer the reader to the full version [45] for formal treatments of Theorem 1.1 and 1.3.

2 Preliminaries

We let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for any integer $n \in \mathbb{N}$. For any two strings x, y, let $x||y$ denote the concatenation of x and y; whenever it is clear from context, we sometimes also use xy to denote the concatenation of x and y. In this work, we sometimes consider strings that contain a special symbol $\perp$ (besides 0 and 1). We will use the following standard encoding scheme – which we refer to as simple the *standard encoding scheme* $\mathsf{enc}_\perp$) to transform a string that may contain $\perp$ into a binary string: $\mathsf{enc}_\perp(x)$, of a string $x \in \{0, 1, \perp\}^*$ is a $2|x|$-bit binary string where we replace each bit in x by 00 for 0, 01 for 1, and 11 for $\perp$.

2.1 Set Cover

Let n be an integer and $S_1, S_2, \ldots, S_\ell, T$ be sets $\subseteq [n]$. We say that the sets $S_1, S_2, \ldots, S_\ell$ cover T if $T \subseteq S_1 \cup S_2 \cup \ldots \cup S_\ell$. Let $\mathcal{S}$ be a collection of sets. We define $\mathsf{cover}(T, \mathcal{S})$ to be the minimum number of sets in $\mathcal{S}$ necessary to cover T.

We recall the γ-Bounded Set Cover Problem:

- Input: $(1^n, 1^\ell, \mathcal{S})$ where n, ℓ are integers $\in \mathbb{N}$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ is a collection of subsets $\subseteq [n]$. It is guaranteed that all the sets in $\mathcal{S}$ covers $[n]$ together and for all $i \in [r]$, $|S_i| \leq \gamma$.
- Decide: Is $\mathsf{cover}([n], \mathcal{S}) \leq \ell$.

We also consider the approximate version of the γ-Bounded Set Cover problem. The α-approximate γ-Bounded Set Cover Problem is a promise problem $(\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}})$ where Π_{yes} contains $(1^n, 1^\ell, \mathcal{S})$ such that $\mathsf{cover}([n], \mathcal{S}) \leq \ell$ and Π_{no} consists of $(1^n, 1^\ell, \mathcal{S})$ such that $\mathsf{cover}([n], \mathcal{S}) > \alpha \cdot \ell$.

Trevisan [60] showed that approximating the γ-Bounded Set Cover Problem within a constant factor is NP-hard:

▶ **Theorem 2.1** ([60]). *For every constant $\alpha \geq 1$, there exists a constant $\gamma \in \mathbb{N}$ such that the α-approximate γ-Bounded Set Cover Problem is NP-hard. More concretely, for any language $L \in \mathsf{NP}$, there exists a polynomial-time algorithm R such that on input $x \in L$, $R(x)$ outputs an instance in Π_{yes}; on input $x \notin L$, $R(x)$ outputs an instance in Π_{no}, where $(\Pi_{\mathsf{yes}}, \Pi_{\mathsf{no}})$ denotes the α-approximate γ-Bounded Set Cover Problem.*

2.2 The RAM Model

A RAM program $\Pi = (M, y)$ consists of a CPU "next-step" Turing machine M, and some initial input $y \in \{0,1\}^*$. Let $\mathsf{state} = 0$ be an initial state. The execution of this RAM program Π on input $z \in \{0,1\}^*$ (which may be empty) proceeds as follows.

- At initialization, the memory is set to $y||\perp||z$, and the "read bit" b^{read} is set to $\perp$. (For simplicity, we assume that each memory position contains a symbol $\in \{0,1,\perp\}$.[10] We assume that the memory is of infinite length and the rest of the positions in the memory are filled with $\perp$.)
- At each CPU step, M receives as input $\mathsf{state} \in \{0,1\}^*$, the most recently read bit b^{read}, and outputs a new state $\mathsf{state}' \in \{0,1\}^*$, a read position i^{read}, a write position i^{write} and some bit b^{write} (to be written to position i^{write}).[11]
- The execution of this step replaces state with state', sets b^{read} to the content of memory position i^{read}, and replaces the content of memory position i^{write} by b^{write}.
- When $\mathsf{state} = \varepsilon$ (i.e., the empty string), the computation ends and the output of the of the computation is defined as the content of the memory tape up to the symbol $\perp$.[12]
- The running time of Π is defined to be the sum of the running time of M in all CPU steps.

Note that any polynomial-time Turing machine can be simulated by a polynomial-time RAM program by simply copying the content of the memory into state, next letting M run the original Turing machine using state as its tape, and finally copying the content of state back into the memory.

2.3 Time-bounded Conditional Kolmogorov Complexity

We introduce the notion of time-bounded conditional Kolmogorov complexity with respect to RAM programs. Roughly speaking, the *t-time-bounded Kolmogorov complexity*, $K^t(x \mid z)$, of a string $x \in \{0,1\}^*$ conditioned on a string $z \in \{0,1\}^*$ is the length of the shortest RAM program $\Pi = (M, y)$ such that $\Pi(z)$ outputs x in $t(|x|)$ steps.

Let U be some fixed Universal Turing machine that can emulate any RAM program Π with polynomial overhead. Let $U(\Pi(z), 1^t)$ denote the output of $\Pi(z)$ when emulated on U for t steps. We now define the notion of t-time-bounded conditional Kolmogorov complexity.

▶ **Definition 2.2.** *Let t be a polynomial. For all $x \in \{0,1\}^*$ and $z \in \{0,1\}^*$, define*

$$K^t(x \mid z) = \min_{\Pi \in \{0,1\}^*} \{|\Pi| : U(\Pi(z), 1^{t(|x|)}) = x\}$$

where $|\Pi|$ is referred to as the description length *of Π. When there is no time bound, we define*

$$K(x \mid z) = \min_{\Pi \in \{0,1\}^*} \{|\Pi| : U(\Pi(z), 1^{t'}) = x \text{ for some finite } t'\}$$

[10] When we implement this, we always use the standard encoding scheme, $\mathsf{enc}_\perp$. We also note that the string y and z can never contain the symbol $\perp$ (since they exclusively consist of 0s and 1s). When we load y and z into the memory, instead of storing y and z directly, we store the standard encoding of y and z (where 0 becomes 00 and 1 becomes 01).

[11] Formally, the inputs and outputs of M are separated by the $\perp$ symbol so that state can be of variable length.

[12] In a real execution, the content of the memory is encoded by the standard encoding scheme. The output of the computation is then defined by the *decoded* content of the memory.

We also consider the decisional variant of the minimum t-time-bounded conditional Kolmogorov complexity problem. Let t, ζ be two polynomials, and let $\mathsf{McK}^t\mathsf{P}[\zeta]$ denote the language of triples (x, z, k), having the property that $K^t(x \mid z) \leq k$, where $z \in \{0,1\}^{\zeta(|x|)}$ and $k \in \{0,1\}^{\lceil \log n \rceil}$.

We note that for any string $z \in \{0,1\}^*, x \in \{0,1\}^*$, for any polynomial $t(\cdot)$, $K^t(x \mid z)$, is always upper bounded by $|x| + O(1)$.

▶ **Fact 2.3.** *There exists a constant $c \in \mathbb{N}$ such that for all polynomial $t(\cdot)$, for all string $z \in \{0,1\}^*, x \in \{0,1\}^*$, $K^t(x \mid z) \leq |x| + c$.*

Proof. Consider the RAM program $\Pi = (M, x)$ where M is a Turing machine that directly sets $\mathsf{state} = \varepsilon$. Note that in the execution of Π, x will be put into the memory and Π will halt immediately. Thus Π will output the string x. Note that M is a constant-size machine, so the description length of Π is at most $|x| + c$ for some constant c. ◀

We finally remark that for any polynomials $t(\cdot), \zeta(\cdot)$, $\mathsf{McK}^t\mathsf{P}[\zeta] \in \mathsf{NP}$.

▷ Claim 2.4. For all polynomials $t(\cdot), \zeta(\cdot)$, $\mathsf{McK}^t\mathsf{P}[\zeta] \in \mathsf{NP}$.

Proof. On input an instance $(x, z, k) \in \mathsf{McK}^t\mathsf{P}[\zeta]$, and a witness Π, checking if $|\Pi| \leq k$, $|z| = \zeta(|x|)$ and $U(\Pi(z), 1^{t(|x|)}) = x$ can be done in polynomial time. ◁

2.4 One-way Functions

We recall the definition of one-way functions [15]. Roughly speaking, a function f is one-way if it is polynomial-time computable, but hard to invert for PPT attackers.

▶ **Definition 2.5.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be a* one-way function (OWF) *if for every PPT algorithm $\mathcal{A}$, there exists a negligible function μ such that for all $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) \in f^{-1}(f(x))] \leq \mu(n)$$

We may also consider a weaker notion of a *weak one-way function* [61], where we only require all PPT attackers to fail with probability noticeably bounded away from 1:

▶ **Definition 2.6.** *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. f is said to be a α-*weak one-way function (α-weak OWF) *if for every PPT algorithm $\mathcal{A}$, for all sufficiently large $n \in N$,*

$$\Pr[x \leftarrow \{0,1\}^n; y = f(x) : A(1^n, y) \in f^{-1}(f(x))] < 1 - \alpha(n)$$

2.5 Average-case Hard Languages

We turn to defining what it means for a language to be average-case hard (for PPT algorithms). We will be considering languages that are only defined on some input lengths (such as $\mathsf{McK}^t\mathsf{P}[\zeta]$). We say that a language L is *defined over inputs lengths $s(\cdot)$* if $L \subseteq \cup_{n \in \mathbb{N}} \{0,1\}^{s(n)}$. For concreteness, note that $\mathsf{McK}^t\mathsf{P}[\zeta]$ is defined on input lengths $s(n) = n + \zeta(n) + \lceil \log n \rceil$.

We now turn to defining average-case hardness.

▶ **Definition 2.7.** *We say that a language L defined over inputs lengths $s(\cdot)$ is $\alpha(\cdot)$* hard-on-average (α-HoA) *if for all PPT heuristic $\mathcal{H}$, for all sufficiently large $n \in \mathbb{N}$,*

$$\Pr[x \leftarrow \{0,1\}^{s(n)} : \mathcal{H}(x) = L(x)] < 1 - \alpha(n)$$

In other words, there does not exist a PPT "heuristic" $\mathcal{H}$ that decides L with probability $1 - \alpha(n)$ on infinitely many input lengths $n \in N$ over which L is defined.

We refer to a language L as being *mildly HoA* if there exists a polynomial $p(\cdot) > 0$ such that L is $\frac{1}{p(\cdot)}$-HoA.

3 NP-Hardness of McK^{t}P[ζ]

In this section, we prove our main theorem: We show that there exists a reduction from the approximate γ-Bounded Set Cover Problem to McK^{t}P[ζ] when t, ζ are sufficiently large.

▶ **Theorem 3.1.** *For all polynomial $t(n) \geq n^2$, there exists a polynomial $\zeta(n)$ such that* McK^{t}P[ζ] *is* NP*-hard under many-one randomized polynomial-time reductions.*

Proof. The theorem follows from Proposition 3.3 and Proposition 3.4 (stated and proved in Section 3.2), and Theorem 2.1. ◀

In fact, we note that the reduction only has one-sided errors:

▶ **Theorem 3.2.** *For all polynomial $t(n) \geq n^2$, if there exists a polynomial $\zeta(n)$ such that* McK^{t}P[ζ] $\in$ coRP*, then* NP $\subseteq$ coRP*.*

Proof. By Proposition 3.3, our reduction succeeds with probability 1 on YES instances. By Proposition 3.4, our reduction succeeds with high probability ($\geq \frac{1}{2}$) on NO instances. Finally, the corollary follows from Theorem 2.1. ◀

3.1 A Reduction from the γ-Bounded Set Cover Problem to McK^{t}P

Let γ be a constant, let $t(n) \geq n^2$ be a polynomial, and consider $\zeta(n) = (t(n))^4 n^{2\gamma}$. We will show that there exists a randomized reduction from the γ-Bounded Set Cover Problem to McK^{t}P[ζ].

Given an instance $(1^n, 1^\ell, \mathcal{S})$ where $\mathcal{S} = \{S_1, S_2, \ldots, S_r\}$ of the γ-Bounded Set Cover Problem, we proceed as follows:

- Let $m = n^3$; for each $i \in [n]$, sample a random string $A_i \in \{0,1\}^m$, and consider the length-$(n \times m)$ concatenation $A = A_1 || A_2 || \ldots || A_n$ of the sampled strings. Think of A_i as a randomized encoding of the element i in the Set Cover problem. See Figure 1 for an illustration of these strings.

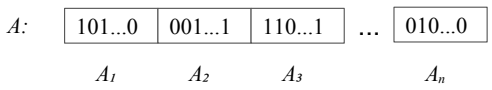


Figure 1 An illustrative example for the string A.

- For each $i \in [r]$, we construct a "gadget" string $W_i \in (\{0,1\}^m)^n$ (for set S_i). We partition W_i into n blocks $W_{i,1}, W_{i,2}, \ldots, W_{i,n}$ where each block is of size m. We let $W_{i,j} = A_j$ if $j \in S_i$, and otherwise $W_{i,j} = 0^m$. In other words, W_i reveals the strings A_j for all $j \in S_i$; think of W_i as a randomized encoding of the set S_i. See Figure 2 for an illustration of there strings.
- Let $\lambda = 4 \log r + 4 \log t(nm)$. For each $i \in [r]$, we sample a "key" $k_i \in \{0,1\}^\lambda$ for W_i. For simplicity, we assume that the sampled keys are distinct with each other. (If this is not the case, the reduction just aborts; since this happens only with negligible probability we may ignore this event in the analysis.)
- We are finally ready to describe the "auxiliary input" z. The idea is to hide the gadgets $\{W_i\}$ in z at random locations specified by the keys so as to ensure that the only way for a t-time bounded program to recover W_i is to essentially hard-code the key k_i as part of its description. In more detail, we consider a string z of length $2^\lambda \times n \times m$; partition z

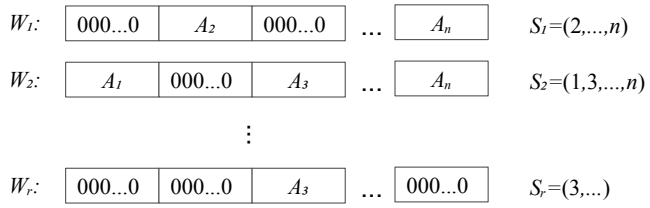


Figure 2 An illustrative example for the gadget strings W_i. Note that if we have a Set Cover $(i_1, i_2, \ldots, i_\ell)$, then the bitwise OR of the strings $W_{i_1}, W_{i_1}, \ldots W_{i_\ell}$ equals A.

into 2^λ blocks $z_{0^\lambda}, z_{0^{\lambda-1}1}, \ldots, z_{1^{\lambda-1}0}, z_{1^\lambda}$ where for all $p \in \{0,1\}^\lambda$, $|z_p| = n \times m$. For all $p \in \{0,1\}^{2^\lambda}$, let $z_p = W_i$ if $p = k_i$ for some $i \in [r]$, and otherwise, let $z_p = 0^{n \times m}$. See Figure 3 for an illustration of these strings.

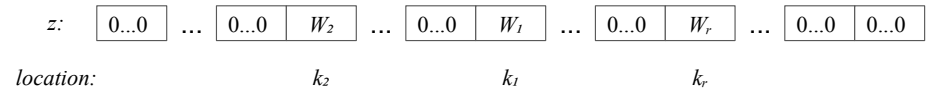


Figure 3 An illustrative example of the "auxiliary" input z.

Finally, the reduction will output YES if $K^t(A \mid z) \leq 2\lambda\ell$. Note that the length of z is upper bounded by $\zeta(|A|)$, and thus this is a syntactically valid reduction to an $\mathsf{McK}^t\mathsf{P}[\zeta]$ instance.

We turn to analyzing the success probability of the reduction.

3.2 Analyzing the Reduction

We will prove that the above reduction gives us a 4-approximation of the γ-Bounded Set Cover Problem. We first show that if $[n]$ can be covered by a small number ($\leq \ell$) of sets, the time-bounded Kolmogorov complexity of A conditioned on the string z will be small ($\leq 2\lambda\ell$): the program computing A simply needs to hard-code the keys k_i corresponding to the ℓ sets in the set cover; it can look into z at the positions specified by the keys and output the bitwise OR of the content of those positions.

▶ **Proposition 3.3.** *If* $\mathsf{cover}([n], \mathcal{S}) \leq \ell$ *then* $K^t(A \mid z) \leq K^{t'}(A \mid z) \leq 2\lambda\ell$ *where* $t'(n) = n^2$.

Proof. Let $S_{i_1}, S_{i_2}, \ldots, S_{i_\ell}$ be the ℓ sets in $\mathcal{S}$ that cover $[n]$. (Since the sets are γ-Bounded, it follows that $\ell \geq n/\gamma$.) Let Π be a RAM program with n, m, λ and the keys $k_{i_1}, \ldots, k_{i_\ell}$ hardwired in it. For each $j \in [\ell]$, Π first reads $W'_{i_j} = z_{k_{i_j}}$ from the k_{i_j}-th block of the string z (where $|z_{k_{i_j}}| = n \times m$). (Recall that z is partitioned into 2^λ blocks and each block is of size $n \times m$.) Π then obtains $W'_{i_1}, \ldots, W'_{i_\ell}$ and Π simply outputs

$$W'_{i_1} \vee W'_{i_2} \vee \ldots \vee W'_{i_\ell}$$

where $\vee$ denotes the bitwise OR for binary strings.

We first show that Π indeed outputs the string A. Note that by the construction of string z, it holds that

$$(W'_{i_1}, \ldots, W'_{i_\ell}) = (z_{k_{i_1}}, \ldots, z_{k_{i_\ell}}) = (W_{i_1}, \ldots, W_{i_\ell}).$$

Recall that in the construction of the gadget string W_{i_j} (for each $j \in [\ell]$), W_{i_j} is partitioned into n blocks $W_{i_j,1}, \ldots, W_{i_j,n}$. And for each block $b \in [n]$, $W_{i_j,b} = A_b$ if $b \in S_{i_j}$, and otherwise $W_{i_j,b} = 0^m$. Since the sets $S_{i_1}, \ldots, S_{i_\ell}$ cover $[n]$, for all $b \in [n]$, there exists an index j such that the b-th block of the gadget string W_{i_j} matches A_b. Thus, $W_{i_1} \vee W_{i_2} \vee \ldots \vee W_{i_\ell} = A$.

We then show that Π can be described within $2\lambda\ell$ bits. Recall that Π contains the values n, m, λ (which takes $O(\log n)$ bits to describe), the keys $k_{i_1}, \ldots, k_{i_\ell}$ (which takes $\lambda\ell$ bits), and the code of Π (which takes $O(1)$ bits). We will provide a more fine-grained analysis in the full version [45] to show that the code of Π is of constant-bit length in the RAM model. Thus, Π can be represented using $\lambda\ell + O(\log n) \leq 2\lambda\ell$ bits.

Finally, note that Π runs in time $O(\ell nm\mathsf{poly}\log n) \leq (nm)^2 = t'(|A|) \leq t(|A|)$ (since in each CPU step, the CPU next-step machine takes $O(\mathsf{poly}\log n)$ time). (We refer the reader to the full version [45] for a more detailed running time analysis.) Thus, we conclude that $K^{t'}(A \mid z) \leq K^t(A \mid z) \leq 2\lambda\ell$. ◀

The key part of the analysis is showing that if $K^t(A \mid z) \leq 2\lambda\ell$ then $\mathsf{cover}([n], \mathcal{S}) \leq 4\ell$:

▶ **Proposition 3.4.** *With probability at least $1 - 2/n$ over the random choice of $k_1, k_2, \ldots, k_r$ (which determines z) and A, it holds that if $K^t(A \mid z) \leq 2\lambda\ell$ then $\mathsf{cover}([n], \mathcal{S}) \leq 4\ell$.*

The proof of Proposition 3.4 is provided in Section 3.3. Proposition 3.3 together with Proposition 3.4 concludes that our reduction achieves a 4-approximation.

3.3 Proof of Proposition 3.4

Let Π be a RAM program such that $|\Pi| \leq 2\lambda\ell$ and $\Pi(z)$ prints A in $\leq t = t(|A|)$ CPU steps (where t is the running time bound associated with the problem $\mathsf{McK}^t\mathsf{P}[\zeta]$). The existence of such Π is implied by the assumption that $K^t(A \mid z) \leq 2\lambda\ell$. We will now show how to use Π, z to extract out a Set Cover of size 4ℓ. Towards this, recall that when executing $\Pi(z)$, in each CPU step, $\Pi(z)$ will read one bit from the memory. Let

$$q_1, q_2, \ldots, q_t$$

be the memory positions that $\Pi(z)$ reads in the execution of $\Pi(z)$ (such that in CPU step i, $\Pi(z)$ reads the content of memory position q_i). Note that the string z will be stored in the memory of $\Pi(z)$, and we are interested in the memory positions where the string z is stored. So, we let d be the memory position such that z is stored from position d to position $d + |z| - 1$. In addition, most of the bits in z are just zeros and $z_{k_1}, z_{k_2}, \ldots, z_{k_r}$ are the only informative blocks. (Recall that z is partitioned into 2^λ blocks of size $n \times m$.) Thus, let

$$p_i = \lfloor (q_i - d)/(n \times m) \rfloor$$

be the index of the block in z from which $\Pi(z)$ reads one bit in CPU step i. When p_i matches some key k_j, $z_{p_i} = z_{k_j} = W_j$. When p_i does not match any of the keys, $z_{p_i} = 0^{n \times m}$.[13]

We say that $\Pi(z)$ makes a useful access to the string z in CPU step i if there exists $j \in [r]$ such that $p_i = k_j$ and for all $i' < i$, $p_i \neq p_{i'}$. In other words, $\Pi(z)$ makes a useful access when it first reads some bit in the block z_{k_j} for some $j \in [r]$. We say that $\Pi(z)$ hits some block z_p if in some CPU step i, $\Pi(z)$ reads one bit from z_p.

[13] Here we discuss the string z constructed by the reduction, instead of the one stored in the memory. (So Π can not manipulate values in z.) Thus, when p_i is out of the range (e.g., $p_i < 0$), it still holds that $z_{p_i} = 0^{n \times m}$.

Bounding the number of useful accesses. We first present an upper-bound on the number of useful accesses. The following central claim shows that if the number of useful accesses is large, then the Kolmogorov complexity of Keys must be small.

▷ Claim 3.5. Let $\mathsf{Keys} = k_1||k_2||\ldots||k_r$ be the concatenation of $k_1, k_2, \ldots, k_r$. If $\Pi(z)$ makes α (or more) useful accesses to the string z, then

$$K(\mathsf{Keys} \mid A, \mathcal{S}) \leq |\Pi| + (r-\alpha)\lambda + \alpha(\log t + \log r) + O(\log n)$$

We defer the proof of Claim 3.5 to Section 3.4

We observe that since Keys are picked at random, their (conditional) Kolmogorov complexity is high.

▷ Claim 3.6. For all $A \in \{0,1\}^{n\times m}$, with probability $1 - 1/n$ (over the random choice of Keys), it holds that

$$K(\mathsf{Keys} \mid A, \mathcal{S}) \geq |\mathsf{Keys}| - \log n = r\lambda - \log n.$$

Proof. Note that the total number of RAM programs with description length $< r\lambda - \log n$ is at most $2^{r\lambda - \log n} \leq \frac{2^{r\lambda}}{n}$, while the total number of the choices of Keys is $2^{r\lambda}$); thus the claim follows. ◁

By combining Claim 3.5 and Claim 3.6 we get the following bound on the number of useful accesses.

▶ **Corollary 3.7.** *With probability* $1 - 1/n$ *over the random choice of* Keys*, if* $|\Pi| \leq 2\lambda\ell$*, it holds that* $\Pi(z)$ *makes at most* 4ℓ useful accesses

Proof. Assume not. Then by Claim 3.5,

$$\begin{aligned}
K(\mathsf{Keys} \mid A, \mathcal{S}) &\leq |\Pi| + (r-4\ell)\lambda + 4\ell(\log t + \log r) + O(\log n) \\
&\leq 2\lambda\ell + r\lambda - 4\lambda\ell + 4\ell(\log t + \log r) + O(\log n) \\
&\leq r\lambda - (2\lambda\ell - 4\ell(\log t + \log r) - O(\log n)) \\
&\leq r\lambda - (2 \cdot 4(\log t + \log r) \cdot \ell - 4\ell(\log t + \log r) - O(\log n)) \\
&\leq r\lambda - (\frac{n}{\gamma} - O(\log n)) \\
&< r\lambda - \log n
\end{aligned}$$

which contradicts Claim 3.6. ◀

Extracting a small Set Cover. We now turn to showing that we can extract a Set Cover from Π, z which is bounded in size by the number of useful accesses. We first show that if $\Pi(z)$ manages to output the string A, yet does not make useful accesses such that the union of all the blocks that are hit by $\Pi(z)$ equal A, then the Kolmogorov complexity of A must be small.

▷ Claim 3.8. Assume that

- $\Pi(z)$ makes α useful accesses;
- $\Pi(z)$ outputs the string A.
- $z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} \neq A$; [14]

[14] When $p_i < 0$ or $p_i \geq 2^\lambda$, we assume that z_{p_i} is an all-zero string and $z_{p_i} = 0^{n\times m}$.

Then,

$$K(A \mid \mathcal{S}) \leq |\Pi| + (n-1)m + \alpha(\log t + \log r) + O(\log n)$$

We defer the proof of Claim 3.8 to Section 3.4.

We observe that since A is a random string, its (conditional) Kolmogorov complexity must be high.

▷ Claim 3.9. With probability $1 - 1/n$ (over the random choice of A), it holds that

$$K(A \mid \mathcal{S}) \geq |A| - \log n \geq nm - \log n.$$

Proof. Note that the total number of RAM programs with description length $< nm - \log n$ is at most $2^{nm - \log n} \leq \frac{2^{nm}}{n}$ (while the total number of the choices of A is 2^{nm}); thus the claim follows. ◁

Combining Claim 3.8 and Claim 3.9, we conclude that the union of all the blocks hit by $\Pi(z)$ must equal A (provided that $\Pi(z)$ prints the string A and makes at most 4ℓ useful accesses).

▶ **Corollary 3.10.** *With probability $1 - 1/n$ over the random choice of A, if $|\Pi| \leq 2\lambda\ell$, $\Pi(z) = A$, and $\Pi(z)$ makes at most 4ℓ useful accesses, it holds that $z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} = A$.*

Proof. Assume not. Then by Claim 3.8,

$$\begin{aligned} K(A \mid \mathcal{S}) &\leq |\Pi| + (n-1)m + 4\ell(\log t + \log r) + O(\log n) \\ &\leq 2\lambda\ell + (n-1)m + 4\ell(\log t + \log r) + O(\log n) \\ &= nm - (m - (2\lambda\ell + 4\ell(\log t + \log r) + O(\log n))) \\ &< nm - \log n \quad (\text{since } m = n^3, \lambda \leq n, \ell \leq n) \end{aligned}$$

which contradicts to Claim 3.9. ◀

We finally show that if the union of all the blocks hit by $\Pi(z)$ matches A, then we can extract out a Set Cover whose size is bounded by the number of useful accesses $\Pi(z)$ made.

▷ Claim 3.11. If $\Pi(z)$ makes at most 4ℓ useful accesses and $z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} = A$, then $\mathsf{cover}([n], \mathcal{S}) \leq 4\ell$.

Proof. Let α be the number of useful accesses made by $\Pi(z)$. Let

$$i_1, i_2, \ldots, i_\alpha$$

be the CPU steps when $\Pi(z)$ makes a useful access; that is, $i_1, i_2, \ldots, i_\alpha$ is a sequence of CPU step indices such that for each $l \in [\alpha]$, $\Pi(z)$ will make a useful access in CPU step i_l. (Recall that except for $z_{k_1}, \ldots, z_{k_r}$, the blocks in the string z are all-zero strings.) Recall that $\Pi(z)$ makes a useful access when it first reads some bit in the block z_{k_j} for some $j \in [r]$.) Thus, by the definition of useful access, it follows that

$$z_{p_{i_1}} \vee z_{p_{i_2}} \vee \ldots \vee z_{p_{i_\alpha}} = z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} = A$$

Since $\Pi(z)$ makes a useful access in CPU step i_l, p_{i_l} must equal some key. We let $j_1, j_2, \ldots, j_\alpha \in [r]$ be a sequence of indices of the keys such that

$$(p_{i_1}, p_{i_2}, \ldots, p_{i_\alpha}) = (k_{j_1}, k_{j_2}, \ldots, k_{j_\alpha})$$

Note that (by the construction of the string z)

$$(W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}) = (z_{p_{i_1}}, z_{p_{i_2}}, \ldots, z_{p_{i_\alpha}})$$

Thus, it follows that

$$W_{j_1} \vee W_{j_2} \vee \ldots \vee W_{j_\alpha} = z_{p_{i_1}} \vee z_{p_{i_2}} \vee \ldots \vee z_{p_{i_\alpha}} = A$$

Finally, we argue that $S_{j_1}, S_{j_2}, \ldots, S_{j_\alpha}$ cover $[n]$, which concludes the proof (since $\alpha \leq 4\ell$). We recall that for each $l \in [\alpha]$, W_{j_l} is the gadget string for S_{j_l}. Furthermore, W_{j_l} is partitioned into n blocks, $W_{j_l,1}, W_{j_l,2}, \ldots, W_{j_l,n}$. For each block $b \in [n]$, $W_{j_l,b} = A_b$ if $b \in S_{j_l}$, and otherwise $W_{j_l,b} = 0^m$. Since $W_{j_1} \vee W_{j_2} \vee \ldots \vee W_{j_\alpha} = A$, it follows that for all blocks $b \in [n]$,

$$W_{j_1,b} \vee W_{j_2,b} \vee \ldots \vee W_{j_\alpha,b} = A_b.$$

Thus, for all $b \in [n]$, there must exist $l \in [\alpha]$ such that $b \in S_{j_l}$. We conclude that the sets $S_{j_1}, S_{j_2}, \ldots, S_{j_\alpha}$ indeed cover $[n]$. ◁

We can now conclude the proof of Proposition 3.4:

Proof of Proposition 3.4. By Corollary 3.7, with probability $1-1/n$, $\Pi(z)$ makes at most 4ℓ useful accesses. By Corollary 3.10, with probability $1-1/n$, it holds that $z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} = A$. Finally by Claim 3.11, it holds that $\mathsf{cover}([n], \mathcal{S}) \leq 4\ell$, which happens with probability at least $1 - 2/n$ (by a union bound). ◀

3.4 Proof of Claim 3.5 and Claim 3.8

In both Claim 3.5 and Claim 3.8, the goal is to compress some strings (either Keys or A) provided that $\Pi(z)$ prints A. Towards doing this, we need be able to find a short representation of the information needed to perform the execution of $\Pi(z)$. Towards this, it will be helpful to track when $\Pi(z)$ makes a useful access. Furthermore, note that every useful access corresponds to some key k_j such that z_{k_j} stores the gadgets W_j of the set S_j. For each such useful access, we will also track this "key index" j. As we shall see, given Π, A and S, as well as the sequence of CPU steps and key indexes (of useful accesses), the whole execution of $\Pi(z)$ can be emulated without having access to z. In fact, as we shall formalize now, we actually do not even need the full content of A and S, but rather just the gadgets W_j corresponding to the sets hit by the useful accesses.

To formalize this, let $t = t(|A|)$ be the maximum number of CPU steps that $\Pi(z)$ can run, and let α be some integer bounded by the number of useful accesses made by $\Pi(z)$. We refer a pair of sequences of CPU steps and key indexes $\omega = ((i_1, i_2, \ldots, i_\alpha), (j_1, j_2, \ldots, j_\alpha)) \in [t]^\alpha \times [r]^\alpha$ as a configuration. We say that $\Pi(z)$ *matches* ω if the first time $\Pi(z)$ makes a useful access is in CPU step i_1 and $\Pi(z)$ reads one bit from the block $z_{k_{j_1}}$ (and recall that $z_{k_{j_1}} = W_{j_1}$), and the second time $\Pi(z)$ makes a useful access is in CPU step i_2 and $\Pi(z)$ reads one bit from the block $z_{k_{j_2}}$, and so on.

▶ **Lemma 3.12.** *Let $\alpha \in \mathbb{N}$, and $\omega = ((i_1, \ldots, i_\alpha), (j_1, \ldots, j_\alpha))$ be a* configuration *in $[t]^\alpha \times [r]^\alpha$. If $\Pi(z)$ matches ω then one can emulate $\Pi(z)$ for i_α CPU steps using the code of Π, the configuration ω, and $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$ (without having access to z).*

Proof. We now describe how to emulate the execution of $\Pi(z)$ for i_α steps using the code of Π, the configuration ω, and $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$. Recall that d is the memory position where z starts at; that is, z is stored in memory positions d to $d + |z| - 1$.

Given the code of Π, we start to emulate $\Pi(z)$ with the content of memory positions $d, d+1, \ldots, d+|z|-1$ (which are supposed to store z) set to 0. In the simulation, we keep track of all memory positions that $\Pi(z)$ has written to. In each CPU step i, if i matches some value in $\{i_1, i_2, \ldots, i_\alpha\}$ (and suppose $i = i_l$), we proceed as follows:

- Let q_i be the memory position which Π will read from in CPU step i and proceed as follows.
- Let $p_i = \lfloor (q_i - d)/(n \times m) \rfloor$. Put the string $W_{j_l} \in \{0,1\}^{n \times m}$ into the memory from position $d + p_i \times nm$ to position $d + p_i \times nm + nm - 1$, with the following exception: If Π has ever previously written into a memory between position $d + p_i \times nm$ and $d + p_i \times nm + nm - 1$, we keep those bits unchanged.
- Finally, let Π will read the bit from the memory (just as if the string z had been there), and we continue to emulate the execution of $\Pi(z)$ in the rest of CPU step i.

If i does not appear in $\{i_1, i_2, \ldots, i_\alpha\}$, we simply emulate the execution honestly. When $i = i_\alpha$, we stop to emulate $\Pi(z)$.

We argue, by induction, that the above procedure perfectly emulates the execution of $\Pi(z)$ in the first i_α CPU steps. For the base case, we consider CPU step $i = 0$, in which $\Pi(z)$ has not started yet, so the statement is trivially true. For any $i \leq i_\alpha$, we now assume that in all the steps $\leq i - 1$, our simulation perfectly emulates $\Pi(z)$, and we will prove that also in CPU step i, the simulation does so as well. First note that if, in CPU step i, Π attempts to read from a memory position q_i that has (1) previously been written or read from, (2) the memory position is not within the range $[d, d+|z|-1]$, or (3) the memory access to q_i is not a useful access, then the induction step directly follows from the induction hypothesis and the fact that the step is performed in exactly the same way in the simulation as in the real execution. We thus only need to consider the case when the memory access to q_i is a useful access. But whenever this happens, by the induction hypothesis, the simulation will produce exactly the same content in the block of z where q_i is contained, as in the real execution of $\Pi(z)$. It thus follows that also this step is perfectly emulated.

Thus, we conclude that $\Pi(z)$ can be emulated for i_α steps using the code of Π, the configuration ω, and $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$. ◀

We are now ready to prove Claim 3.5, which we restate for the convenience of the reader.

▷ Claim 3.13 (Claim 3.5, restated). Let $\mathsf{Keys} = k_1||k_2||\ldots||k_r$ be the concatenation of $k_1, k_2, \ldots, k_r$. If $\Pi(z)$ makes α (or more) useful accesses to the string z, then

$$K(\mathsf{Keys} \mid A, \mathcal{S}) \leq |\Pi| + (r-\alpha)\lambda + \alpha(\log t + \log r) + O(\log n)$$

Proof. If $\Pi(z)$ makes at least α useful accesses, $\Pi(z)$ must match some configuration

$$\omega = ((i_1, i_2, \ldots, i_\alpha), (j_1, j_2, \ldots, j_\alpha))$$

where $\omega \in [t]^\alpha \times [r]^\alpha$. We let $\{j'_1, j'_2, \ldots, j'_{r-\alpha}\} = [r] - \{j_1, j_2, \ldots, j_\alpha\}$ be the set of key indices that do *not* appear in ω.

We consider the following program Π' that prints the string $\mathsf{Keys} = k_1||k_2||\ldots||k_r$ with the string A and the collection of sets $\mathcal{S}$ as auxiliary information. Π' has the values n, m, λ, α, t, r hardwired in it, and the code of Π' also includes the configuration ω, the code of Π, and the $r - \alpha$ keys $k_{j'_1}, k_{j'_2}, \ldots, k_{j'_{r-\alpha}}$. Π' first computes $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$ from A and $\mathcal{S}$. Π' then emulates the execution of $\Pi(z)$ (using the code of Π, the configuration ω, and $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$, using the method described in Lemma 3.12) for i_α CPU steps (recall that i_α is the CPU step when $\Pi(z)$ makes its α'th useful access). Let d be the index such that z is initially stored in the memory from position d to the position $d + |z| - 1$. Let

$$q_1, q_2, \ldots, q_{i_\alpha}$$

be the memory positions that $\Pi(z)$ reads (such that in CPU step i, $\Pi(z)$ reads one bit from memory position q_i) in the first i_α CPU steps. We will decode $k_{j_1}, k_{j_2}, \ldots, k_{j_\alpha}$ from $q_1, q_2, \ldots, q_{i_\alpha}$ as follows: For each $i \leq i_\alpha$, let

$$p_i = \lfloor (q_i - d)/(n \times m) \rfloor$$

Since $\Pi(z)$ matches ω, it follows that

$$p_{i_1} = k_{j_1}, p_{i_2} = k_{j_2}, \ldots, p_{i_\alpha} = k_{j_\alpha}$$

Thus, Π' has access to $k_{j'_1}, k_{j'_2}, \ldots, k_{j'_{r-\alpha}}$ (hardwired) and can compute $k_{j_1}, k_{j_2}, \ldots, k_{j_\alpha}$ as specified above. Thus, Π' can recover and output the string $\mathsf{Key} = k_1 || k_2 || \ldots || k_r$.

Finally, we show that the description length of Π' is at most $|\Pi| + (r-\alpha)\lambda + \alpha(\log t + \log r) + O(\log n)$. To describe Π', we require:

- $|\Pi|$ bits to store the code of Π;
- $(r-\alpha)\lambda$ bits to store the $r-\alpha$ keys $k_{j'_1}, k_{j'_2}, \ldots, k_{j'_{r-\alpha}}$;
- $\alpha(\log t + \log r)$ bits to store the configuration ω.
- $O(\log n)$ bits to store the values n, m, λ, α, t, r.
- $O(1)$ bits to describe the CPU next-step machine.

Thus, the description length of Π' is at most $|\Pi| + (r-\alpha)\lambda + \alpha(\log t + \log r) + O(\log n)$, and from this we conclude that

$$K(\mathsf{Keys} \mid A, \mathcal{S}) \leq |\Pi| + (r-\alpha)\lambda + \alpha(\log t + \log r) + O(\log n)$$

which completes the proof. ◁

We next proceed to prove Claim 3.8, which we first restate:

▷ Claim 3.14 (Claim 3.8, restated). Assume that

- $\Pi(z)$ makes α useful accesses;
- $\Pi(z)$ outputs the string A.
- $z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} \neq A$; [15]

Then,

$$K(A \mid \mathcal{S}) \leq |\Pi| + (n-1)m + \alpha(\log t + \log r) + O(\log n)$$

Proof. Consider some Π, z satisfying the pre-conditions of the claim. Since $\Pi(z)$ has the property that

$$z_{p_1} \vee z_{p_2} \vee \ldots \vee z_{p_t} \neq A,$$

and recalling that each z_{p_i} is divided n m-size blocks, $z_{p_i,1}, \ldots, z_{p_i,n}$, it follows that there exists a block index $b \in [n]$ such that for each block $z_{p_i} \in \{0,1\}^{n \times m}$ that $\Pi(z)$ reads, $z_{p_i,b} = 0^m$. In addition, note that $\Pi(z)$ makes α useful accesses, so $\Pi(z)$ must match some configuration

$$\omega = ((i_1, i_2, \ldots, i_\alpha), (j_1, j_2, \ldots, j_\alpha))$$

where $\omega \in [t]^\alpha \times [r]^\alpha$. Since $\Pi(z)$ matches ω, we know that

$$(W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}) = (z_{p_{i_1}}, z_{p_{i_2}}, \ldots, z_{p_{i_\alpha}})$$

[15] When $p_i < 0$ or $p_i \geq 2^\lambda$, we assume that z_{p_i} is an all-zero string and $z_{p_i} = 0^{n \times m}$.

Thus,

$$W_{j_1} \vee W_{j_2} \vee \ldots \vee W_{j_\alpha} \neq A$$

It follows that for all $l \in [\alpha]$, $W_{j_l,b} = 0^m$. From this, we can conclude that the gadget strings $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$ can be constructed from $\mathcal{S}$ and all randomized encodings $A_1, \ldots, A_{b-1}, A_{b+1}, \ldots, A_n$ *excluding* A_b.

Based on this observation, let us show how to construct a program Π' that outputs the string A given $\mathcal{S}$ as auxiliary information. The program Π' embeds the values n, m, λ, α, r, t, the value of b, the code of Π, the configuration ω, and strings $A_1, \ldots, A_{b-1}, A_{b+1}, \ldots, A_n$ into its code. Π' first computes $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$ from $A_1, \ldots, A_{b-1}, A_{b+1}, \ldots, A_n$ and $\mathcal{S}$. Π' then simulates the execution of $\Pi(z)$ using the code of Π, the configuration ω, and the gadget strings $W_{j_1}, W_{j_2}, \ldots, W_{j_\alpha}$ (making use of Lemma 3.12), and finally outputs whatever $\Pi(z)$ outputs. Note that since $\Pi(z)$ makes exactly α useful accesses, Π' can emulate $\Pi(z)$ all the way until it terminates. Furthemore, recall that by assumption $\Pi(z)$ outputs A, so Π' will do so as well.

We finally show that the description length of Π' is at most $|\Pi| + (n-1)m + \alpha(\log t + \log r) + O(\log n)$. To see this, note that to specify Π', we require:

- $|\Pi|$ bits to include the code of Π;
- $(n-1)m$ bits to store strings $A_1, \ldots, A_{b-1}, A_{b+1}, \ldots, A_n$;
- $\alpha(\log t + \log r)$ bits to save the configuration ω.
- $O(\log n)$ bits to strore the values n, m, λ, α, r, t, b
- $O(1)$ bits to implement the CPU next-step machine:

Thus, we have that the description length of Π' is at most $|\Pi| + (n-1)m + \alpha(\log t + \log r) + O(\log n)$. From this we conclude that

$$K(A \mid \mathcal{S}) \leq |\Pi| + (n-1)m + \alpha(\log t + \log r) + O(\log n).$$

which proves the claim. ◁

References

1. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996. doi:10.1145/237814.237838.
2. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. 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 284–293. ACM, 1997. doi:10.1145/258533.258604.
3. Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *STOC '06*, pages 701–710, 2006. doi:10.1145/1132516.1132614.
4. Eric Allender, Harry Buhrman, Michal Kouckỳ, Dieter Van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.
5. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021. Revision 2; October 19, 2021.
6. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021. Revision 1; April 18, 2021.
7. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021.

8 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.

9 Manuel Blum. Coin flipping by telephone - A protocol for solving impossible problems. In *COMPCON'82, Digest of Papers, Twenty-Fourth IEEE Computer Society International Conference, San Francisco, California, USA, February 22-25, 1982*, pages 133–137. IEEE Computer Society, 1982.

10 Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

11 Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on NP-hardness. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015.

12 Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for np problems. In *FOCS '03*, pages 308–317, 2003.

13 Gilles Brassard. Relativized cryptography. *IEEE Transactions on Information Theory*, 29(6):877–893, 1983.

14 Gregory J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers. *J. ACM*, 16(3):407–422, 1969.

15 Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

16 Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pages 416–426, 1990. doi:10.1145/100216.100272.

17 Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 305–313. IEEE Computer Society, 2000.

18 Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electronic Colloquium on Computational Complexity (ECCC)*, 2022. URL: https://eccc.weizmann.ac.il/report/2022/007.

19 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.

20 Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

21 S. Dov Gordon, Hoeteck Wee, David Xiao, and Arkady Yerukhimovich. On the round complexity of zero-knowledge proofs based on one-way permutations. In *LATINCRYPT*, pages 189–204, 2010.

22 Iftach Haitner, Mohammad Mahmoody, and David Xiao. A new sampling protocol and applications to basing cryptographic primitives on the hardness of NP. In *IEEE Conference on Computational Complexity*, pages 76–87, 2010.

23 J. Hartmanis. Generalized kolmogorov complexity and the structure of feasible computations. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 439–445, November 1983. doi:10.1109/SFCS.1983.21.

24 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.

25 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 247–258, 2018.

26 Shuichi Hirahara. Unexpected hardness results for kolmogorov complexity under uniform reductions. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1038–1051. ACM, 2020.

27 Shuichi Hirahara. Symmetry of information in heuristica. Manuscript, 2021.

28 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD 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 5:1–5:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CCC.2018.5.

29 Rahul Ilango. $AC^0[p]$ lower bounds and NP-hardness for variants of MCSP. *Electron. Colloquium Comput. Complex.*, 26:21, 2019. URL: https://eccc.weizmann.ac.il/report/2019/021.

30 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and AC^0[p]. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, pages 34:1–34:26, 2020.

31 Rahul Ilango. Personal communication, 2021.

32 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference, CCC 2020*, pages 22:1–22:36, 2020.

33 Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory '95*, pages 134–147, 1995.

34 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989.

35 Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 236–241. IEEE Computer Society, 1989.

36 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79, 2000.

37 Ker-I Ko. On the notion of infinite pseudorandom sequences. *Theor. Comput. Sci.*, 48(3):9–33, 1986. doi:10.1016/0304-3975(86)90081-2.

38 Ker-I Ko. On the complexity of learning minimum time-bounded turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

39 A. N. Kolmogorov. Several theorems about algorithmic entropy and algorithmic amount of information (a talk at a moscow math. soc. meeting 10/31/67). *An abstract in Usp. Mat. Nauk*, 23(2):201, 1968.

40 A. N. Kolmogorov. Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968.

41 L. A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, 2003. doi:10.1023/A:1023634616182.

42 Leonid A. Levin. Universal search problems (russian), translated into English by BA Trakhtenbrot in [59]. *Problems of Information Transmission*, 9(3):265–266, 1973.

43 Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020.

44 Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. Cryptology ePrint Archive, Report 2021/513, 2021. ; received on April 19, 2021. URL: https://ia.cr/2021/513.

45 Yanyi Liu and Rafael Pass. On one-way functions from np-complete problems. *Electron. Colloquium Comput. Complex.*, 28:59, 2021. URL: https://eccc.weizmann.ac.il/report/2021/059.

46 Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on EXP $\neq$ BPP. In *CRYPTO*, 2021.

47 Noam Livne. On the construction of one-way functions from average case hardness. In *ICS*, pages 301–309. Citeseer, 2010.

48 Luc Longpré and Sarah Mocas. Symmetry of information and one-way functions. In Wen-Lian Hsu and Richard C. T. Lee, editors, *ISA '91 Algorithms, 2nd International Symposium on Algorithms, Taipei, Republic of China, December 16-18, 1991, Proceedings*, volume 557 of *Lecture Notes in Computer Science*, pages 308–315. Springer, 1991. doi:10.1007/3-540-54945-5_75.

49 Luc Longpré and Osamu Watanabe. On symmetry of information and polynomial time invertibility. In *Algorithms and Computation*, pages 410–419, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

50 R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

51 Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

52 A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *In Cryptology and Computational Number Theory*, pages 75–88. A.M.S, 1990.

53 Rafael Pass. Parallel repetition of zero-knowledge proofs and the possibility of basing cryptography on NP-hardness. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 96–110. IEEE Computer Society, 2006.

54 Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. *Electron. Colloquium Comput. Complex.*, 28:57, 2021. URL: https://eccc.weizmann.ac.il/report/2021/057.

55 Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983. doi:10.1145/357980.358017.

56 John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

57 Michael Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 330–335. ACM, 1983.

58 R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964. doi:10.1016/S0019-9958(64)90223-2.

59 Boris A Trakhtenbrot. A survey of Russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

60 Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. 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 453–461. ACM, 2001. doi:10.1145/380752.380839.

61 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

62 A. K. Zvonkin and L. A. Levin. the Complexity of Finite Objects and the Development of the Concepts of Information and Randomness by Means of the Theory of Algorithms. *Russian Mathematical Surveys*, 25(6):83–124, December 1970. doi:10.1070/RM1970v025n06ABEH001269.

Derandomization from Time-Space Tradeoffs

Oliver Korten
Columbia University, New York, NY, USA

Abstract

A recurring challenge in the theory of pseudorandomness and circuit complexity is the explicit construction of "incompressible strings," i.e. finite objects which lack a specific type of structure or simplicity. In most cases, there is an associated **NP** search problem which we call the "compression problem," where we are given a candidate object and must either find a compressed/structured representation of it or determine that none exist. For a particular notion of compressibility, a natural question is whether an efficient algorithm for the compression problem would aide us in the construction of incompressible objects. Consider the following two instances of this question:

1. Does an efficient algorithm for circuit minimization imply efficient constructions of hard truth tables?
2. Does an efficient algorithm for factoring integers imply efficient constructions of large prime numbers?

In this work, we connect these kinds of questions to the long-standing challenge of proving time-space tradeoffs for Turing machines, and proving stronger separations between the RAM and 1-tape computation models. In particular, one of our main theorems shows that modest time-space tradeoffs for deterministic exponential time, or separations between basic Turing machine memory models, would imply a positive answer to both (1) and (2). These results apply to the derandomization of a wider class of explicit construction problems, where we have some efficient compression scheme that encodes n-bit strings using $< n$ bits, and we aim to construct an n-bit string which cannot be recovered from its encoding.

2012 ACM Subject Classification Theory of computation → Complexity theory and logic

Keywords and phrases Pseudorandomness, circuit complexity, total functions

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.37

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2022/025/`

Funding This research is supported by NSF Grant CCF-1763970.

Acknowledgements The author would like to thank Christos Papadimitriou and Mihalis Yannakakis for their support and guidance throughout the completion of this work.

1 Introduction

Mathematicians have long been familiar with the curious phenomenon of a *non-constructive proof*: an argument which demonstrates the existence of an object satisfying some special property, but which fails to indicate a particular example of such an object. The advent of complexity theory has provided us with a formal way of defining the level of "inherent constructivity" in a theorem: we can say that an existence theorem is constructive if there is an accompanying polynomial time algorithm supplying an example of one of the objects the theorem proves to exist, and is inherently non-constructive if no such algorithm exists. Papadimitriou initiated a formal complexity-theoretic treatment of this topic three decades ago [18], where he provided a taxonomy of total search problems (problems that always have solutions) in **NP** by classifying them based on the strength of the lemma guaranteeing the existence of a solution on all instances.

When a particular theorem appears to be inherently non-constructive, in that no polynomial time algorithm can witness the solutions it guarantees, one perspective we can take is that this theorem is "effectively false" in a certain context, despite being irrefutably true in general. This is essentially the outlook presented by Yao in his seminal paper introducing the basis of theoretical cryptography [24]. Here, Yao focuses on the central tenets of Shannon's information theory. He argues that although Shannon's theorems are of course provably correct, in many scenarios it appears computationally infeasible to witness their truth. For example, an output of some function $f\colon \{0,1\}^n \to \{0,1\}^{2n}$ under the uniform distribution on $\{0,1\}^n$ has entropy n, and thus by a result of Shannon can be encoded on average using n bits. However for an observer who sees only the outputs of this distribution, there seems to be no efficient way to generate such a code without the ability to efficiently invert f. If we posit that there are some specific functions f for which it is in fact impossible to efficiently realize Shannon's theorem in this sense, one might venture to say that Shannon's theorem *effectively fails* for f in the computational realm, perhaps allowing us to carry out tasks which would, in the absence of computational constraints, be deemed impossible. Indeed it is widely conjectured that there are efficiently computable functions f of this form, and this conjecture underpins the security of many cryptographic protocols. Similar situations are abundant in the field of cryptography, where the computational infeasibility of *witnessing* impossibility theorems from information theory allows us to effectively bypass them.

With this perspective in mind, let us now turn our attention to a special family of non-constructive proofs of great interest to complexity theorists, proofs which guarantee the existence of "pseudorandom objects." Key examples include the existence of truth tables of high circuit complexity and of pseudorandom generators capable of derandomizing algorithms. The task of making these proofs constructive is often referred to as an "explicit construction problem," since the goal is to print one explicit example of an object possessing some pseudorandom property. In [14] it is shown that a broad collection of such problems can be reduced to the following more general task: given some efficiently computable function $f\colon \{0,1\}^{n-1} \to \{0,1\}^n$, find an n-bit string outside the range of f. The existence of a solution is guaranteed by the "dual weak pigeonhole principle," and indeed this principle guarantees that a *randomly chosen* string is a solution with high probability. The question at hand is whether this principle is *inherently nonconstructive*. Unlike the case of Shannon's theorems on information transmission, the prevailing wisdom in complexity theory is this theorem *can be made constructive*: it is widely conjectured that exponential time requires exponential circuit size, and by [14], this would in fact imply a generic non-trivial[1] "witnessing" algorithm for the dual weak pigeonhole principle, i.e. an algorithm which produces n-bit strings outside the range of any such f.

In search of evidence for this widely-conjectured belief that the weak pigeonhole principle can be made more constructive, the starting point of this work is to imagine what the computational landscape would look like if it were false, i.e. if we lived in a world where there was an "effective counterexample" to the weak pigeonhole principle. In particular, let us model this scenario by supposing that there are a pair of efficiently computable functions $\mathcal{G} = \{g_n\colon \{0,1\}^n \to \{0,1\}^{n-1}\}_{n\in\mathbb{N}}$, $\mathcal{F} = \{f_n\colon \{0,1\}^{n-1} \to \{0,1\}^n\}_{n\in\mathbb{N}}$, such that no polynomial time algorithm is able to construct an n-bit string x such that $f_n(g_n(x)) \neq x$ in $poly(n)$ time (for more then finitely many n). Note that this is a slightly different version of the weak pigeonhole principle then what we defined in the previous paragraph: here we are given both the length-increasing function f and a supposed inverse g, and the pigeonhole

[1] The implied witnessing algorithm runs in $\mathbf{P}^{\mathbf{NP}}$, whereas the trivial upper bound for this problem is $\mathbf{\Sigma_2^P}$.

principle tells us that $f \circ g$ cannot be the identity. In this hypothetical world where $\mathcal{F}, \mathcal{G}$ are an "effective counterexample" to the weak pigeonhole principle, we have access to an efficient compression scheme which allows us to encode any n-bit string using $n-1$ bits. What improbable feats can be accomplished in such a world?

In this work we give one answer to this question: if the weak pigeonhole principle *effectively fails* in the above sense, we can utilize this failure to construct an efficient data structure which allows us to simulate RAM computations in low space and near-linear time on a 1-tape Turing machine. Stated in contrapositive, mild time-space tradeoffs for simulating RAM machines on a 1-tape Turing machine would imply a generic algorithm to witness the weak pigeonhole principle, and in turn would have significant consequences in the theory of pseudorandomness and circuit complexity.

1.1 Our Contributions

1.1.1 Derandomization from Time-Space Tradeoffs

Let $\mathcal{C}, \mathcal{D} = \{C_n \colon \{0,1\}^n \to \{0,1\}^{n-1}\}_{n\in\mathbb{N}}, \{D_n \colon \{0,1\}^{n-1} \to \{0,1\}^n\}_{n\in\mathbb{N}}$. We call such a pair a "uniform compression scheme," where $\mathcal{C}$ is the "compressor" which encodes n-bit "messages" by $n-1$-bit "codewords," and $\mathcal{D}$ is the "decompressor" which maps $n-1$-bit codewords to the messages they represent[2]. We use the term "uniform" since in the relevant cases, $\mathcal{C}$ and $\mathcal{D}$ will each be described by a single Turing machine which computes C_n (resp. D_n) for all n. We will say that a string $x \in \{0,1\}^n$ is "incompressible" for such a scheme if $D_n(C_n(x)) \neq x$, i.e. if $\mathcal{D}$ cannot be used to recover the message x from the codeword assigned to it by $\mathcal{C}$. In this paper, we study the complexity of the *explicit construction task* of generating incompressible strings for uniform compression schemes. This task can be viewed naturally as a derandomization problem, since a randomly chosen n-bit string will be incompressible with high probability with respect to any fixed compression scheme.

Our results then show how to derive efficient derandomized algorithms for generating incompressible strings, assuming certain *uniform* lower bounds. The lower bounds we consider are time-space tradeoffs for simulating RAM by 1-tape machines. Roughly, they posit that there are problems solvable in time T on a RAM machine which cannot be solved on a 1-tape machine using $T^{0.01}$ space and $T^{1.01}$ time. Depending on certain features of the compression scheme and the resources available to the explicit construction algorithm, the specifics of the tradeoff assumption will vary, as we shall explain in Table 1.

Initially, we consider the case of "poly-time compression schemes", where both $\mathcal{C}$ and $\mathcal{D}$ are computable in polynomial time. We show that for such schemes, incompressible strings can be constructed deterministically in polynomial time assuming time-space tradeoffs for deterministic exponential time:

▶ **Theorem** (Theorem 31). *Suppose there is some exponential time bound $T(n) \geq 2^{\Omega(n)}$ and some $\epsilon > 0$ such that it is impossible to simulate $T(n)$-time RAM computations on 1-tape Turing machines that use $T(n)^{1+\epsilon}$ time and $T(n)^{\epsilon}$ space. Then for any poly-time compression scheme, there is a polynomial time algorithm that produces incompressible n-bit strings for this scheme on input 1^n (for infinitely many n).*

Constructing incompressible strings for poly-time compression schemes is a natural and quite broad derandomization problem. Theorem 31 gives a novel connection between derandomizing this task and proving *uniform* time-space tradeoffs– in contrast, the assumptions

[2] The formal definition given in Section 3 is slightly more general, but we will focus on this special case for now.

previously required to derandomize this sort of problem require a lower bound against *non-uniform* algorithms, such as circuits. To put this derandomization task in a more familiar context, in Section 3.2 we show that several well-studied explicit construction problems, such as the construction of large prime numbers or the construction of truth tables of high circuit complexity/formula size, can be reduced to the problem of finding incompressible strings for some uniform compression scheme. In each case, the compression scheme will either be computable efficiently, or efficiently with access to an oracle for some search problem in **FNP** which is not known to be **NP**-complete.

To state our other main results in their most interesting form, we focus our attention now on the construction of strings/truth tables of high complexity with respect to some non-uniform complexity measure, such as formula size, circuit size, or time-bounded Kolmogorov complexity. In each case there is an associated **NP** search problem, where we are given a string/truth table and must find a small formula/circuit/program computing it (if one exists); we call this the "compression problem." The following table shows how our three main theorems relate various time-space tradeoff hypotheses to such problems:

Table 1 Main Results. We use the shorthand $T = T(n)$.

Hypothesis: For some exponential time bound T and some $\epsilon > 0$...	**Implication** :
$\mathbf{RAM\text{-}TIME}[T] \not\subseteq \mathbf{1\text{-}TISP}[T^{1+\epsilon}, T^{\epsilon}]$	If the compression problem has a polynomial time algorithm, then incompressible strings can be constructed in polynomial time.
$\mathbf{NTIME}[T] \not\subseteq \mathbf{1\text{-}NTISPG}[T^{1+\epsilon}, T^{\epsilon}, T^{\epsilon}]$	Incompressible strings can be constructed in polynomial time with an **NP**-oracle. In particular, $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{size}[2^n/2n]$.
$\mathbf{RAM\text{-}TIME}[T] \not\subseteq \mathbf{1\text{-}NTISPG}[T^{1+\epsilon}, T^{\epsilon}, T^{\epsilon}]$	Incompressible strings can be constructed in polynomial time with an oracle for the compression problem.

The classes seen on the left hand side will be defined formally in Section 2.2, but we give a brief explanation here so that the table can be interpreted appropriately. $\mathbf{TIME}[T(n)]$ and $\mathbf{NTIME}[T(n)]$ are defined in the standard way using multitape machines; the prefixes **1** and **RAM** indicate, respectively, either a restriction of the model to 1-tape machines or a strengthening to random access machines. The class $\mathbf{TISP}[T(n), S(n)]$ consists of problems decidable simultaneously in time $T(n)$ and space $S(n)$. Finally, $\mathbf{NTISPG}[T(n), S(n), G(n)]$ consists of problems decidable by a nondeterministic machine running in time $T(n)$ which, on every computation path, uses at most $S(n)$ space and $G(n)$ nondeterministic guesses.

For a concrete example of how to apply these results, lets focus on the compression problem Circuit Synthesis (given a truth table find a small circuit for it if one exists), and the explicit construction problem of producing truth tables of exponential circuit complexity. The first hypothesis in the above table implies that if Circuit Syntheses is in **P** then there is a polynomial time construction of hard truth tables (i.e. **E** has exponential circuit complexity). The second hypothesis implies unconditionally that there is an efficient **NP** oracle construction of hard truth tables (i.e. $\mathbf{E}^{\mathbf{NP}}$ has exponential circuit complexity). Finally, the third and strongest hypothesis tells us that there is a polynomial time construction of hard truth tables using an oracle for Circuit Synthesis.

The case of large prime construction does not fit in as neatly to the above picture, as the scheme we devise for this problem will require a factoring oracle for both the compressor and decompressor (while the above problems have a polynomial-time computable decompressor). In this case we get the following result:

▶ **Theorem** (Corollary 35). *Under the hypothesis in the top row of the above table, a polynomial time algorithm for factoring implies a polynomial time algorithm to construct* $32n$*-bit primes of magnitude* $> 2^n$ *for infinitely many* n*.*

If we forgo the questionable assumption that factoring lies in **P**, we get:

▶ **Theorem** (Corollary 36). *One of the following is true:*

1. *For every exponential time bound* T *and every* $\epsilon > 0$*, every language decidable in time* $T(n)$ *on a RAM machine can be decided in time* $T(n)^{1+\epsilon}$ *and space* $T(n)^\epsilon$ *by a 1-tape machine with a factoring oracle, which makes oracle calls of length at most* $T(n)^\epsilon$*.*
2. *There is a polynomial time algorithm with a factoring oracle that generates* $32n$*-bit primes of magnitude* $> 2^n$ *for infinitely many* n*.*

The problem of deterministically generating large primes has been investigated previously in several works, and was notably the subject of the Polymath 4 project [20]. In the public discussion forums for this project, it was explicitly asked whether a polyomial time algorithm for factoring, or more generally an oracle for factoring, would help. More recently, Oliviera and Santhanam gave a subexponential time "pseudodeterministic" construction of large primes [17], using only the fact that primality is testable in **P** [1] and that primes occur with non-negligible frequency.

1.1.2 BPP and the Weak Pigeonhole Principle

In Section 6 we briefly consider the relationship between various search problems associated with the weak pigeonhole principle, and the "full derandomization task" characterized by the class **prBPP**. Observe that the problem introduced above of finding incompressibe strings for uniform compression schemes can be generalized to a **TFNP** search problem – instead of considering compression schemes generated by uniform Turing machines, we can consider the search problem where a compression scheme of some fixed length is given as input in the form of a pair of boolean circuits:

▶ **Definition 1.** *In* Lossy Code*, we are given as input a pair of circuits* $C\colon \{0,1\}^n \to \{0,1\}^{n-1}$*,* $D\colon \{0,1\}^{n-1} \to \{0,1\}^n$*, and must output some* $x \in \{0,1\}^n$ *such that* $D(C(x)) \neq x$*.*

While finding a deterministic algorithm for Lossy Code appears to be a quite generic derandomization problem, it would be a major breakthrough to show that Lossy Code captures the "full derandomization problem:" since Lossy Code lies in **TFNP**, if **prBPP** reduces to Lossy Code then **BPP** $\subseteq$ **NP**, which is a notorious open problem. In Section 6, we show that a natural generalization of Lossy Code, where we allow the compressor C to be randomized, is indeed strong enough to characterize **prBPP** precisely. The formal problem is as follows:

▶ **Definition 2.** *In* R-Lossy Code*, we are given as input circuits* $C\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^{n-1}$*,* $D\colon \{0,1\}^{n-1} \to \{0,1\}^n$*, and must find some* x *such that* $Pr_r[D(C(x,r)) = x] < \frac{1}{2}$*.*

In this work we demonstrate:

▶ **Theorem** (Theorem 40). R-Lossy Code *is complete for* **prBPP** *under deterministic Turing reductions.*

The relation between pigeonhole principle search problems and more standard derandomization problems is summarized in Figure 1.

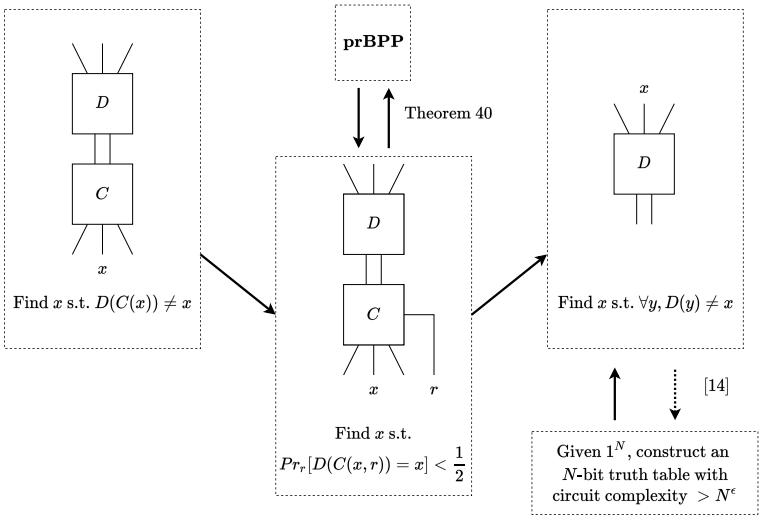


Figure 1 Relations between the total search problems associated with various weak pigeonhole principles, and standard derandomization problems. Solid arrows represent deterministic polynomial time reductions, while dotted arrows represent **NP**-oracle reductions.

The proof of Theorem 40 is mostly standard, following quite directly from Yao's next bit predictor lemma. From the proof of this theorem we are able to extract the following interesting corollary:

▶ **Corollary** (Corollary 41, Informal). *If the fixing of the leftover bits in Yao's hybrid argument can be derandomized, then* **BPP** $\subseteq$ **NP**. *Indeed, under this assumption, every problem in* **prBPP** *reduces to the search problem* Lossy Code *in* **PPP** $\subseteq$ **TFNP**.

In other words, derandomizing a particular step in Yao's classical argument implies a quite universal derandomization of **prBPP**.

1.2 Relation to Prior Work

Hardness vs. Randomness

As mentioned above, the task of constructing incompressible strings can be naturally viewed as a derandomization problem, and is thus amenable to the standard hardness/randomness paradigm. In particular, when the compression scheme is polynomial time computable as in Theorem 31, the standard hardness assumptions used to derandomize **BPP** (e.g. [8]) would suffice to yield a polynomial time construction of incompressible strings. Such hardness assumptions require a lower bound for a language in **E** against some non-uniform model of computation, such as boolean circuits. In contrast, Theorem 31 gives a hardness/randomness connection for this problem which only requires a *uniform lower bound* for a a language in **E**, in particular a lower bound against low-space algorithms running in slightly more time using a weaker memory model.

The Easy Witness Lemma

Perhaps the closest prior work to our results is the "Easy Witness Method," initiated by [11] and furthered in [7] and [23], which roughly says that assuming $\mathbf{E}^{\mathbf{NP}}$ has small circuits, any nondeterministic exponential time computation must have witnesses of low circuit complexity. This immediately implies that unless $\mathbf{E}^{\mathbf{NP}}$ requires large circuits, we can efficiently simulate $\mathbf{NTIME}[2^n]$ using limited nondeterminism by "guessing a small circuit." In the full version, we show that this old method (along with one other well-known tool) is in fact enough to prove the second implication in Table 1, although this connection to time-space tradeoffs does not seem to have been noted previously. However, this proof heavily utilizes the distinctive power of nondeterminism, whereby additional "guesses" allow us to vastly simplify the verification procedure, and does not seem to extend to our other two main results.

1.3 Known Time-Space tradeoffs

Finally, we cover the known results on time-space tradeoffs and separations between the RAM and 1-tape models. We first emphasize the following: all three time-space tradeoff hypotheses stated on the left-hand side of Table 1 are known to hold *unconditionally* when the time bound T is $O(n)$. In particular, an old result of Maass [15] shows that the set of palindromes is recognizable in quasilinear time on a deterministic RAM machine, but requires $\Omega(n^2)$ time on a nondeterministic 1-tape machine. However, Maass's proof is essentially a counting argument which crucially relies on the entropy of the input being comparable in magnitude to the total computation time, which is no longer the case for exponential time computations. When it comes to separating the RAM and 1-tape models for generic time bounds, a result of [21] shows that there must be *some* slowdown when simulating a RAM on a 1-tape machine for *any time bound* greater then $n \log \log n$, but the slow-down is an astronomically small multiplicative factor, on the order of $\log \log T(n)$.

Another sequence of works ([2], [3], [22] among others) has shown unconditionally that nondeterministic linear time cannot be solved in $\mathbf{TISP}[n^{1+\epsilon}, n^{\epsilon}]$ for certain fixed values of $\epsilon > 0$, even when the simulating machine is given access to RAM. These results can be scaled to larger time bounds as follows: for any time-constructible T, $\mathbf{\Sigma_2 TIME}[T(n)]$ is not contained in $\mathbf{TISP}[T(n)^{1+\epsilon}, T(n)^{\epsilon}]$ for certain fixed values of $\epsilon > 0$. Such results are obtained by combining hierarchy theorems with a fast simulation result, whereby a $\mathbf{\Sigma_2}$ machine can simulate a $\mathbf{TISP}[T(n)^{1+\epsilon}, T(n)^{\epsilon}]$ computation in time $T(n)^{1-\delta}$ time for some fixed $\epsilon, \delta > 0$. It is evident that such results will not be sufficient for our purposes, as they only rule out efficient low-space simulations of $\mathbf{\Sigma_2}$ computations by deterministic machines.

1.4 Main Tool: the J-tree

The basis of our main proofs is a construction which we call the "J-tree." Roughly speaking, the J-tree is a data structure which allows us to store an array of T elements, subject to fast update/query operations, using significantly less then T bits. While this task is information-theoretically impossible (by the weak pigeonhole principle), the J-tree uses a compression scheme C, D to perform its operations, and whenever it fails in its role as a data structure, it is able to print out an incompressible string for this scheme. Thus, assuming no polynomial time algorithm can witness the weak pigeonhole principle for C, D, no polynomial time algorithm can witness the failure of this data structure. This will allow our low-space simulations to operate a RAM memory using low space on a 1-tape machine, in such a way that if the simulation fails, then a polynomial time algorithm can witness this failure, and thus witness the weak pigeonhole principle for C, D. The core construction underlying the

J-tree has a long and intriguing history, occurring in some form or another in [16, 5, 19, 9], and [14]. The details of this history and its relation to our results are explained in the full version; our main novel contribution to this line of work is the "J-tree Update Lemma" (Lemma 25).

2 Preliminaries

2.1 Basics

Our notation for basic concepts is standard, e.g. all logarithms are base 2, we use $[n]$ to denote $\{1, \ldots, n\}$, $|x|$ to denote length of a binary string. We will define the following precise notion of an "exponential time bound:"

▶ **Definition 3.** *We say a function $T\colon \mathbb{N} \to \mathbb{N}$ is an "exponential time bound" if T is time constructible, and there exist constants $0 < \beta < B$ such that for sufficiently large n, $2^{\beta n} \leq T(n) \leq 2^{Bn}$.*

This is in contrast to the more general notion of exponential growth where we have $2^{n^\beta} \leq T(n) \leq 2^{n^B}$.

2.2 Machine Models

We start by precisely defining the various machine models and complexity classes that will be used. We begin with a definition of "random access memory" or "RAM" machines:

▶ **Definition 4** (RAM machines)**.** *A RAM machine is a turing machine equipped with two binary tapes, one called the "auxilliary tape" and the other called the "addressing tape." We collectively refer to these as the "linear tapes." Both linear tapes admit the same operations as a standard Turing machine tape, where in one step we can move the head left/right and read or modify a cell of the tape. In addition, there is an associated "RAM memory" consisting of a sequence of binary variables $A_1, A_2, \ldots$. The addressing tape has two additional operations that allow it to interact with the RAM memory. There is an* Update *operation, which in one step sets $A_i = b$, where i is the integer whose binary representation lies to the left of the addressing tapehead, and b is the bit currently read by the auxiliary tape. There is also a* Load *operation, which in one step sets the cell pointed to by the auxiliary tapehead to the value A_i, where again i is the integer whose binary representation lies to the left of the addressing tapehead. At the start of the computation on input x, the RAM cells $A_1, \ldots, A_{|x|}$ are initialized to the bits of x in order, and $A_{|x|+1}, \ldots$ are initialized to 0. The addressing tape is then initialized to $|x|$, so that the machine knows where the input ends.*

We will use the following simplification lemma for RAM computations later, which follows easily from the use of balanced binary search trees:

▶ **Lemma 5** ([6])**.** *A RAM machine running in time $T(n)$ can be simulated in time $T'(n) = \tilde{O}(T(n))$ by a RAM machine that uses $O(\log T'(n))$ space on its addressing and auxiliary tapes, and uses only its first $T'(n)$ RAM cells.*

We will also make reference to k-tape Turing machines, for which we omit a formal definition as this model is standard. However, we will at some points concern ourselves with multi-tape machines equipped with oracles for languages/functions, and here we will need to be precise about the oracle access mechanism:

▶ **Definition 6** (Oracle Turing Machines). *For a function $F\colon \{0,1\}^* \to \{0,1\}^*$, an F-oracle k-tape Turing machine has k standard read/write "work tapes," in addition to an oracle tape where in one step the machine can replace the leading cells of the oracle tape with $F(x)$, where x is the string lying to the left of the oracle tapehead prior to the oracle call; after this the oracle tapehead is moved to the first cell of the oracle tape. In some cases we will give a machine access to several oracles $F_1, F_2, \ldots, F_k$ (for a fixed constant k), in which case each gets its own oracle tape. The oracle tape(s) and the first work tape share a single tapehead, and in one move this tapehead can move from the first cell of the work tape to the first cell of one of the oracle tapes, or vice-versa.*

The space usage of an oracle k-tape machine is defined to be the sum of the space used on all of its work tapes and oracle tapes.

The sharing of a single tapehead between the oracle tapes and the first work tape is only relevant for 1-tape machines, where we don't want the machine to "cheat" and use it's oracle tapes as additional work tapes.

▶ **Definition 7** (Time Bounded Classes). *For a function $T\colon \mathbb{N} \to \mathbb{N}$, let $\mathbf{TIME}[T(n)]$ denote the class of languages which are decidable by a deterministic multi-tape Turing machine running in time $O(T(n))$. Let $\mathbf{RAM\text{-}TIME}[T(n)]$ be defined analogously for RAM machines.*

▶ **Definition 8** (Time-Space Classes). *For functions $T, S\colon \mathbb{N} \to \mathbb{N}$, let $\mathbf{TISP}[T(n), S(n)]$ denote the class of languages which are decidable by a deterministic multi-tape Turing machine running simultaneously in time $O(T(n))$ and total space $O(S(n))$. Let $\mathbf{1\text{-}TISP}[T(n), S(n)]$ be defined identically, but with the additional restriction that the machine uses only one tape.*

Finally, we define an analogous "time-space" class for nondeterministic time:

▶ **Definition 9** (Nondeterministic Time-Space Classes). *For functions $T, S, G\colon \mathbb{N} \to \mathbb{N}$, let $\mathbf{NTISPG}[T(n), S(n), G(n)]$ ("nondeterministic time, space, guess") denote the class of languages decidable by a nondeterministic multi-tape Turing machine such that on any computation path on an input of length n, the machine spends time $O(T(n))$, uses space at most $O(S(n))$, and makes at most $O(G(n))$ non-deterministic guesses. Let $\mathbf{1\text{-}NTISPG}[T(n), S(n), G(n)]$ be defined identically, but with the additional restriction that the machine uses only one tape. We define $\mathbf{NTIME}[T(n)]$ in the standard way, which is analogous to the above but with no restriction on space usage or nondeterminism.*

We make note of the following result, which tells us that for nondeterministic computations the multi-tape and RAM models are roughly equivalent:

▶ **Lemma 10** ([6]). *Any time-$T(n)$ computation on a nondeterministic RAM machine can be simulated in $\tilde{O}(T(n))$ time on a nondeterministic multi-tape machine.*

We will thus not bother to explicitly define a nondeterministic RAM model, though the definition for the deterministic case naturally extends.

3 Pigeonhole Principles and Compression Schemes

We now define the basic formalization of the weak pigeonhole principle we will investigate in this work, and introduce the relevant terminology.

3.1 Compressors and Decompressors

▶ **Definition 11.** *Let $D\colon \{0,1\}^m \to \{0,1\}^n$, with $m < n$. We call such a map which extends its input length a "decompresser." The "code length" of this decompressor is m, and its "message length" is n.*

The terminology should be interpreted as follows: for a string x such that $D(y) = x$, y functions as an m-bit compressed representation (or "codeword") for the n-bit "message" x, and D functions as an an algorithm which lets us "decompress" this codeword into the message it represents.

By the *dual weak pigeonhole principle*, if $n > m$, any function mapping 2^m "pigeons" to 2^n "holes" must leave some hole empty. We thus know there must exist an $x \in \{0,1\}^n$ such that $\forall y \in \{0,1\}^m$, $D(y) \neq x$. We call such a string x an "empty pigeonhole" for the decompressor D.
The primary subject of [14] was the problem EMPTY, originally introduced in [13], where we are given a decompressor specified by a boolean circuit and must output one of its empty pigeonholes. In this work, we will study a slight modification of this problem, where we are given both a compressor and a decompressor, and must and must find a witness to the fact that some message is not recoverable from its codeword. This was referred to by Jeřábek as the "retractive pigeonhole principle."[10].

▶ **Definition 12.** *Let $C\colon \{0,1\}^n \to \{0,1\}^m, D\colon \{0,1\}^m \to \{0,1\}^n$, where $m < n$; we call such a C a "compressor", and collectively we will refer to C, D as a "compression scheme." Again, m and n are the "code length" and "message length" respectively. By the pigeonhole principle, we know that there must exist some $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$; we call such an x "incompressible" with respect to the scheme C, D.*

Note that when $x \in \{0,1\}^n$ is an empty pigeonhole for a decompressor D, it is necessarily incompressible for *all schemes* C, D which use D as the decompressor. The converse is not true in general, but when it is we use the following terminology:

▶ **Definition 13.** *We call C a "proper compressor" for D if C, D satisfy the following for all $x \in \{0,1\}^n$: if there exists an $y \in \{0,1\}^m$ such that $D(y) = x$, then $D(C(x)) = x$.*

By definition, when C is a proper compressor for D, the incompressible strings for the scheme C, D correspond exactly to the empty pigeonholes for D. It should be noted that when D is polynomial time computable, or is specified by some boolean circuit, there always exists a proper compressor for D computable efficiently with an **NP** oracle, which simply finds the lexicographically first preimage of a string under D or else outputs something arbitrary when none exist. In this sense, the work of [14], which studies the complexity of EMPTY with respect to **NP**-oracle reductions, was essentially studying a special case of this problem, where the compressor is the "canonical proper compressor" which searches for the lexicographically first preimage.

For most of this work it will be convenient to focus on the special case of functions which exactly double their input length:

▶ **Definition 14.** *When a compression scheme has code length n and message length $2n$, we will say that it has "stretch 2."*

We will utilize the following lemma, which tells us that if our goal is to find an incompressible string with respect to some scheme, we can assume it has stretch 2 without loss of generality:

▶ **Lemma 15.** *Let C, D be a compression scheme with code length n and message length m. Then there is another scheme C', D' of code length n and message length $2n$, such that C' is computable in poly(m) time with an oracle for C, D' is computable in poly(m) time with an oracle for D, and such that given an incompressible string for C', D', we can construct an incompressible string for C, D in poly(m) time with oracles for C, D.*

Proof. This is proven in [14] for the special case when C is a proper compressor for D, but the exact same proof extends to the more general case stated here. ◀

In [14], the problem EMPTY was studied as a search problem, where the decompressor D is provided as input (in the form of a circuit), and the goal is to find one of its empty pigeonholes. We could equivalently define such a search problem for the retractive pigeonhole principle, where we are given circuits computing C, D as input, and must output an incompressible string for this scheme (this search problem is considered in Section 6). However, for the purposes of our main results it will be more suitable to study a *uniform* version of this problem, where we have a sequence of schemes C_n, D_n of increasing size, and each are computable within some uniform complexity class.

▶ **Definition 16.** *A uniform compression scheme is a pair $\mathcal{C} = \{C_n \colon \{0,1\}^{\ell(n)} \to \{0,1\}^{m(n)}\}_{n\in\mathbb{N}}$, $\mathcal{D} = \{D_n \colon \{0,1\}^{m(n)} \to \{0,1\}^{\ell(n)}\}_{n\in\mathbb{N}}$ for some pair of time constructible functions $m, l \colon \mathbb{N} \to \mathbb{N}$ such that $m(n) < \ell(n)$ for all n, and $\ell(n)$ is bounded above by a polynomial in n.*

In this work we will concern ourselves with uniform compression schemes where $\mathcal{C}, \mathcal{D}$ are computable in polynomial time, each with access to different oracles, hence the name "uniform."

3.2 Particular Compression Schemes of Interest

We now introduce some uniform compression schemes whose incompressible strings have certain desirable properties for which no explicit constructions are currently known.

3.2.1 Compression Schemes for Non-Uniform Complexity Measures

We start with the case of hard truth tables. The following lemma is a well-known folklore result, for which a formal proof can be found in [14]:

▶ **Lemma 17.** *For sufficiently large $N \in \mathbb{N}$, there is function $\mathsf{f}_N \colon \{0,1\}^{N-1} \to \{0,1\}^N$ computable uniformly in $\tilde{O}(N^2)$ time such that if $x \in \{0,1\}^N$ has circuits of size at most $\frac{N}{2\log N}$, then x is in the range of f_N.*

We now define the search problem associated with finding preimages of strings under f_N, which has commonly been referred to as the "circuit synthesis problem:"

▶ **Definition 18** (Circuit Synthesis)**.** *Given a string $x \in \{0,1\}^N$, output an $N-1$ bit description of a circuit C of size at most $\frac{N}{2\log N}$ on $\lceil \log N \rceil$ variables such that $C(i) = x_i$ for all $0 \leq i < N$ if such a circuit exists, or else determine that no such circuit exists.*

This problem is the search variant of the more well-studied "Minimum Circuit Size Problem" [12], which is not known to admit a search-to-decision reduction. By the same arguments underlying Lemma 17, given an instance $x \in \{0,1\}^N$ of Circuit Synthesis, any circuit of the stated size can be represented using at most $N-1$ bits via a standard encoding so this search problem is well defined. Further, requiring the output to be specified in such an encoding does not increase the complexity of the problem, since the standard encoding can be computed efficiently from any description of such a circuit.

By definition, we see that there is a proper compressor for f_N computable in polynomial time with a Circuit Synthesis oracle[3], and thus any incompressible string for this scheme is an N-bit string with high circuit complexity.

We similarly have the following:

▶ **Lemma 19.** *For any fixed polynomial p, there is a uniform compression scheme whose decompressor is computable in polynomial time, and whose compressor is computable with an oracle for $K^{p(n)}$* Minimization *(given $x \in \{0,1\}^n$ find a short program $y \in \{0,1\}^{n-2}$ that prints x in $p(n)$ steps if one exists). The incompressible strings for this scheme have $K^{p(n)}$ complexity $\geq n-1$.*

▶ **Lemma 20.** *There is a uniform compression scheme whose decompressor is computable in polynomial time, and whose compressor is computable with an oracle for* Formula Synthesis *(given a truth table find a short formula if one exists). The incompressible strings for this scheme are truth tables of exponential formula size.*

We provide these as basic examples without defining the problems too formally; more generally it can be verified that for most reasonable non-uniform measures of complexity (which are bounded above by K^{poly}), such a uniform scheme exists whose decompressor is computable efficiently, whose compressor is computable with access to the relevant "compression problem," and whose incompressible strings have high complexity with respect to this measure.

3.2.2 Large Primes

We next construct a compression scheme related to the large prime construction problem. This scheme is unlike the previous ones, in that both the compressor and decompressor have the same complexity: each will require an oracle for factoring.

▶ **Theorem 21.** *There is a compression scheme R, P with code length $n + \lceil \log n \rceil + 3$ and message length $n + \lceil \log n \rceil + 4$, such that R, P are each computable uniformly in polynomial time with a factoring oracle, and given an incompressible string for this scheme, a $32n$-bit prime of magnitude $> 2^n$ can be constructed in polynomial time with a factoring oracle.*

A proof of Theorem 21 can be found in the full version. This theorem is an algorithmic analogue of a well known result of Paris, Wilkie, and Woods [19], and our proof follows from analysing the computational resources needed to carry out their construction.

4 J-Trees

In this section we develop the core tool used in our main result, namely the "J-tree." The J-tree is, informally, a data structure "solving" the following information-theoretically impossible task: store an array of T elements, subject to efficient updates and queries, using significantly fewer then T bits. While such a data structure cannot exist unconditionally, the J-tree will take as input a compression scheme C, D, and will be set up so that when it fails in its capacity as a data structure, it prints out an incompressible string for C, D. In other words, if it is hard to witness the weak pigeonhole principle for the scheme C, D, then it is hard to find a sequence of updates/queries which causes our data structure to fail.

[3] There is some subtly when considering oracles for search problems like Circuit Synthesis which have multiple valid solutions on each input, see the full version for a discussion of how our results are robust to this potential issue.

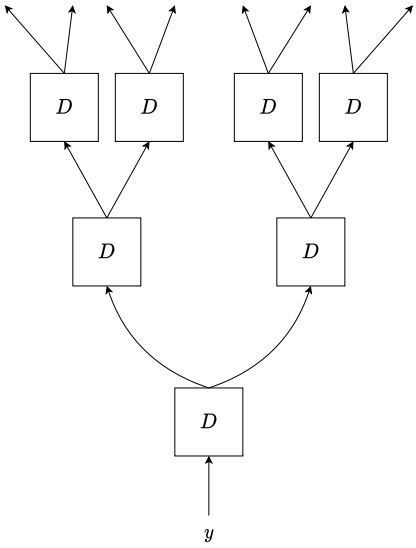


Figure 2 A J-tree of depth 3. Each arrow consists of n wires, where n is the code length of the decompressor D.

▶ **Definition 22** (J-trees). *Let $D\colon \{0,1\}^n \to \{0,1\}^{2n}$ be a decompressor of code length n and stretch 2. We define $D_0, D_1\colon \{0,1\}^n \to \{0,1\}^n$ to be the maps obtained by computing D and taking the first n and last n bits of output respectively. Now, for any binary string $x \in \{0,1\}^*$, we define $D_x\colon \{0,1\}^n \to \{0,1\}^n$ as follows. When $|x| = 0$, D_x is the identity, and when $|x| = 1$, D_0, D_1 are defined as above. In the general case, when $x = b_1 \cdots b_k$ for some $b_1, \ldots, b_k \in \{0,1\}$, D_x is defined as:*

$$D_x = D_{b_k} \circ \cdots \circ D_{b_1}$$

Now, for a particular input $y \in \{0,1\}^n$, we can define the function $D[y]\colon \{0,1\}^* \to \{0,1\}^n$ as follows. For each $x \in \{0,1\}^*$:

$$D[y](x) = D_x(y)$$

For an integer k, we then define $D^k[y]\colon \{0,1\}^k \to \{0,1\}^n$, which is simply the restriction of $D[y]$ to the domain $\{0,1\}^k$.

We will refer to $D^k[y]$ as a "J-tree," D as its decompressor, y as its "seed," and k as its "depth."

It will be useful to visualize a J-tree as a binary tree (hence the name). Refer to Figure 2 which illustrates a J-tree $D^k[y]$ where $k = 3$. For a given D and depth k, we can construct a tree-like circuit which starts with one copy of D, then feeds each of its two n-bit output blocks into another copy of D, and so on for k iterations, ultimately yielding a circuit with n inputs and $2^k n$ outputs which has the structure of a perfect binary tree of depth k. If we now fix the inputs to some value y (the seed), by passing y through this tree-like circuit we obtain 2^k n-bit values at the output. As seen in the figure, each of the 2^k n-bit values is associated uniquely with a leaf in this binary tree of depth k, and thus can be specified by a k-bit index indicating whether to move left or right at each step along a root-to-leaf path. $D^k[y]$ is then the function which, given the description of such a path, returns the value at the corresponding leaf.

We will think of a J-tree $D^k[y]$ operationally as a data structure which stores an array of 2^k n-bit values, one for each for each of its 2^k "leaves", where the i^{th} value is simply $D^k[y](i)$. The state of the data structure is described purely by y and D, which in our use cases will require far fewer bits then storing 2^k n-bit strings explicitly. In the following, we now prove that the J-tree data structure admits fast query and update operations, i.e. operations that allow us to read one entry of the data structure, or update the value of one entry. While the query operation will be computable efficiently by evaluating only the decompressor D $O(k)$ times, the update operation will require evaluating some compressor C, and might "fail" in a certain well-defined sense. However, when the update does not fail, there is a deterministic algorithm which can verify, using $O(k)$ evaluations of D, that the resulting J-tree is in fact the true updated version of the original. We begin with the query or "access" operation:

▶ **Lemma 23** (J-tree Access Lemma). *Let $D^k[y]$ be a J-tree. Then for any $i \in \{0,1\}^k$ we can compute $D^k[y](i)$ in time $O(nk)$ given i, y and using $O(k)$ evaluations of D.*

Proof. This follows directly from the definition of $D^k[y]$. Letting $b_1, \ldots, b_k$ be the individual bits of i, we have that:

$$D^k[y](i) = D_{b_k} \circ D_{b_{k-1}} \circ \cdots \circ D_{b_1}(y)$$

So we can compute $D^k[y](i)$ by evaluating D k times successively starting with the input y. ◀

▶ **Definition 24** (J-tree Modifications). *Let $D^k[y]$, $D^k[y']$ be J-trees of depth k and code length n. We say that the relation* $\mathsf{Modified}(y, y', i, s, D)$ *holds if:*

1. $D^k[y'](i) = s$
2. *For all $i' \in \{0,1\}^k$, $i' \neq i$, $D^k[y'](i') = D^k[y](i')$*

In other words, $\mathsf{Modified}(y, y', i, s, D)$ asserts that the J-tree $D^k[y']$ represents a local modification of the J-tree $D^k[y]$ which changes its i^{th} value to s and leaves all other values the same.

▶ **Lemma 25** (J-tree Update Lemma). *There exist algorithms* Find *and* Verify *satisfying the following.*

Verify *takes as input a decompressor D (specified as an oracle) of code length n and stretch 2, a pair of seeds $y, y' \in \{0,1\}^n$, an index $i \in \{0,1\}^k$, and a value $s \in \{0,1\}^n$, and either accepts or rejects.* Verify *runs in deterministic time $O(nk)$ using $O(k)$ evaluations of D, and satisfies the property that if* $\mathsf{Verify}(y, y', i, s, D)$ *accepts, then* $\mathsf{Modified}(y, y', i, s, D)$ *must hold.*

Find *takes as input a compression scheme C, D of code length n and stretch 2 (again specified as oracles), a seed $y \in \{0,1\}^n$, an index $i \in \{0,1\}^k$, and a value $s \in \{0,1\}^n$. It either "succeedes" and outputs a string $y' \in \{0,1\}^n$, or else outputs* $\mathsf{FAIL}\langle e \rangle$*, where $e \in \{0,1\}^{2n}$.* Find *runs in $O(nk)$ time using $O(k)$ evaluations of C and D, and satisfies the property that for every input y, i, s, C, D, one of the following holds:*

1. $\mathsf{Find}(y, i, s, C, D)$ *outputs a string y' such that* $\mathsf{Verify}(y, y', i, s, D)$ *accepts.*
2. $\mathsf{Find}(y, i, s, C, D)$ *outputs* $\mathsf{FAIL}\langle e \rangle$*, where e is incompressible with respect to C, D.*

Proof. We start by defining the procedure Find. Given inputs y, i, s, C, D, Find begins by computing a sequence of k values $z_1, \ldots, z_k \in \{0,1\}^n$ as follows. Let $b_1, \ldots, b_k$ denote the bits of i in order. For each $j \in [k]$, we set

$$z_j = D[y](b_1 b_2 \ldots b_{j-1} \neg b_j)$$

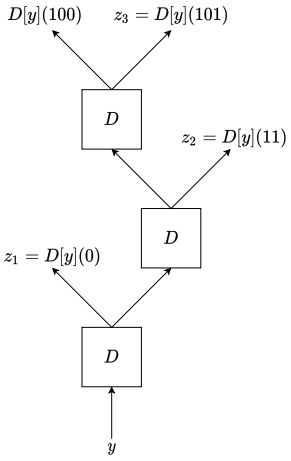


Figure 3 "Forward phase" of $\mathsf{Find}(y, 100, s, C, D)$ (where $k = 3$), during which the $z_1, \ldots, z_k$ are computed.

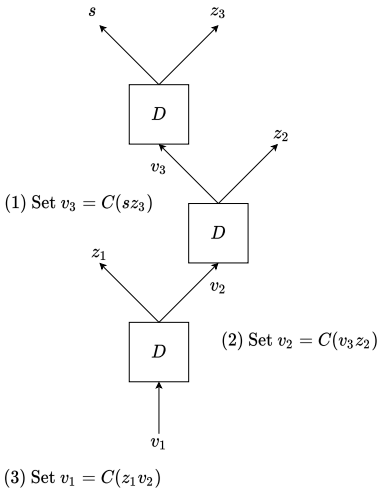


Figure 4 "Backward phase" of $\mathsf{Find}(y, 100, s, C, D)$ (where $k = 3$), during which the $v_1, \ldots, v_k$ are computed.

It is clear that the list of z_j be computed in $O(nk)$ time using $O(k)$ evaluations of D, by storing the intermediate values of $D[y](b_1 \cdots b_j)$ for each j and computing the z_j in increasing order of j.

Now, given this list of values, we compute a second sequence $v_1, \ldots, v_k \in \{0,1\}^n$. We compute the v_j for each $j \in [k]$ in decreasing order of k as follows. First we set:

$$v_k = \begin{cases} C(sz_k) & \text{if } b_k = 0 \\ C(z_k s) & \text{if } b_k = 1 \end{cases}$$

We then check that $D(C(sz_k)) = sz_k$ (resp. $D(C(z_k s)) = z_k s$). If this check fails we abort and return $\mathsf{FAIL}\langle sz_k \rangle$ (resp. $\mathsf{FAIL}\langle z_k s \rangle$). Now, for each $j \in \{k-1, k-2, \ldots, 1\}$, we set:

$$v_j = \begin{cases} C(v_{j+1} z_j) & \text{if } b_j = 0 \\ C(z_j v_{j+1}) & \text{if } b_j = 1 \end{cases}$$

Again, each time we evaluate C on some input, we check that that input is indeed compressible with respect to C, D, and if not then we return $\mathsf{FAIL}\langle e \rangle$ where e is the incompressible string we found. If we get to the end of this process without returning failure, we return the string v_1. This completes the description of the Find procedure, which overall requires at most $O(nk)$ time to compute using at most $O(k)$ evaluations of C and D. Figures 3 and 4 illustrate this procedure for a J-tree of depth 3.

We now describe the Verify procedure on input y, y', i, s, D, which simply verifies that a given string y' is a possible successful output of $\mathsf{Find}(y, i, s, C, D)$ for some C. Given y, y', i, s, D, again let $b_1, \ldots, b_k$ denote the bits of i in order. First, Verify iterates over each value of $j \in [k]$, and checks that $D[y'](b_1, \ldots, b_{j-1}, \neg b_j) = D[y](b_1, \ldots, b_{j-1}, \neg b_j)$. Next, it checks that $D[y'](b_1, \ldots, b_k) = D^k[y'](i) = s$. If all these conditions hold, the Verify procedure accepts, and otherwise it rejects. It is clear that these conditions can be verified in $O(nk)$ time using $O(k)$ evaluations of D, by storing the intermediate values $D[y](b_1, \ldots, b_j)$,$D[y'](b_1, \ldots, b_j)$ at each step.

We now show that if $\mathsf{Find}(y, i, s, C, D)$ succeeds and returns a string y', then $\mathsf{Verify}(y, y', i, s, D)$ accepts. By definition, if $\mathsf{Find}(y, i, s, C, D)$ doesn't fail, then it is able to compute some list of values $v_1, \ldots, v_k \in \{0,1\}^n$ such that for all $j < k$, $D_{b_j}(v_j) = v_{j+1}$ and $D_{\neg b_j}(v_j) = z_j$, and it returns v_1 as its output. So then we have that $D[v_1](b_1, \ldots, b_{j-1}, \neg b_j) = D[v_j](\neg b_j) = z_j$ for all $j \in [k]$. Further, we have that $D[v_1](b_1, \ldots, b_{k-1}, b_k) = D[v_k](b_k) = s$. So overall $\mathsf{Verify}(y, v_1, i, s, D)$ must accept if $\mathsf{Find}(y, i, s, C, D)$ succeeds and returns v_1.

It remains only to show that if $\mathsf{Verify}(y, y', i, s, D)$ accepts, then $\mathsf{Modified}(y, y', i, s, D)$ must hold. Recall that $\mathsf{Modified}(y, y', i, s, D)$ asserts that the two J-trees $D^k[y]$, $D^k[y']$ agree an all indices $i' \neq i$, and that $D^k[y'](i) = s$. If Verify accepts then this second condition holds trivially, since Verify explicitly checks that $D^k[y'](i) = s$ and rejects if this does not hold. Now consider the first condition of $\mathsf{Modified}$. Let $i' \in \{0,1\}^k$, $i' \neq i$. Let $\ell_1, \ldots, \ell_k$ denote the bits of i' in order, and let $t \in [k]$ be the smallest index such that $\ell_t \neq b_t$. By our assumption that Verify accepted, we have that

$$D[y](\ell_1, \ldots, \ell_t) = D[y](b_1, \ldots, b_{t-1}, \neg b_t) = D[y'](b_1, \ldots, b_{t-1}, \neg b_t) = D[y'](\ell_1, \ldots, \ell_t)$$

But by definition we also know that

$$D[y](\ell_1, \ldots, \ell_t, \ell_{t+1}, \ldots, \ell_k) = D[D[y](\ell_1, \ldots, \ell_t)](\ell_{t+1}, \ldots, \ell_k)$$

and similarly

$$D[y'](\ell_1, \ldots, \ell_t, \ell_{t+1}, \ldots, \ell_k) = D[D[y'](\ell_1, \ldots, \ell_t)](\ell_{t+1}, \ldots, \ell_k)$$

so if $D[y](\ell_1, \ldots, \ell_t) = D[y'](\ell_1, \ldots, \ell_t)$ then it must be that $D[y](\ell_1, \ldots, \ell_k) = D[y'](\ell_1, \ldots, \ell_k)$. So we have established that $D^k[y](i') = D^k[y'](i')$, which completes the proof. ◀

In addition to the update and access lemmas, we will need the following "initialization" lemma, which lets us efficiently set all leaves of a J-tree to a common value in time proportional to its depth:

▶ **Lemma 26** (J-Tree Initialization Lemma)**.** *There is an algorithm* Initialize *which takes as input a compression scheme C, D of code length n and stretch 2 (specified as oracles), a value $s \in \{0,1\}^n$, and a depth parameter k. It either succeeds and returns a seed $y \in \{0,1\}^n$ such that for all $i \in \{0,1\}^k$, $D^k[y](i) = s$, or else outputs $\mathsf{Fail}\langle e \rangle$ where e is incompressible with respect to C, D. $\mathsf{Initialize}(s, k, C, D)$ runs in time $O(nk)$ using $O(k)$ evaluations of C an D.*

Finally, we utilize the following "iterated compression" lemma.

▶ **Lemma 27** (J-Tree Iterated Compression Lemma)**.** *Let C, D be a compression scheme of code length n, and let $S = (s_1, s_2, \ldots, s_\ell)$ be a sequence of strings, where $s_i \in \{0,1\}^n$. There exists an algorithm* Iter-Compress *running in time $\mathrm{poly}(n, \ell)$ and using $\mathrm{poly}(\ell)$ evaluations of C and D which, given C, D and S, either "succedes" and outputs a seed $y \in \{0,1\}^n$ such that $D^{\lceil \log \ell \rceil}[y](i) = s_i$ for all $i \in \{0,1\}^{\lceil \log \ell \rceil}, i \leq \ell$, or else "fails" and outputs a string e which is incompressible with respect to C, D.*

The proofs of both the iterated compression and initialization lemmas follow the same format as the update lemma, and can be found in the full version.

5 Low-Space Simulations

In this section we prove our main set of theorems. In each case, we show how to simulate an exponential time computation with a drastic reduction in certain resources, using short oracle calls to some uniform compression scheme. We then show that either this simulation is successful, or there is an explicit construction algorithm that prints incompressible strings for this compression scheme.

5.1 Proofs of Main Theorems

▶ **Theorem 28.** *Let $\mathcal{C}, \mathcal{D} = \{C_n\}_{n\in\mathbb{N}}, \{D_n\}_{n\in\mathbb{N}}$ be a uniform compression scheme. Then one of the following must hold:*

1. *There is polynomial time algorithm with oracle access to $\mathcal{C}, \mathcal{D}$ which, for infinitely many n, outputs an incompressible string for C_n, D_n on input 1^n.*
2. *For every exponential time bound T, every language $L \in$ **RAM-TIME**$[T(n)]$, and every $\epsilon > 0$, there is 1-tape Turing machine with oracle access to $\mathcal{C}, \mathcal{D}$ which decides L in time $T(n)^{1+\epsilon}$, uses space at most $T(n)^\epsilon$, and makes oracle calls of length at most $T(n)^\epsilon$.*

Proof. Let $L \in$ **RAM-TIME**$[T(n)]$, and let $\mathcal{M}$ be the deterministic random access oracle machine witnessing this inclusion. Let $\mathcal{C}, \mathcal{D}$ be a uniform compression scheme. By Lemma 15, we can assume C_n, D_n have stretch 2 for all n. We will define a "simulator machine" which attempts to decide L using low space and small oracle calls to $\mathcal{C}, \mathcal{D}$. We then define a second "checker machine" which checks the work of the simulator. We show that whenever the simulator fails to decide L, the checker will be able to witness this failure in the form of an incompressible string. We will thus conclude that if simulation of L fails for infinitely many inputs, the "checker" will constitute an explicit construction algorithm which prints incompressible strings for $\mathcal{C}, \mathcal{D}$.

Step 1 – Defining the Simulator. Let $0 < \epsilon < 1$ be a fixed rational constant. We define a machine $\mathcal{S}_\epsilon$ which will attempt to efficiently simulate $\mathcal{M}$ using low space, short $\mathcal{C}, \mathcal{D}$-oracle queries, and which operates on a 1-tape oracle Turing machine. Let $x \in \{0,1\}^n$ be an input. For the remainder of thi s section we will keep a particular input length n fixed in our mind and thus drop the dependence of other terms on n in our notation; in particular we will use the abbreviation $T = T(n)$. Our machine will now fix a particular instance of $\mathcal{C}, \mathcal{D}$, in particular $C_{\lceil 2^{\epsilon n}\rceil}, D_{\lceil 2^{\epsilon n}\rceil}$; again we simplify our notation from here on and simply write C, D for this scheme. By definition of a uniform compression scheme, we see that the code length of C, D will by $poly(2^{\epsilon n}) = T(n)^{O(\epsilon)}$, and by assumption its message length is twice its code length. We will use W to denote its code length for the remainder of the proof.

We start by invoking Lemma 5, which lets us assume without loss of generality that $\mathcal{M}$ uses its first T cells of RAM and uses at most $O(\log T)$ space on its linear tapes. This simplification will come at a $\log T$ multiplicative cost to run time, which is negligible here. Now, our simulating machine will initialize a variable $\texttt{Mem} \in \{0,1\}^W$ which will be used as a seed in a J-tree with decompressor D and depth $\lceil \log T \rceil$. From now on we will fix $k = \lceil \log T \rceil$. The simulation $\mathcal{S}_\epsilon$ will run in T phases, and will maintain the invariant that if $\mathcal{M}$'s i^{th} memory cell has value $b \in \{0,1\}$ at the start of $\mathcal{M}$'s t^{th} time step, then $D^k[\texttt{Mem}](i) = b^W$ at the start of $\mathcal{S}_\epsilon$'s t^{th} phase. Aside from $\texttt{Mem}$, $\mathcal{S}_\epsilon$ will also explicitly store a copy of $\mathcal{M}$'s linear tapes (which have total length $O(k)$), the input x, and descriptions of $\mathcal{M}$'s state and tape-head pointers, each requiring at most k bits. Thus the total space required to store all local variables between phases in the simulation is at most $O(W(n)^2 + n) = T(n)^{O(\epsilon)}$ (recall that T is an exponential time bound).

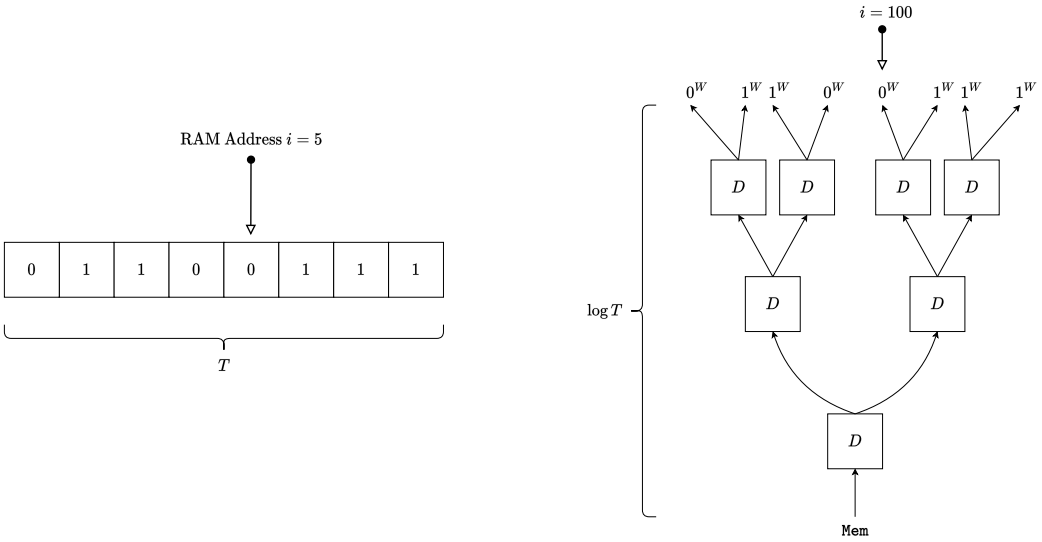


Figure 5 Simulating a RAM memory with a J-tree. While the RAM memory on the left requires T bits to store explicitly, the virtual RAM on the right can be stored with $|\texttt{Mem}| = W = T^{O(\epsilon)}$ bits.

At the start of the first phase, $\mathcal{S}_\epsilon$ initializes $\texttt{Mem}$ to a value such that $D^k[\texttt{Mem}](i) = 0^W$ for all $i \in \{0,1\}^k$, which matches the initial state of $\mathcal{M}$'s RAM memory at the beginning of its computation. By Lemma 26, this can be accomplished in time $O(kW) = T^{O(\epsilon)}$ with oracle access to C, D; if the initialization procedure fails and returns an incompressible string for C, D, the simulation halts and rejects its input x. Now, in phase $t \in [T]$, $\mathcal{S}_\epsilon$ will simulate the t^{th} step of $\mathcal{M}$ as follows:

1. Read the values at the current tapehead positions on all linear tapes, and the current state of $\mathcal{M}$.
2. Based on these values, determine which of $\mathcal{M}$'s transition rules to apply. Every rule involves a state update, which we can perform manually as we explicitly store $\mathcal{M}$'s state. If the new state is an accept/reject state for $\mathcal{M}$, then $\mathcal{S}_\epsilon$ halts and accepts/rejects accordingly. Otherwise:
   a. First, say the rule only involves updates to the linear tapes. In this case we just carry out the rule explicitly, which requires at most $poly(k) = T^{o(\epsilon)}$ operations since the linear tapes have length $O(k)$.
   b. Next, say the rule involves a RAM operation. In this case we start by reading the entire contents of the addressing tape, which we denote $i \in \{0,1\}^k$. Now, if the operation is a Load, we compute the first bit of $D^k[\texttt{Mem}](i)$ and update the auxiliary tape at its current tapehead position to this value. If the operation is an Update, we read the value at the current position of the auxiliary tape, call it $s \in \{0,1\}$. We then compute $\mathsf{Find}(\texttt{Mem}, i, s^W, C, D)$. If this procedure fails and returns an incompressible string, $\mathcal{S}_\epsilon$ aborts its entire simulation and rejects its input. Otherwise, we update $\texttt{Mem}$ to the value returned by this Find call.

If we get through all T phases without halting, we reject the input.

We now show that $\mathcal{S}_\epsilon$ operates within the required resource bounds. First, we note that all oracle calls are of length $W = T^{O(\epsilon)}$. Second, it is clear that the $\mathsf{Find}/\mathsf{Initialize}$ operations and the evaluations of $D^k[\texttt{Mem}]$ can be carried out in $poly(W)$ space, since in particular they require at most $poly(W)$ time by Lemmas 23, 25, and 26. So the space used within a phase is at most $poly(W) = T^{O(\epsilon)}$ and the size of oracle calls are bounded identically. To bound

the time complexity, we note that by the same arguments each phase can be completed in time $T^{O(\epsilon)}$, and overall there are T phases, so the total time complexity is $T^{1+O(\epsilon)}$. In the above description of each phase, we were informal about the number of work tapes required to carry out these computations. However, since any multi-tape machine running in space S and time T can be simulated on a one-tape machine in time $poly(S, T)$ and space $poly(S)$, we see that we can modify the algorithm within each phase to operate on a 1-tape machine with at most a polynomial blowup in space and time. So the above bounds still hold on a 1-tape machine, where the space is bounded by $poly(T^\epsilon) = T^{O(\epsilon)}$, and the time within each phase is bounded identically.

Step 2 – If Find Never Fails Then the Simulator Works. We now show that if $\mathcal{S}_\epsilon$ completes its computation on an input x without any Initialize or Find operation failing, then $\mathcal{S}_\epsilon$ accepts x if and only if $x \in L$. To prove this, we show by induction on $t \in [T]$ that at the beginning of the t^{th} phase of $\mathcal{S}_\epsilon$'s simulation on x, $\mathcal{S}_\epsilon$'s simulated configuration matches the configuration of $\mathcal{M}$ on input x at the beginning of time step t. This is clearly true for $t = 1$, since at the start of the simulation we initialize explicit representations of $\mathcal{M}$'s linear tapes to all zeroes, and do the same for the RAM memory using the Initialize operation, which succeeds by assumption.

Now assume the inductive hypothesis up to step t. So at the beginning of this time step on input x, $\mathcal{M}$'s linear tapes, tapehead pointers, machine state and RAM memory at the start of time step t are faithfully represented by $\mathcal{S}_\epsilon$ at the start of phase t. Thus, $\mathcal{S}_\epsilon$ is able to choose the correct transition rule to apply during this phase. It then updates the linear tapes, tapehead pointers, and machine state accordingly; all of these are stored explicitly so $\mathcal{S}_\epsilon$ has no potential for failure here. Next, if $\mathcal{M}$ uses a Load operation at this step, since we assumed the previous state of the virtual RAM is accurate, the correct value will be found by $\mathcal{S}_\epsilon$. Finally, if $\mathcal{M}$ uses a Update operation at this step, $\mathcal{S}_\epsilon$ makes the associated call to the Find procedure. By Lemma 25, if this does not fail, then the state of the virtual RAM after this step will accurately represent $\mathcal{M}$'s ram at the end of time step t. This completes the inductive case.

Step 3 – Failure of the Simulator Witnesses the Pigeonhole Principle for $\mathcal{C}, \mathcal{D}$. We now show that if the above simulation $\mathcal{S}_\epsilon$ fails to decide L for infinitely many inputs, then there is an algorithm which, given 1^n, outputs an incompressible string for C_n, D_n in polynomial time using oracles for $\mathcal{C}, \mathcal{D}$, for infinitely many n.

In particular, assume that there is an infinite set $R \subseteq \mathbb{N}$ such that for all $n \in R$, $\mathcal{S}$ makes a mistake on some length n input in its attempt to decide L. Recall that $\mathcal{S}_\epsilon$ uses the compression scheme $C_{\lceil 2^{\epsilon n} \rceil}, D_{\lceil 2^{\epsilon n} \rceil}$ in its simulation on inputs of length n. Now, let $\mathbb{I} = \{\lceil 2^{\epsilon n} \rceil \mid n \in R\}$. We will give a construction algorithm that succeeds for all input lengths in $\mathbb{I}$.

Given an input 1^m, our construction algorithm $\mathcal{H}$ operates as follows. It starts by computing an n such that $\lceil 2^{\epsilon n} \rceil = m$; by construction we see $n = O(\log m)$. Next, $\mathcal{H}$ runs the simulation $\mathcal{S}_\epsilon$ on all inputs of length n one after the other; by construction we see that $\mathcal{S}_\epsilon$ uses the same compression scheme C_m, D_m on all such inputs. During each simulation, if $\mathcal{S}_\epsilon$ fails on some Find or Initialize operation and returns an incompressible string for C_m, D_m, $\mathcal{H}$ halts and outputs that string. If $\mathcal{H}$ gets through all inputs of length n without any operation failing, it outputs something arbitrary, say 1^m. By definition we see that when $m \in \mathbb{I}$, $\mathcal{H}$ cannot get through all simulations without $\mathcal{S}_\epsilon$ failing, since by definition of $\mathbb{I}$ we know that $\mathcal{S}_\epsilon$ fails to compute L on some input of length n, and by the previous section we know this can

only happen when $\mathcal{S}_\epsilon$ fails on some Find/Initialize operation. So overall we have that $\mathcal{H}$ runs in $poly(2^n) = 2^{O(n)} = 2^{O(\log m)} = poly(m)$ time, and successfully finds an incompressible string for C_m, D_m on input 1^m for infinitely many m.

Step 4 – Wrapping Things Up. We now have that for any language $L \in$ **RAM-TIME**$[T(n)]$ and every $\epsilon > 0$, there is a machine S_ϵ running in time $T(n)^{1+O(\epsilon)}$, space $T^{O(\epsilon)}$, and making $\mathcal{C}, \mathcal{D}$ oracle calls of length $T^{O(\epsilon)}$, such that one of the following holds:

1. $\mathcal{S}_\epsilon$ correctly decides L on all but finitely many inputs.
2. There is a polynomial time construction algorithm which finds incompressible strings for C_n, D_n given oracles for $\mathcal{C}, \mathcal{D}$, which works for infinitely many inputs.

Clearly if the first possibility holds then we can get a simulation of the same complexity which decides L exactly. In addition, if the first possibility holds for all $\epsilon > 0$, then we can replace the $O(\epsilon)$ terms with ϵ as we take ϵ to zero. This yields the stated theorem. ◀

We now prove a variant of the above theorem, which allows us to replace the oracle for the compressor $\mathcal{C}$ in our low-space simulation with nondeterminism. This will be relevant for schemes $\mathcal{C}, \mathcal{D}$ where $\mathcal{C}$ is significantly harder to compute then $\mathcal{D}$.

▶ **Theorem 29.** *Let T be an exponential time bound, and let $\mathcal{C}, \mathcal{D}$ be a uniform compression scheme.*

Then one of the following must hold:

1. *There is polynomial time algorithm with oracle access to $\mathcal{C}, \mathcal{D}$ which, for infinitely many n, outputs an incompressible string for C_n, D_n on input 1^n.*
2. *For every language $L \in$ **RAM-TIME**$[T(n)]$ and every $\epsilon > 0$, there is nondeterministic 1-tape Turing machine with oracle access to $\mathcal{D}$ which decides L in time $T(n)^{1+\epsilon}$, uses space at most $T(n)^\epsilon$, and makes at most $T(n)^\epsilon$ nondeterministic guesses on all computation paths.*

Proof. We follow the same simulation $\mathcal{S}_\epsilon$ of some language L in **RAM-TIME**$[T(n)]$, with a slight modification. The key observation is the following: the procedure Find is proven to work with access to both $\mathcal{C}$ and $\mathcal{D}$ oracles by Lemma 25. However, recall that Lemma 25 also defines separate procedure Verify, which is able to verify that a certain J-tree seed is the updated form of another, and this verification only needs the $\mathcal{D}$ oracle. Recall also the key property relating Find and Verify, which says that Verify will accept any successful output of a Find operation.

Thus, if we make our simulator $\mathcal{S}_\epsilon$ nondeterministic, it can replace the Find operation by a nondeterministic guess as to the new value of the seed Mem during each phase, and then use Verify to check that this guess is correct, which requires only the $\mathcal{D}$ oracle. This immediately gives us a low space nondeterministic simulator $\mathcal{S}_\epsilon$ with similar properties as that in the proof of Theorem 28. However, the total nondeterminism used by this simulation is $T(n)^{1+O(\epsilon)}$, since it has to guess a seed of length $T^{O(\epsilon)}$ during each of the T phases.

To get away with $T^{O(\epsilon)}$ bits of nondeterminism, we use the following trick[4]. At the beginning of $\mathcal{S}_\epsilon$'s simulation, it guesses a single seed for a separate J-tree of the same code length, call this seed Up. Now, during phase t, to simulate an Update operation which sets RAM cell $i \in \{0,1\}^k$ to value $s \in \{0,1\}$, we first set $\texttt{Mem}' = D^k[\texttt{Up}](t)$, and then check if $\mathsf{Verify}(\texttt{Mem}, \texttt{Mem}', i, s^W, D)$ holds. If so we then set $\texttt{Mem} = \texttt{Mem}'$ and continue to the next phase,

[4] This trick is essentially the "easy witness method" of [11].

and if not then we halt the simulation and reject outright. It is straightforward to see that this simulation runs in time $T(n)^{1+O(\epsilon)}$, uses space $T(n)^{O(\epsilon)}$, and guesses at most $T(n)^{O(\epsilon)}$ nondeterministic bits.

By the same arguments as in Theorem 28, we have that if every Verify call accepts during $\mathcal{S}_\epsilon$'s computation on input x, then $\mathcal{S}_\epsilon$ correctly decides if $x \in L$. To finish the proof, it suffices to show that given some x such that $\mathcal{S}_\epsilon$ fails to decide if $x \in L$, we can construct an incompressible string for $C_{2^{\epsilon n}}, D_{2^{\epsilon n}}$ in $poly(T(n))$ time with $\mathcal{C}, \mathcal{D}$ oracles; from here the theorem follows directly using the arguments at the end of the proof of Theorem 28.

So, say we have such an x. Since our simulation errs on the side of rejecting, we know that $x \in L$ but $\mathcal{S}_\epsilon$ rejects x. We start by running the *deterministic* low space simulation from Theorem 28 (corresponding to L, ϵ) on x, which can be done in $poly(T(n))$ time with $\mathcal{C}, \mathcal{D}$ oracles. During this simulation, we keep track of all updated memory seeds constructed using Initialize and Find. If at any step a Find or Initialize procedure fails, we find an incompressible string and output it. Otherwise, we have found a sequence of seeds $\mathtt{Mem}_1, \ldots, \mathtt{Mem}_T \in \{0,1\}^W$ (where W is the code length used by the simulation), such that $\mathtt{Mem}_{t+1}$ represents a correct update to the RAM memory previously represented by $\mathtt{Mem}_t$ after time step t. Now, we use the Iter-Compress procedure to construct a seed $\mathtt{Up} \in \{0,1\}^W$ such that for all $t \in \{0,1\}^k$, $D^k[\mathtt{Up}](t) = \mathtt{Mem}_t$. By Lemma 27, in $poly(T(n))$ time we can either find such a seed Up, or else find an incompressible string for our scheme. But if such a string Up exists, this is precisely the nondeterministic guess which would cause our nondeterministic simulation $\mathcal{S}_\epsilon$ to accept x. So by assumption that it rejects x, if we get to this point then Iter-Compress must find an incompressible string. ◀

Finally, we show that if the explicit construction algorithm in the above theorem is also granted access to an **NP** oracle, we can get the same result, but where we simulate languages in the larger class $\mathbf{NTIME}[T(n)]$.

▶ **Theorem 30.** *Let T be an exponential time bound, and let $\mathcal{C}, \mathcal{D}$ be a uniform compression scheme.*

Then one of the following must hold:

1. *There is polynomial time algorithm with oracle access to $\mathcal{C}, \mathcal{D}$, and SAT which, for infinitely many n, outputs an incompressible string for C_n, D_n on input 1^n.*
2. *For every language $L \in \mathbf{NTIME}[T(n)]$ and every $\epsilon > 0$, there is nondeterministic 1-tape Turing machine with oracle access to $\mathcal{D}$ which decides L in time $T(n)^{1+\epsilon}$, uses space at most $T(n)^\epsilon$, and makes at most $T(n)^\epsilon$ nondeterministic guesses on all computation paths.*

Proof. This will follow quite directly from the proof of the previous theorem. Let $\mathcal{M}$ be some $\mathbf{NTIME}[T(n)]$ machine which we are attempting to simulate. At the outset of its computation, in addition to guessing one seed Up which encodes a sequence of RAM-seed updates, our new simulator $\mathcal{S}_\epsilon$ also guesses a second seed Wit which will encode the nondeterministic choices made by $\mathcal{M}$ during its computation. In particular, after guessing this seed, at each phase t of our simulation the simulator reads the first $O(1)$ bits of $D^k[\mathtt{Wit}](t)$, and uses this to determine which transition rule to apply at that step (in a nondeterministic machine there can be $O(1)$ such rules, which will be smaller then the code length of the decompressor for sufficiently large input lengths). Otherwise we run the simulation exactly as in the above proof.

By the same reasoning as in the previous proof, we see that if this machine accepts an input x then $x \in L$ (since the simulation errs on the side of rejection), but it is possible that $x \in L$ and the simulation fails to determine this. In such a case, the only possibility is that for all sequences $z \in \{0,1\}^{O(T)}$ of nondeterministic guesses causing $\mathcal{M}$ to accept x, there is no pair $\mathtt{Up}, \mathtt{Wit} \in \{0,1\}^W$ such that Wit encodes z (as described above), and Up encodes a

sequence of valid RAM seeds representing the deterministic computation of $\mathcal{M}$ on x with witness z. Thus, if we have some input x on which our simulation fails, we can use an **NP** oracle to compute a witness z causing $\mathcal{M}$ to accept x. We then run the deterministic low space simulation $\mathcal{M}$ on x with witness z, and either find an incompressible string for $\mathcal{C}, \mathcal{D}$, or else find a sequence of valid RAM seeds $\mathtt{Mem}_1, \ldots, \mathtt{Mem}_T$ for this computation. We then proceed as in the previous Theorem, attempting to compress the $\mathtt{Mem}_t$ to a single seed `Up` and the bits of z to another seed `Wit`, both using the Iter-Compress procedure. By assumption that our main nondeterministic simulation rejected x, if we get to this point we know that one of these Iter-Compress procedures must return an incompressible string. ◀

5.2 Implications

Consider the following three hypotheses:

▶ **Hypothesis 1.** *There exists some exponential time bound T and some $\epsilon > 0$ such that:*

$$\mathbf{RAM\text{-}TIME}[T(n)] \not\subseteq \mathbf{1\text{-}TISP}[T(n)^{1+\epsilon}, T(n)^{\epsilon}]$$

▶ **Hypothesis 2.** *There exists some exponential time bound T and some $\epsilon > 0$ such that:*

$$\mathbf{NTIME}[T(n)] \not\subseteq \mathbf{1\text{-}NTISPG}[T(n)^{1+\epsilon}, T(n)^{\epsilon}, T(n)^{\epsilon}]$$

▶ **Hypothesis 3.** *There exists some exponential time bound T and some $\epsilon > 0$ such that:*

$$\mathbf{RAM\text{-}TIME}[T(n)] \not\subseteq \mathbf{1\text{-}NTISPG}[T(n)^{1+\epsilon}, T(n)^{\epsilon}, T(n)^{\epsilon}]$$

The first Hypothesis asserts that deterministic exponential time RAM computations cannot be simulated on 1-tape machines using low space and near-linear blowup in time. The second Hypothesis asserts roughly the same for nondeterministic computations, claiming that such a simulation cannot occur which simultaneously uses a small amount of nondeterminism. Recall that for nondeterministic time, the multi-tape and RAM models are roughly equivalent, which is why we don't make a distinction here on the left-hand side. Finally, the third hypothesis says that deterministic RAM computations cannot be recognized by machines with short "proofs" verifiable in low space and near-linear time on a 1-tape machine.

While 1 and 2 seem incomparable and both quite reasonable, we see that 3 is formally stronger then the previous two, and we have significantly less intuition as to whether or not it should be true. However, recall from Section 1.3 that when the time bound T is linear, all three of these hypotheses are known to hold unconditionally (indeed they hold for *all* $\epsilon < 1$). In any case, the results of the previous section give us the following:

▶ **Theorem 31.** *If Hypothesis 1 holds, then for any uniform compression scheme $\mathcal{C}, \mathcal{D}$ where both $\mathcal{C}$ and $\mathcal{D}$ are computable in polynomial time, there is a polynomial time algorithm which, given 1^n, prints an incompressible string for C_n, D_n for infinitely many n.*

▶ **Theorem 32.** *If Hypothesis 2 holds, then for any uniform decompressor $\mathcal{D}$ computable in polynomial time, there exists there is a polynomial time $\mathbf{NP}$-oracle algorithm which, given 1^n, prints an empty pigeonhole for D_n for infinitely many n. In particular, $\mathbf{E}^{\mathbf{NP}} \not\subseteq \mathbf{size}[2^n/2n]$.*

▶ **Theorem 33.** *If Hypothesis 3 holds, then for any uniform compression scheme $\mathcal{C}, \mathcal{D}$ where $\mathcal{D}$ is computable in polynomial time, there is a polynomial time algorithm using a $\mathcal{C}$ oracle which, given 1^n, prints an incompressible string for C_n, D_n for infinitely many n.*

In each case, the stated result follows from the fact that oracle calls of length $T^{O(\epsilon)}$ to a problem computable in polynomial time can be simulated directly without effecting the resource bounds of the simulation as we take $\epsilon \to 0$. To illustrate the use of these theorems, we first consider their implications for the compression schemes related to non-uniform complexity measures described in Section 3.2.1. In these cases, we have some complexity measure on strings/truth tables, such as circuit complexity, formula size, or K^{poly} complexity, and wish to construct strings/truth tables of high complexity (for infinitely many input lengths), which we call the "construction problem." In each case there is an associated "compression problem" in **FNP**, where we are given a string/truth table and wish to find a small circuit/formula/program computing it if one exists. For these sorts of problems, we now have the following:

▶ **Corollary 34.** *For each of the above mentioned uniform compression schemes related to non-uniform complexity measures, we have:*

1. *If Hypothesis 1 holds, then an efficient algorithm for the compression problem implies an efficient algorithm for the construction problem.*
2. *If Hypothesis 2 holds, then there is an efficient* **NP***-oracle algorithm for the construction problem.*
3. *If Hypothesis 3 holds, then there is an efficient algorithm for the construction problem using an arbitrary oracle for the compression problem. In other words, the construction problem reduces to the compression problem.*

We see here that the conclusion of the third implication implies the conclusion of the previous two. As noted in the introduction, the second implication can be proved by simpler techniques, utilizing the "Easy Witness Lemma." We give an alternate proof using this method in the full version for the interested reader. However, for the first and third implication, the J-tree update lemma seems necessary.

The case of large prime construction does not quite fit in with the others, as both the compressor and decompressor given in Theorem 21 require a factoring oracle. In addition, the second implication of Corollary 34 above is trivial when applied to the construction of large primes, since primality testing is in **P** [1]. However we can still derive the following from Theorem 31:

▶ **Corollary 35.** *If Hypothesis 1 holds, then a polynomial time algorithm for factoring implies a polynomial time algorithm to construct* $16n$*-bit primes of magnitude* $> 2^n$ *(for infinitely many* n*).*

More generally we can conclude the following directly from Theorem 28:

▶ **Corollary 36.** *One of the following is true:*

1. *For every exponential time bound* T *and every* $\epsilon > 0$*, every language decidable in time* $T(n)$ *on a RAM machine can be decided in time* $T(n)^{1+\epsilon}$ *and space* $T(n)^{\epsilon}$ *by a 1-tape machine with a factoring oracle, which makes oracle calls of length at most* $T(n)^{\epsilon}$*.*
2. *There is a polynomial time algorithm with a factoring oracle that generates* $32n$*-bit primes of magnitude* $> 2^n$ *for infinitely many* n*.*

6 BPP, TFNP, and the Weak Pigeonhole Principle

Throughout this paper we have studied the problem of finding incompressible strings for *uniform* compression schemes. As mentioned in Section 3, it would also be natural to study the more general search problem, where a compression scheme of a fixed message/code length is given as input in the form of a boolean circuit:

▶ **Definition 37.** Lossy Code *is the following problem: given circuits* $C\colon \{0,1\}^n \to \{0,1\}^{n-1}$ *and* $D\colon \{0,1\}^{n-1} \to \{0,1\}^n$*, find some* x *such that* $D(C(x)) \neq x$*.*

It follows from the definition that Lossy Code lies in **TFNP** (more specifically the class **PPP** defined in [18]), and reduces to the problem Empty studied in [13] and [14]. Further, we have seen that Lossy Code admits a randomized algorithm that outputs a solution with high probability. Since solutions can be verified efficiently, this fits it into a family of search problems studied by Goldreich [4], whose results imply the following:

▶ **Lemma 38.** Lossy Code *is polynomial time Turing reducible to* CAPP*.*

Here, CAPP denotes the canonical complete problem for **prBPP**, where we are given as input a circuit $E\colon \{0,1\}^n \to \{0,1\}$, and must output an approximation of $Pr_x[E(x) = 1]$ that is accurate to within $\pm\frac{1}{6}$. A natural question is whether the converse holds, i.e. whether CAPP reduces deterministically to Lossy Code. As hinted towards in Section 3.2, a polynomial time algorithm for Circuit Synthesis would imply such a reduction, but an unconditional reduction would be a major breakthrough as it would place $\mathbf{BPP} \subseteq \mathbf{NP}$. We show here that if the compressor C in Lossy Code is allowed to be randomized, the associated total search problem is in fact Turing-equivalent to CAPP, and thus complete for **prBPP**. The problem is defined formally as follows:

▶ **Definition 39.** R-Lossy Code *is the following problem: given* $C\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^{n-1}$*,* $D\colon \{0,1\}^{n-1} \to \{0,1\}^n$*, find some* $x \in \{0,1\}^n$ *such that* $Pr_r[D(C(x,r)) = x] < \frac{1}{2}$*.*

To clarify, this is precisely the same problem as Lossy Code, except that the "compressor circuit" C now uses some random coins. In this case, we seek a string which has a $< \frac{1}{2}$ probability of being recoverable from its description (where the probability is taken over the random coins of the compressor C). Just like Lossy Code this search problem is total, but verifying a solution is no longer in **P**. Indeed, it may be **PP**-hard to determine if any particular string is a solution, but there is a efficient randomized verification procedure that accepts only valid solutions, and accepts a random solution with high probability.

In the full version we show the following:

▶ **Theorem 40.** R-Lossy Code *and* CAPP *are Turing reducible to one another in deterministic polynomial time.*

An interesting takeaway from the proof of the above theorem is the following: choosing a good fixing of the "leftover bits" in Yao's lemma is in some sense a *universal* probabilistic search problem, since derandomizing this step would give a reduction from **prBPP** to Lossy Code $\in$ **TFNP** (and in particular would imply $\mathbf{BPP} \subseteq \mathbf{NP}$). More formally:

▶ **Corollary 41.** *Consider the following promise problem: we are given a circuit* $E\colon \{0,1\}^n \to \{0,1\}$*, a matrix* $A \in \{0,1\}^{m \times n}$*, and an index* $i \in [n]$*, such that* A *fails Yao's next-bit test with respect to* E *at index* i *with bias* $\epsilon = poly(\frac{1}{m})$*, and we must produce some* $z \in \{0,1\}^{n-i}$ *such that fixing the last* $n - i$ *bits of* E *to* z *preserves a non-negligible bias for this next-bit test. If this problem can be solved in deterministic polynomial time, then* **prBPP** *reduces to* Lossy Code*.*

References

1 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Ann. of Math*, 2:781–793, 2002.

2 Lance Fortnow. Time–space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, 2000. doi:10.1006/jcss.1999.1671.

3 Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, November 2005. doi:10.1145/1101821.1101822.

4 Oded Goldreich. *In a World of P=BPP*, pages 191–232. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_20.

5 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986. doi:10.1145/6490.6503.

6 Yuri Gurevich and Saharon Shelah. Nearly linear time. In *Proceedings of the Symposium on Logical Foundations of Computer Science: Logic at Botik '89*, pages 108–118, Berlin, Heidelberg, 1989. Springer-Verlag.

7 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002. Special Issue on Complexity 2001. doi:10.1016/S0022-0000(02)00024-7.

8 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 '97, pages 220–229, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258590.

9 Emil Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1):1–37, 2004. doi:10.1016/j.apal.2003.12.003.

10 Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *Journal of Logic and Computation*, 17(3):587–604, 2007.

11 Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001. doi:10.1006/jcss.2001.1763.

12 Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 73–79, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335314.

13 Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total Functions in the Polynomial Hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2021.44.

14 Oliver Korten. The hardest explicit construction. In *62nd Annual Symposium on Foundations of Computer Science*, 2021.

15 Wolfgang Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 401–408, New York, NY, USA, 1984. Association for Computing Machinery. doi:10.1145/800057.808706.

16 Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

17 Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 665–677, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055500.

18 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

19 J. Paris, A. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53:1235–1244, 1988.

20 D. H. J. Polymath. Deterministic methods to find primes. *arXiv*, 2010. doi:10.48550/ARXIV.1009.3956.

21 J.M. Robson. Deterministic simulation of a single tape turing machine by a random access machine in sub-linear time. *Information and Computation*, 99(1):109–121, 1992. doi:10.1016/0890-5401(92)90026-C.

22 Ryan Williams. Inductive time-space lower bounds for sat and related problems. *Computational Complexity*, 15:433–470, 2005.

23 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.

24 Andrew C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

Improved Low-Depth Set-Multilinear Circuit Lower Bounds

Deepanshu Kush
Department of Computer Science, University of Toronto, Canada

Shubhangi Saraf
Department of Mathematics and Department of Computer Science, University of Toronto, Canada

Abstract

In this paper, we prove strengthened lower bounds for constant-depth set-multilinear formulas. More precisely, we show that over any field, there is an explicit polynomial f in VNP defined over n^2 variables, and of degree n, such that any product-depth Δ set-multilinear formula computing f has size at least $n^{\Omega\left(n^{1/\Delta}/\Delta\right)}$. The hard polynomial f comes from the class of Nisan-Wigderson (NW) design-based polynomials.

Our lower bounds improve upon the recent work of Limaye, Srinivasan and Tavenas (STOC 2022), where a lower bound of the form $(\log n)^{\Omega(\Delta n^{1/\Delta})}$ was shown for the size of product-depth Δ set-multilinear formulas computing the iterated matrix multiplication (IMM) polynomial of the same degree and over the same number of variables as f. Moreover, our lower bounds are novel for any $\Delta \geq 2$.

The precise quantitative expression in our lower bound is interesting also because the lower bounds we obtain are "sharp" in the sense that any asymptotic improvement would imply general set-multilinear circuit lower bounds via depth reduction results.

In the setting of general set-multilinear formulas, a lower bound of the form $n^{\Omega(\log n)}$ was already obtained by Raz (J. ACM 2009) for the more general model of multilinear formulas. The techniques of LST (which extend the techniques of the same authors in (FOCS 2021)) give a different route to set-multilinear formula lower bounds, and allow them to obtain a lower bound of the form $(\log n)^{\Omega(\log n)}$ for the size of general set-multilinear formulas computing the IMM polynomial. Our proof techniques are another variation on those of LST, and enable us to show an improved lower bound (matching that of Raz) of the form $n^{\Omega(\log n)}$, albeit for the same polynomial f in VNP (the NW polynomial). As observed by LST, if the same $n^{\Omega(\log n)}$ size lower bounds for unbounded-depth set-multilinear formulas could be obtained for the IMM polynomial, then using the self-reducibility of IMM and using hardness escalation results, this would imply super-polynomial lower bounds for general algebraic formulas.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases algebraic circuit complexity, complexity measure, set-multilinear formulas

Digital Object Identifier 10.4230/LIPIcs.CCC.2022.38

Related Version *Full Version*: https://arxiv.org/abs/2205.00611
Full Version: https://eccc.weizmann.ac.il/report/2022/064/

Funding *Shubhangi Saraf*: Research partially supported by a Sloan research fellowship and an NSERC Discovery Grant.

Acknowledgements We would like to thank Swastik Kopparty, Mrinal Kumar, and Ben Rossman for several helpful discussions.

37th Computational Complexity Conference (CCC 2022).
Editor: Shachar Lovett; Article No. 38; pp. 38:1–38:16

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

1 Introduction

1.1 Background

An *algebraic circuit* over a field $\mathbb{F}$ for a multivariate polynomial $P(x_1, \ldots, x_N)$ is a directed acyclic graph (DAG) whose internal vertices (called gates) are labeled as either $+$ (sum) or $\times$ (product), and leaves (vertices of in-degree zero) are labeled by the variables x_i or constants from $\mathbb{F}$. A special output gate (the root of the DAG) represents the polynomial P. If the DAG happens to be a tree, such a resulting circuit is called an *algebraic formula*. The size of a circuit is the number of nodes in the DAG. We also consider the product-depth of the circuit, which is the maximum number of product gates on a root-to-leaf path.

An algebraic circuit is therefore a computational model, which solves the computational task of evaluating P on a given input $(x_1, \ldots, x_N)$. The complexity of this model is measured by the size of the circuit, which serves as an indicator of the time complexity of computing the polynomial. The product-depth measures the degree to which this computation can be made parallel. As an algebraic circuit is supposed to construct a formal polynomial P, it is a *syntactic* model of computation. This is unlike a Boolean circuit, which is only required to model specific truth-table constraints. The problem of proving algebraic circuit lower bounds is therefore widely considered to be easier than its Boolean counterpart. Indeed, we know that proving VP $\neq$ VNP, the algebraic analog of the P vs. NP problem, is implied by the latter separation, in the non-uniform setting ([3]). We refer the reader to [28] for a much more elaborate survey of this topic.

1.2 The LST breakthrough

Much like in the Boolean setting, the problem of showing lower bounds for *general* algebraic circuits (or even formulas) has remained elusive. However, some remarkable progress has been made very recently by Limaye, Srinivasan, and Tavenas ([20]) who in a spectacular breakthrough, showed the first super-polynomial lower bounds for algebraic circuits of *all* constant depths. Prior to their work, the best known lower bound ([14]) even for product-depth 1 (or $\Sigma\Pi\Sigma$ circuits) was only almost-cubic. This is in stark contrast with the Boolean setting, in which we have known strong constant-depth lower bounds for many decades [2, 8, 31, 9, 27, 29]. Constant-depth circuits are critical to the study of algebraic complexity theory, as unlike the Boolean setting, strong enough bounds against them are known to yield VP $\neq$ VNP ([1]). This helps put into perspective the importance of the work [20].

The crucial step in the proof of their result is to first establish super-polynomial lower bounds for a certain restricted class of (low-depth) algebraic circuits, namely *set-multilinear* circuits which we now define along with other important circuit models. A polynomial is said to be homogeneous if each monomial has the same total degree and *multilinear* if every variable occurs at most once in any monomial. Now, suppose that the underlying variable set is partitioned into d sets $X_1, \ldots, X_d$. Then the polynomial is said to be *set-multilinear* with respect to this variable partition if each monomial in P has *exactly* one variable from each set. We also define different models of computation corresponding to these variants of polynomials classes. An algebraic formula (circuit) is set-multilinear with respect to a variable partition $(X_1, \ldots, X_d)$ if each internal node in the formula (circuit) computes a set-multilinear polynomial. Multilinear/homogeneous circuits and formulas are defined analogously.

Several well-studied and interesting polynomials happen to be set-multilinear. For example, the Determinant and the Permanent polynomials, the study of which is profoundly consequential to the field of algebraic complexity theory, are set-multilinear (with respect

to the column variables). Another well-studied polynomial, namely the Iterated Matrix Multiplication polynomial, is also set-multilinear. The polynomial $\mathrm{IMM}_{n,d}$ is defined on $N = dn^2$ variables, where the variables are partitioned into d sets $X_1, \ldots, X_d$ of size n^2, each of which is represented as an $n \times n$ matrix with distinct variable entries. The polynomial $\mathrm{IMM}_{n,d}$ is defined to be the polynomial that is the $(1,1)$-th entry of the product matrix $X_1 X_2 \cdots X_d$. This polynomial has a simple divide-and-conquer-based set-multilinear formula of size $n^{O(\log d)}$, and more generally for every $\Delta \leq \log d$, a set-multilinear formula of product-depth Δ and size $n^{O(\Delta d^{1/\Delta})}$, and circuit[1] of size $n^{O(d^{1/\Delta})}$. Even without the set-multilinearity constraint, no significantly better upper bound is known. It is reasonable to conjecture that this simple upper bound is tight up to the constant in the exponent.

The lower bounds in [20] for general constant-depth algebraic circuits are shown in the following sequence of steps:

1. It is shown that general low-depth algebraic circuits can be transformed to set-multilinear algebraic circuits of low depth, and without much of a blow-up in size (as long as the degree is small). More precisely, any product-depth Δ circuit of size s computing a polynomial that is set-multilinear with respect to the partition $(X_1, \ldots, X_d)$ where each $|X_i| \leq n$, can be converted to a set-multilinear circuit[2] of product-depth 2Δ and size $\mathrm{poly}(s) \cdot d^{O(d)}$. Such a "set-multilinearization" of general formulas of small degree was already shown before in [25] (which we describe soon in more detail); however, the main contribution of [20] here is to prove this *depth-preserving* version of it.
2. Strong lower bounds are then established for low-depth set-multilinear circuits (of small enough degree). More precisely, any set-multilinear circuit C computing $\mathrm{IMM}_{n,d}$ (where $d = O(\log n)$) of product-depth Δ must have size at least $n^{d^{\exp(-O(\Delta))}}$. This combined with the first step yields the desired lower bound for general constant-depth circuits.

Given Raz's set-multilinearization of formulas of small degree that we alluded to, and this description of the set-multilinear formula lower bounds from [20] when $d = O(\log n)$, it is evident the "small degree" regime is inherently interesting to study - as it provides an avenue, via "hardness escalation", for tackling one of the grand challenges of algebraic complexity theory, namely proving super-polynomial lower bounds for general algebraic formulas. However, we shall now see that even the large degree regime can be equally consequential in this regard.

1.3 The large degree regime

Consider a polynomial P that is set-multilinear with respect to the variable partition $(X_1, \ldots, X_d)$ where each $|X_i| \leq n$. The main focus of this paper is to study set-multilinear circuit complexity in the regime where d and n are *polynomially* related (as opposed to say, the assumption $d = O(\log n)$ described above). We now provide some background and motivation for studying this regime.

In follow-up work [21], the same authors showed the first super-polynomial lower bound against unbounded-depth set-multilinear formulas computing $\mathrm{IMM}_{n,n}$[3]. As is astutely described in [21], studying the set-multilinear formula complexity of IMM is extremely interesting and consequential even in the setting $d = n$ because of the following reasons:

[1] In this paper, when speaking of constant-depth models of computation at a high level, we shall often use the terms circuit and formula interchangeably as a product-depth Δ circuit of size s can be simulated by a product-depth Δ formula of size $s^{2\Delta}$.

[2] There is also an intermediate "homogenization" step which we skip describing here for the sake of brevity.

[3] Note that for $\mathrm{IMM}_{n,n}$, each X_i has size n^2, not n. But the important thing for us here is that the degree, n, is polynomially related to this parameter.

- $\mathrm{IMM}_{n,n}$ is a *self-reducible* polynomial i.e., it is possible to construct formulas for $\mathrm{IMM}_{n,n}$ by recursively using formulas for $\mathrm{IMM}_{n,d}$ (for any $d < n$). In particular, if we had formulas of size $n^{o(\log d)}$ for $\mathrm{IMM}_{n,d}$ (for some $d < n$), this would imply formulas of size $n^{o(\log n)}$ for $\mathrm{IMM}_{n,n}$. In other words, an optimal $n^{\Omega(\log n)}$ lower bound for $\mathrm{IMM}_{n,n}$ implies $n^{\omega_d(1)}$ lower bounds for $\mathrm{IMM}_{n,d}$ for any $d < n$.
- Raz in [25] showed that if an N-variate set-multilinear polynomial of degree d has an algebraic formula of size s, then it also has a set-multilinear formula of size $\mathrm{poly}(s)\cdot(\log s)^d$. In particular, for a set-multilinear polynomial P of degree $d = O(\log N/\log\log N)$, it follows that P has a formula of size $\mathrm{poly}(N)$ if and only if P has a set-multilinear formula of size $\mathrm{poly}(N)$. Thus, having $N^{\omega_d(1)}$ set-multilinear formula size lower bounds for such a low degree would imply super-polynomial lower bounds for general formulas.

In particular, proving the optimal $n^{\Omega(\log n)}$ set-multilinear formula size lower bound for $\mathrm{IMM}_{n,n}$ would have dramatic consequences. To this end, the authors in [21] are able to show a weaker bound of the form $(\log n)^{\Omega(\log n)}$ instead. Even though it is the case that "simply" improving the base of this exponent from $\log n$ to n yields general formula lower bounds, it seems that we are still far from achieving it. Indeed, as is observed in [21], we do not even have the optimal $n^{\Omega(\sqrt{n})}$ lower bound[4] when product-depth $\Delta = 2$. Moreover, we do not know how to obtain a lower bound of the form $n^{\Omega(\sqrt{n})}$ for product-depth 2 set-multilinear circuits for *any* explicit polynomial of degree n and in $\mathrm{poly}(n)$ variables. For product-depths $\Delta \leq \log n$, [21] shows a set-multilinear formula size lower bound of $(\log n)^{\Omega(\Delta n^{1/\Delta})}$ for $\mathrm{IMM}_{n,n}$, which is in fact the best set-multilinear lower bound we know for any polynomial of degree n and in $\mathrm{poly}(n)$ variables, and for any $\Delta \geq 2$. As far as we know, the previous best lower bound of $\exp(\Omega(n^{1/\Delta}))$, also for $\mathrm{IMM}_{n,n}$, followed from the work of Nisan and Wigderson ([23]). It is therefore an interesting challenge to improve the base of this exponent from $\log n$ to n i.e., establish a near-optimal $n^{\Omega(n^{1/\Delta})}$ lower bound in the constant (or low) depth setting.

1.4 Our Results

In this paper, we obtain these "optimal" lower bounds, albeit not for $\mathrm{IMM}_{n,n}$, but rather for another explicit polynomial in VNP. We show the following:

▶ **Theorem 1.** *Let N be a growing parameter and Δ be an integer such that $1 \leq \Delta \leq \log N/\log\log N$. There is an explicit polynomial P_N defined over $N = n^2$ variables with degree $d = n$ that is set-multilinear with respect to the variable partition $X = (X_1, \ldots, X_d)$ where each $|X_i| = n$ and such that any set-multilinear formula of product-depth Δ computing $P_N(X)$ must have size at least $N^{\Omega(d^{1/\Delta}/\Delta)}$.*

Notice that obtaining this precise bound is interesting also when viewed through the lens of *depth reduction*. Tavenas ([30]), building on several prior works ([1, 16]), showed that any algebraic circuit of $\mathrm{poly}(N)$ size computing a homogeneous N-variate polynomial of degree d can be converted to a homogeneous circuit of product-depth[5] Δ of size $(Nd)^{O(d^{1/\Delta})}$. It easily follows from the proof that this depth reduction preserves syntactic restrictions. That is, if we start with a syntactically set-multilinear circuit, the resulting product-depth Δ circuit is also syntactically set-multilinear. Therefore, the precise bound in Theorem 1 is *sharp* in the

[4] This is known for set-multilinear (and even multilinear) $\Sigma\Pi\Sigma\Pi$ circuits (see [7, 15]), but those are only special cases of general product-depth 2 circuits, which are $\Sigma\Pi\Sigma\Pi\Sigma$.

[5] The result is stated in [30] for $\Sigma\Pi\Sigma\Pi$ circuits but the proof can be appropriately modified for larger product-depths.

sense that any asymptotic improvement in its exponent would imply super-polynomial set-multilinear circuit lower bounds, which would be quite a strong and interesting consequence. Another very intriguing direction is to consider the problem of *improved* depth reduction for set-multilinear circuits. If an asymptotic improvement in the exponent on the bound for general circuits from [30] could be shown to hold for set-multilinear circuits in the setting of Theorem 1 (i.e., when $N = d^2$), this would again imply super-polynomial set-multilinear circuit lower bounds. There is some evidence towards this possibility, as [17] shows such an improvement in a certain regime of parameters for multilinear circuits (see the discussion in Section 4 for more details).

▶ Remark 2. The lower bound in Theorem 1 is actually $d^{\Omega(d^{1/\Delta}/\Delta)}$, where d is the degree of the underlying polynomial, and it holds as long as degree $d \leq n$ (the details are deferred to the proof of Theorem 9 in Section 3). Observe that for constant Δ this bound already nearly matches the bound $(\log n)^{\Omega(\Delta d^{1/\Delta})}$ in [21] (which was shown for $\mathrm{IMM}_{n,d}$) when $d = (\log n)^{\Omega(1)}$ and exceeds it as soon as d becomes super-polylogarithmic in n. Moreover for $d < \log n / \log\log n$, both the bounds are trivial even for $\Delta = 1$.

We also remark that in several lower bounds for algebraic circuit classes in the past, the lower bound was initially shown for a polynomial in VNP and then with additional effort, was shown to also hold for a polynomial in VP (in particular, the IMM polynomial). A strong candidate for the choice of this polynomial family in VNP has been the Nisan-Wigderson (NW) design-based ([22]) family of polynomials. For instance, [13] showed a lower bound of $n^{\Omega(\sqrt{n})}$ for the top fan-in of a $\Sigma\Pi^{[O(\sqrt{n})]}\Sigma\Pi^{[\sqrt{n}]}$ circuit computing the NW polynomial, which was subsequently shown for IMM by [7]. Similarly, [11] showed an $n^{\Omega(\sqrt{d})}$ size lower bound for homogeneous depth-4 algebraic formulas for the NW polynomial, which was then shown for IMM later in [19]. Much like these examples, our hard polynomial family in Theorem 1 is also indeed the NW polynomial family, as we shall see in Section 3. Our motivation to study constant-depth set-multilinear formula complexity was to prove the optimal lower bounds for the IMM polynomial. Although we are presently able to show it only for the NW polynomial instead of IMM, we are hopeful that this is an important step in its direction.

In addition to our lower bound for bounded-depth set-multilinear formulas, we observe that the same proof technique also implies a lower bound of the form $n^{\Omega(\log n)}$ for unbounded-depth set-multilinear formulas. [21] showed a weaker bound of the form $(\log n)^{\Omega(\log n)}$ but for $\mathrm{IMM}_{n,n}$.

▶ **Theorem 3.** *For a given integer N, there is an explicit polynomial P_N defined over $N = n^2$ variables with degree $d = n$ that is set-multilinear with respect to the variable partition $X = (X_1, \ldots, X_d)$ where each $|X_i| = n$ such that any set-multilinear formula computing $P_N(X)$ must have size at least $N^{\Omega(\log N)}$.*

The hard polynomial in Theorem 3 is also the NW polynomial, which if "improved" to $\mathrm{IMM}_{n,n}$, then as discussed, would yield super-polynomial general formula lower bounds. However, we note that in this case, our result is in some sense subsumed by the result of Raz ([24]) who showed an $n^{\Omega(\log n)}$ lower bound for the $n \times n$ permanent (or determinant) polynomial for unbounded-depth multilinear formulas.

1.5 Other Related Work

In the bounded-depth setting, other than the works [20, 21, 23] already mentioned, there have been several lower bounds for the class of low-depth *multilinear* circuits ([26, 5, 4, 12]). In the unbounded-depth setting, apart from the works [21, 24] already mentioned for set-multilinear formulas, there have also been several strong lower bounds of the form $n^{\Omega(\log n)}$ against

multilinear formulas ([6, 10, 15]). However, in both settings of depth, several of these works are not even applicable to the set-multilinear setting as the corresponding hard polynomial does not happen to be set-multilinear.

1.6 Proof overview

Our overall proof techniques are similar to that of many known lower bounds. We work with a measure that we show to be small for all polynomials computed by small enough set-multilinear formulas (appropriately so in the bounded and unbounded-depth settings) and large for the NW polynomial. These *partial derivative measures* were introduced by Nisan and Wigderson in [23], who used them to prove the constant-depth set-multilinear formula lower bounds we discussed earlier. [20, 21] use a particular variant of this measure and our measure is in turn inspired from these works.

Given a variable partition $(X_1, \ldots, X_d)$, we label each set of variables X_i as "positive" or "negative" uniformly at random. Let $\mathcal{P}$ and $\mathcal{N}$ denote the set of positive and negative indices respectively, and let $\mathcal{M}^{\mathcal{P}}$ and $\mathcal{M}^{\mathcal{N}}$ denote the sets of all set-multilinear monomials over $\mathcal{P}$ and $\mathcal{N}$ respectively. For a polynomial that is set-multilinear over the given variable partition $(X_1, \ldots, X_d)$, our measure then is simply the rank of the "partial derivative matrix" whose rows are indexed by the elements of $\mathcal{M}^{\mathcal{P}}$ and columns indexed by $\mathcal{N}^{\mathcal{P}}$, and the entry of this matrix corresponding to a row m_1 and a column m_2 is the coefficient of the monomial $m_1 \cdot m_2$ in the given polynomial.

In contrast, the measure used in [20] is deterministic and moreover, it is *asymmetric* with respect to the positive and negative variable sets, in the sense that while keeping the positive variable sets as is, it first reduces the size of the negative variable sets by arbitrarily setting a few of these variables to field constants, and then works with the resulting polynomial. On the other hand, [21] does use a randomized measure, but one that is still asymmetric, relying on randomly setting a few of the variables inside each set to constants. The way they control the discrepancy between the sizes of the positive and negative variable sets (which is indeed crucial for obtaining the claimed lower bounds) is by imposing a Martingale-like distribution. The lower bound of [23] also uses random restrictions to enable them to effectively "simplify" the circuit and upper bound its complexity. Our symmetric, randomized measure avoids random restrictions altogether, and though it is mainly inspired by the measure and the techniques from [20], it is also reminiscent of the measures used in [24, 26] to prove multilinear formula lower bounds.

2 Preliminaries

We begin by defining the hard polynomial of our main result (Theorem 1). As is done in previous lower bounds using the NW polynomials (for example, see [13]), we will identify the set of the first n integers as elements of $\mathbb{F}_n$ via an arbitrary correspondence $\phi : [n] \to \mathbb{F}_n$. If $f(z) \in \mathbb{F}_n[z]$ is a univariate polynomial, then we abuse notation to let $f(i)$ denote the evaluation of f at the i-th field element via the above correspondence i.e., $f(i) \coloneqq \phi^{-1}(f(\phi(i)))$. To simplify the exposition, in the following definition, we will omit the correspondence ϕ and identify a variable $x_{i,j}$ by the point $(\phi(i), \phi(j)) \in \mathbb{F}_n \times \mathbb{F}_n$.

▶ **Definition 4** (Nisan-Wigderson Polynomials). *For a prime power n, let $\mathbb{F}_n$ be a field of size n. For an integer $d \leq n$ and the set X of nd variables $\{x_{i,j} : i \in [n], j \in [d]\}$, we define the degree d homogeneous polynomial $NW_{n,d}$ over any field as*

$$NW_{n,d}(X) = \sum_{\substack{f(z) \in \mathbb{F}_n[z] \\ \deg(f) < d/2}} \prod_{j \in [d]} x_{f(j),j}.$$

Next, we turn to the measure that we shall use to prove Theorems 1 and 3. For the purpose of setting it up, we follow the notation of [20] in the following definition. However, we do remark that we do not need it in its full generality as we will eventually work with a simpler, *symmetric* notion that was alluded to in Section 1. Nevertheless, employing the same notation has the advantage that the reader is quite possibly already familiar with it in the context of proving set-multilinear circuit lower bounds.

▶ **Definition 5** (Relative Rank Measure of [20, 21])**.** *Let f be a polynomial that is set-multilinear with respect to the variable partition $(X_1, X_2, \ldots, X_d)$ where each set is of size n. Let $w = (w_1, w_2, \ldots, w_d)$ be a tuple (or word) of non-zero real numbers such that $2^{|w_i|} \in [n]$ for all $i \in [d]$. For each $i \in [d]$, let $X_i(w)$ be the variable set obtained by removing arbitrary variables from the set X_i such that $|X_i(w)| = 2^{|w_i|}$, and let $\overline{X}(w)$ denote the tuple of sets of variables $(X_1(w), \ldots, X_d(w))$. Corresponding to a word w, define $\mathcal{P}_w := \{i \mid w_i > 0\}$ and $\mathcal{N}_w := \{i \mid w_i < 0\}$. Let $\mathcal{M}_w^{\mathcal{P}}$ be the set of all set-multilinear monomials over a subset of the variable sets $X_1(w), X_2(w), \ldots, X_d(w)$ indexed by $\mathcal{P}_w$, and similarly let $\mathcal{M}_w^{\mathcal{N}}$ be the set of all set-multilinear monomials over these variable sets indexed by $\mathcal{N}_w$.*

Define the 'partial derivative matrix' matrix $\mathcal{M}_w(f)$ whose rows are indexed by the elements of $\mathcal{M}_w^{\mathcal{P}}$ and columns indexed by the elements of $\mathcal{N}_w^{\mathcal{P}}$ as follows: the entry of this matrix corresponding to a row m_1 and a column m_2 is the coefficient of the monomial $m_1 \cdot m_2$ in f. We define

$$\mathrm{relrk}_w(f) := \frac{\mathrm{rank}(\mathcal{M}_w(f))}{\sqrt{|\mathcal{M}_w^{\mathcal{P}}| \cdot |\mathcal{M}_w^{\mathcal{N}}|}} = \frac{\mathrm{rank}(\mathcal{M}_w(f))}{2^{\frac{1}{2}\sum_{i\in[d]} |w_i|}}.$$

▶ **Definition 6.** *For any tuple $w = (w_1, \ldots, w_t)$ and a subset $S \subseteq [t]$, we shall refer to the sum $\sum_{i\in S} w_i$ by w_S. And by $w|_S$, we will refer to the tuple obtained by considering only the elements of w that are indexed by S. We denote by $\mathbb{F}_{\mathrm{sm}}[\mathcal{T}]$ the set of set-multilinear polynomials over the tuple of sets of variables $\mathcal{T}$.*

The following is a simple result that establishes various useful properties of the relative rank measure.

▷ Claim 7 ([20]).

1. (Imbalance) Say $f \in \mathbb{F}_{\mathrm{sm}}[\overline{X}(w)]$. Then, $\mathrm{relrk}_w(f) \leq 2^{-|w_{[d]}|/2}$.
2. (Sub-additivity) If $f, g \in \mathbb{F}_{\mathrm{sm}}[\overline{X}(w)]$, then $\mathrm{relrk}_w(f+g) \leq \mathrm{relrk}_w(f) + \mathrm{relrk}_w(g)$.
3. (Multiplicativity) Say $f = f_1 f_2 \cdots f_t$ and assume that for each $i \in [t]$, $f_i \in \mathbb{F}_{\mathrm{sm}}[\overline{X}(w|_{S_i})]$, where $(S_1, \ldots, S_t)$ is a partition of $[d]$. Then

$$\mathrm{relrk}_w(f) = \prod_{i\in[t]} \mathrm{relrk}_{w|_{S_i}}(f_i).$$

3 Main Result

We are now ready to prove our main result. We start by showing that the *symmetric* relative rank is large for the NW polynomial.

▷ Claim 8. For an integer $n = 2^k$ and $d \leq n$, let $w \in \{k, -k\}^d$ with $w_{[d]} = 0$. Then $\mathrm{relrk}_w(NW_{n,d}) = 1$ i.e., $\mathcal{M}_w(NW_{n,d})$ has full rank.

Proof. Fix $n = 2^k$ and d, so that we can also write NW for $NW_{n,d}$, and let $n' = d/2$. The condition on w implies that $|\mathcal{P}_w| = |\mathcal{N}_w| = n'$. Observe that $\mathcal{M}_w(NW)$ is a square matrix of dimension $|\mathcal{M}_w^{\mathcal{P}}| = |\mathcal{M}_w^{\mathcal{N}}| = n^{n'}$. Consider a row of $\mathcal{M}_w(NW)$ indexed by a monomial

$m_1 = x_{i_1,j_1} \cdots x_{i_{n'},j_{n'}} \in \mathcal{M}_w^{\mathcal{P}}$. m_1 can be thought of as a map from $S = \{j_1, \ldots, j_{n'}\}$ to $\mathbb{F}_n$ which sends j_ℓ to i_ℓ for each $\ell \in [n']$. Next, by interpolating the pairs $(j_1, i_1), \ldots, (j_{n'}, i_{n'})$, we know that there exists a unique polynomial $f(z) \in \mathbb{F}_n(z)$ of degree $< n'$ for which $f(j_\ell) = i_\ell$ for each $\ell \in [n']$. As a consequence, there is a unique "extension" of the monomial $x_{i_1,j_1} \cdots x_{i_{n'},j_{n'}}$ that appears as a term in NW, which is precisely $m_1 \cdot \prod_{j \in \mathcal{N}_w} x_{f(j),j}$. Therefore, all but one of the entries in the row corresponding to m_1 must be zero, and the remaining entry must be 1. Applying the same argument to the columns of $\mathcal{M}_w(NW)$, we deduce that $\mathcal{M}_w(NW)$ is a permutation matrix, and so has full rank. ◁

The following is a more precise and general version of Theorem 1 that is stated in Section 1. We also incorporate Remark 2 here and show our lower bound for any degree $d \leq n$. Theorem 1 follows from the special case $d = n$.

▶ **Theorem 9.** *For an integer $n = 2^k$, let $\mathbb{F}_n$ be a field of size n. Let d, Δ be integers such that $d \leq n$ is large enough*[6] *and $\Delta \leq \log d / \log\log d$. Let X_i denote the set of n variables $\{x_{i,j} : j \in [d]\}$ and X be the tuple $(X_1, \ldots, X_d)$. Then, any set-multilinear formula family of product-depth Δ computing $NW_{n,d}(X)$ must have size at least $d^{\Omega(d^{1/\Delta}/\Delta)}$.*

Proof. We show that the symmetric relative rank of low-depth set-multilinear formulas is small with high probability in the lemma below, and then combine it with Claim 8 above to prove the desired bound.

▶ **Lemma 10.** *Let C be a set-multilinear formula of product-depth $1 \leq \Delta \leq \log d / \log\log d$ of size at most s which computes a polynomial that is set-multilinear with respect to the partition $(X_1, \ldots, X_d)$ where each $|X_i| = n$. Let $w \in \{k, -k\}^d$ be chosen uniformly at random. Then, we have*

$$\mathrm{relrk}_w(C) \leq s \cdot 2^{-\frac{kd^{1/\Delta}}{20}}$$

with probability at least $1 - s \cdot d^{-\frac{d^{1/\Delta}}{12\Delta}}$.

Proof. We prove the statement by induction on Δ.

If $\Delta = 1$, then $C = C_1 + \cdots + C_t$ where each C_i is a product of linear forms. So, for all $i \in [t]$, by Claim 7,

$$\mathrm{relrk}_w(C_i) = \prod_{i=1}^{d} 2^{-\frac{1}{2}|w_j|} = 2^{-\frac{kd}{2}}$$

where in the last step, we used the observation that regardless of the choice of w, $|w_j| = k$ for all $j \in [n]$. Hence, by the sub-additivity of relrk_w, with probability 1, we have

$$\mathrm{relrk}_w(C) \leq s \cdot 2^{-\frac{kd}{2}} \leq s \cdot 2^{-\frac{kd}{20}}.$$

Next, we assume the statement is true for all formulas of product-depth $\leq \Delta$. Let C be a formula of product-depth $\Delta + 1$. So, C is of the form $C = C_1 + \cdots + C_t$. Following an overall proof strategy similar to the one in [20], we say that a sub-formula C_i of size s_i is of type 1 if one of its factors has degree at least $T_\Delta = d^{\frac{\Delta}{\Delta+1}}$, otherwise we say it is of type 2.

[6] We only need d to be larger than some absolute constant.

Suppose $C_i = C_{i,1} \cdot \cdots \cdot C_{i,t_i}$ is of type 1 with, say, $C_{i,1}$ having degree at least T_Δ. Let $w^{i,1}$ be the corresponding word i.e., $w^{i,1} = w|_{S_1}$ if $C_{i,1}$ is set-multilinear with respect to $S_1 \subsetneq [d]$. If it has size $s_{i,1}$, then since it has product-depth at most Δ, it follows by induction that

$$\mathrm{relrk}_w(C_i) \leq \mathrm{relrk}_{w^{i,1}}(C_{i,1}) \leq s_{i,1} \cdot 2^{-\frac{kT_\Delta^{1/\Delta}}{20}} \leq s_i \cdot 2^{-\frac{kd^{1/(\Delta+1)}}{20}}$$

with probability at least

$$1 - s_{i,1} \cdot T_\Delta^{-\frac{T_\Delta^{1/\Delta}}{12\Delta}} \geq 1 - s_i \cdot d^{-\frac{d^{1/(\Delta+1)}}{12\Delta} \cdot \frac{\Delta}{\Delta+1}} = 1 - s_i \cdot d^{-\frac{d^{1/(\Delta+1)}}{12(\Delta+1)}}.$$

Now suppose that $C_i = C_{i,1} \cdot \cdots \cdot C_{i,t_i}$ is of type 2 i.e., each factor $C_{i,j}$ has degree $< T_\Delta$. Note that this forces $t_i > d/T_\Delta = d^{\frac{1}{\Delta+1}}$. As the formula is set-multilinear, $(S_1, \ldots, S_{t_i})$ form a partition of $[d]$ where each $C_{i,j}$ is set-multilinear with respect to $(X_\ell)_{\ell \in S_j}$ and C_i is set-multilinear with respect to $(X_\ell)_{\ell \in S}$. Let $w^{i,1}, \ldots, w^{i,t_i}$ be the corresponding decomposition, whose respective sums are denoted simply by $w_{S_1}, \ldots, w_{S_{t_i}}$.

From the properties of relrk_w (Claim 7), we have

$$\mathrm{relrk}_w(C_i) = \prod_{j=1}^{t_i} \mathrm{relrk}_{w^{i,j}}(C_{i,j}) \leq \prod_{j=1}^{t_i} 2^{-\frac{1}{2}|w_{S_j}|} = 2^{-\frac{1}{2}\sum_{j=1}^{t_i} |w_{S_j}|},$$

from which we observe that the task of upper bounding $\mathrm{relrk}_w(C)$ can be reduced to the task of lower bounding the sum $\sum_{j=1}^{t_i} |w_{S_j}|$, which is established in the following claim. For the sake of convenience, the choice of the alphabet for w below is scaled down to $\{-1, 1\}$.

▷ Claim 11. For large enough d, suppose $(S_1, \ldots, S_\ell)$ is a partition of $[d]$ such that each $|S_j| < T_\Delta = d^{\frac{\Delta}{\Delta+1}}$. Then, we have

$$\mathbb{P}_{w \sim \{-1,1\}^d} \left[\sum_{j=1}^{\ell} |w_{S_j}| < \frac{d^{1/(\Delta+1)}}{10} \right] \leq d^{-\frac{d^{1/(\Delta+1)}}{12}}.$$

Proof. We first show that without loss of generality, we may assume that each S_j has size "roughly" T_Δ. To see this, we apply the following *clubbing* procedure to the sets in the partition $(S_1, \ldots, S_\ell)$:

- Start with the given partition $(S_1, \ldots, S_\ell)$. At each step in the procedure, we shall "club" two of the sets in the partition according to the following rule.
- If there are two distinct sets S' and S'' in the current partition each of size $< T_\Delta/2$, we remove both of them and add their union $S' \cup S''$ to the partition.
- If the rule above is no longer applicable, then we have at most one set in the current partition of size $< T_\Delta/2$. If there is none, then we halt the procedure here. Otherwise, we union this set with any one of the other sets and then halt.

After the clubbing procedure, we are left with a partition $(S'_1, \ldots, S'_{\ell'})$ of $[d]$ such that $\frac{T_\Delta}{2} \leq |S'_j| \leq \frac{3T_\Delta}{2}$ for each $j \in [\ell']$, also implying that $\frac{2d^{1/(\Delta+1)}}{3} \leq \ell' \leq 2d^{1/(\Delta+1)}$. Through a repeated use of the triangle inequality, we see that $\sum_{j=1}^{\ell'} |w_{S'_j}| \leq \sum_{j=1}^{\ell} |w_{S_j}|$. Therefore, upper bounding the latter sum is a "smaller" event than upper bounding the former sum. Hence, it suffices to prove the statement of the claim with the assumption that $\frac{T_\Delta}{2} \leq |S_j| \leq \frac{3T_\Delta}{2}$ for each $j \in [\ell]$ (we henceforth drop the primed notation).

Now, in the event that the sum $\sum_{j=1}^{\ell} |w_{S_j}|$ is at most $\frac{d^{1/(\Delta+1)}}{10}$, since $\ell \geq \frac{2d^{1/(\Delta+1)}}{3}$, it follows that for at least half of the sets S_j, $w_{S_j} = 0$ (as $\frac{2}{3} - \frac{1}{10} = \frac{17}{30} > \frac{1}{2}$). By Stirling's approximation, it follows that for a fixed j, the probability

$$\mathop{\mathbb{P}}_{w\sim\{-1,1\}^d}\left[w_{S_j} = 0\right] \leq \sqrt{\frac{2}{\pi|S_j|}} \leq \sqrt{\frac{4}{\pi T_\Delta}} = \sqrt{\frac{4}{\pi}} \cdot \frac{1}{d^{\frac{\Delta}{2(\Delta+1)}}} < \frac{2}{d^{1/3}},$$

where in the final step, we used $\Delta \geq 2$. Therefore, the probability that this happens for $\ell/2$ distinct j is bounded by

$$\binom{\ell}{\ell/2} \cdot \left(\frac{2}{d^{1/3}}\right)^{\frac{\ell}{2}} < 2^{\ell} \cdot \left(\frac{2}{d^{1/3}}\right)^{\frac{\ell}{2}} = \left(\frac{2\sqrt{2}}{d^{1/6}}\right)^{\ell} \leq \left(\frac{2}{d^{1/9}}\right)^{d^{1/(\Delta+1)}} < d^{-\frac{d^{1/(\Delta+1)}}{12}},$$

where we used the bound $\ell \geq \frac{2d^{1/(\Delta+1)}}{3}$. ◁

The claim above and the preceding calculation immediately implies that for a sub-formula C_i of type 2,

$$\mathrm{relrk}_w(C_i) \leq s_i \cdot 2^{-\frac{kd^{1/(\Delta+1)}}{20}}$$

with probability at least $1 - d^{-\frac{d^{1/(\Delta+1)}}{12}} \geq 1 - s_i \cdot d^{-\frac{d^{1/(\Delta+1)}}{12(\Delta+1)}}$.

Next, by a union bound over $i \in [t]$ and the sub-additivity property of relrk_w, it follows that

$$\mathrm{relrk}_w(C) \leq \mathrm{relrk}_w(C_1) + \cdots + \mathrm{relrk}_w(C_t) \leq s_1 \cdot 2^{-\frac{kd^{1/(\Delta+1)}}{20}} + \cdots + s_t \cdot 2^{-\frac{kd^{1/(\Delta+1)}}{20}} = s \cdot 2^{-\frac{kd^{1/(\Delta+1)}}{20}}$$

with probability at least $1 - s \cdot d^{-\frac{d^{1/(\Delta+1)}}{12(\Delta+1)}}$, which concludes the proof of the lemma. ◀

Returning to the proof of the theorem, let C be a set-multilinear formula of product depth Δ of size s computing $NW_{n,d}(X)$. Suppose $s < d^{\frac{d^{1/\Delta}}{24\Delta}}$. Then, by Lemma 10, with probability at least $1 - d^{-\frac{d^{1/\Delta}}{24\Delta}}$,

$$\mathrm{relrk}_w(C) \leq s \cdot 2^{-\frac{kd^{1/\Delta}}{20}}.$$

But now, we can condition on the event that $w_{[d]} = 0$ (which occurs with probability $\Theta(\frac{1}{\sqrt{d}})$) to establish the existence of a word $w \in \{-k, k\}^d$ with $w_{[d]} = 0$ such that w satisfies $\mathrm{relrk}_w(C) \leq s \cdot 2^{-\frac{kd^{1/\Delta}}{20}}$. This is because of the asymptotic bound $\frac{1}{\sqrt{d}} \gg d^{-\frac{d^{1/\Delta}}{24\Delta}}$, which follows from the given constraints on the parameters d, Δ. Therefore, by Claim 8,

$$s \geq 2^{\frac{kd^{1/\Delta}}{20}} \cdot \mathrm{relrk}_w(C) = n^{\frac{d^{1/\Delta}}{20}}$$

which contradicts the assumption that $s < d^{\frac{d^{1/\Delta}}{24\Delta}}$. Thus, we conclude that $s \geq d^{\frac{d^{1/\Delta}}{24\Delta}} = d^{\Omega(d^{1/\Delta}/\Delta)}$. ◀

Next, we show the supplementary result (Theorem 3) mentioned in Section 1, stated more precisely below.

▶ **Theorem 12.** *For an integer $n = 2^k$, let $\mathbb{F}_n$ be a field of size n and suppose $d \leq n$ is large enough. Let X_i denote the set of n variables $\{x_{i,j} : j \in [n]\}$ and X be the tuple $(X_1, \ldots, X_d)$. Then, any set-multilinear formula family computing $NW_{n,d}(X)$ must have size at least $d^{\Omega(\log d)}$.*

Proof. We first need the following structural result, whose proof can be immediately extrapolated from [28] (see Lemma 13.3), where it is shown for multilinear and homogeneous formulas.

▶ **Lemma 13** (Product Lemma). *Assume that F is a formula with at most s leaves, and is set-multilinear with respect to the set partition $(X_1, \dots, X_d)$. Then, we can write*

$$F = \sum_{i=1}^{s} \prod_{j=1}^{\ell} F_{i,j}$$

where $\ell \geq \log_3 d$ and for each $i \in [s]$, the product $F_i = \prod_{j=1}^{\ell} F_{i,j}$ is also set-multilinear. Furthermore, the degrees of $F_{i,j}$ satisfy the following geometric decay property:

$$\left(\frac{1}{3}\right)^j d \leq \deg(F_{i,j}) \leq \left(\frac{2}{3}\right)^j d, \text{ and } \deg(F_{i,\ell}) = 1.$$

▶ **Lemma 14.** *Let F be a set-multilinear formula of size at most s which computes a polynomial that is set-multilinear with respect to the partition $(X_1, \dots, X_d)$ where each $|X_i| = n$. Let $w \in \{k, -k\}^d$ be chosen uniformly at random. Then, we have*

$$\operatorname{relrk}_w(C) \leq s \cdot 2^{-\frac{k \log d}{20}}$$

with probability at least $1 - s \cdot d^{-\frac{\log d}{60}}$.

Proof. We begin by writing F in the form that is given by Lemma 13. Now, because of the geometric decay of the degrees of $F_{i,j}$, we observe that for each $i \in [s]$, at least for the first $\frac{3\ell}{4}$ many values of j, $\deg(F_{i,j}) \geq d^{1/4}$. In other words, at least a *constant* fraction of the $F_{i,j}$s have their degrees at least *polynomially large* in d. This observation will be instrumental in establishing the following claim, which is akin to Claim 11 used in the proof of Lemma 10.

▷ Claim 15. For large enough d, suppose $(S_1, \dots, S_\ell)$ is a partition of $[d]$ such that $\left(\frac{1}{3}\right)^j d \leq |S_j| \leq \left(\frac{2}{3}\right)^j d$ for all $j \in [\ell]$, and $|S_\ell| = 1$. Then, we have

$$\mathop{\mathbb{P}}_{w \sim \{-1,1\}^d} \left[\sum_{j=1}^{\ell} |w_{S_j}| < \frac{\log d}{10} \right] \leq d^{-\frac{\log d}{60}}.$$

Proof. Consider the given event that $\frac{\log d}{10}$ exceeds the sum $\sum_{j=1}^{\ell} |w_{S_j}|$. As $\ell \geq \frac{\log d}{\log 3} > \frac{5 \log d}{8}$, it follows that for at least half of the sets S_j, $w_{S_j} = 0$ (since $\frac{5}{8} - \frac{1}{10} = \frac{21}{40} > \frac{1}{2}$). By the observation above, it also follows that at least for $\frac{\ell}{4}$ many of the *first* $\frac{3\ell}{4}$ values of j, $w_{S_j} = 0$. But for a fixed such j, since $|S_j| \geq d^{1/4}$, the probability

$$\mathop{\mathbb{P}}_{w \sim \{-1,1\}^d} \left[w_{S_j} = 0 \right] \leq \sqrt{\frac{2}{\pi |S_j|}} < \frac{1}{\sqrt{|S_j|}} \leq \frac{1}{d^{1/8}},$$

Therefore, the probability that this happens for $\ell/4$ distinct j amongst the first $\frac{3\ell}{4}$ values of j is bounded by

$$\binom{3\ell/4}{\ell/4} \cdot \left(\frac{1}{d^{1/8}}\right)^{\frac{\ell}{4}} < 2^{3\ell/4} \cdot \left(\frac{1}{d^{1/8}}\right)^{\frac{\ell}{4}} < \left(\frac{2}{d^{1/32}}\right)^{\ell} < d^{-\frac{\log d}{60}}. \qquad \triangleleft$$

By sub-additivity of relrk_w (Claim 7), we have

$$\mathrm{relrk}_w(F) \leq \mathrm{relrk}_w(F_1) + \cdots + \mathrm{relrk}_w(F_s). \tag{1}$$

So, fix an $i \in [s]$. As the formula is set-multilinear, let $(S_1, \ldots, S_\ell)$ be the partition of $[d]$ such that each $F_{i,j}$ is set-multilinear with respect to $(X_t)_{t \in S_j}$. Let $w^{i,1}, \ldots, w^{i,\ell}$ be the corresponding decomposition, whose respective sums are denoted by $w_{S_1}, \ldots, w_{S_\ell}$. Then, by Claim 15,

$$\mathrm{relrk}_w(F_i) = \prod_{j=1}^{\ell} \mathrm{relrk}_{w^{i,j}}(F_{i,j}) \leq \prod_{j=1}^{\ell} 2^{-\frac{1}{2}|w_{S_j}|} = 2^{-\frac{1}{2}\sum_{j=1}^{\ell}|w_{S_j}|} \leq 2^{-\frac{k \log d}{20}}$$

with probability at least $1 - d^{-\frac{\log d}{60}}$. Therefore, by a union bound over $i \in [s]$ and (1), we conclude that

$$\mathrm{relrk}_w(F) \leq s \cdot 2^{-\frac{k \log d}{20}}$$

with probability at least $1 - s \cdot d^{-\frac{\log d}{60}}$. ◀

Returning to the proof of the theorem, let F be a set-multilinear formula of size s computing $NW_{n,d}$. Suppose $s < d^{\frac{\log d}{120}}$. Then, by Lemma 14, with probability at least $1 - d^{-\frac{\log d}{120}}$,

$$\mathrm{relrk}_w(F) \leq s \cdot 2^{-\frac{k \log d}{20}}.$$

But now, we can condition on the event that $w_{[d]} = 0$ (which occurs with probability $\Theta(\frac{1}{\sqrt{d}})$) to establish the existence of a word $w \in \{-k, k\}^d$ with $w_{[d]} = 0$ such that w satisfies $\mathrm{relrk}_w(F) \leq s \cdot 2^{-\frac{k \log d}{20}}$. This is because of the trivial asymptotic bound $\frac{1}{\sqrt{d}} \gg d^{-\frac{\log d}{120}}$. Therefore, again by Claim 8,

$$s \geq 2^{\frac{k \log d}{20}} \cdot \mathrm{relrk}_w(F) = n^{\frac{\log d}{20}}$$

which contradicts the assumption that $s < d^{\frac{\log d}{120}}$. Thus, we conclude that $s \geq d^{\frac{\log d}{120}} = d^{\Omega(\log d)}$. ◀

4 Discussion and Open Problems

We conclude by mentioning some interesting directions for future work.

- The most interesting and natural question is to make the hard polynomial in our main result $\mathrm{IMM}_{n,n}$. This would imply super-polynomial algebraic formula lower bounds. As far as we know, it is conceivable that our complexity measure could be used to prove the lower bound for the $\mathrm{IMM}_{n,n}$ polynomial. While the relative rank of $\mathrm{IMM}_{n,n}$ itself is low, there might be a suitable "restriction" of it such that for a randomly chosen $w \in \{-k, k\}^n$, with reasonably high probability the restriction has large rank. This could then be used to prove the lower bound for $\mathrm{IMM}_{n,n}$ (using Lemma 10 or Lemma 14). The result from [20] also showed its lower bound for the IMM polynomial by first analyzing a suitable restriction of IMM (although unfortunately that very same restriction idea does not work for us; please see the discussion in the appendix). Perhaps an intermediate question is to make the hard polynomial computationally simpler, for instance to find any hard polynomial that lies in VP.

- Another interesting question is to prove an improved depth hierarchy theorem for constant-depth set-multilinear formulas. [20] shows a depth hierarchy theorem for low-depth set-multilinear formulas. However, since their lower bounds only hold for small degrees, the depth hierarchy theorem in [20] only gives a quasi-polynomial separation of successive product-depths. It would be very interesting to obtain exponential separations (which for instance have been shown for low-depth multilinear circuits in [4]) using our measure.
- Another interesting direction could be to obtain lower bounds for general set-multilinear circuits via improved depth reduction results. The work of Kumar, Oliveira, and Saptharishi ([17]) provides some insight in this context, which shows an improved depth reduction to product-depth Δ with a size blow-up of $N^{O(\Delta\cdot(N/\log N)^{1/\Delta})}$ for multilinear circuits (regardless of degree). If a similar improvement (or any asymptotic improvement in the exponent) on the bound for general circuits from [30] could be shown to hold for set-multilinear circuits in the setting of Theorem 1 or Theorem 9 (i.e., when $N \geq d^2$), then combined with our lower bounds, this would imply super-polynomial set-multilinear circuit lower bounds. We should note that [7] rules out the possibility of obtaining a stronger reduction to depth-4, or $\Sigma\Pi\Sigma\Pi$ circuits, as it shows an $n^{\Omega(\sqrt{n})}$ size lower bound for set-multilinear depth-4 circuits computing $\mathrm{IMM}_{n,n}$, which of course has small polynomial-sized set-multilinear circuits. Nevertheless, there is still the possibility of obtaining improved depth reduction statements for product-depths 2 (which as noted earlier, is $\Sigma\Pi\Sigma\Pi\Sigma$ and hence more general than depth-4) or higher, and combining it with our Theorem 1 to obtain unbounded-depth set-multilinear circuit lower bounds. [18] shows a quasi-polynomial separation between the strength of homogeneous $\Sigma\Pi\Sigma\Pi$ and $\Sigma\Pi\Sigma\Pi\Sigma$ circuits, which could be considered as some evidence towards the validity of this possibility.

References

1. 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.
2. Miklós Ajtai. $\sum^1_1$-formulae on finite structures. *Ann. Pure Appl. Log.*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.
3. Peter Bürgisser. Cook's versus valiant's hypothesis. *Theor. Comput. Sci.*, 235(1):71–88, 2000. doi:10.1016/S0304-3975(99)00183-8.
4. 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. doi:10.1109/FOCS.2018.00092.
5. 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. doi:10.1137/18M1191567.
6. 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.
7. 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. doi:10.1137/140990280.

8 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

9 Johan Håstad. Almost optimal lower bounds for small depth circuits. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986. doi:10.1145/12130.12132.

10 Pavel Hrubes and Amir Yehudayoff. Homogeneous formulas and symmetric polynomials. *Comput. Complex.*, 20(3):559–578, 2011. doi:10.1007/s00037-011-0007-3.

11 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. doi:10.1137/151002423.

12 Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between read-once oblivious algebraic branching programs (roabps) and multilinear depth-three circuits. *ACM Trans. Comput. Theory*, 12(1):2:1–2:27, 2020. doi:10.1145/3369928.

13 Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 146–153. ACM, 2014. doi:10.1145/2591796.2591847.

14 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. An almost cubic lower bound for depth three arithmetic circuits. 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 33:1–33:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.33.

15 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. On the size of homogeneous and of depth-four formulas with low individual degree. *Theory Comput.*, 14(1):1–46, 2018. doi:10.4086/toc.2018.v014a016.

16 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

17 Mrinal Kumar, Rafael Oliveira, and Ramprasad Saptharishi. Towards optimal depth reductions for syntactically multilinear circuits. 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 78:1–78:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.78.

18 Mrinal Kumar and Ramprasad Saptharishi. Finer separations between shallow arithmetic circuits. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 38:1–38:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.38.

19 Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. *SIAM J. Comput.*, 46(1):336–387, 2017. doi:10.1137/140999335.

20 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 804–814. IEEE, 2021. doi:10.1109/FOCS52979.2021.00083.

21 Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Set-multilinear and non-commutative formula lower bounds for iterated matrix multiplication. *To appear in STOC*, 2022. URL: https://eccc.weizmann.ac.il/report/2021/094.

22 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.

23 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Comput. Complex.*, 6(3):217–234, 1997. doi:10.1007/BF01294256.

24 Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2):8:1–8:17, 2009. doi:10.1145/1502793.1502797.

25 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40:1–40:15, 2013. doi:10.1145/2535928.

26 Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Comput. Complex.*, 18(2):171–207, 2009. doi:10.1007/s00037-009-0270-8.

27 Alexander A. 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*, 41:333–338, 1987.

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

29 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.

30 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. doi:10.1016/j.ic.2014.09.004.

31 Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 1–10. IEEE Computer Society, 1985. doi:10.1109/SFCS.1985.49.

A Word Polynomials from [20, 21] and Our Measure

Both [20, 21] show their set-multilinear formula lower bounds for $\mathrm{IMM}_{n,d}$ by showing that small enough set-multilinear formulas have low relative rank and that a certain "restriction" of $\mathrm{IMM}_{n,d}$ has large relative rank. This restriction possesses the desirable property that if there is a small low-depth set-multilinear circuit computing $\mathrm{IMM}_{n,d}$, then there is one for this restriction as well. It is then natural to wonder if we can use these same restrictions for our *symmetric* measure and deduce strong lower bounds for IMM (in order to show super-polynomial general formula lower bounds as discussed), in addition to obtaining them for the NW polynomial. Unfortunately, it is straightforward to show that this is not possible, as we shall now see.

▶ **Definition 16** (Word Polynomials of [20, 21]). *Let $w \in \mathbb{R}^d$ be any word with non-zero entries. Say $X(w) = (X_1, \ldots, X_d)$ where each X_i has size $2^{|w_i|}$; we assume that the variables of X_i are labelled by strings in $\{0,1\}^{|w_i|}$.*

Given any monomial $m \in \mathbb{F}_{\mathrm{sm}}[\overline{X}(w)]$, let m_+ denote the corresponding "positive" monomial from $\mathcal{M}_w^{\mathcal{P}}$ and m_- the corresponding "negative" monomial from $\mathcal{M}_w^{\mathcal{N}}$. As each variable of $\overline{X}(w)$ is labelled by a Boolean string and each set-multilinear monomial over any subset of $\overline{X}(w)$ is associated with a string of variables, we can associate any such monomial m' with a Boolean string $\sigma(m')$. More precisely, if $j_1 < \cdots < j_t$ and $m' = x_{\sigma_1}^{(j_1)} x_{\sigma_1}^{(j_1)} \ldots x_{\sigma_t}^{(j_t)}$ with $x_{\sigma_i}^{(j_i)} \in X_{j_i}$ and $\sigma_i \in \{0,1\}^{|w_{j_i}|}$ for each $i \in [t]$, then $\sigma(m')$ is defined to be $\sigma_1 \cdots \sigma_t$. We will write $\sigma(m_+) \sim \sigma(m_-)$ when the shorter one is a prefix of the other one. The polynomial P_w is defined as follows

$$P_w = \sum_{\substack{m \in \mathbb{F}[\overline{X}(w)], \\ \sigma(m_+) \sim \sigma(m_-)}} m.$$

Clearly, the matrices $\mathcal{M}_w(P_w)$ are full-rank (i.e., have rank equal to either the number of rows or the number of columns, whichever is smaller). So, $\mathrm{relrk}_w(P_w) = 2^{-|w_{[d]}|/2}$.

In our measure, $w \in \{k, -k\}^d$ with $w_{[d]} = 0$ i.e., there is an equal number of positive and negative variable sets and an equal number of variables $n = 2^k$ in each set. Thus, in the sum above, $\sigma(m_+) \sim \sigma(m_-)$ gets replaced with $\sigma(m_+) = \sigma(m_-)$. The sum is indexed over all Boolean strings of length $kd/2$, and so there are $n^{d/2}$ terms in all. Moreover, there is a canonical bijection between the positive and negative variables: since $|\mathcal{P}_w| = |\mathcal{N}_w| = d/2$, if an element $j \in \mathcal{P}_w$ is the k-th largest element in $\mathcal{P}_w$, it corresponds to the k-th largest element j' in $\mathcal{N}_w$ such that $x_{i,j}$ appears in a monomial of P_w if and only if so does $x_{i,j'}$. Let $\phi : \mathcal{P}_w \to \mathcal{N}_w$ denote this correspondence. Then, we see that

$$P_w = \prod_{j \in \mathcal{P}_w} \sum_{i=1}^{n} x_{i,j} \cdot x_{i,\phi(j)},$$

implying that P_w actually has small depth-3 set-multilinear formulas.