Linear Algebraic Techniques in Algorithms and Complexity

by

Josh Alman

S.B., Massachusetts Institute of Technology (2014)
M.S., Stanford University (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Author . . . .
Department of Electrical Engineering and Computer Science
August 30, 2019

Certified by . . . .
R. Ryan Williams
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by . . . .
Virginia Vassilevska Williams
Steven and Renee Finn Career Development Associate Professor of
Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Linear Algebraic Techniques in Algorithms and Complexity

by

Josh Alman

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2019, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

We develop linear algebraic techniques in algorithms and complexity, and apply them to a variety of different problems. We focus in particular on *matrix multiplication algorithms*, which have surprisingly fast running times and can hence be used to design fast algorithms in many settings, and *matrix rank methods*, which can be used to design algorithms or prove lower bounds by analyzing the ranks of matrices corresponding to computational tasks.

First, we study the design of matrix multiplication algorithms. We define a new general method, called the Universal Method, which subsumes all the known approaches to designing these algorithms. We then design a suite of techniques for proving lower bounds on the running times which can be achieved by algorithms using many tensors and the Universal Method. Our main limitation result is that a large class of tensors generalizing the Coppersmith-Winograd tensors (the family of tensors used in all record-holding algorithms for the past 30+ years) cannot achieve a better running time for multiplying n by n matrices than $O(n^{2.168})$.

Second, we design faster algorithms for batch nearest neighbor search, the problem where one is given sets of data points and query points, and one wants to find the most similar data point to each query point, according to some distance measure. We give the first subquadratic time algorithm for the exact problem in high dimensions, and the fastest known algorithm for the approximate problem, for various distance measures including Hamming and Euclidean distance. Our algorithms make use of new probabilistic polynomial constructions to reduce the problem to the multiplication of low-rank matrices.

Third, we study rigid matrices, which cannot be written as the sum of a low rank matrix and a sparse matrix. Finding explicit rigid matrices is an important open problem in complexity theory with applications in many different areas. We show that the Walsh-Hadamard transform, previously a leading candidate rigid matrix, is in fact not rigid. We also give the first nontrivial construction of rigid matrices in a certain parameter regime with applications to communication complexity, using an efficient algorithm with access to an NP oracle.

Thesis Supervisor: R. Ryan Williams
Title: Associate Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Virginia Vassilevska Williams
Title: Steven and Renee Finn Career Development Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I want to thank my advisors Ryan Williams and Virginia Vassilevska Williams. They've supported me in every stage of my grad school career, from helping me to find research topics that I'm excited about, to encouraging me when I'm totally stuck on research problems, to teaching me how to write and communicate and apply for jobs. Working together with them has been a great experience, not only because of their endless streams of great research ideas, but also because of their strong passion for theoretical computer science and their desire to share it with others. Outside of work, Ryan and Virginia made life a pleasure, between hosting weekly music nights, feeding me chocolate, having long sports discussions with me in the middle of meetings, and just being great friends. I doubt any of the work in this thesis would have been possible without their support, and I hope this is only the beginning of our collaborations and friendships.

I also want to thank the other mentors I have had throughout my research career so far. Piotr Indyk, my third thesis committee member, also supervised my first computer science research experience as part of a final project for sublinear algorithms class. Greg Valiant was my rotation advisor during my first few months at Stanford; he helped get me acquainted with grad student life, and introduced Chebyshev polynomials to me. Vitaly Feldman and T.S. Jayram mentored me during a fruitful summer at IBM Almaden where I learned about some practical applications of theory. Pavlo Pylyavskyy advised me in the amazing REU in combinatorics at the University of Minnesota, Twin Cities, which has probably been the most productive couple of months of my life. Lionel Levine and Lorenzo Orecchia mentored me in research projects while I was an undergraduate at MIT, and taught me me how exciting research can be. All of these mentors helped to shape who I am as a researcher.

I want to thank everyone I've collaborated with, both for their contributions and for everything they've taught me: Arkadev Chattopadhyay, Timothy M. Chan, Lijie Chen, Timothy Chu, Cesar Cuenca, Vitaly Feldman, Jiaoyang Huang, Robin Hui, T.S. Jayram, Carl Lian, Dylan McKay, Matthias Mnich, Liat Peterfreund, Aaron Schild, Zhao Song, Brandon Tran, Virginia Vassilevska Williams, Greg Valiant, Nikhil Vyas, Joshua Wang, Ryan Williams, and Huacheng Yu. Research is much more enjoyable with such great people to work with.

I've had the unique experience of spending my grad school career at two wonderful schools: Stanford for the first half, and MIT for the second half. Both the Stanford and MIT theory groups are warm and welcoming communities full of great people to work, learn, and hang out with. I especially want to thank my colleagues and friends at both schools and throughout the TCS community, who made my time in grad school so memorable, including: Amir Abboud, Greg Bodwin, Vaggos Chatziafratis, Lijie Chen, Tobias Christiani, Michael Cohen, Ofir Geri, Daniel Grier, Robin Hui, Gautam Kamath, Michael Kim, Jerry Li, Andrea Lincoln, Quanquan Liu, Alex Lombardi, Nathan Manohar, Dylan McKay, Saeed Mehraban, Yonatan Naamad, Govind Ramnarayan, Luke Schaeffer, Adam Sealfon, Vatsal Sharan, Zhao Song, Warut Suksompong, Paris Syminelakis, Joshua Wang, Nicole Wein, and Huacheng Yu. Special shoutout to Dylan McKay, my only friend who would write a paper about League of

Legends with me [AM17].

Finally, I want to thank my parents Zena and Ben, my sister Sophie, my girlfriend DD, and my whole family, for always supporting me, even when they don't necessarily understand what I'm doing.

Contents

Chapter 1

Introduction

In this Chapter, we give a high-level introduction to this dissertation. After this Introduction and the Preliminaries in Chapter 2, we get into the technical components of the dissertation, which are divided into three main Parts: Part I is about matrix multiplication algorithms, Part II is about the polynomial method and nearest neighbor search algorithms, and Part III is about matrix rigidity. Each of those Parts begins with its own, more technical overview of the results therein.

1.1 Matrix Multiplication and Matrix Rank

Linear algebra is used throughout computer science, including in areas like error correcting codes, signal analysis, graphics, optimization algorithms, secure encryption and secret sharing schemes, graph analysis algorithms like PageRank, and a number of important machine learning algorithms. Understanding and improving the linear algebraic ideas underlying these different applications is imperative to the theory and practice of computation.

In this dissertation, we develop novel bridges between linear algebra and computer science. We will use linear algebraic techniques which were originally designed for certain tasks, and repurpose them to solve new problems. At the same time, rather than just using known linear algebraic techniques, many of our results will require new linear algebraic concepts and constructions. Although a number of known concepts from linear algebra will make appearances, there are two in particular which will play a central role: *matrix multiplication*, and *matrix rank*.

Matrix Multiplication Matrix multiplication is one of the most basic algebraic operations, and most computational tasks in linear algebra can be performed in the same number of arithmetic operations as matrix multiplication, including computing the determinant [Str69] or the inverse [Str69, BH74] of a matrix, computing various matrix factorizations [BH74], solving systems of linear equations [Str69], and even solving linear programs [CLS18]. Thus, almost every algorithmic application of linear algebra makes use of matrix multiplication algorithms, and the 'bottleneck' in designing faster algorithms for these applications is frequently matrix multiplication.

It was widely believed that two $n \times n$ matrices cannot be multiplied using fewer than n^3 arithmetic operations until 1969, when Strassen [Str69] published a breakthrough algorithm which uses only $O(n^{2.81})$ arithmetic operations. We measure the running time of matrix multiplication algorithms in terms of the number of required arithmetic operations since, when multiplying matrices with very large entries, the time to multiply and add those entries can also significantly contribute to the final running time. That said, typically in applications, the entries of the matrices are small enough that this contribution is negligible.

Since Strassen's algorithm, an enormous amount of work has gone into speeding up matrix multiplication. The best known theoretical algorithm for matrix multiplication uses a multitude of clever ideas coming from algorithm design, algebra, and combinatorics (see Figure 1-1 below). Matrix multiplication in practice has received as much if not more attention, with a number of hardware and software optimizations which yield fast practical algorithms.

Reducing other algorithmic problems to matrix multiplication gives a way to apply the same ideas and algorithms to a wide variety of computational tasks. Algorithmic problems from areas as diverse as parsing, graph algorithms, cryptography, statistics, and learning theory, which a priori have nothing to do with matrix multiplication, have been sped up by clever reductions to matrix multiplication. Hence, understanding the computational complexity of matrix multiplication is one of the most central and applicable problems in computer science.

Matrix Rank The rank of a matrix M measures the 'intrinsic dimensionality' of the rows and columns of M. In today's era of 'big data' and high-dimensional datasets, it is no surprise that matrix rank can play a role in *algorithm design*: the observation that some high-dimensional matrix actually has low rank can often be used to perform computations faster on that matrix. Indeed, most basic operations can be performed faster on low-rank matrices (see e.g. [CKL13]).

Matrix rank can also be helpful for proving *lower bounds*. Roughly, the idea is to show that if a matrix M has high rank (or a variant on rank), then M is so complicated that computations related to M cannot be done efficiently. Rank methods like this can be used in the most evident way to show lower bounds for algebraic problems directly related to M. For example, if M has high rank, then computing the linear transformation defined by M applied to an input vector cannot be done with small, constant-depth linear circuits. In fact, almost every known lower bound in arithmetic complexity theory has been proved via rank methods; see e.g. [EGOW18].

Rank methods can also be used to show lower bounds for functions and models of computation that are seemingly unrelated to linear algebra. Often one can associate a matrix M_f with a function f, and show that if the rank of M_f is high, then f itself cannot be computed by efficient algorithms or protocols. For example, if one can show that the *truth table matrix* of a Boolean function f hash high rank, then this implies f does not have efficient deterministic communication protocols [MS82].

1.2 Summary of Results

In this dissertation, we develop linear algebraic tools in algorithms and complexity, especially developing the theory of matrix multiplication algorithms, and introducing new rank methods in algorithms and complexity. We apply these tools, often in conjunction with each other, to solve a number of computer science problems, including problems where linear algebraic tools haven't been used before.

We now describe the main contributions of this dissertation. The main body of the dissertation is arranged into three Parts.

Part I: Limitations on Matrix Multiplication Algorithms

Since Strassen published his algorithm in 1969, there has been a long line of work developing many different tools for designing faster matrix multiplication algorithms, leading to the best known running time of about $O(n^{2.373})$ arithmetic operations [Wil12, LG14]. In Figure 1-1, we show the history of the best known exponent of matrix multiplication over time, i.e. the best upper bound on the constant ω such that one can multiply two $n \times n$ matrices using about $O(n^\omega)$ arithmetic operations. It is popularly conjectured that one can design an algorithm achieving $\omega = 2$, and this conjecture is very appealing since it would mean that matrix multiplication, along with many different applications, can be solved in nearly linear time in the input size.

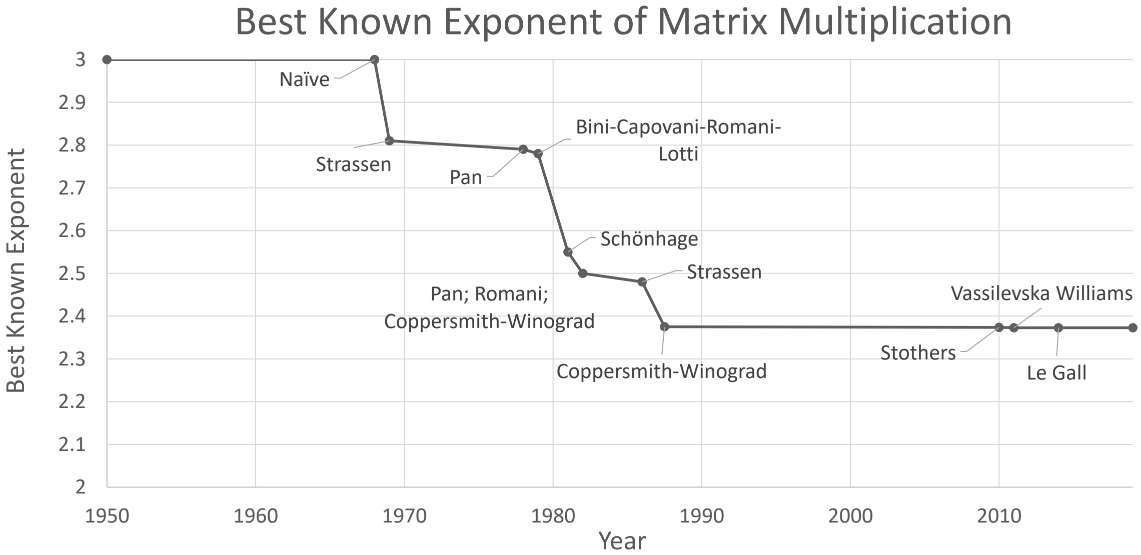

Figure 1-1: The best known exponent of matrix multiplication over time. For instance, Strassen's breakthrough algorithm from 1969 [Str69] multiplies $n \times n$ matrices using $O(n^{2.81})$ arithmetic operations, so the point $(1969, 2.81)$ is plotted above.

One striking feature of Figure 1-1 is that, although there was a flurry of improvements to ω in the first 20 years after Strassen's result, the best known value has remained nearly unchanged for more than 30 years. In Part I of this dissertation, we help to explain why progress has been mostly stagnant for so long.

The known approaches to designing matrix multiplication algorithms follow a formula which uses two main components:

1. An efficient algorithm for evaluating some order-3 tensor T. Matrix multiplication can be seen as the task of evaluating a prescribed set of bilinear polynomials on a given input, which can in turn be seen as evaluating a certain order-3 tensor on a given input; T corresponds to such a task but with a different set of bilinear polynomials.

2. A method of reducing from one tensor to another, so that algorithms for evaluating the latter can be converted into algorithms for evaluating the former.

Combined, these two components give a matrix multiplication algorithm, by using the method to reduce to T, and then applying the algorithm for T. Both major approaches to designing matrix multiplication algorithms — the 'Laser Method' spearheaded by Strassen [Str87] which is used to achieve the best current bound on ω, as well as the more recent Group-theoretic Method introduced by Cohn and Umans [CU03] — follow this formula, but with restrictions on what tensor T may be used in component (1), and only a limited type of reduction used in component (2). In particular, all the record-holding algorithms for the past 30+ years have come from such an approach where T is the 'Coppersmith-Winograd tensor' CW_q introduced by [CW90].

In Chapter 4, we define a new generalization of all the known approaches to designing matrix multiplication algorithms, which we call the *Universal Method*. It makes use of the most general type of reduction between tensors which is known to be applicable in component (2) of the above formula, called a *degeneration*.

Then, in Chapter 5, we prove *lower bounds* on the algorithms one can design using the Universal Method. We show that if the Universal Method is applied to any tensor in a big family generalizing CW_q, then the resulting upper bound on ω cannot be better than 2.168. Our limitation result is quite general, and also applies to all other record-holding tensors in the history of matrix multiplication algorithms. Hence, in order to prove a better bound on ω, one must take a radically different approach, either by starting with a tensor T which is very different from those which have led to the best algorithms, or else by analyzing tensors to yield matrix multiplication algorithms in an entirely new way.

Our limitation result, which at first seems to be a negative one, actually leads to a number of interesting algorithmic ideas. First, in defining the Universal Method itself, we highlight steps in the current best matrix multiplication algorithms where more powerful techniques may be possible but aren't being used; while our result rules out achieving a running time of $O(n^2)$ using these techniques, it doesn't rule out an improved running time of, say, $O(n^{2.2})$. Second, in the process of proving our limitation result, we also identify a large number of fundamentally different algorithms, arising from new, different tensors, which are able to *match* the best-known bound of $\omega \leq 2.373$. Perhaps one of these different algorithms will help improve our matrix multiplication algorithms.

The proof of our limitation result makes use of a measure of complexity of a tensor called its *slice rank*. When one generalizes the notion of matrix rank to tensors, there are a number of natural ways to do so; slice rank is one such generalization. Roughly, we observe that any tensor whose slice rank is not high enough is too simple to be used with the Universal Method to design very fast matrix multiplication algorithms. We then give slice rank upper bounds for CW_q and a wide variety of other tensors.

Part II: Probabilistic Polynomials and Hamming Nearest Neighbors

The *polynomial method* has been a powerful tool for studying Boolean functions, at least since Minsky and Papert's 1969 book [MP69]. The method concerns representing Boolean functions by 'simple' polynomials. Minsky and Papert first used the polynomial method as a way to prove limitations on different models of computation (they focused in particular on 'perceptrons'). Roughly, the idea is to show that

1. any function computed by a certain model of computation can also be computed by a simple polynomial, and
2. some particular Boolean function f *cannot* be computed by a simple polynomial.

Combined, this means the model of computation cannot compute the function f. This method is still one of the most popular approaches today for proving lower bounds in complexity theory.

There are numerous ways to measure the 'simplicity' of a polynomial, but the most common is to use the polynomial's degree. Consider, for instance, the Boolean OR function on n inputs from $\{0,1\}$. It can be computed exactly by the polynomial

$$p(x_1, x_2, \dots, x_n) = 1 - (1 - x_1) \cdot (1 - x_2) \cdots (1 - x_n).$$

Indeed, if all the x_i are 0, then p evaluates to 0, but if any of them is 1, then p evaluates to 1. However, p has degree n, which is as big as possible: *every* Boolean function on n inputs can be computed by some polynomial of degree at most n.

One way around this high degree is to weaken the constraints on what it means for a polynomial to 'compute' a function. For instance, instead of aiming for an exact polynomial for OR, we can instead design a *probabilistic polynomial*: A distribution $\mathcal{P}$ on n-input polynomials such that for every $(x_1, x_2, \dots, x_n) \in \{0,1\}^n$ we have

$$\Pr_{p \sim \mathcal{P}}[p(x_1, x_2, \dots, x_n) = \mathsf{OR}(x_1, x_2, \dots, x_n)] \geq 1 - \varepsilon$$

for some error parameter $\varepsilon > 0$. We can design such a probabilistic polynomial over $\mathbb{F}_2$ (the field with two elements, i.e. the integers mod 2) as follows: to draw a polynomial from $\mathcal{P}$, pick $k = \lceil \log(1/\varepsilon) \rceil$ independent uniformly random subsets $I_1, I_2, \dots, I_k \subseteq \{1, 2, \dots, n\}$, and output

$$p(x_1, x_2, \dots, x_n) = 1 - \prod_{\ell=1}^{k} \left(1 - \sum_{i \in I_\ell} x_i \right).$$

If all the x_i are 0, then p always evaluates to 0. Otherwise, for each $\ell \in \{1, 2, \dots, k\}$, the sum $\sum_{i \in I_\ell} x_i$ is odd with probability 1/2, and so the polynomial p evaluates to 1 (over $\mathbb{F}_2$) with error only $2^{-k} \leq \varepsilon$. $\mathcal{P}$ is a distribution on polynomials of degree only $O(\log(1/\varepsilon))$; for constant $\varepsilon > 0$, this is constant degree!

The polynomial method is concerned with trade-offs like the above: between the guarantees of a polynomial, and the degree or simplicity that can be achieved. Probabilistic polynomials, for instance, can achieve much lower degrees than exact polynomials, but they have a chance of outputting the wrong value.

While complexity theorists see these low-degree polynomial representations as a weakness of the computational model, algorithms designers can instead view them as algorithmic tools: If a critical subroutine in an algorithm can be converted into a low-degree polynomial, then a fast algorithm for manipulating polynomials can sometimes be applied to speed up that subroutine, and solve the original problem faster. This viewpoint has led to the same polynomials, which complexity theorists designed for proving lower bounds, being used in the design of faster algorithms for many problems, including in learning theory [Val15], constraint satisfaction [Wil14c], and graph algorithms [Wil14a].

In Part II of this dissertation, we apply the polynomial method in novel ways to design new algorithms and prove new lower bounds. Our results critically make use of a connection between low-degree polynomials and low-rank matrices: if the entries of a matrix can be computed by a low-degree polynomial, then that polynomial can be used to construct a low-rank representation of the matrix. This can allow us to use fast matrix multiplication as the "fast algorithm for manipulating polynomials" from the previous paragraph. In other words, the polynomial method can be seen as a way to design faster algorithms for many different algorithmic problems by giving reductions to matrix multiplication, allowing us to take advantage of fast matrix multiplication algorithms.

In Chapter 7, we design new low-degree polynomial representations of Boolean functions. We focus in particular on polynomial representations of *threshold functions* like the majority function MAJ, although our results will extend to symmetric Boolean functions as well as a number of classes of Boolean circuits. Threshold functions arise naturally in many settings, including in linear programming, in machine learning algorithms like perceptrons and neural networks, and in nearest neighbor search (as we will discuss shortly).

We first construct a probabilistic polynomial for MAJ on n inputs with error ε and degree $O(\sqrt{n \log(1/\varepsilon)})$. This matches a classical $\Omega(\sqrt{n \log(1/\varepsilon)})$ degree lower bound due to Razborov [Raz87] and Smolensky [Smo87]; they originally introduced probabilistic polynomials and proved this degree lower bound in order to show a circuit lower bound, that MAJ cannot be computed by AC^0 circuits.

We then show it is possible to circumvent Razborov and Smolensky's lower bound and achieve even lower degree polynomials. To do this, we consider a new generalization of a probabilistic polynomial which we call a *probabilistic polynomial threshold function* (probabilistic PTF). While a probabilistic polynomial for a function f must exactly compute f on any given input with high probability, a probabilistic PTF must only output a positive real number when f is true, and a negative real number when f is false, with high probability. It is easy to construct a degree 1 probabilistic PTF for MAJ (in fact, randomness isn't even needed), but we aim to design a probabilistic

PTF for an OR of many MAJ functions, which is much less straightforward. One way to do this is to sum together independent copies of our probabilistic polynomial for MAJ, resulting in a probabilistic PTF for an OR of $O(1/\varepsilon)$ different MAJs, each on n inputs, with degree $O(\sqrt{n \log(1/\varepsilon)})$. However, we are able to improve on this construction and achieve degree only $O(n^{1/3} \log^{2/3}(n/\varepsilon))$. Our construction combines ideas from the design of randomized algorithms with 100-year-old constructions from polynomial approximation theory, especially the Chebyshev polynomials [Che99].

New Nearest Neighbor Search Algorithms Next, in Chapter 8, we apply our polynomial constructions to design new algorithms as well as prove new lower bounds. Our main algorithmic application is for *nearest neighbor search problems*. In the (batch) nearest neighbor search problem, one is given as input n data points, and n query points, and the goal is to find the nearest data point to each query point. Nearest neighbor search has applications in almost every domain, including computational geometry, coding theory, pattern recognition, and DNA sequencing. There are, of course, many different settings of this problem depending on the specific details.

First, one needs to specify what types of data points can be input, and how one should measure the distance between them. Some natural choices include:

- points from the d-dimensional Boolean hypercube, $\{0,1\}^d$, with distance measured by the Hamming distance, which counts the number of the d entries in which two points differ,
- points from d-dimensional Euclidean space, $\mathbb{R}^d$, with distance measured by the standard Euclidean metric,
- points which are d-dimensional vectors of real numbers, from $\mathbb{R}^d$, along with a *different* distance measure like the ℓ_1 distance (also known as Manhattan distance), and
- points which are subsets of a large universe, with distance measured by the Jaccard index, which equals the ratio of the size of the symmetric difference and the size of the union of the two sets.

Second, one needs to choose whether *exact* nearest neighbors are necessary, or whether *approximate* nearest neighbors are sufficient. In the $(1+\varepsilon)$-approximate nearest neighbor problem for some small constant $\varepsilon > 0$, it is sufficient to find, for each query point, a data point which is within a $(1+\varepsilon)$ factor of the distance to the actual nearest data point. In many applications, finding approximate nearest neighbors is sufficient, and such a relaxation can allow for substantially faster algorithms.

In all these choices of settings, there is a brute force quadratic-time algorithm, which simply iterates over all pairs of points and computes their distances. However, quadratic time can be too slow in applications with many data points! Using the polynomial method, we give the fastest known, subquadratic time algorithm for

- the approximate problem for all of the aforementioned distance measures, and
- the exact problem for most of the distance measures, in high dimensions where subquadratic time algorithms weren't previously known.

To give two examples:

For the exact batch nearest neighbor search problem with Hamming distance, in dimension $d = c \log n$, we design an algorithm with running time $n^{2-1/\widetilde{O}(\sqrt{c})}$. For any constant c, this is a truly subquadratic running time (i.e. time $O(n^{2-\delta})$ for some constant $\delta > 0$). Previously, no truly subquadratic time algorithm was known even in dimension, say, $d = 2 \log n$.

For the $(1+\varepsilon)$-approximate batch nearest neighbor search problem with Hamming distance, in any dimension d, we design an algorithm with running time $dn+n^{2-\widetilde{\Omega}(\varepsilon^{1/3})}$. This algorithm runs in subquadratic time, even up to dimension $d = n^{1-\delta}$. For small enough constant $\varepsilon > 0$, this improves on the previous best running time of $dn + n^{2-\Omega(\varepsilon^{1/2})}$ [Val15], as well as running times of $dn + n^{2-\Omega(\varepsilon)}$ which one can achieve using techniques like Locality-Sensitive Hashing [IM98].

We also achieve nearly identical results for Euclidean distance, ℓ_1 distance, Jaccard distance, and many other choices of distance measure instead of Hamming distance. We complement our algorithms with new conditional lower bounds, showing that if one can design algorithms which are significantly faster than the ones we design here, it would refute a popular conjecture from complexity theory (the 'Strong Exponential Time Hypothesis' [IPZ01]) about the time required to solve the Boolean satisfiability problem.

Because threshold functions are so versatile, we apply our polynomial constructions to other applications as well, including basic problems in data analysis and statistics, constraint satisfaction problems like MAX-SAT, and new circuit lower bounds for circuits with threshold gates.

Part III: Probabilistic Rank and Matrix Rigidity

Informally, a matrix is called *rigid* if it has high rank, and one must change many of its entries before it has low rank. Of course there are parameters involved: the rank-r rigidity of a $N \times N$ matrix M, denoted $\mathscr{R}_M(r)$, is the minimum number of entries of M which one must change in order to make its rank at most r.

Consider, for example, the $N \times N$ identity matrix I_N. Although I_N has full rank, it is not particularly rigid: each time a 1 on the diagonal changes to a 0, the rank of I_N decreases by one. Hence, for all ranks r, we have $\mathscr{R}_{I_N}(r) \leq N - r$. In fact, since changing one entry of a matrix can never decrease the rank by more than one, the identity matrix I_N is as non-rigid as a full rank matrix can be!

The above example can be summarized by saying that the identity matrix is not rigid because it is *sparse*. We could try to get around this to find a rigid matrix by considering simple dense matrices instead, like perhaps the $N \times N$ upper triangular matrix U_N with all 1s above the diagonal. However, with some work one can show that U_N isn't very rigid either. We could then move on to even more complicated matrices like a Vandermonde or Fourier matrix, and although these seem more rigid, it's hard to prove that this is the case. In fact, it's an open problem to show that any *explicit* matrices, like these, are rigid (for certain parameters we describe in the next paragraph).

Finding explicit rigid matrices has been a central open challenge in complexity ever since the notion of rigidity was introduced by Leslie Valiant in 1977 [Val77]. At a high level, rigid matrices are of interest because they are "inherently complicated":

following the outline of using rank methods for proving lower bounds, complexity theorists have shown that explicit rigid matrices would yield new lower bounds for several different models of computation. The two most interesting rigidity parameter regimes for a family $\{M_N\}_{N \in \mathbb{N}}$ of matrices, where M_N is a $N \times N$ matrix, are as follows:

- $\{M_N\}_{N \in \mathbb{N}}$ is called *Valiant-rigid* if there is a constant $\varepsilon > 0$ such that

  $$\mathscr{R}_{M_N}(N/\log\log N) \geq \Omega(N^{1+\varepsilon}).$$

  Valiant [Val77] showed that the linear transformations corresponding to Valiant-rigid matrices cannot be computed by $O(N)$-size $O(\log N)$-depth arithmetic circuits. There are currently no known lower bounds showing that such circuits cannot compute any explicit families of matrices.

- $\{M_N\}_{N \in \mathbb{N}}$ is called *Razborov-rigid* if there is any super-constant function $\alpha(N) = \omega(1)$ such that

  $$\mathscr{R}_{M_N}(2^{(\log\log N)^{\alpha(N)}}) \geq \Omega(N^2).$$

  Razborov [Raz89] (see also [Wun12]) showed that if the communication matrix M_f of a Boolean function f is Razborov-rigid, then f is not in $\mathsf{PH}^{\mathsf{cc}}$, the communication analogue of the polynomial hierarchy. There are currently no explicit Boolean functions known to be outside $\mathsf{PH}^{\mathsf{cc}}$.

In other words, M_N is Valiant-rigid if you have to change a super-linear number of its entries to make its rank drop to *barely* sublinear, and M_N is Razborov-rigid if reducing it to a *tiny* rank requires changing a constant fraction of its entries. Showing that an *explicit* family of matrices $\{M_N\}_{N \in \mathbb{N}}$ is rigid in either regime would imply new, breakthrough lower bounds in complexity theory.

We have used the word ‘explicit’ a number of times; what does it mean? We say $\{M_N\}_{N \in \mathbb{N}}$ is explicit if there is a deterministic algorithm which, on input N, outputs the matrix M_N in poly(N) time. Aiming for a deterministic algorithm is important, since random matrices are known to be very rigid. For instance, for any rank $r = o(N)$, a random $\{0,1\}$ matrix $M_N \in \{0,1\}^{N \times N}$ over any field has rigidity $\mathscr{R}_{M_N}(r) \geq \Omega(N^2)$ with high probability, and is hence both Valiant-rigid and Razborov-rigid. This is not particularly exciting, since random functions are already known to require large circuits and inefficient communication protocols. Aiming for a poly(N) time construction is also important since, with enough time, an algorithm could simply brute force over, say, every $N \times N$ matrix over $\mathbb{F}_2$, and compute the rigidity of every one. Beyond these details, finding and understanding explicit rigid matrices is important, since these are matrices which are efficiently computable in one sense, but inherently inefficient in other (e.g. can’t be computed by super-linear size circuits or $\mathsf{PH}^{\mathsf{cc}}$ protocols).

To summarize: although rigidity was defined more than 40 years ago, and explicit rigid matrices are known to have many important applications throughout complexity theory, there are still no known constructions of explicit rigid matrices. In Part III of this dissertation, we make new progress on this problem, both by explaining the weakness of current attempts at proving matrices are rigid, and by giving new nontrivial constructions of rigid matrices. We approach the problem by combining linear algebra with techniques from algorithm design and complexity theory which haven't been used before in this setting, including fast matrix multiplication algorithms, the polynomial method, and a new generalization of matrix rank which we call probabilistic rank.

Hadamard Matrices Are Not Rigid Among the many attempts to prove lower bounds via rigidity, perhaps the most commonly studied explicit matrix has been the Walsh-Hadamard transform [PS88, Alo90, Gri, Nis, KR98, Cod00, Lok01, LTV03, Mid05, dW06b, Ras16]. The Walsh-Hadamard transform is a family of matrices $\{H_n\}_{n\in\mathbb{N}}$ with $H_n \in \{-1,1\}^{2^n\times 2^n}$, defined recursively as:

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \text{ and } H_{n+1} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix} \text{ for } n \geq 1.$$

H_n is a Hadamard matrix (i.e. its rows are mutually orthogonal) corresponding to the discrete Fourier transform for the power of the cyclic group C_2^n, and it has applications in areas like quantum computing, signal processing, and data compression. These properties were believed to imply that H_n must be rigid; in fact, many of the references above were working toward proving that *every* Hadamard matrix is rigid. The best known rigidity lower bounds for H_n for rank r are that $\mathscr{R}_{H_n}(r) \geq \Omega(4^n/r)$, which is insufficient to show they are Valiant-rigid or Razborov-rigid.

In Chapter 10, we partially explain why these rigidity lower bounds are not stronger: we *refute* the popular conjecture that H_n is rigid, and show that it is *not* Valiant-rigid, by giving a new rigidity upper bound. More precisely, letting $N = 2^n$ be the side-length of H_n, we show that for every field $\mathbb{F}$, and all sufficiently small constants $\varepsilon > 0$,

$$\mathscr{R}_{H_n}(N^{1-\tilde{\Theta}(\varepsilon^2)}) \leq N^{1+\varepsilon}$$

over $\mathbb{F}$. The choice of field $\mathbb{F}$ can sometimes make a difference in how rigid a matrix is (for instance, H_n has constant rank over $\mathbb{F}_2$), but our rigidity upper bound works over any field.

Our proof makes use of a new generalization of matrix rank we introduce, called *probabilistic rank*. Generalizing the notion of a probabilistic polynomial, we say that a matrix H has probabilistic rank r for error ε if there is a distribution $\mathcal{M}$ on matrices of the same dimensions as H and rank at most r such that, for every entry $H[i,j]$ of the matrix H, we have

$$\Pr_{M\sim\mathcal{M}}[H[i,j] = M[i,j]] \geq 1-\varepsilon.$$

It is not hard to see that a 'typical' matrix M from the probabilistic matrix $\mathcal{M}$ is a rank r matrix which differs from H in only an ε fraction of its entries, hence giving a rigidity upper bound for H. However, in principle, there could be better rigidity upper bounds than this, since a rigidity upper bound only requires a small total number of errors, whereas a probabilistic matrix is required to have a low probability of error on *every* entry of H. In fact, we show that the two notions are *equivalent* for many families of matrices including the Walsh-Hadamard transform H_n, so it suffices to focus on giving probabilistic rank upper bounds for H_n.

Our new probabilistic rank upper bounds are inspired by our earlier use of the polynomial method in algorithm design. In order to show probabilistic rank upper bounds, we show that probabilistic rank also generalizes probabilistic polynomial degree. When using the polynomial method to design new algorithms, we used the fact that low-degree polynomials which compute the entries of a matrix give rank upper bounds for that matrix. Similarly, if the entries of a matrix are computed by low-degree probabilistic polynomials, then the same connection gives a probabilistic rank upper bound for that matrix. Our rigidity upper bound for H_n critically makes use of a new probabilistic polynomial construction for most of the rows and columns of H_n.

Efficient Construction of Rigid Matrices in P^{NP} Finally, in Chapter 11, we give a new construction of rigid matrices. We give a family of matrices $\{M_N\}_{N\in\mathbb{N}}$, with $M_N \in \{0,1\}^{N\times N}$, which is Razborov-rigid (infinitely often), and which can be constructed in deterministic poly(N) time *with access to an NP oracle.* More precisely, for infinitely many N, our $N \times N$ matrix M_N has the rigidity bound

$$\mathscr{R}_{M_N}(2^{(\log N)^{1/4-\varepsilon}}) \geq \Omega(N^2)$$

for any $\varepsilon > 0$ over any constant-sized finite field $\mathbb{F}_q$.

This is the first nontrivial construction of Razborov-rigid matrices which doesn't use randomness. Although it doesn't qualify as an explicit construction in the sense we previously described (since the construction uses an NP oracle), it still implies a number of new lower bounds in complexity theory in conjunction with the various known applications of rigid matrices, including:

- There is a function in $\mathsf{TIME}[2^{(\log n)^{\omega(n)}}]^{\mathsf{NP}}$ which is not in $\mathsf{PH}^{\mathsf{cc}}$. Here, $\mathsf{PH}^{\mathsf{cc}}$ is the communication complexity analogue of the polynomial hierarchy, consisting of functions with efficient communication protocols that can make use of alternating nondeterministic and co-nondeterministic guesses by the two players. It was previously even open whether every function in the larger class $\mathsf{TIME}[2^{O(n)}]^{\mathsf{NP}}$ is also in $\mathsf{AM}^{\mathsf{cc}}$, an important subclass of $\mathsf{PH}^{\mathsf{cc}}$.

- Depth-2 linear circuits for computing the linear transformation defined by the $N \times N$ matrix M_N described above require size $\Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$. The previous best nontrivial such lower bounds for non-randomly constructed matrices were at best $\Omega(N \cdot \log^2 N/\log\log N)$.

Our construction takes a very different approach from past attempts at constructing rigid matrices. While past constructions have mainly used tools from algebra and combinatorics, our construction is instead inspired by circuit complexity theory. The main idea is to view low-rank expressions for matrices as a special type of 'circuit class' for computing matrices. In this way, finding a rigid matrix is equivalent to proving a certain average-case lower bound against these 'circuits'.

In order to prove this circuit lower bound, we use the *algorithmic* approach introduced by Williams [Wil13]. Very roughly, Williams' approach shows that fast, deterministic algorithms for analyzing circuits (e.g. for counting the number of satisfying assignments to a circuit) can be used to prove lower bounds against those circuits. Tools from linear algebra, algorithm design, and complexity theory come into play as we design the appropriate circuit analysis algorithm and use it to prove our rigidity bound. For instance, we use an algorithm for counting the number of 1s in a low-rank matrix by Chan and Williams [CW16], which applies the polynomial method and fast matrix multiplication.

1.3 Bibliographic Details

This dissertation is based off of the results in eight previously published papers:

- 'Further Limitations of the Known Approaches for Matrix Multiplication' with Virginia Vassilevska Williams [AW18a], which appeared in ITCS 2018,
- 'Limits on All Known (and Some Unknown) Approaches to Matrix Multiplication' with Virginia Vassilevska Williams [AW18b], which appeared in FOCS 2018,
- 'Limits on the Universal Method for Matrix Multiplication' [Alm19b], which appeared in CCC 2019 and won the Best Student Paper Award,
- 'Probabilistic Polynomials and Hamming Nearest Neighbors' with Ryan Williams [AW15], which appeared in FOCS 2015,
- 'Polynomial Representations of Threshold Functions and Algorithmic Applications' with Timothy M. Chan and Ryan Williams [ACW16], which appeared in FOCS 2016,
- 'An Illuminating Algorithm for the Light Bulb Problem' [Alm19a], which appeared in SOSA 2019,
- 'Probabilistic Rank and Matrix Rigidity' with Ryan Williams [AW17], which appeared in STOC 2017, and
- 'Efficient Construction of Rigid Matrices Using an NP Oracle' with Lijie Chen [AC19], which will appear in FOCS 2019.

See the Introduction to each Part for more specific details about which results correspond to each reference.

Chapter 2

Preliminaries

We assume familiarity with basic facts about algorithms, complexity, combinatorics, probability, and algebra (especially linear algebra and properties of polynomials). That said, in this Chapter, we will review some of the less common notions from these areas which will play important roles throughout this dissertation. We begin first by discussing the notation we will use.

2.1 Notation

Sets, Vectors, and Matrices We use the standard notation for common sets of numbers: $\mathbb{Z}$ is the set of integers, $\mathbb{N} := \{1, 2, 3, \ldots\}$ is the set of natural numbers, $\mathbb{R}$ is the set of real numbers, and $\mathbb{C}$ is the set of complex numbers. For $n \in \mathbb{N}$, we write $[n] := \{1, 2, \ldots, n\}$, and $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ to denote the integers mod n.

For any set S, $n \in \mathbb{N}$, $i \in [n]$, and n-dimensional vector $v \in S^n$, we write $v_i \in S$ for the ith entry of v. We may sometimes write $v(i)$ or $v[i]$ instead of v_i to avoid ambiguity with other subscripts; the meaning should be clear from context. When we say $v \in S^n$ is 'indexed by X' for a set X of size $|X| = n$, we implicitly define a bijection $m_X : X \to [n]$, and for $x \in X$ define $v_x := v_{m_X(x)}$.

Similarly, for a $n \times m$ matrix $M \in S^{n \times m}$ over set S, and for $i \in [n]$ and $j \in [m]$, we write $M(i, j)$ (or sometimes $M[i, j]$) for the (i, j)th entry of M. We may also write $M[i, :]$ for the ith row of M, or $M[:, j]$ for the jth column of M. If X, Y are sets of size $|X| = n$ and $|Y| = m$, then we can index the entries of M by X, Y and refer to entry $M(x, y)$ for $x \in X$ and $y \in Y$ (using similar implicit bijections m_X and m_Y as above).

Logarithms and Asymptotics We write $\log_b$ for the base b logarithm. We also write log for $\log_2$ and ln for $\log_e$ for short.

We use the standard symbols from asymptotic analysis: O, o, Ω, ω, and Θ. We write $f = \text{poly}(n)$ if there is a constant $c \geq 0$ such that $f(n) = O(n^c)$, and write $\text{polylog}(n) := \text{poly}(\log(n))$. If f, g are functions of many variables including n, we write $f = O_n(g)$ to mean that $f = O(g)$ when n grows and all variables other than n

are fixed to any constant values. We write $f = \tilde{O}(g)$ to ignore polylog factors, i.e. if there is a constant $c \in \mathbb{Z}$ such that $f = O(g \cdot \log^c(g))$. Define $\tilde{\Omega}$ similarly.

Boolean functions A Boolean function is a function $f : \{0,1\}^n \to \{0,1\}$ for some $n \in \mathbb{N}$, where we think of 0 as 'false' and 1 as 'true'. Some simple Boolean functions which will recur include:

- OR, which outputs 1 unless all its inputs are 0. We sometimes also write OR_n to emphasize the number n of inputs.

- AND, which outputs 1 when all its inputs are 1.

- XOR, which outputs 1 when an odd number of its inputs are 1. For $x, y \in \{0,1\}$ we will write $x \oplus y := \mathsf{XOR}(x,y)$, and for $x \in \{0,1\}^n$ we write $\bigoplus_{i=1}^n x_i := \mathsf{XOR}(x_1, \ldots, x_n)$. (The symbol $\oplus$ will also be used for direct sum, but it will be clear from context which of the two meanings we are using.)

- MOD_m for $m \in \mathbb{N}$, which outputs 1 when a multiple of m of its inputs are 1. In particular, for any $x \in \{0,1\}^n$, $\mathsf{MOD}_2(x) = 1 - \mathsf{XOR}(x)$.

- MAJ (MAJORITY), which outputs 1 when at least half of its inputs are 1.

For $x \in \{0,1\}^n$, we write $|x| := \sum_{i=1}^n x_i \in \mathbb{Z}$ to denote the *Hamming weight* of x. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called *symmetric* if $f(x) = f(y)$ for any $x, y \in \{0,1\}^n$ such that $|x| = |y|$. All the aforementioned Boolean functions are symmetric.

For a logical predicate P, we use Iverson bracket notation $[P]$ to denote the function which outputs 1 when P is true, and 0 when P is false. For instance, for $x \in \{0,1\}^n$, we have $\mathsf{MAJ}(x) = [|x| \geq n/2]$. Brackets '[,]' will also be used as parentheses for emphasis in some places; the meaning should be clear from context.

Groups, Rings, and Fields We will typically use multiplicative notation for the group operation of groups. Two particular groups which will arise frequently are, for $n \in \mathbb{N}$, the cyclic group C_n of order n, and the symmetric group S_n of permutations on n elements.

Every ring R has two distinguished elements: the additive identity 0 and the multiplicative identity 1. When using R to represent a Boolean function, we will use 0 to denote 'false' and 1 to denote 'true'. The characteristic of a ring R is the smallest $n \in \mathbb{N}$ such that

$$\underbrace{1 + 1 + \cdots + 1}_{n \text{ times}} = 0$$

over R if such an n exists, and 0 if there is no such n. When $0 \neq 1$ (i.e. when R is not the trivial ring with one element) then the characteristic is never 1. In this case, when c is the characteristic of R, there is a natural (and unique) ring homomorphism from $\mathbb{Z}$ to R whose range is $\mathbb{Z}_c$ (or $\mathbb{Z}$ when $c = 0$). Hence, polynomials over $\mathbb{Z}$ can also be viewed as polynomials over any commutative ring R where we take all outputs mod c (and in particular, do not change the values 0 and 1).

When R is a commutative ring, for $n \in \mathbb{N}$ and $x, y \in R^n$, we write $\langle x, y \rangle_R := \sum_{i=1}^{n} x_i y_i$. When the ring R is clear from context, we omit it and simply write $\langle x, y \rangle$.

When $q \in \mathbb{N}$ is a power of a prime, we write $\mathbb{F}_q$ for the finite field of order q.

2.2 Boolean and Arithmetic Circuits

We study two types of circuits:

- *Boolean circuits*, whose inputs are Boolean ($\{0, 1\}$) values, and whose gates compute Boolean functions of their inputs (the default gates are AND, OR, and NOT), and
- *Arithmetic circuits* over a field $\mathbb{F}$, whose inputs are values from $\mathbb{F}$, and whose gates compute $\mathbb{F}$-valued functions of their inputs (the default gates are $+$ and $\times$). Arithmetic circuits may also take constant values from $\mathbb{F}$ as input in addition to the usual inputs. One can more generally consider arithmetic circuits over a ring.

Boolean Circuits The depth, size, fan-in, and set of allowed gates can drastically change what functions can be computed by a given class of circuits. The classes we will encounter in this dissertation include:

- AC^0: functions computable by families of constant-depth unbounded fan-in polynomial-size circuits over the basis (set of gates) $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$
- $\mathsf{AC}^0[m]$: functions computable by families of constant-depth unbounded fan-in polynomial-size circuits over the basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}, \mathsf{MOD}_m\}$
- ACC^0: the union of $\mathsf{AC}^0[m]$ for all $m \geq 2$
- TC^0: functions computable by families of constant-depth unbounded fan-in polynomial-size circuits over the basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}, \mathsf{MAJ}\}$

It is known that

$$\mathsf{AC}^0 \subsetneq \mathsf{AC}^0[m] \subsetneq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0.$$

It is believed that $\mathsf{MAJ} \notin \mathsf{ACC}^0$, and hence that $\mathsf{ACC}^0 \subsetneq \mathsf{TC}^0$, but this is still an open problem.

We will also study the class LTF of linear threshold functions, i.e. Boolean functions $f : \{0, 1\}^n \to \{0, 1\}$ of the form $f(x) = [\sum_{i=1}^{n} a_i x_i \geq t]$ for constants $a \in \mathbb{R}^n$ and $t \in \mathbb{R}$. For instance, $\mathsf{MAJ} \in \mathsf{LTF}$.

For classes of circuits $\mathcal{C}$ and $\mathcal{D}$, we write $\mathcal{C} \circ \mathcal{D}$ to denote the class of circuits consisting of a single circuit $C \in \mathcal{C}$ whose inputs are the outputs of some circuits from $\mathcal{D}$. That is, $\mathcal{C} \circ \mathcal{D}$ is simply the composition of circuits from $\mathcal{C}$ and $\mathcal{D}$. For instance, $\mathsf{LTF} \circ \mathsf{LTF}$ denotes "depth-two linear threshold circuits", $\mathsf{LTF} \circ \mathsf{MOD}_2$ denotes "linear threshold function of parities", etc.

Arithmetic Circuits Since arithmetic circuits only use $+$ and $\times$ gates, they always compute sets of polynomials in their inputs. It is natural to ask why arithmetic circuits do not typically allow for $\div$ gates. In fact, it is known that allowing $\div$ gates can only save polynomial factors in arithmetic circuit size:

Proposition 2.1 ([Str73, HY09])**.** *If a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree r can be computed by an arithmetic circuit of size s using $+, \times, \div$, then it can be computed by an arithmetic circuit of size $poly(s, r, n)$ using only $+, \times$.*

This answer is unsatisfying in situations where we care about precise polynomial factors, such as in the design of matrix multiplication algorithms. However, Strassen [Str73] also showed that divisions do not help, even by constant additive or multiplicative factors in the circuit size, when computing sets of quadratic forms (such as matrix multiplication).

One type of arithmetic circuit which will be of particular interest to us is a *Linear circuit*. In such a circuit, each gate computes a $\mathbb{F}$-linear combination of its inputs. Hence, linear circuits can only compute linear transformations of their inputs. In other words, linear circuits with n inputs and m outputs correspond to matrices $A \in \mathbb{F}^{m \times n}$, such that given as input $x \in \mathbb{F}^n$, the circuit outputs Ax. It is known that any arithmetic circuit for computing a linear transformation can be converted into a linear circuit for the same linear transformation with only constant factor increases in the size and depth (see e.g. [BCS13, Theorem 13.1]).

2.3 Models of Communication

In a communication protocol Π for a function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, two players, each given one of $x, y \in \{0,1\}^n$, send each other messages in order to compute $F(x, y)$. The number of bits of communication used in Π is the maximum, over all $x, y \in \{0,1\}^n$, of the sum of the lengths of the messages (as binary strings) that the players send to each other. See [KN97] for more details.

Starting with [BFS86], a growing line of work has studied the communication complexity analogues of different classical complexity classes. The most relevant communication complexity classes for us will be:

- P^{cc}: Functions $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ which can be computed by a deterministic communication protocol using only $\text{polylog}(n)$ bits of communication.
- $\mathsf{NP}^{\mathsf{cc}}$: Functions $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ which can be written as $\bigvee_{i=1}^{k} R_i(x, y)$, where $k \leq \text{poly}(n)$, and each R_i is a *rectangle*, i.e. a function of the form $R_i(x, y) = [x \in S_X \wedge y \in S_y]$ for subsets $S_X, S_Y \subseteq \{0,1\}^n$. (This can be viewed as a communication protocol where the two players nondeterministically guess which rectangle is satisfied by their inputs.)
- $\mathsf{AM}^{\mathsf{cc}}$: Functions $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ such that there is a distribution $\mathcal{D}$ on $\mathsf{NP}^{\mathsf{cc}}$ protocols Π, such that for any $x, y \in \{0,1\}^n$ we have $\Pr_{\Pi \sim \mathcal{D}}[F(x, y) = \Pi(x, y)] \geq 2/3$.

- $\mathsf{PH}^{\mathsf{cc}}$: Functions $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ which can be computed by a formula which is a poly(n)-ary tree of constant depth, where each gate computes an AND or an OR, and each leaf computes a rectangle of the inputs.

Similar to the classical complexity setting, we know that

$$\mathsf{P}^{\mathsf{cc}} \subseteq \mathsf{NP}^{\mathsf{cc}} \subseteq \mathsf{AM}^{\mathsf{cc}} \subseteq \mathsf{PH}^{\mathsf{cc}}.$$

Further communication complexity classes can similarly be defined in a natural way. For instance, $\mathsf{MOD}_m\mathsf{P}^{\mathsf{cc}}$ consists of functions $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ which can be written as a MOD_m of poly(n) many rectangles, and $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}^{\mathsf{cc}}$ consists of functions which can be computed with high probability on every input by distributions on $\mathsf{MOD}_m\mathsf{P}^{\mathsf{cc}}$ protocols. See [GPW18], for instance, for more about the known relationships between these and other communication complexity classes.

2.4 Tail Bounds and Probabilistic Tools

We assume familiarity with standard tools from probability, including the union bound, Markov's inequality, Chernoff bounds, and Chebyshev's inequality. We will occasionally pay particular attention to the constants in tail bounds on Binomial distributions, by applying the following instantiation:

Lemma 2.1 (Hoeffding's Inequality for Binomial Distributions [Hoe63, Theorem 1])**.** *If m independent random draws $x_1, \ldots, x_m \sim \{0,1\}$ are made with $\Pr[x_i = 1] = p$ for all i, then for any $k \leq mp$ we have*

$$\Pr\left[\sum_{i=1}^{m} x_i \leq k\right] \leq \exp\left(-\frac{2(mp-k)^2}{m}\right),$$

where $\exp(x) = e^x$.

When designing deterministic algorithms, we will also need a Chernoff bound for samples with limited independence:

Lemma 2.2 ([SSS95, Theorem 5 (I)(b)])**.** *If X is the sum of k-wise independent random variables, each of which is confined to the interval $[0,1]$, with $\mu = \mathbb{E}[X]$, $\delta \leq 1$, and $k = \lfloor \delta^2 \mu e^{-1/3} \rfloor$, then*

$$\Pr[|X - \mu| \geq \delta\mu] \leq e^{-\delta^2\mu/3}.$$

2.5 Bounds on Binomial Coefficients

We now present some standard bounds on the growth of binomial and multinomial coefficients. We will make use of these bounds throughout this dissertation.

Proposition 2.2. *For all $n, k \in \mathbb{N}$ with $1 \leq k \leq n$, we have*

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} < \left(\frac{n \cdot e}{k}\right)^k .$$

Proof. The lower bound follows since:

$$\binom{n}{k} = \prod_{i=0}^{k-1} \frac{n-i}{k-i} \geq \prod_{i=0}^{k-1} \frac{n}{k} = \left(\frac{n}{k}\right)^k .$$

For the upper bound, first recall from the Taylor series for e^x that

$$e^k = \sum_{i=0}^{\infty} \frac{k^i}{i!} > \frac{k^k}{k!}.$$

Rearranging gives that $k! > (k/e)^k$. It thus follows that:

$$\binom{n}{k} = \frac{\prod_{i=0}^{k-1} n - i}{k!} \leq \frac{\prod_{i=0}^{k-1} n}{k!} = \frac{n^k}{k!} < \left(\frac{n \cdot e}{k}\right)^k .$$

□

The bound from Proposition 2.2, which shows that $\binom{n}{k} = \Theta(n/k)^k$, will be sufficient in most cases where we need to use bounds on binomial coefficients. However, in some cases where n and k are both large, we will need a tighter bound on the constant hidden by the Θ.

Definition 2.1. *The* binary entropy function $H : [0, 1] \to [0, 1]$ *is given by*

$$H(x) = x \log \frac{1}{x} + (1 - x) \log \frac{1}{1 - x},$$

where we define $H(0) = H(1) = 0$. Hence, $2^{H(p)} = \frac{1}{p^p \cdot (1-p)^{1-p}}$.

Proposition 2.3. *For any $n \in \mathbb{N}$ and any $p \in [0, 1]$ such that $p \cdot n$ is an integer, we have*

$$\frac{1}{n+1} 2^{n \cdot H(p)} \leq \binom{n}{p \cdot n} \leq 2^{n \cdot H(p)}.$$

Proof. We can verify that the claim is true when $p = 0$ or $p = 1$, so assume $p \in (0, 1)$. Define $T : \{0, 1, \ldots, n\} \to \mathbb{R}$ by $T(k) = \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k}$. Notice in particular that

$$T(p \cdot n) = \binom{n}{p \cdot n} \cdot p^{pn} \cdot (1 - p)^{(1-p)n} = \binom{n}{p \cdot n} \cdot 2^{-n \cdot H(p)}.$$

Our goal is hence to prove that

$$\frac{1}{n+1} \leq T(p \cdot n) \leq 1.$$

We have that $T(k) > 0$ for all k, and the binomial theorem says that

$$\sum_{k=0}^{n} T(k) = (p + (1-p))^n = 1.$$

In particular, the upper bound $T(p \cdot n) \leq 1$ immediately follows.

Next, to prove the lower bound, it is sufficient to show that $T(p \cdot n) \geq T(k)$ for all k. This will imply that $T(p \cdot n)$ is at least the average value of all the $T(k)$'s, which by the binomial theorem above, is at least $1/(n+1)$.

More precisely, we will prove that $T(k) - T(k+1)$ is nonnegative when $k \geq p \cdot n$, and nonpositive when $k \leq p \cdot n$. This will imply that $(T(k))_{0 \leq k \leq n}$ is a unimodal sequence with maximum at $p \cdot n$. To see this, note that

$$\begin{aligned} T(k) - T(k+1) &= \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} - \binom{n}{k+1} \cdot p^{k+1} \cdot (1-p)^{n-k-1} \\ &= \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \cdot \left(1 - \frac{n-k}{k+1} \cdot \frac{p}{1-p}\right). \end{aligned}$$

Hence, $T(k) - T(k+1) \geq 0$ if and only if $(n-k) \cdot p \leq (k+1) \cdot (1-p)$, which rearranges to $k \geq p \cdot n - (1-p)$. Since $(1-p) \in (0,1)$, but the argument k to T must be an integer, this means that $T(k) - T(k+1) \geq 0$ if and only if $k \geq p \cdot n$, as desired. □

We will also use the following estimate of $H(p)$ when p is close to $1/2$:

Proposition 2.4. *For $\varepsilon \in (0, 1/2)$ we have $H(\frac{1}{2} - \varepsilon) = 1 - \Theta(\varepsilon^2)$.*

Proof. This follows from the Taylor expansion of $H(p)$ about $p = 1/2$:

$$H\left(\frac{1}{2} - \varepsilon\right) = 1 - \frac{1}{2 \ln 2} \sum_{i=1}^{\infty} \frac{(2\varepsilon)^{2i}}{i \cdot (2i-1)} = 1 - \Theta(\varepsilon^2).$$

□

Corollary 2.1. *We have*

$$\binom{n}{(1/2 - \varepsilon)n} = 2^{n - \Theta(\varepsilon^2 n)}$$

for $\varepsilon \in (0, 1/2)$ and $n \in \mathbb{N}$ such that $(1/2 - \varepsilon)n$ is an integer.

2.5.1 Multinomial Coefficients

Definition 2.2. *For $m \in \mathbb{N}$ and nonnegative integers $k_1, k_2, \ldots, k_m$ with $n = k_1 + k_2 + \cdots + k_m$, the* multinomial coefficient *$\binom{n}{k_1, k_2, \ldots, k_m}$ is given by*

$$\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1! \cdot k_2! \cdots k_m!} = \prod_{i=1}^{m} \binom{k_1 + k_2 + \cdots + k_i}{k_i}.$$

Multinomial coefficients have a combinatorial interpretation similar to that of binomial coefficients: $\binom{n}{k_1,k_2,\ldots,k_m}$ counts the number of ways to put n distinct objects into m distinct buckets such that k_i objects are put into the ith bucket for each $i \in [m]$. We will make use of the following approximation, which follows directly by applying Proposition 2.3:

Proposition 2.5. *For any $m \in \mathbb{N}$ and any constants $p_1, p_2, \ldots, p_m \in [0,1]$ such that $p_1 + p_2 + \cdots + p_m = 1$, we have*

$$\binom{n}{p_1 \cdot n,\ p_2 \cdot n,\ \ldots,\ p_m \cdot n} = \left(\frac{1}{p_1^{p_1} \cdot p_2^{p_2} \cdots p_m^{p_m}}\right)^{n+o(n)}$$

for all $n \in \mathbb{N}$ such that $p_i \cdot n$ is an integer for all $i \in [m]$.

Part I

Limitations on Matrix Multiplication Algorithms

Chapter 3

Background and Overview

One of the biggest open questions in computer science asks how quickly one can multiply two matrices. Progress on this problems is measured by giving bounds on ω, the *exponent of matrix multiplication*, defined as the smallest real number such that two $n \times n$ matrices over a field can be multiplied using $n^{\omega+\varepsilon}$ field operations for any $\varepsilon > 0$. Trivially, $2 \leq \omega \leq 3$. Many have conjectured over the years that $\omega = 2$, and this conjecture is very attractive: a near-linear time algorithm for MM would immediately imply near-optimal algorithms for many problems.

Almost 50 years have passed since Strassen [Str69] first showed that $\omega \leq 2.81 < 3$. Since then, an impressive toolbox of techniques has been developed to obtain faster MM algorithms, culminating in the current best bound $\omega < 2.373$ [LG14, Wil12]. Unfortunately, this bound is far from 2, and the current methods seem to have reached a standstill. Recent research has turned to proving limitations on the two main MM algorithmic techniques: the Laser Method of Strassen [Str86] and the Group-theoretic Method of Cohn and Umans [CU03].

Both Coppersmith and Winograd [CW90] and Cohn et al. [CKSU05] proposed conjectures which, if true, would imply that $\omega = 2$. The first conjecture works in conjunction with the Laser Method, and the second with the Group-theoretic method. The first "technique limitation" result was by Alon, Shpilka and Umans [ASU13] who showed that both conjectures would contradict the widely believed Sunflower conjecture of Erdös and Rado.

Ambainis, Filmus and Le Gall [AFLG15] formalized the specific implementation of the Laser Method proposed by Coppersmith and Winograd [CW90] which is used in the recent papers on MM. They gave limitations of this implementation, and in particular showed that the exact approach used in [CW90, DS13, LG14, Wil12] cannot achieve a bound on ω better than 2.3078. The analyzed approach, the "Laser Method with Merging", is a bit more general than the approaches in [CW90, DS13, LG14, Wil12]: in a sense it corresponds to a dream implementation of the exact approach.

Blasiak et al. [BCC$^+$17a] considered the Group-theoretic Method for developing MM algorithms proposed by Cohn and Umans [CU03], and showed that this approach cannot prove $\omega = 2$ using any fixed abelian group. In follow-up work, Sawin [Saw18] extended this to any fixed non-abelian group, and Blasiak et al. [BCC$^+$17b] extended it to a host of families of non-abelian groups.

All these limitations proven so far are for very specific attacks on proving $\omega = 2$. While the proofs of [AFLG15] apply directly to CW_q, they only apply to the restricted Laser Method with Merging, and no longer apply to slight changes to this. The proofs in [BCC$^+$17a] and [BCC$^+$17b] are tailored to the Group-theoretic Method and do not apply (for instance) to the Laser Method on "non-group" tensors.

3.1 Our Results

3.1.1 The Universal Method

The key to Strassen's algorithm is an algebraic identity showing how $2 \times 2 \times 2$ matrix multiplication can be computed surprisingly efficiently. In particular, Strassen showed that the $2 \times 2 \times 2$ matrix multiplication tensor has *tensor rank* at most 7; see the beginning of Chapter 4 for precise definitions). Arguing about the ranks of larger matrix multiplication tensors has proven to be quite difficult – in fact, even the rank of the $3 \times 3 \times 3$ matrix multiplication tensor isn't currently known. Progress on bounding ω since Strassen's algorithm has thus taken the following approach: Pick a tensor (trilinear form) T, typically not a matrix multiplication tensor, such that

- Powers $T^{\otimes n}$ of T can be efficiently computed (i.e. T has low asymptotic rank), and
- T is useful for performing matrix multiplication, since large matrix multiplication tensors can be 'reduced to' powers of T.

Combined, these give an upper bound on the rank of matrix multiplication itself, and hence ω.

In Chapter 4, we define a new very general method for analyzing tensors to give MM algorithms, which we call the *Universal Method*. In the Universal Method, the notion of 'reduction' between tensors we use is a *degeneration*. Degenerations are the most general type of reduction known to preserve the ranks of tensors as required for the above approach. In other words, the Universal Method captures the most general sense in which one can use T to design MM algorithms. We write $\omega_u(T)$ to denote the best bound one can prove on ω by applying the Universal Method to T.

We also define two weaker methods: *the Galactic Method applied to* T, in which the 'reduction' must be a more restrictive *monomial degeneration*, resulting in the bound $\omega_g(T)$ on ω, and *the Solar Method applied to* T, in which the 'reduction' must be an even more restrictive *zeroing out*, resulting in the bound $\omega_s(T)$ on ω. Since monomial degenerations and zeroing outs are successively more restrictive types of degenerations, we have that for all tensors T,

$$\omega \leq \omega_u(T) \leq \omega_g(T) \leq \omega_s(T).$$

These methods are *very general*; there are no known methods for computing $\omega_u(T)$, $\omega_g(T)$, or $\omega_s(T)$ for a given tensor T, and these quantities are even unknown for very well-studied tensors T.

The two main approaches to designing matrix multiplication algorithms are the Laser Method of Strassen [Str87] and the Group-Theoretic Method of Cohn and Umans [CU03]. Both of these approaches show how to give upper bounds on $\omega_s(T)$ for particular structured tensors T (and hence upper bound ω itself). We will conclude Chapter 4 by giving an overview of these two methods, and describing how they are special cases of the Solar Method, emphasising that the Universal Method affords an algorithm designer much more power than is taken advantage of by these methods.

3.1.2 Limits on the Universal Method

The Coppersmith-Winograd Tensor

Both the Laser Method and the Group-theoretic Method give ways to find zeroing outs of tensors into matrix multiplication tensors, but not necessarily the best zeroing outs. In fact, it is known that the Laser Method does not always give the best zeroing out for a particular tensor T, since the improvements from [CW90] to later works [DS13, Wil12, LG14] can be seen as giving slight improvements to the Laser Method to find better and better zeroing outs[1]. The Group-Theoretic Method, like the Solar Method, is very general, and it is not clear how to optimally apply it to a particular group or family of groups.

All of the improvements on bounding ω for the past 30+ years have come from studying the Coppersmith-Winograd family of tensors $\{CW_q\}_{q\in\mathbb{N}}$. The Laser Method applied to powers of CW_5 gives the bound $\omega_s(CW_5) \leq 2.3729$. The Group-Theoretic Method can also prove the best known bound $\omega \leq 2.3729$, by simulating the Laser Method analysis of CW_q (see e.g. [AW18a] for more details). Despite a long line of work on matrix multiplication, there are no known tensors[2] which seem to come close to achieving the bounds one can obtain using CW_q. This leads to the first main question of this Part of the dissertation:

Question 3.1. *How much can we improve our bound on ω using a more clever analysis of the Coppersmith-Winograd tensor?*

To resolve Question 3.1, we prove a new lower bound for the Coppersmith-Winograd tensor in Chapter 5:

Theorem 3.1. $\omega_u(CW_q) \geq 2.16805$ *for all q.*

Thus, if one starts with the CW tensor which has led to all improvements on ω for the last 30+ years, even if one uses the Universal method which vastly generalizes all known approaches, one cannot prove a better bound than 2.16805 on ω. We also give stronger lower bounds for particular tensors

[1] These works apply the Laser Method to higher powers of the tensor $T = CW_q$, a technique which is still captured by the Solar Method.

[2] The author and Vassilevska Williams [AW18b] study a generalization of CW_q which can tie the best known bound, but its analysis is identical to that of CW_q. Our lower bounds in this paper will apply equally well to this generalized class as to CW_q itself.

in the family. For instance, for the specific tensor CW_5 which yields the current best bound on ω, we show $\omega_u(CW_5) \geq 2.21912\ldots$.

Our proof of Theorem 3.1 proceeds by upper bounding $\tilde{S}(CW_q)$, the *asymptotic slice rank* of CW_q; we will show that for any tensor T, non-trivial upper bounds on $\tilde{S}(T)$ imply non-trivial lower bounds on $\omega_u(T)$. The slice rank of a tensor, denoted $S(T)$, was first introduced by Blasiak et al. [BCC+17a] in the context of lower bounds against the Group-Theoretic Method. In order to study degenerations of *powers* of tensors, rather than just tensors themselves, we need to study an *asymptotic* version of slice rank, $\tilde{S}$. This is important since the slice rank of a product of two tensors can be greater than the product of their slice ranks, and as we will show, $S(CW_q^{\otimes n})$ is much greater than $S(CW_q)^n$ for big enough n.

We will give three different tools for proving upper bounds on $\tilde{S}(T)$ for many different tensors T. They will imply our lower bound on the Universal Method for CW_q as well as many other tensors of interest, including: the same lower bound $\omega_u(CW_{q,\sigma}) \geq 2.16805$ for any *generalized Coppersmith-Winograd tensor* $CW_{q,\sigma}$ (a new class of tensors we define which slightly modify the structure of CW_q), a similar lower bound for $cw_{q,\sigma}$, the generalized 'simple' Coppersmith-Winograd tensor missing its 'corner terms', and a lower bound for T_q, the structural tensor of the cyclic group C_q, matching the lower bounds obtained by [BCC+17a]. In Section 5.5 we give tables of our precise lower bounds for these and other tensors.

We briefly note that our lower bound of $2.16805 > 2 + \frac{1}{6}$ in Theorem 3.1 may be significant when compared to the recent algorithm of Cohen, Lee and Song [CLS18] which solves n-variable linear programs in time about $O(n^\omega + n^{2+1/6})$.

The Laser Method is "Complete"

The second main question of this part concerns the Laser Method. The Laser Method upper bounds $\omega_s(T)$ for any tensor T with certain structure (which we describe in detail in Section 5.4), and has led to every improvement on ω since its introduction by Strassen [Str87].

Question 3.2. *When the Laser Method applies to a tensor T, how close does it come to optimally analyzing T?*

We call T *laser-ready* if the Laser Method (as used by [CW90] on CW_q) applies to it; see Definition 5.1 for the precise definition. Tensors need certain structure to be laser-ready, but tensors T with this structure are essentially the only ones for which successful techniques for upper bounding $\omega_u(T)$ are known. In fact, every record-holding tensor in the history of matrix multiplication algorithm design has been laser-ready.

As discussed, we know the Laser Method does not always give a tight bound on $\omega_s(T)$ for laser-ready T. For instance, Coppersmith-Winograd [CW90] applied the Laser Method to CW_q to prove $\omega_s(CW_q) \leq 2.376$, and then later work [DS13, Wil12, LG14] analyzed higher and higher powers of CW_q to show $\omega_s(CW_q) \leq 2.373$. Ambainis, Filmus and Le Gall [AFLG15] showed that analyzing higher and higher powers of CW_q itself with the Laser Method cannot yield an upper bound better than

$\omega_s(CW_q) \leq 2.3725$. What about for other tensors? Could there be a tensor such that applying the Laser Method to T yields $\omega_s(T) \leq c$ for some $c > 2$, but applying the Laser Method to high powers $T^{\otimes n}$ of T yields $\omega_s(T) = 2$? Could applying an entirely different method to such a T, using arbitrary degenerations and not just zeroing outs, show that $\omega_u(T) = 2$?

We show that for any laser-ready tensor T, the Laser Method can be used to prove a *lower bound* on $\tilde{S}(T)$. Moreover, we will see that this lower bound *matches* the upper bound on $\tilde{S}(T)$ implied by one of our tools, Theorem 5.3. We will use this to give an intriguing answer to Question 3.2:

Theorem 3.2. *If T is a laser-ready tensor, and the Laser Method applied to T yields the bound $\omega_u(T) \leq c$ for some $c > 2$, then $\omega_u(T) > 2$.*

To reiterate: If T is any tensor to which the Laser Method applies (as in Definition 5.1), and the Laser Method does not yield $\omega = 2$ when applied to T, then in fact $\omega_u(T) > 2$, and even the substantially more general Universal Method applied to T cannot yield $\omega = 2$. Hence, the Laser Method, which was originally used as an algorithmic tool, can also be seen as a lower bounding tool. Conversely, Theorem 3.2 shows that the Laser Method is "complete", in the sense that it cannot yield a bound on ω worse than 2 when applied to a tensor which is able to prove $\omega = 2$.

Another consequence of our proof is that, whenever T is a laser-ready tensor, we will be able to prove *matching* upper and lower bounds on $\tilde{S}(T)$. As mentioned, this includes every record-holding tensor in the history of MM algorithms, including CW_q, cw_q, and all the other tensors we study in Section 5.5. Hence, for these tensors T, no better lower bound on $\omega_u(T)$ is possible by arguing only about $\tilde{S}(T)$.

Theorem 3.2 explains and generalizes a number of phenomena:

- The fact that Coppersmith-Winograd [CW90] applied the Laser Method to the tensor CW_q and achieved an upper bound greater than 2 on ω *implies that* $\omega_u(CW_q) > 2$, and no *arbitrary degeneration* of powers of CW_q can yield $\omega = 2$.

- As mentioned above, it is known that applying the Laser Method to higher and higher powers of a tensor T can successively improve the resulting upper bound on ω. Theorem 3.2 shows that if the Laser Method applied to the first power of any tensor T did not yield $\omega = 2$, then this sequence of Laser Method applications (which is a special case of the Universal method) must converge to a value greater than 2 as well. This generalizes the result of Ambainis, Filmus and Le Gall [AFLG15], who proved this about applying the Laser Method to higher and higher powers of the specific tensor $T = CW_q$.

- Our result also generalizes the result of Kleinberg, Speyer and Sawin [Kle97], where it was shown that (what can be seen as) the Laser Method achieves a tight lower bound on $\tilde{S}(T_q^{lower})$, matching the upper bound of Blasiak et al. [BCC$^+$17a]. Indeed, T_q^{lower}, the lower triangular part of T_q, is a laser-ready tensor.

3.1.3 Additional Results

In our study of tensors and slice rank, we will also prove a number of new, complementary results.

Asymptotic Subrank Equals Asymptotic Slice Rank for Laser-Ready Tensors

Our proof of Theorem 3.2 also sheds light on a notion related to the asymptotic slice rank $\tilde{S}(T)$ of a tensor T, called the *asymptotic subrank* $\tilde{Q}(T)$ of T. $\tilde{Q}$ is a "dual" notion of asymptotic rank, and it is important in the definition of Strassen's asymptotic spectrum of tensors [Str87]. While the asymptotic rank of T can be thought of as the 'cost' of T, the asymptotic subrank can be thought of as its 'value'.

It is not hard to see that $\tilde{Q}(T) \leq \tilde{S}(T)$ for all tensors T. However, there are no known separations between the two notions; whether there exists a tensor T such that $\tilde{Q}(T) < \tilde{S}(T)$ is an open question. As a Corollary of Theorem 3.2, we prove:

Corollary 3.1. *Every laser-ready tensor T has $\tilde{Q}(T) = \tilde{S}(T)$.*

Since, as discussed above, almost all of the most-studied tensors are laser-ready, this might help explain why we have been unable to separate the two notions.

The Structural Tensors of Group Algebras

We also study the relationship between the generalized CW tensors and the structural tensors of group algebras (the tensors which arise in the Group-theoretic Method). Our new results include:

1. **All Finite Groups Suffice for Current ω Bounds.** We show that every finite group G has a monomial degeneration to some generalized CW tensor of parameter $q = |G| - 2$. Thus, applying the Galactic method on T_G for *every* G (with sufficiently small asymptotic rank, i.e. $\tilde{R}(T_G) = |G|$) can yield the current best bounds on ω.

2. **New Tri-Colored Sum-Free Set Constructions.** Tri-Colored Sum-Free Sets are subsets of a group G which arise in extremal combinatorics. We show that, for every finite group G, there is a constant $c_{|G|} > 2/3$ depending only on $|G|$ such that its nth tensor power G^n has a tri-colored sum-free set of size at least $|G|^{c_{|G|}n-o(n)}$. For moderate $|G|$, the constant $c_{|G|}$ is quite a bit larger than 2/3. To our knowledge, such a general result was not known until now.

3.2 Other Related Work

Probabilistic Tensors and Support Rank Cohn and Umans [CU13] introduced the notion of the *support rank* of tensors, and showed that upper bounds on the support rank of matrix multiplication tensors can be used to design faster *Boolean* matrix multiplication algorithms. Recently, Karppa and Kaski [KK19] used 'probabilistic tensors' as another way to design Boolean matrix multiplication algorithms.

In fact, our tools for proving asymptotic slice rank upper bounds can be used to prove lower bounds on these approaches as well. For instance, our results imply that finding a 'weighted' matrix multiplication tensor as a degeneration of a power of CW_q (in order to prove a support rank upper bound) cannot result in a better exponent for Boolean matrix multiplication than 2.16805.

This is because 'weighted' matrix multiplication tensors can degenerate into independent tensors just as large as their unweighted counterparts. Similarly, if a probabilistic tensor $\mathcal{T}$ is degenerated into a (probabilistic) matrix multiplication tensor, Karppa and Kaski show that this gives a corresponding support rank expression for matrix multiplication as well, and so upper bounds on $\tilde{S}(T)$ for any T in the support of $\mathcal{T}$ also result in lower bounds on this approach.

Rectangular Matrix Multiplication Our tools can also be used to prove lower bounds on approaches to designing rectangular matrix multiplication algorithms. For instance, the best known rectangular matrix multiplication algorithms [LGU17] show that powers of CW_q zero out into large rectangular matrix multiplication tensors. Using the fact that CW_q is *variable-symmetric*, this implies a corresponding upper bound on $\omega_u(CW_q)$, which our tools give a lower bound against; see Section 4.5 for details.

Slice Rank Upper Bounds Our limitation results critically make use of upper bounds on the *asymptotic slice rank* of CW_q and other tensors of interest. Slice rank was first introduced by Tao [Tao16] in a symmetric formulation of the recent proof of the capset bound [CLP17, EG17], which shows how to prove slice rank upper bounds using the 'polynomial method'. Since then, a number of papers have focused on proving slice rank upper bounds for many different tensors. Sawin and Tao [TS16, Proposition 6] show slice rank upper bounds by studying the combinatorics of the support of the power of a fixed tensor, and Naslund and Sawin [NS17] use that approach to study sunflower-free sets[3]; one of our slice rank bounding tools, Theorem 5.3, uses this type of approach applied to blocked tensors. Slice rank was first used in the context of matrix multiplication by Blasiak et al. [BCC+17a], and this line of work has led to more techniques for proving slice rank upper bounds, including connections to the notion of instability from geometric invariant theory [BCC+17a], and a generalization of the polynomial method to the nonabelian setting [BCC+17b].

Concurrent Work Building off of our work [AW18b], Christandl, Vrana and Zuiddam [CVZ19] independently proved lower bounds on ω_u, including a bound matching Theorem 3.1. Their bounds use the seemingly more complicated machinery of Strassen's asymptotic spectrum of tensors [Str91]. They thus phrase their results in

[3] In fact, the tensor T whose slice rank is bounded in [NS17, Section 3] can be viewed as a change of basis of a Generalized Simple Coppersmith-Winograd tensor $cw_{q,\sigma}$ which we study below in Section 5.5.2

terms of the asymptotic subrank $\tilde{Q}(T)$ of tensors rather than the asymptotic slice rank $\tilde{S}(T)$, and the fact that their bounds are often the same as ours is related to the fact we prove, in Corollary 3.1, that $\tilde{Q}(T) = \tilde{S}(T)$ for all of the tensors we study.

3.3 Bibliographic Details

This Part of the dissertation is based off of the results in three previously published papers:

- 'Further Limitations of the Known Approaches for Matrix Multiplication' with Virginia Vassilevska Williams [AW18a], which appeared in ITCS 2018,
- 'Limits on All Known (and Some Unknown) Approaches to Matrix Multiplication' with Virginia Vassilevska Williams [AW18b], which appeared in FOCS 2018, and
- 'Limits on the Universal Method for Matrix Multiplication' [Alm19b], which appeared in CCC 2019 and won the Best Student Paper Award.

Chapter 4 primarily presents results from [AW18b], and Chapter 5 primarily presents results from [Alm19b], except that Subsection 5.5.4 and Section 5.6 come from [AW18b], and Theorem 5.5 follows the proof of [AW18a, Lemma 4.1].

Chapter 4

The Universal Method

In this chapter, we introduce the relevant notions and notation related to tensors and matrix multiplication (MM) algorithms. We will give an overview of the known approaches to designing MM algorithms, including the Laser Method and the Group-theoretic Method, and then we will define a new, vast generalization of these approaches which we call the *Universal Method.* In the next Chapter, we will prove new *limitation results* against the Universal Method.

4.1 Tensors and Tensor Rank

The mathematical objects of interest in the study of MM algorithms are called tensors (or 3-tensors). Recall that a $q \times r$ matrix M over a field $\mathbb{F}$ can be viewed in many equivalent ways, including:

- a 2-dimensional grid of numbers $(M_{ij})_{i\in[q],j\in[r]} \in \mathbb{F}^{q\times r}$,
- a linear map $M : \mathbb{F}^q \to \mathbb{F}^r$,
- a bilinear map $M : \mathbb{F}^q \times \mathbb{F}^r \to \mathbb{F}$.

Analogously, a $q \times r \times s$ tensor T can be viewed in a number of different ways, including:

- a hypermatrix, i.e. a 3-dimensional grid of numbers $(T_{ijk})_{i\in[q],j\in[r],k\in[s]} \in \mathbb{F}^{q\times r\times s}$,
- a bilinear map $T : \mathbb{F}^q \times \mathbb{F}^r \to \mathbb{F}^s$,
- a trilinear map $T : \mathbb{F}^q \times \mathbb{F}^r \times \mathbb{F}^s \to \mathbb{F}$.

In this dissertation, we will focus on the equivalent view of tensors which I find simplest: as a trilinear polynomial.

For sets $X = \{x_1, \dots, x_q\}$, $Y = \{y_1, \dots, y_r\}$, and $Z = \{z_1, \dots, z_s\}$ of formal variables, a *tensor over* X, Y, Z is a trilinear form

$$T = \sum_{x_i\in X,y_j\in Y,z_k\in Z} \alpha_{ijk} x_i y_j z_k,$$

where the α_{ijk} coefficients come from an underlying field $\mathbb{F}$ (the field $\mathbb{F}$ can typically be thought of as the complex numbers $\mathbb{C}$). The *terms*, which we write as $x_i y_j z_k$, are sometimes written as $x_i \otimes y_j \otimes z_k$ in the literature. We say T is *minimal for* X, Y, Z if, for each $x_i \in X$, there is a term involving x_i with a nonzero coefficient in T, and similarly for Y and Z (i.e. T can't be seen as a tensor over a strict subset of the variables). For two tensors T_1, T_2, we write $T_1 = T_2$, if they are equal up to renaming or reindexing variables.

The main measure of the complexity of a tensor is its rank. A tensor T has *rank one* if there are values $a_i \in \mathbb{F}$ for each $x_i \in X$, $b_j \in \mathbb{F}$ for each $y_j \in Y$, and $c_k \in \mathbb{F}$ for each $z_k \in Z$, such that the coefficient α_{ijk} of $x_i y_j z_k$ in T is $a_i b_j c_k$, or in other words,

$$T = \sum_{x_i \in X, y_j \in Y, z_k \in Z} a_i b_j c_k \cdot x_i y_j z_k = \left(\sum_{x_i \in X} a_i x_i \right) \left(\sum_{y_j \in Y} b_j y_j \right) \left(\sum_{z_k \in Z} c_k z_k \right). \tag{4.1}$$

The *rank* of a tensor T, denoted $R(T)$, is the smallest number of rank one tensors whose sum (as polynomials, i.e. summing the coefficient of each term individually) is T. This is analogous to the rank of a matrix: a matrix M has rank one if it can be written as the outer product of two vectors (the expression (4.1) can be seen as the outer product of three vectors), and more generally the rank of M is the minimum number of rank one matrices whose sum gives M.

4.2 Matrix Multiplication Tensors

We now define the primary family of tensors of interest in the study of MM algorithms. For positive integers a, b, c, the *matrix multiplication tensor* $\langle a, b, c \rangle$ is a tensor over $\{x_{ij}\}_{i \in [a], j \in [b]}$, $\{y_{jk}\}_{j \in [b], k \in [c]}$, $\{z_{ki}\}_{k \in [c], i \in [a]}$ given by

$$\langle a, b, c \rangle = \sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{c} x_{ij} y_{jk} z_{ki}. \tag{4.2}$$

The tensors $\langle a, b, c \rangle$ can be seen as a 'generating function' for $a \times b \times c$ matrix multiplication: the coefficient of z_{ki} in $\langle a, b, c \rangle$ is exactly the (i, k) entry in the matrix product

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1b} \\ x_{21} & x_{22} & \cdots & x_{2b} \\ \vdots & \vdots & \ddots & \vdots \\ x_{a1} & x_{a2} & \cdots & x_{ab} \end{pmatrix} \times \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1c} \\ y_{21} & y_{22} & \cdots & y_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ y_{b1} & y_{b2} & \cdots & y_{bc} \end{pmatrix}.$$

In light of this fact, we can design MM algorithm which make use of rank upper bounds on MM tensors, following the main recursive idea of Strassen's original algorithm [Str69] (the key identities in Strassen's algorithm can be interpreted as showing that $R(\langle 2, 2, 2 \rangle) \leq 7$). We state the result here for square matrix multiplication for simplicity, but the analogous algorithm works for rectangular matrix multiplication as well.

Proposition 4.1 ([Str69]). *For any constant $q \in \mathbb{N}$, if $R(\langle q,q,q\rangle) \leq r$ (over a field $\mathbb{F}$), there is an algorithm which performs $n \times n \times n$ matrix multiplication over $\mathbb{F}$ using $O(n^{\log_q(r)})$ field operations.*

Proof. Since $R(\langle q,q,q\rangle) \leq r$, we can write

$$\langle q,q,q\rangle = \sum_{\ell=1}^{r} \left(\sum_{i,j\in[q]} a_{ij\ell} x_{ij} \right) \left(\sum_{j,k\in[q]} b_{jk\ell} y_{jk} \right) \left(\sum_{k,i\in[q]} c_{ki\ell} z_{ki} \right), \tag{4.3}$$

for some coefficients $a_{ij\ell}, b_{jk\ell}, c_{ki\ell} \in \mathbb{F}$.

We design a recursive algorithm for multiplying $A, B \in \mathbb{F}^{n\times n}$. We may assume that n is a power of q by padding the input matrices with 0s, which increases the dimension n by less than a multiplicative factor of q.

First, we partition A into a $q \times q$ block matrix, where each block is a $n/q \times n/q$ matrix; call the blocks A_{ij} for $i,j \in [q]$. We similarly partition B into a $q \times q$ block matrix, and call the blocks B_{jk} for $j,k \in [q]$. The algorithm first computes, for each $\ell \in [r]$, the linear combination

$$A'_\ell = \sum_{i,j\in[q]} a_{ij\ell} A_{ij},$$

and the linear combination

$$B'_\ell = \sum_{j,k\in[q]} b_{jk\ell} B_{jk}.$$

Next, for each $\ell \in [r]$, the algorithm computes the $(n/q)\times(n/q)$ matrix $C'_\ell := A'_\ell \times B'_\ell$, by *recursively* performing $(n/q) \times (n/q) \times (n/q)$ matrix multiplication. Finally, for each $i,k \in [q]$, the algorithm computes the linear combination

$$C_{ki} = \sum_{k,i\in[q]} c_{ki\ell} C'_{ki}.$$

These are the blocks of the $n \times n$ matrix C which we output; indeed, we can see from (4.3) that for all $k,i \in [q]$, the matrix C_{ki} is the coefficient of z_{ki} in $\langle q,q,q\rangle$ when the substitutions $x_{ij} \leftarrow A_{ij}$ and $y_{jk} \leftarrow B_{jk}$ are made, and from the definition of $\langle q,q,q\rangle$ that these are exactly the desired output blocks.

Throughout the algorithm, we performed $O(n^2)$ field operations to compute linear combinations when constructing the A'_ℓ, B'_ℓ, and C_{ki} matrices, and we performed r recursive $(n/q) \times (n/q) \times (n/q)$ matrix multiplications. Thus, the total number $T(n)$ of field operations satisfies

$$T(n) = r \cdot T(n/q) + O(n^2),$$

which solves[1] to $T(n) = O(n^{\log_q(r)})$. □

[1] Here we use the known bound $r > q^2$ [CW82, BI13] to avoid additional $\log(n)$ factors.

Note that in Proposition 4.1, we gave a bound on the number of field operations performed by the algorithm, rather than on the running time of the algorithm. Over a field like $\mathbb{F}_2$ where field operations can be performed in constant time, this distinction is unimportant, but over larger fields, the field operations may take super-constant time and contribute to the final running time. Throughout Part I of this dissertation, we will abstract away this issue by focusing on a model of computation, such as the arithmetic circuit model, where field operations can be performed in constant time. Later, in Part II, we will be multiplying matrices of integers, and we will need to return to this issue.

In light of Proposition 4.1, we define ω, the *exponent of matrix multiplication*, as

$$\omega := \liminf_{q \in \mathbb{N}} \log_q(R(\langle q, q, q \rangle)).$$

It follows from Proposition 4.1 that, for any $\varepsilon > 0$, $n \times n \times n$ matrix multiplication can be performed in $O(n^{\omega+\varepsilon})$ field operations. In fact, it is known that in the arithmetic circuit model, any algorithm for MM (which does not necessarily come from a tensor rank upper bound, and which may use division over $\mathbb{F}$) can be converted into a tensor rank upper bound which yields asymptotically the same running time when combined with Proposition 4.1 (see e.g. [Blä13, Theorem 4.7]). Hence, ω exactly captures the arithmetic circuit complexity of MM, and so our goal for designing MM algorithms is to give upper bounds on the ranks of MM tensors.

Before moving on, we make two notes about the definition of ω. First, using a $\liminf$ rather than just a min (which would, for instance, allow us to omit the $\varepsilon > 0$ in the previous paragraph) is known to be required: Coppersmith and Winograd [CW82] showed that ω is a limit point that cannot be achieved by any single algorithm. Second, our notation in defining ω is somewhat sloppy, since the rank $R(\langle q, q, q \rangle)$ may depend on the field $\mathbb{F}$ of coefficients. It is known that ω only depends on the characteristic of $\mathbb{F}$ [Sch81], but for instance, it may be the case that ω over $\mathbb{F}_2$ is different from ω over $\mathbb{C}$. That said, the best known upper bounds on ω hold equally well for all fields, so we will simply refer to ω without reference to the field $\mathbb{F}$ for simplicity.

What bounds on the ranks of MM tensors are known? Strassen [Str69] showed in 1969 that $R(\langle 2, 2, 2 \rangle) \leq 7$, yielding $\omega \leq \log_2(7) \approx 2.81$ (see Figure 4-1 below).

The next improved bound came in 1978, when Victor Pan [Pan78] showed that $R(\langle 70, 70, 70 \rangle) \leq 143640$, yielding $\omega \leq \log_2(7) \approx 2.80$. Since then, a long line of work has led to the best known bound $\omega \leq 2.372864$ [CW82, DS13, Wil12, LG14], which comes from a bound on $R(\langle q, q, q \rangle)$ for a *very* large value of q.

In order to handle rank expressions for such large tensors, the known approaches to designing matrix multiplication algorithms make use of two two key techniques which we describe next: *tensor powers*, which allow us to prove properties of large tensors by arguing only about smaller tensors, and *rank-preserving reductions* between tensors, which allow us to argue about tensors T other than MM tensors as long we can find a reduction from MM to T.

$$
\begin{aligned}
\langle 2,2,2\rangle =&(x_{11}+x_{22})(y_{11}+y_{22})(z_{11}+z_{22})\\
&+(x_{21}+x_{22})y_{11}(z_{21}-z_{22})\\
&+x_{11}(y_{12}-y_{22})(z_{12}+z_{22})\\
&+x_{22}(y_{21}-y_{11})(z_{11}+z_{21})\\
&+(x_{11}+x_{12})y_{22}(z_{12}-z_{11})\\
&+(x_{21}-x_{11})(y_{11}+y_{12})z_{22}\\
&+(x_{12}-x_{22})(y_{21}+y_{22})z_{11}
\end{aligned}
$$

Figure 4-1: Strassen's algorithm as a rank expression, showing that $\langle 2,2,2\rangle$ is a sum of 7 rank one tensors, and thus $R(\langle 2,2,2\rangle) \leq 7$.

4.3 Tensor Powers and Asymptotic Rank

We first introduce the tensor product. If T_1 is a tensor over X_1, Y_1, Z_1, and T_2 is a tensor over X_2, Y_2, Z_2, then the *tensor product* $T_1 \otimes T_2$ is a tensor over $X_1 \times X_2, Y_1 \times Y_2, Z_1 \times Z_2$ such that, for any $(x_1, x_2) \in X_1 \times X_2$, $(y_1, y_2) \in Y_1 \times Y_2$, and $(z_1, z_2) \in Z_1 \times Z_2$, the coefficient of $(x_1, x_2)(y_1, y_2)(z_1, z_2)$ in $T_1 \otimes T_2$ is the product of the coefficient of $x_1 y_1 z_1$ in T_1, and the coefficient of $x_2 y_2 z_2$ in T_2. In other words, if

$$T_1 = \sum_{\substack{x_1 \in X_1\\ y_1 \in Y_1\\ z_1 \in Z_1}} \alpha_{x_1 y_1 z_1} x_1 y_1 z_1, \qquad \text{and} \qquad T_2 = \sum_{\substack{x_2 \in X_2\\ y_2 \in Y_2\\ z_2 \in Z_2}} \beta_{x_2 y_2 z_2} x_2 y_2 z_2,$$

$$\text{then, } T_1 \otimes T_2 = \sum_{\substack{(x_1,x_2) \in X_1 \times X_2\\ (y_1,y_2) \in Y_1 \times Y_2\\ (z_1,z_2) \in Z_1 \times Z_2}} \alpha_{x_1 y_1 z_1} \beta_{x_2 y_2 z_2} (x_1, x_2)(y_1, y_2)(z_1, z_2).$$

The tensor product $T_1 \otimes T_2$ is exactly the product of T_1 and T_2 as polynomials, except that instead of viewing the result as a degree 6 polynomial, we continue to view it as a degree 3 polynomial by merging variables $x_1 x_2 \to (x_1, x_2)$ and similarly for the y and z variables.

The nth tensor power of a tensor A, denoted $A^{\otimes n}$, is the result of taking the tensor product of n copies of A together, so $A^{\otimes 1} = A$, and $A^{\otimes n} = A \otimes A^{\otimes (n-1)}$. Hence, if T is over X, Y, Z, then $T^{\otimes n}$ is over X^n, Y^n, Z^n, and its variables are n-tuples of variables from T. We will use this view in some of our proofs in Chapter 5.

Tensor products interact very nicely with the notions and tensors we have already introduced. First, for $a_1, a_2, b_1, b_2, c_1, c_2 \in \mathbb{N}$, we have $\langle a_1, b_1, c_1\rangle \otimes \langle a_2, b_2, c_2\rangle = \langle a_1 a_2, b_1 b_2, c_1 c_2\rangle$. This corresponds to performing $a_1 a_2 \times b_1 b_2 \times c_1 c_2$ matrix multiplication using block matrices, reducing the problem to the multiplication of $a_1 \times b_1 \times c_1$ matrices whose entries are $a_2 \times b_2$ and $b_2 \times c_2$ matrices. It particular, it follows that for all $q, n \in \mathbb{N}$, we have $\langle q, q, q\rangle^{\otimes n} = \langle q^n, q^n, q^n\rangle$.

Second, for any two tensors A, B, we always have $R(A \otimes B) \leq R(A) \cdot R(B)$.

This follows by combining the distributive property with the fact that the product of two rank one tensors also has rank one. However, this inequality is often not tight. For instance, it is known that $R(\langle 2,2,2\rangle) = 7$ (the lower bound was shown by Winograd [Win71]), but for large n, we can see that $R(\langle 2,2,2\rangle^{\otimes n}) \leq 2^{\omega n + o(n)} \leq 5.18^{n+o(n)}$.

This motivates defining the *asymptotic rank*[2] of a tensor T as

$$\tilde{R}(T) := \liminf_{n \in \mathbb{N}} (R(T^{\otimes n}))^{1/n}.$$

Because of the tensor product properties above, we can alternatively define ω in a number of ways:

$$\omega = \liminf_{q \in \mathbb{N}} \log_q R(\langle q,q,q\rangle) = \liminf_{q \in \mathbb{N}} \log_q \tilde{R}(\langle q,q,q\rangle) = \log_2(\tilde{R}(\langle 2,2,2\rangle)).$$

4.4 Reductions Between Tensors

We now describe four different ways to reduce between tensors. The key property we would like from a type of reduction is that, if A reduces to B, then $\tilde{R}(A) \leq \tilde{R}(B)$. Thus, if we can show that a MM tensor reduces to some tensor T, then in order to prove upper bounds on ω, it suffices to prove upper bounds on $\tilde{R}$. The four types of reductions we will define are summarized in Figure 4-2.

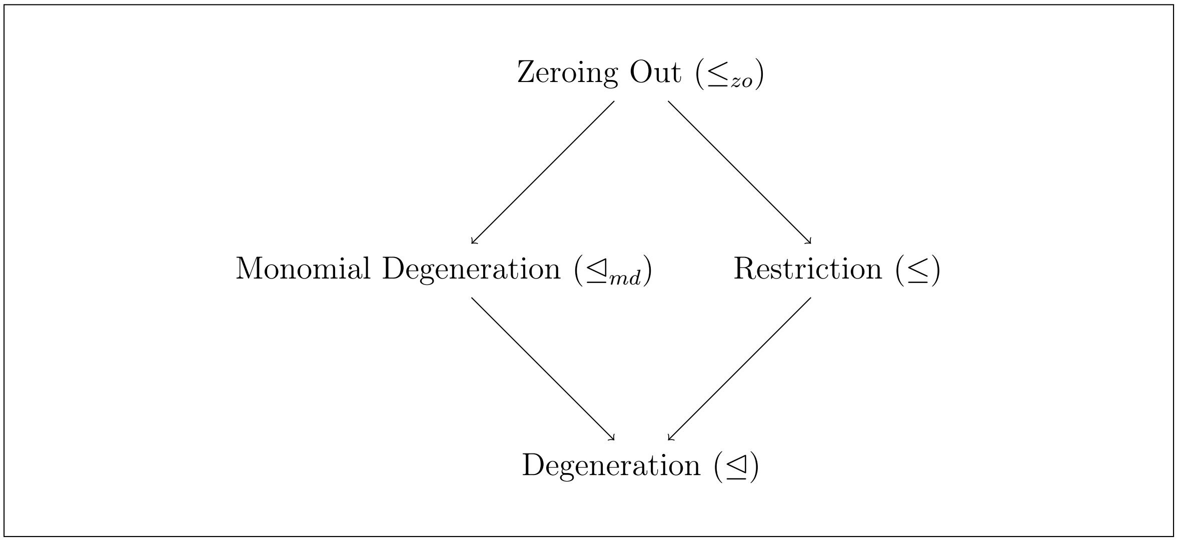
Zeroing Out ($\leq_{zo}$)
Monomial Degeneration ($\trianglelefteq_{md}$)
Restriction ($\leq$)
Degeneration ($\trianglelefteq$)

Figure 4-2: The four notions of a reduction between tensors. An arrow $\leq_0 \;\rightarrow\; \leq_1$ means that $\leq_1$ subsumes $\leq_0$, meaning that for any tensors A, B, if $A \leq_0 B$, then $A \leq_1 B$.

[2] The asymptotic rank is often written as $\underset{\sim}{R}$ in the literature, but we instead write $\tilde{R}$ for ease of notation.

Zeroing Out The simplest type of reduction is a zeroing out (also called a *combinatorial restriction*). Let B be a tensor over X, Y, Z, and A be a tensor over X', Y', Z' where $X' \subseteq X$, $Y' \subseteq Y$, and $Z' \subseteq Z$. A is a zeroing out of B, denoted $A \leq_{zo} B$, if for all $x \in X'$, $y \in Y'$, and $z \in Z'$, the coefficient of xyz in A equals the coefficient of xyz in B. In other words, A is obtained by setting to zero all $x \in X \setminus X', y \in Y \setminus Y'$, and $z \in Z \setminus Z'$. It is not hard to see that if $A \leq_{zo} B$ then $R(A) \leq R(B)$ and $\tilde{R}(A) \leq \tilde{R}(B)$, by applying the same zeroing out to the (asymptotic) rank expression for B.

Example 4.1. *For the tensors*

$$\begin{aligned} B &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1 + x_0y_1z_1, \\ A &= x_0y_0z_0 + x_1y_1z_0, \end{aligned}$$

we see that $A \leq_{zo} B$ by setting $z_1 \leftarrow 0$ in B, since the terms in A are exactly the terms in B that do not contain z_1. By comparison, for the tensor

$$C = x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1,$$

there is no zeroing out from B to C, since the term $x_0y_1z_1$ from B we would like to remove shares each of its variables with a term in C that we need to keep.

Although zeroing outs are quite simple, we will see that the best known approaches to designing MM algorithms only make use of zeroing outs to reduce to MM tensors, rather than the following more powerful methods.

Monomial Degeneration Let A, B be tensors over X, Y, Z. We say that A is a monomial degeneration of B, denoted $A \trianglelefteq_{md} B$, if the following type of transformation from B to A is possible. For a formal variable λ, let $Mon := \{\lambda^p \mid p \in \mathbb{Z}^{\geq 0}\}$. Pick a map $m : X \cup Y \cup Z \to Mon$, then from the tensor

$$B = \sum_{x \in X, y \in Y, z \in Z} \beta_{xyz} xyz,$$

with coefficients from $\mathbb{F}$, consider the transformed tensor

$$B' = \sum_{x \in X, y \in Y, z \in Z} m(x) \cdot m(y) \cdot m(z) \cdot \beta_{xyz} xyz,$$

with coefficients from $\mathbb{F}[\lambda]$. B' can alternatively be viewed as a polynomial in λ whose coefficients are tensors over X, Y, Z with coefficients from $\mathbb{F}$. If h is the smallest integer for which the coefficient of λ^h in B' is nonzero, and A is the coefficient of λ^h in B', then this is a monomial degeneration from B to A.

If $A \leq_{zo} B$, then $A \trianglelefteq_{md} B$ as well: if A is a zeroing out of B by setting the variables in $S \subseteq X \cup Y \cup Z$ to zero, then A is also a monomial degeneration of B by picking $m(w) = \lambda$ for all $w \in S$, and $m(w) = 1$ for all $w \in X \cup Y \cup Z \setminus S$, since then A will be the constant coefficient of the resulting B'. We will see in Example 4.2 below that monomial degeneration *strictly* subsumes zeroing out. It is not evident that monomial degenerations should preserve any notion of tensor rank, but Bini [Bin80] showed that for any tensors A, B, if $A \leq_{md} B$, then $\tilde{R}(A) \leq \tilde{R}(B)$.

Example 4.2. *For the tensors*

$$\begin{aligned} B &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1 + x_0y_1z_1, \\ C &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1, \end{aligned}$$

we see that $C \trianglelefteq_{md} B$ by picking $m(x_0) = m(y_1) = m(z_1) = \lambda$, and $m(x_1) = m(y_0) = m(z_0) = 1$. The resulting tensor B' is

$$B' = \lambda x_0y_0z_0 + \lambda x_1y_1z_0 + \lambda x_1y_0z_1 + \lambda^3 x_0y_1z_1,$$

so we have $B' = \lambda C + \lambda^3 T$ for some tensor T. B and C are the same tensors from Example 4.1, showing that monomial degeneration is strictly more powerful than zeroing out.

Restriction Let A be a tensor over X', Y', Z', and B be a tensor over X, Y, Z. We say A is a restriction of B, denoted $A \leq B$, if there are linear maps $M_X : \mathbb{F}^X \to \mathbb{F}^{X'}$, $M_Y : \mathbb{F}^Y \to \mathbb{F}^{Y'}$, $M_Z : \mathbb{F}^Z \to \mathbb{F}^{Z'}$ such that $A = B \circ (M_X, M_Y, M_Z)$. In other words, if

$$B = \sum_{x \in X, y \in Y, z \in Z} \beta_{xyz} xyz,$$

then

$$A = \sum_{x \in X, y \in Y, z \in Z} \beta_{xyz} M_X(x) M_Y(y) M_Z(z).$$

Unlike in zeroing outs and monomial degenerations, we do not need that $X' \subseteq X$, and it might even be the case that $|X'| > |X|$. If $A \leq_{zo} B$, then $A \leq B$, since a zeroing out corresponds to a restriction where, for each $x \in X$, we either pick $M_X(x) = x$ or $M_X(x) = 0$ (and similarly for Y and Z).

Example 4.3. *For the tensors*

$$\begin{aligned} D &= x_+y_+z_+ + x_-y_-z_-, \\ B &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1 + x_0y_1z_1, \end{aligned}$$

we see that $B \leq D$ by picking

$$x_+ \leftarrow x_0 + x_1, \quad y_+ \leftarrow y_0 + y_1, \quad z_+ = \frac{1}{2}(z_0 + z_1),$$

$$x_- \leftarrow x_0 - x_1, \quad y_- \leftarrow y_0 - y_1, \quad z_- = \frac{1}{2}(z_0 - z_1).$$

The resulting transformation of D is

$$\frac{(x_0+x_1)(y_0+y_1)(z_0+z_1)+(x_0-x_1)(y_0-y_1)(z_0-z_1)}{2},$$

which when expanded gives B.

The transformation of D in Example 4.3 is the sum of two rank one tensors, showing that B has rank at most 2. In fact, restrictions can be used in this way to exactly characterize rank.

For $q \in \mathbb{N}$, write $\langle q \rangle := \sum_{i=1}^{q} x_i y_i z_i$ for the *independent tensor of size* q, which has q terms which do not share any variables with each other. For instance, in Example 4.3, $D = \langle 2 \rangle$. Since $\langle 1 \rangle$ can be restricted to any rank one tensor, and restrictions act independently on each term of $\langle q \rangle$, we see that:

Proposition 4.2. *For any $q \in \mathbb{N}$ and tensor T, there is a restriction $T \leq \langle q \rangle$ if and only if $R(T) \leq q$.*

We can see that $A \leq B \leq C$ implies that $A \leq C$ by simply composing the corresponding linear maps; it follows from Proposition 4.2 that $A \leq B$ implies $R(A) \leq R(B)$.

Proposition 4.2 illustrates how difficult it can be to determine whether there is a restriction between two given tensors, since tensor rank is hard to compute even for small explicit tensors. For instance, although $R(\langle 2,2,2 \rangle) = 7$ is known, determining the value of $R(\langle 3,3,3 \rangle)$ is *open*. The best known upper bound is $R(\langle 3,3,3 \rangle) \leq 23$, and so we do not know whether there is a restriction $\langle 3,3,3 \rangle \leq \langle 22 \rangle$. For more on the computational difficulty of determining whether there is a restriction between given tensors, see e.g. [GQ19].

Before continuing, we note that the independent tensor $\langle q \rangle$ will appear again numerous times throughout this dissertation. For one example, it can be used to capture the disjoint sum of a tensor with itself:

Definition 4.1. *For any tensors T_1 over X_1, Y_1, Z_1 and T_2 over X_2, Y_2, Z_2, their* direct sum *$T_1 \oplus T_2$ is a tensor over $X_1 \sqcup X_2, Y_1 \sqcup Y_2, Z_1 \sqcup Z_2$ given by the sum (as polynomials) of T_1 and T_2. For $q \in \mathbb{N}$ and tensor T, we write $q \odot T$ for the disjoint sum of q copies of T. Note that $q \odot T = \langle q \rangle \otimes T$.*

Degeneration The most powerful known asymptotic rank-preserving reduction between tensors is a degeneration[3]. It combines the power of the formal variable λ from monomial degenerations with the linear transformations from restrictions.

Let A be a tensor over X', Y', Z', and B be a tensor over X, Y, Z. We say A is a degeneration of B, denoted $A \trianglelefteq B$, if there are linear maps $M_X : \mathbb{F}^X \to \mathbb{F}[\lambda]^{X'}$, $M_Y : \mathbb{F}^Y \to \mathbb{F}[\lambda]^{Y'}$, $M_Z : \mathbb{F}^Z \to \mathbb{F}[\lambda]^{Z'}$, whose ranges are linear combinations of the variables of A whose coefficients are *polynomials in* λ such that: when $B' = B \circ (M_X, M_Y, M_Z)$

[3] Slightly more powerful notions, like degenerations with multiple λ variables, can be captured by straightforward modifications to the notion of degeneration which preserve all the results about degenerations which we prove below.

is viewed as a polynomial in λ, and h is the smallest integer such that the coefficient of λ^h in B' is nonzero, then the coefficient of λ^h in B' is A.

Example 4.4. *For the tensors*

$$\begin{aligned} D &= x_+y_+z_+ + x_-y_-z_-, \\ C &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1, \\ B &= x_0y_0z_0 + x_1y_1z_0 + x_1y_0z_1 + x_0y_1z_1, \end{aligned}$$

we showed in Example 4.2 that $C \trianglelefteq_{md} B$, and we showed in Example 4.3 that $B \leq D$. Composing the two transformations shows that $C \trianglelefteq D$. Although composing a monomial degeneration and a restriction like this is one way to give a degeneration, there are more degenerations not captured by this approach as well.

The result of Bini [Bin80] implies that if $A \trianglelefteq B$ then $\tilde{R}(A) \leq \tilde{R}(B)$, just as it did for monomial degenerations. In particular, if $A \trianglelefteq \langle q \rangle$, then $\tilde{R}(A) \leq q$.

4.5 The Universal Method

We now have all the ingredients in place to define the *Universal Method* for designing MM algorithms, which subsumes and greatly generalizes the known approaches for designing MM algorithms. The **Universal Method** applied to a tensor T consists of two components:

(1) a bound $\tilde{R}(T) \leq r$ on the asymptotic rank of T, and

(2) a degeneration $\langle q, q, q \rangle \trianglelefteq T^{\otimes n}$, which reduces MM to a tensor power of T.

Combined, the two components imply that $\tilde{R}(\langle q, q, q \rangle) \leq r^n$, and hence that $\omega \leq n \log_q r$. We write $\omega_u(T)$ for the lim inf over all bounds on ω which can be proved in this way, including picking r to be the true asymptotic rank of T, and for each n picking the largest q such that the degeneration required for step (2) exists.

The Asymptotic Sum Inequality? Readers familiar with Schönhage's Asymptotic Sum Inequality may wonder why step (2) in the Universal Method requires a degeneration to a single MM tensor rather than a disjoint sum of many. Indeed, Schönhage shows that degenerations to disjoint sums of MM tensors are sufficient to bound ω:

Theorem 4.1 (Asymptotic Sum Inequality [Sch81])**.** *If $\tilde{R}\left(f \odot \langle q, q, q \rangle\right) \leq r$, then $\omega \leq \log(r/f)/\log(q)$.*[4]

[4] Schönhage's Asymptotic Sum Inequality allows for a disjoint sum of MM tensors of many different shapes (values of q) as well, but the first step of its proof shows that, by taking large tensor powers, one can assume without loss of generality that all the MM tensors have the same shape.

It may seem like one could prove a better bound on ω by degenerating to a disjoint sum of MM tensors rather than a single MM tensor. However, this technique is actually captured by the Universal Method as well; one can always find degenerations from powers of T to a *single* MM tensor which achieve an equally good bound on ω:

Proposition 4.3. *If T is a tensor with $\tilde{R}(T) = r$ and $f \odot \langle q, q, q \rangle \trianglelefteq T$, then $\omega_u(T) \leq \log(r/f)/\log(q)$.*

Proof. By definition of ω, for every $\varepsilon > 0$, there is a $m \in \mathbb{N}$ such that $\langle a, a, a \rangle \trianglelefteq \langle f^m \rangle$ and $a \geq f^{m/(\omega+\varepsilon)} \geq f^{m/(\omega_u(T)+\varepsilon)}$ (where the second inequality holds because $\omega_u(T) \geq \omega$). In particular, we have that

$$T^{\otimes m} \trianglerighteq \langle f^m \rangle \otimes \langle q^m, q^m, q^m \rangle \trianglerighteq \langle aq^m, aq^m, aq^m \rangle,$$

which yields the bound

$$\omega_u(T) \leq \frac{\log(r^m)}{\log(aq^m)} \leq \frac{\log(r^m)}{\log(f^{m/(\omega_u(T)+\varepsilon)} q^m)} = \frac{\log(r^{\omega_u(T)+\varepsilon})}{\log(f q^{\omega_u(T)+\varepsilon})}.$$

Rearranging yields

$$f \cdot q^{\omega_u(T)+\varepsilon} \leq r^{1+\varepsilon/\omega_u(T)} \leq r^{1+\varepsilon},$$

and hence

$$\omega_u(T) \leq \frac{\log(r^{1+\varepsilon}/f)}{\log(q)} - \varepsilon = \frac{\log(r/f)}{\log(q)} + \varepsilon \cdot \left(\frac{\log(r)}{\log(q)} - 1 \right).$$

Since this holds for all sufficiently small $\varepsilon > 0$, it implies that $\omega_u(T) \leq \log(r/f)/\log(q)$ as desired. □

Rectangular MM Tensors? Step (2) in the Universal Method requires a degeneration to a *square* MM tensor, but degenerations to rectangular MM tensors also give bounds on ω. Indeed, if $\tilde{R}(\langle a, b, c \rangle) \leq r$, then by the symmetry of $\langle a, b, c \rangle$ we also get that $\tilde{R}(\langle b, c, a \rangle) \leq r$ and $\tilde{R}(\langle c, a, b \rangle) \leq r$, which combined mean that $\tilde{R}(\langle abc, abc, abc \rangle) \leq r^3$, and hence $\omega \leq 3 \log(r)/\log(abc)$.

If tensor T is '*variable-symmetric*', then a similar argument shows that, in the Universal Method applied to T, allowing for degenerations to rectangular MM tensors cannot help.

Definition 4.2. *If T is a tensor over X, Y, Z, then the* rotation *of T, denoted $rot(T)$, is the tensor over Y, Z, X such that for any $(x_i, y_j, z_k) \in X \times Y \times Z$, the coefficient of $x_i y_j z_k$ in T is equal to the coefficient of $y_j z_k x_i$ in $rot(T)$. Tensor T is* variable-symmetric *if $T = rot(T)$.*

The symmetrized *version of T, denoted $sym(T)$, is given by $sym(T) = T \otimes rot(T) \otimes rot(rot(T))$. For any tensor T, the tensor $sym(T)$ is always variable-symmetric. Moreover, if T was variable-symmetric, then $sym(T) = T^{\otimes 3}$.*

For any tensor T, if $\tilde{R}(T) = r$ and $T^{\otimes n} \trianglerighteq \langle a, b, c \rangle$ to yield $\omega \leq 3n \log(r)/\log(abc)$, then it follows that $sym(T)^{\otimes n} \trianglerighteq \langle abc, abc, abc \rangle$, meaning

$\omega_u(sym(T)) \leq 3n\log(r)/\log(abc)$. In other words, any bound on ω which we could achieve if we allowed for rectangular MM tensors in step (2) of the Universal Method applied to T, can also be achieved by applying the Universal Method as written to $sym(T)$. Notably, almost every tensor we will consider in the remainder of this chapter is variable-symmetric, for which $sym(T) = T$.

In the previous paragraph we used the fact that $\tilde{R}(sym(T)) \leq \tilde{R}(T)^3 = r^3$, but if T is such that $\tilde{R}(sym(T)) < \tilde{R}(T)^3$ then we get an even better bound of $\omega_u(sym(T)) \leq n\log(\tilde{R}(sym(T)))/\log(abc) < 3n\log(r)/\log(abc)$. To prove the best bounds on ω when using a tensor T which is not variable-symmetric, one should always apply the Universal Method to $sym(T)$ rather than T.

Example 4.5. *For any $q \in \mathbb{N}$, consider the tensor $T = \langle q, 1, 1\rangle$. A 'flattening' argument*[5] *shows that $\tilde{R}(T) = q$. However, $sym(T) = \langle q, q, q\rangle$, and so $\tilde{R}(sym(T)) = q^\omega < q^3 = \tilde{R}(T)^3$ when $q > 1$.*

4.5.1 The Solar and Galactic Methods

The Universal Method is very general, and it is typically unclear how to apply it optimally to a given tensor T. In fact, all known approaches to designing MM algorithms use a substantially restricted method, which uses zeroing outs instead of degenerations, which we call the Solar Method. The **Solar Method** applied to a tensor T consists of two components:

(1) a bound $\tilde{R}(T) \leq r$ on the asymptotic rank of T, and

(2) a zeroing out $f \odot \langle q, q, q\rangle \leq_{zo} T^{\otimes n}$.

The resulting bound on ω is that $\omega \leq \log(r^n/f)/\log(q)$, and the lim inf over all bounds on ω which can be achieved in this way is denoted $\omega_s(T)$. Here, we need to allow for a zeroing out into a disjoint sum of multiple MM tensors in step (2), unlike in the Universal Method, since Proposition 4.3 critically makes use of degenerations, and not just zeroing outs, in the first step of the proof.

One can also define an intermediate method, the **Galactic Method**, which is identical to the Solar Method except that the zeroing out $\leq_{zo}$ in step (2) is replaced by the more powerful monomial degeneration $\trianglelefteq_{md}$, and the best resulting bound on ω is denoted $\omega_g(T)$. One could also consider an incomparable intermediate method which makes use of *restrictions* instead of monomial degenerations, but since such a method has not been studied much, and is captured by the Universal Method, it doesn't yet have a name.

Because each of degeneration, monomial degeneration, and zeroing out (strictly) subsumes the previous, we get that for all tensors T,

$$\omega \leq \omega_u(T) \leq \omega_g(T) \leq \omega_s(T).$$

[5] When T is viewed as a matrix by removing the z-variable (i.e. setting $z_1 \leftarrow 1$), then $T^{\otimes n}$ becomes the $q^n \times q^n$ identity matrix, which has rank q^n. If it were the case that $R(T^{\otimes n}) < q^n$, then similarly removing z_1 from the rank expression would give an upper bound of $R(T^{\otimes n})$ on the rank of the $q^n \times q^n$ identity matrix as well.

To be clear, all three of these methods are very general, and we don't know the values of $\omega_s(T)$, $\omega_g(T)$, or $\omega_u(T)$ for almost any nontrivial tensors T. In fact, all the known approaches to bounding ω proceed by giving upper bounds on $\omega_s(T)$ for some carefully chosen tensors T; the most successful has been the Coppersmith-Winograd family of tensors $T = CW_q$, which has yielded all the best known bounds on ω since the 80's [CW90, DS13, Wil12, LG14]. In particular, it is not generally believed that our current upper bounds on $\omega_s(CW_q)$ are optimal, since the techniques from [DS13, Wil12, LG14] seem able to further improve on the current bounds (by a small amount) with more effort.

Finally, we remark that the tensor T which these methods apply to is very crucial. The three different methods will trivially give the same bound, $\omega_s(T) = \omega_g(T) = \omega_u(T) = \omega$, when applied to $T = \langle 2, 2, 2\rangle$ itself, but this is not particularly interesting: the point of these different methods is that the asymptotic rank of matrix multiplication tensors is not well-understood, but the methods allow us to prove bounds on ω by studying *other* tensors.

4.6 The Known Approaches to Matrix Multiplication

We conclude this chapter by giving an overview of the two known approaches to designing MM algorithms: the Laser Method and the Group-theoretic Method. Each of these methods applies to particular tensors T to yield bounds in $\omega_s(T)$; in particular, we will see that the Solar Method is a generalization of both of these methods. We only give high-level overviews here, mostly just for context, and we recommend the original papers (cited below) for more details.

4.6.1 The Laser Method

Strassen [Str86] called his approach for reducing MM tensors to other tensors the *Laser Method*. In this method applied to a tensor T over X, Y, Z, we start by partitioning the variable sets: $X = X_1 \cup \ldots \cup X_{k_X}$, $Y = Y_1 \cup \ldots \cup Y_{k_Y}$, $Z = Z_1 \cup \ldots, Z_{k_Z}$. For $i \in [k_X], j \in [k_Y], k \in [k_Z]$, let T_{ijk} be the sub-tensor of T obtained by zeroing out all variables $x \notin X_i$, $y \notin Y_j$, and $z \notin Z_k$; we call T_{ijk} a *block* of T. We thus obtain a partitioning

$$T = \sum_{i\in[k_X], j\in[k_Y], k\in[k_Z]} T_{ijk}.$$

Strassen originally considered tensors where the *constituent* tensors T_{ijk} are each MM tensors; we focus here on this setting, although later work showed how to remove this requirement.

The next step is to consider a large tensor power $T^{\otimes N}$ of T. We can write

$$T^{\otimes N} = \sum_{I \in [k_X]^N, J \in [k_Y]^N, K \in [k_Z]^N} \bigotimes_{\ell=1}^{N} T_{I_\ell J_\ell K_\ell}. \tag{4.4}$$

Each of the terms on the right-hand side of (4.4) is a MM tensor, and the goal is to ultimately apply Schönhage's Asymptotic Sum Inequality (Theorem 4.1 above) to yield a bound on ω. However, sum the right-hand side of (4.4) is not a disjoint sum, or in other words, the different MM tensors *share variables* with each other, so we cannot apply the Asymptotic Sum Inequality directly.

In the Laser Method, we seek to zero out some variables to fix this problem. We aim to choose some $\mathcal{I} \subseteq [k_X]^N$, and then for each $I \in \mathcal{I}$, to zero out all the variables in $\prod_\ell I_\ell$ (and similarly for y and z variables). One must find such a zeroing out which leaves exactly a direct sum of matrix multiplication tensors. Finally, applying the Asymptotic Sum Inequality yields a bound on ω.

We now turn to the most successful implementation of the Laser Method: the **Coppersmith-Winograd** approach. The Coppersmith-Winograd (CW) family of tensors $(CW_q)_{q \in \mathbb{N}}$ is defined as:

$$CW_q = x_0 y_0 z_{q+1} + x_{q+1} y_0 z_0 + x_0 y_{q+1} z_0 + \sum_{i=1}^{q} (x_i y_0 z_i + x_0 y_i z_i + x_i y_i z_0).$$

CW_q is a tensor over $q+2$ x-variables, $q+2$ y-variables, and $q+2$ z-variables, and it is known that $\tilde{R}(CW_q) = q+2$ (The upper bound $\tilde{R}(CW_q) \leq q+2$ was a key contribution of [CW90], and the lower bound follows from a 'flattening' argument).

Coppersmith and Winograd [CW90] followed the laser method. The terms of CW_q have a natural partitioning $CW_q = T_{002} + T_{020} + T_{200} + T_{011} + T_{101} + T_{110}$, where $T_{002} = x_0 y_0 z_{q+1}, T_{200} = x_{q+1} y_0 z_0, T_{020} = x_0 y_{q+1} z_0, T_{101} = \sum_{i=1}^{q} x_i y_0 z_i, T_{011} = \sum_{i=1}^{q} x_0 y_i z_i, T_{110} = \sum_{i=1}^{q} x_i y_i z_0$. This partitioning has three key properties. First, each of these parts is a MM tensor. Second, this partitioning arises from a block-partitioning of the variables where we partition $X = X_0 \cup X_1 \cup X_2$ where $X_0 = \{x_0\}$, $X_1 = \{x_1, x_2, \ldots, x_q\}$, and $X_2 = \{x_{q+1}\}$, and similarly for Y and Z. Third, each sub-tensor T_{ijk} is non-zero if and only if $i + j + k = 2$.

The Coppersmith-Winograd implementation of the laser method uses these properties together with sets excluding 3-term arithmetic progressions (in conjunction with the third property above) to decide which blocks of variables to zero out in $CW_q^{\otimes N}$. The sets excluding 3-term arithmetic progressions can be used to guarantee that the result is a direct sum of many large matrix multiplication tensors, thus obtaining a bound on ω. Coppersmith and Winograd get a different bound on ω for each q, and optimize it by picking $q = 6$. They then achieve a slightly better bound on ω by analyzing the square $CW_q^{\otimes 2}$ in a similar way. In Theorem 5.5 in the next Chapter, we present all the details of Coppersmith and Winograd's application of the Laser Method.

The later improvements on the Coppersmith-Winograd bounds by

Stothers [DS13], Vassilevska Williams [Wil12] and Le Gall [LG14] instead used the laser method with the CW tools starting from $CW_q^{\otimes 4}, CW_q^{\otimes 8}$ and $\{CW_q^{\otimes 16}$ and $CW_q^{\otimes 32}\}$, respectively. Each new analysis used different, but related, blockings and partitionings, and each ultimately optimizes the resulting bound on ω by picking $q = 5$, and hence using CW_5 as the starting tensor. In other words, each ultimately gives an upper bound on $\omega_s(CW_5)$. The constituent tensors of powers of CW_q are sometimes not MM tensors. To deal with this, one (recursively) performs a Coppersmith-Winograd analysis on the constituent tensors, showing that they, themselves, can zero out into large MM tensors at high enough tensor powers. As the power of CW_q which is considered grows, the number of recursive analyses needed becomes very large.

The Coppersmith-Winograd approach doesn't exploit very much about the constituent tensors T_{ijk}. In particular, the analysis remains unchanged if one replaces each T_{ijk} with another tensor T'_{ijk} over the same sets of variables X_i, Y_j, Z_k, as long as T'_{IJK} has the same "value" (i.e. can degenerate to equally large MM tensors in high tensor powers), and the modified tensor T' has the same asymptotic rank as T. In this case, the bound on ω the approach would give is exactly the same! For instance, when T_{ijk} is a matrix multiplication tensor $\langle a, b, c\rangle$, one can replace it with another matrix multiplication tensor $\langle a', b', c'\rangle$ as long as the new tensor uses the same variables and $a'b'c' = abc$, and as long as the the asymptotic rank of the overall tensor has not increased. For instance, if we take $T_{110} = \sum_{i=1}^q \sum_{j=1}^q x_i y_j z_0$ in CW_q and replace it with $\sum_{i=1}^q \sum_{j=1}^q x_i y_{q+1-i} z_0$, then we get the *rotated* CW_q tensor studied in [AW18a]. This tensor still has asymptotic rank $q + 2$, and thus gives the same upper bound on ω using the CW approach.

We can thus define a family of *generalized* CW tensors:

Definition 4.3. *The family $\underline{CW}_q$ of tensors includes, for every permutation $\sigma \in S_q$, the tensor*

$$CW_q^\sigma = (x_0 y_0 z_{q+1} + x_0 y_{q+1} z_0 + x_{q+1} y_0 z_0) + \sum_{i=1}^{q} (x_i y_{\sigma(i)} z_0 + x_i y_0 z_i + x_0 y_i z_i).$$

The family above contains all tensors obtained from CW_q by replacing $\sum_{i=1}^q (x_i y_i z_0 + x_i y_0 z_i + x_0 y_i z_i)$ with $\sum_{i=1}^q (x_{\tau(i)} y_{\sigma(i)} z_0 + x_{\alpha(i)} y_0 z_{\beta(i)} + x_0 y_{\gamma(i)} z_{\delta(i)})$ for any choice of $\alpha, \beta, \gamma, \delta, \sigma, \tau \in S_q$, by a simple renaming of variables.

The constituent tensor T_{110} of CW_q^σ is $\sum_{i=1}^q x_i y_{\sigma(i)} z_0$, which is still a $\langle 1, q, 1\rangle$ tensor. Thus, for any such tensor from the family $\underline{CW}_q$, if its asymptotic rank is $q + 2$, then the Coppersmith-Winograd approach would give exactly the same bound on ω, as with CW_q. Unfortunately, the asymptotic rank lower bounding technique applies equally well to any tensor in $\underline{CW}_q$, showing that they all have asymptotic rank at least $q + 2$, so our upper bounds on ω cannot be improved just in this way.

4.6.2 The Group-theoretic Method

Cohn and Umans [CU03] pioneered a new Group-theoretic Method for matrix multiplication, which works with a finite group G rather than directly with a tensor.

Roughly, they define properties of a group such that, if G has these properties, then it is possible to zero out an underlying tensor corresponding to G into a MM tensor.

Definition 4.4. *For any finite group G, the* group tensor of G*, denoted T_G, is a tensor over X_G, Y_G, Z_G where $X_G := \{x_g \mid g \in G\}$, $Y_G := \{y_g \mid g \in G\}$, and $Z_G := \{z_g \mid g \in G\}$, given by*

$$T_G := \sum_{g,h \in G} x_g y_h z_{gh}.$$

(The group tensor of G is often called the structural tensor of the group algebra $\mathbb{C}[G]$, written as $T_{\mathbb{C}[G]}$, in the literature. We write T_G here for ease of notation.)

The Group-theoretic Method first bounds the asymptotic rank of T_G using representation theory, as follows. Let d_u be the dimension of the uth irreducible representation of G (i.e. the d_us are the character degrees of G). Then T_G can be seen to restrict from $\bigoplus_{u=1}^{\ell} \langle d_u, d_u, d_u \rangle$. In particular, we get that[6]

$$\tilde{R}(T_G) = \tilde{R}\left(\bigoplus_{u=1}^{\ell} \langle d_u, d_u, d_u \rangle\right) = \sum_{u=1}^{\ell} d_u^{\omega}.$$

If we could find any degeneration (e.g. a zeroing out) of T_G into $\langle q, q, q \rangle$, it would imply that

$$q^{\omega} \leq \sum_{u=1}^{\ell} d_u^{\omega},$$

which gives an upper bound on ω.

Cohn and Umans defined two properties of subsets of G which yield a zeroing out of T_G into a MM tensor, called the 'triple product property', and the 'simultaneous triple product property' (which zeroes out into a disjoint sum of MM tensors). Hence, bounds on ω can follow from showing that a group G has large subsets with these properties. Typically one works with a family of groups (as in A_n or S_n for all $n \in \mathbb{N}$), and then one can pick the n that optimizes the bound on ω, or even take $n \to$ '∞, e.g. when the groups correspond to tensor powers of some tensor. We refer the reader to [Lan17, Section 3.5] for more exposition on the Group-theoretic Method and its interpretation as finding a zeroing out of group tensors.

[6] It is more straightforward to see that this holds with inequalities ('$\leq$' instead of '$=$') but in fact equality holds because the corresponding restriction of T_G is invertible.

Chapter 5

Limits on the Universal Method

5.1 Asymptotic Slice Rank

In this chapter, we will prove new *limitation results* against the Universal Method. Our limitations will critically make use of variant on the rank of a tensor, called *slice rank*. We begin in this section by introducing slice rank and its key properties.

We say a tensor T over X, Y, Z has *x-rank* one if it is of the form

$$T = \left(\sum_{x\in X} \alpha_x \cdot x\right) \otimes \left(\sum_{y\in Y}\sum_{z\in Z} \beta_{y,z} \cdot y \otimes z\right) = \sum_{x\in X, y\in Y, z\in Z} \alpha_x \beta_{y,z} \cdot xyz$$

for some choices of the α_x and $\beta_{y,z}$ coefficients over $\mathbb{F}$. More generally, the x-rank of T, denoted $S_x(T)$, is the minimum number of tensors of x-rank one whose sum is T. We can similarly define the y-rank, S_y, and the z-rank, S_z. Then, the *slice rank* of T, denoted $\mathcal{S}(T)$, is the minimum k such that there are tensors T_X, T_Y and T_Z with $T = T_X + T_Y + T_Z$ and $S_x(T_X) + S_y(T_Y) + S_z(T_Z) = k$.

Unlike tensor rank, the slice-rank is not submultiplicative in general, i.e. there are tensors A and B such that $\mathcal{S}(A \otimes B) > \mathcal{S}(A) \cdot \mathcal{S}(B)$. For instance, it is not hard to see that $\mathcal{S}(CW_5) = 3$, but since it is known [Wil12, LG14] that $\omega_s(CW_5) \leq 2.373$, it follows (e.g. from Theorem 5.1 below) that $\mathcal{S}(CW_q^{\otimes n}) \geq 7^{n\cdot 2/2.373-o(n)} \geq 5.15^{n-o(n)}$. We are thus interested in the *asymptotic slice rank*, $\tilde{S}(T)$, of tensors T, which is defined similarly to the asymptotic rank as

$$\tilde{S}(T) := \limsup_{n\in\mathbb{N}} [\mathcal{S}(T^{\otimes n})]^{1/n}.$$

We note a few simple properties of slice rank which will be helpful in our proofs:

Lemma 5.1. *For tensors A and B:*

(1) $\mathcal{S}(A) \leq S_x(A) \leq R(A)$,

(2) $S_x(A \otimes B) \leq S_x(A) \cdot S_x(B)$,

(3) $\mathcal{S}(A + B) \leq \mathcal{S}(A) + \mathcal{S}(B)$, *and* $S_x(A + B) \leq S_x(A) + S_x(B)$,

(4) $S(A \otimes B) \leq S(A) \cdot \max\{S_x(B), S_y(B), S_z(B)\}$*, and*

(5) If A *is a tensor over* X, Y, Z*, then* $S_x(T) \leq |X|$ *and hence* $S(T) \leq \min\{|X|, |Y|, |Z|\}$*.*

Proof. (1) and (2) are straightforward. (3) follows since the sum of the slice rank (resp. x-rank) expressions for A and for B gives a slice rank (resp. x-rank) expression for $A + B$. To prove (4), let $m = \max\{S_x(B), S_y(B), S_z(B)\}$, and note that if $A = A_X + A_Y + A_Z$ such that $S_x(A_X) + S_y(A_Y) + S_z(A_Z) = S(A)$, then

$$A \otimes B = A_X \otimes B + A_Y \otimes B + A_Z \otimes B,$$

and so

$$\begin{aligned} S(A \otimes B) &\leq S(A_X \otimes B) + S(A_Y \otimes B) + S(A_Z \otimes B) \\ &\leq S_x(A_X \otimes B) + S_y(A_Y \otimes B) + S_z(A_Z \otimes B) \\ &\leq S_x(A_X)\, S_x(B) + S_y(A_Y)\, S_y(B) + S_z(A_Z)\, S_z(B) \\ &\leq S_x(A_X)m + S_y(A_Y)m + S_z(A_Z)m = S(A) \cdot m. \end{aligned}$$

Finally, (5) follows since, for instance, any tensor with one only x-variable has x-rank 1. □

5.2 Limits on the Universal Method from Asymptotic Slice Rank

Asymptotic slice rank is interesting in the context of MM algorithms and the Universal Method because of the following facts. First, degenerations cannot increase slice rank:

Proposition 5.1 ([TS16, Corollary 2])**.** *If* A *and* B *are tensors such that* $B \trianglelefteq A$*, then* $S(B) \leq S(A)$*, and hence* $\tilde{S}(B) \leq \tilde{S}(A)$*.*

Second, the independent tensor $\langle q \rangle$ has asymptotic slice rank q:

Proposition 5.2 ([Tao16, Lemma 1]; see also [BCC+17a, Lemma 4.7])**.** *For any positive integer* q*, we have* $S(\langle q \rangle) = \tilde{S}(\langle q \rangle) = q$*.*

Third, MM tensors have (monomial) degenerations to large independent tensors:

Proposition 5.3 ([Str86, Theorem 4])**.** *For any positive integers* a, b, c*, there is a* $q \geq \frac{3}{4}abc/\max\{a, b, d\}$ *such that* $\langle q \rangle \trianglelefteq_{md} \langle a, b, c \rangle$*.*

Proof. Assume first that $a = 2m + 1$, $b = 2n + 1$, and $c = 2p + 1$ are all odd, and assume without loss of generality that $c \geq a, b$. We write

$$\langle a, b, c \rangle = \sum_{i=-m}^{m} \sum_{j=-n}^{n} \sum_{k=-p}^{p} x_{ij} y_{jk} z_{ki}.$$

We define our monomial degeneration (using the notation of Section 4.4) via the map $m : X \cup Y \cup Z \to Mon$ defined as follows:

- $m(x_{ij}) = \lambda^{i^2+2ij+3p^2}$,
- $m(y_{jk}) = \lambda^{j^2+2jk+3p^2}$, and
- $m(z_{ki}) = \lambda^{k^2+2ki+3p^2}$.

For any term $x_{ij}y_{jk}z_{ki} \in \langle a, b, c\rangle$, we thus have $m(x_{ij}) \cdot m(y_{jk}) \cdot m(z_{ki}) = \lambda^{(i+j+k)^2+9p^2}$. This exponent of λ is always at least $9p^2$, and the term $x_{ij}y_{jk}z_{ki}$ is included in the resulting tensor D of the monomial degeneration if and only if $i + j + k = 0$. We can see that if $i + j + k = 0$, then any two of i, j, k determines the third, meaning any one of the variables x_{ij}, y_{jk}, z_{ki} determines the other two, and so D is indeed an independent tensor. Finally, there is a triple of (i, j, k), $|i| \leq n, |j| \leq m, |k| \leq p$ with $i + j + k = 0$ for each pair (i, j), $|i| \leq n, |j| \leq m$ with $|i + j| \leq p$. Since $p \geq n, m$, we can see there are at least $\frac{3}{4}ab$ such pairs, as desired. The cases where a, b, c are not all odd are similar. □

Combined, these three facts show that MM tensors have large asymptotic slice rank. In fact, it is as large as possible given its number of variables:

Corollary 5.1. *For any positive integers a, b, c, we have $\tilde{S}(\langle a, b, c\rangle) = abc/\max\{a, b, c\}$.*

Proof. Assume without loss of generality that $c \geq a, b$. For any positive integer n, we have that $\langle a, b, c\rangle^{\otimes n} = \langle a^n, b^n, c^n\rangle \trianglerighteq \langle 0.75 \cdot a^n b^n\rangle$, meaning $S(\langle a, b, c\rangle^{\otimes n}) \geq 0.75 \cdot a^n b^n$ and hence $\tilde{S}(\langle a, b, c\rangle) \geq (0.75)^{1/n} ab$. Since this holds for all $n \in \mathbb{N}$, we see that $\tilde{S}(\langle a, b, c\rangle) \geq ab$. Meanwhile, $\langle a, b, c\rangle$ has ab different x-variables, so it must have $S_x(\langle a, b, c\rangle) \leq ab$ and more generally, $S(\langle a, b, c\rangle^{\otimes n}) \leq S_x(\langle a, b, c\rangle^{\otimes n}) \leq (ab)^n$, which means $\tilde{S}(\langle a, b, c\rangle) \leq ab$. □

To summarize: we know that degenerations cannot increase asymptotic slice rank, and that matrix multiplication tensors have a high asymptotic slice rank. Hence, if T is a tensor such that $\omega_u(T)$ is 'small', meaning a power of T has a degeneration to a large matrix multiplication tensor, then T itself must have 'large' asymptotic slice rank. To be more precise:

Theorem 5.1. *For any tensor T,*

$$\omega_u(T) \geq 2\frac{\log(\tilde{R}(T))}{\log(\tilde{S}(T))}.$$

Proof. By definition of $\omega_u(T)$, for every $\delta > 0$, there are $n, q \in \mathbb{N}$ with $T^{\otimes n} \trianglerighteq \langle q, q, q\rangle$ and $q \geq \tilde{R}(T)^{n/(\omega_u(T)+\delta)}$. By Proposition 5.1 and Corollary 5.1, it follows that $\tilde{S}(T) \geq q^{2/n} \geq \tilde{R}(T)^{2/(\omega_u(T)+\delta)}$. Rearranging gives that $\omega_u(T) + \delta \geq 2\log(\tilde{R}(T))/\log(\tilde{S}(T))$, and since this holds for all $\delta > 0$, the desired result follows. □

Corollary 5.2. *For any tensor T, if $\omega_u(T) = 2$, then $\tilde{S}(T) = \tilde{R}(T)$. Moreover, for every constant $s < 1$, every tensor T with $\tilde{S}(T) \leq \tilde{R}(T)^s$ must have $\omega_u(T) \geq 2/s > 2$.*

By Theorem 5.1, in order to prove lower bounds on $\omega_u(T)$, it suffices to prove upper bounds on $\tilde{S}(T)$! In Section 5.3, we will give a number of new tools for doing so.

5.3 Combinatorial Tools for Asymptotic Slice Rank Upper Bounds

We now give three general tools for proving upper bounds on $\tilde{S}(T)$ for many tensors T. Each of our tools applies to a large class of tensors, but we will see in particular that all three of them apply to the Coppersmith-Winograd tensor CW_q.

The general idea for the three tools is to find partitions $T = A + B$ of our tensors, such that at least one of $\tilde{S}(A)$ and $\tilde{S}(B)$ is low, and use this to show that $\tilde{I}(T)$ is itself low. If $\tilde{S}$ were subadditive, i.e. if it were the case that $\tilde{S}(T) \leq \tilde{S}(A) + \tilde{S}(B)$ when $T = A + B$, then this would be relatively straightforward. Unfortunately, $\tilde{S}$ is not subadditive in general, and even in many natural situations:

Example 5.1. *Let q be any positive integer, and define the tensors $T_1 := \sum_{i=0}^{q} x_0 y_i z_i$, $T_2 := \sum_{i=1}^{q+1} x_i y_0 z_i$, and $T_3 := \sum_{i=1}^{q+1} x_i y_i z_{q+1}$. We can see that T_1 has only one x-variable, T_2 has only one y-variable, and T_3 has only one z-variable, and so $\tilde{S}(T_1) = \tilde{S}(T_2) = \tilde{S}(T_3) = 1$. However, $T_1 + T_2 + T_3 = CW_q$, so the three tensors give a partition of the Coppersmith-Winograd tensor! Hence, for instance, using the fact that $\omega_u(CW_5) \leq 2.373$, we see that $\tilde{S}(CW_5) \geq \tilde{R}(CW_5)^{2/\omega_u(CW_5)} \geq 7^{2/2.373} \geq 5.15$.*

Throughout this section, we will nonetheless describe a number of general situations where, if T can be written as $T = A + B$, then bounds on $\tilde{S}(A)$ and $\tilde{S}(B)$ are sufficient to give bounds on $\tilde{I}(T)$.

5.3.1 Bounds from Variable-Deficient Partitions

We know that tensors T without many of one type of variable have small $\tilde{S}(T)$. For instance, if T is over X, Y, Z, and $|X|$ is 'small', then $\tilde{S}(T) \leq |X|$ is also small. We begin by showing that if T can be written as a sum of a few tensors, each of which does not have many of one type of variable, then we can still prove an upper bound on $\tilde{S}(T)$.

If X, Y, Z are minimal for T, then the *measure* of T, denoted $\mu(T)$, is given by $\mu(T) := |X| \cdot |Y| \cdot |Z|$. We state two simple facts about μ:

Fact 5.1. *For tensors A and B,*

- *$\mu(A \otimes B) = \mu(A) \cdot \mu(B)$, and*
- *if A is minimal over X, Y, Z, then $S(A) \leq \min\{|X|, |Y|, |Z|\} \leq \mu(A)^{1/3}$.*

Theorem 5.2. *Suppose T is a tensor, and $T_1, \ldots, T_k$ are tensors with $T = T_1 + \cdots + T_k$. Then, $\tilde{S}(T) \leq \sum_{i=1}^{k} (\mu(T_i))^{1/3}$.*

Proof. Note that

$$T^{\otimes n} = \sum_{(P_1,\ldots,P_n) \in \{T_1,\ldots,T_k\}^n} P_1 \otimes \cdots \otimes P_n.$$

It follows that

$$\begin{aligned} S(T^{\otimes n}) &\leq \sum_{(P_1,\ldots,P_n) \in \{T_1,\ldots,T_k\}^n} S(P_1 \otimes \cdots \otimes P_n) \\ &\leq \sum_{(P_1,\ldots,P_n) \in \{T_1,\ldots,T_k\}^n} \mu(P_1 \otimes \cdots \otimes P_n)^{1/3} \\ &= \sum_{(P_1,\ldots,P_n) \in \{T_1,\ldots,T_k\}^n} (\mu(P_1) \cdot \mu(P_2) \cdots \mu(P_n))^{1/3} \\ &= (\mu(T_1)^{1/3} + \cdots + \mu(T_k)^{1/3})^n, \end{aligned}$$

which implies as desired that $\tilde{S}(T) \leq (\mu(T_1)^{1/3} + \cdots + \mu(T_k)^{1/3})$. □

5.3.2 Bounds from Block Partitions

This tool will be the most important in upper bounding the asymptotic slice rank of many tensors of interest. We show that a partitioning method similar to the Laser Method applied to a tensor T can be used to prove upper bounds on $\tilde{S}(T)$. We begin by introducing some notation related to partitioning tensors into blocks, similar to the notation used in Subsection 4.6.1 when describing the Laser Method.

Partition Notation Throughout this subsection, we will be partitioning the terms of tensors into blocks defined by partitions of the three variable sets. Here we introduce some notation for some properties of such partitions; these definitions all depend on the particular partition of the variables being used, which will be clear from context.

Suppose T is a tensor minimal over X, Y, Z, and let $X = X_1 \cup \cdots \cup X_{k_X}$, $Y = Y_1 \cup \cdots \cup Y_{k_Y}$, $Z = Z_1 \cup \cdots \cup Z_{k_Z}$ be partitions of the three variable sets. For $(i, j, k) \in [k_X] \times [k_Y] \times [k_Z]$, let T_{ijk} be T restricted to X_i, Y_j, Z_k (i.e. T with $X \setminus X_i$, $Y \setminus Y_j$, and $Z \setminus Z_k$ zeroed out); T_{ijk} is called a *block* of T. Let $L = \{T_{ijk} \mid (i, j, k) \in [k_X] \times [k_Y] \times [k_Z], T_{ijk} \neq 0\}$ be the set of non-zero blocks. For $i \in [k_X]$ let $L_{X_i} = \{T_{ij'k'} \in L \mid (j', k') \in [k_Y] \times [k_Z]\}$ be the set of blocks involving X_i, and define similarly L_{Y_j} and L_{Z_k}.

We will be particularly interested in probability distributions $p : L \to [0, 1]$. Let $P(L)$ be the set of such distributions. For such a $p \in P(L)$, and for $i \in [k_X]$, let $p(X_i) := \sum_{T_{ijk} \in L_{X_i}} p(T_{ijk})$, and similarly $p(Y_j)$ and $p(Z_k)$. Then, define $p_X \in \mathbb{R}$ by

$$p_X := \prod_{i \in [k_X]} \left(\frac{|X_i|}{p(X_i)} \right)^{p(X_i)},$$

and p_Y and p_Z similarly. We can equivalently write $p_X = 2^{H(p(X))}$, where $H(p(X)) = \sum_{i \in [k_X]} -p(X_i) \log p(X_i)$ is the *entropy* of the marginal distribution of p on the parts of X. This expression, which arises naturally in the Laser Method, will play an important role in our upper bounds and lower bounds on $\tilde{S}$.

The Main Tool We now present the main tool for bounding $\tilde{S}$ in terms of the quantities we just defined.

Theorem 5.3. *For any tensor T and partition of its variable sets,*

$$\tilde{S}(T) \leq \limsup_{p \in P(L)} \min\{p_X, p_Y, p_Z\}.$$

Proof. For any positive integer n, we can write

$$T^{\otimes n} = \sum_{(P_1, \ldots, P_n) \in L^n} P_1 \otimes \cdots \otimes P_n.$$

For a given $(P_1, \ldots, P_n) \in L^n$, let $dist(P_1, \ldots, P_n)$ be the probability distribution on L which results from picking a uniformly random $\alpha \in [n]$ and outputting P_α. For a probability distribution $p : L \to [0, 1]$, define $L_{n,p} := \{(P_1, \ldots, P_n) \in L^n \mid dist(P_1, \ldots, P_n) = p\}$. Note that the number of p for which $L_{n,p}$ is nonempty is only poly(n), since they are the distributions which assign an integer multiple of $1/n$ to each element of L. Let D be the set of these probability distributions.

We can now rearrange:

$$T^{\otimes n} = \sum_{p \in D} \sum_{(P_1, \ldots, P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n.$$

Hence,

$$\begin{aligned} S(T^{\otimes n}) &\leq \sum_{p \in D} S\left(\sum_{(P_1, \ldots, P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n \right) \\ &\leq \text{poly}(n) \cdot \max_{p \in D} S\left(\sum_{(P_1, \ldots, P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n \right). \end{aligned}$$

For any probability distribution $p : L \to [0, 1]$, let us count the number of x-variables used in $\left(\sum_{(P_1, \ldots, P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n\right)$. These are the tuples of the form $(x_1, \ldots, x_n) \in X^n$ where, for each $i \in [k_X]$, there are exactly $n \cdot p(X_i)$ choices of j for which $x_j \in X_i$. The number of these is[1]

$$\binom{n}{n \cdot p(X_1), n \cdot p(X_2), \ldots, n \cdot p(X_{k_X})} \cdot \prod_{i \in [k_X]} |X_i|^{n \cdot p(X_i)}.$$

This is upper bounded by $p_X^{n+o(n)}$. It follows that $S_x \left(\sum_{(P_1,\ldots,P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n \right) \leq p_X^{n+o(n)}$. We can similarly argue about S_y and S_z. Hence,

$$\begin{aligned} S(T^{\otimes n}) &\leq \mathrm{poly}(n) \cdot \max_{p \in D} S \left(\sum_{(P_1,\ldots,P_n) \in L_{n,p}} P_1 \otimes \cdots \otimes P_n \right) \\ &\leq \mathrm{poly}(n) \cdot \max_{p \in D} \min\{p_X, p_Y, p_Z\}^{n+o(n)} \\ &\leq \mathrm{poly}(n) \cdot \limsup_{p \in P(L)} \min\{p_X, p_Y, p_Z\}^{n+o(n)}. \end{aligned}$$

Hence, $S(T^{\otimes n}) \leq \limsup_p \min\{p_X, p_Y, p_Z\}^{n+o(n)}$, and the desired result follows. □

We make one remark about the quantity p_X in Theorem 5.3 before continuing. Suppose T is over X, Y, Z with $|X| = |Y| = |Z| = q$. For any probability distribution p we always have $p_X, p_Y, p_Z \leq q$, and moreover we only have $p_X = q$ when $p(X_i) = |X_i|/q$ for each i. It follows that if no probability distribution p is δ-close (say, in ℓ_1 distance) to having $p(X_i) = |X_i|/q$ for all i, $p(Y_j) = |Y_j|/q$ for all j, and $p(Z_k) = |Z_k|/q$ for all k, simultaneously, then we get $\tilde{S}(T) \leq q^{1-f(\delta)}$ for some increasing function f with $f(\delta) > 0$ for all $\delta > 0$. This gives a simple test for whether Theorem 5.3 can give a nontrivial upper bound on $\tilde{S}(T)$ for a tensor T.

Symmetric Block Partitions We make a remark about applying Theorem 5.3 to variable-symmetric tensors. This remark has implicitly been used in past work on applying the Laser Method, such as [CW90], but we prove it here for completeness. Recall the notation in Section 4.5 about variable-symmetric tensors.

If T is a variable-symmetric tensor minimal over X, Y, Z, then partitions $X = X_1 \cup \cdots \cup X_{k_X}$, $Y = Y_1 \cup \cdots \cup Y_{k_Y}$, $Z = Z_1 \cup \cdots \cup Z_{k_Z}$ of the variable sets are called *T-symmetric* if (using the partition notation above) $k_X = k_Y = k_Z$, $|X_i| = |Y_i| = |Z_i|$ for all $i \in [k_X]$, and the block $T_{jki} = rot(T_{ijk})$ for all $(i,j,k) \in [k_X]^3$. For the L resulting from such a T-symmetric partition, a probability distribution $p \in P(L)$ is called *T-symmetric* if it satisfies $p(T_{ijk}) = p(T_{jki})$ for all $(i,j,k) \in [k_X]^3$, and we write $P^{sym}(L) \subseteq P(L)$ for the set of such T-symmetric distributions. Notice in particular that any $p \in P^{sym}(L)$ satisfies $p_X = p_Y = p_Z$.

Proposition 5.4. *Suppose T is a variable-symmetric tensor over X, Y, Z, and $X = X_1 \cup \cdots \cup X_{k_X}$, $Y = Y_1 \cup \cdots \cup Y_{k_Y}$, $Z = Z_1 \cup \cdots \cup Z_{k_Z}$ are T-symmetric partitions.*

[1] Recall from Proposition 2.5 that, for fixed p_is, we have $\binom{n}{p_1 n, p_2 n, \ldots, p_\ell n} \leq \left(\prod_i p_i^{-p_i}\right)^{n+o(n)}$. Throughout this dissertation we use the convention that $p_i^{p_i} = 1$ when $p_i = 0$.

Then,

$$\tilde{S}(T) \leq \limsup_{p \in P^{sym}(L)} p_X.$$

Proof. We know from Theorem 5.3 that $\tilde{S}(T) \leq \limsup_{p \in P(L)} \min\{p_X, p_Y, p_Z\}$. We will show that for any $p \in P(L)$, there is a $p' \in P^{sym}(L)$ such that $\min\{p_X, p_Y, p_Z\} \leq \min\{p'_X, p'_Y, p'_Z\}$, which means that in fact, $\tilde{S}(T) \leq \limsup_{p \in P^{sym}(L)} \min\{p_X, p_Y, p_Z\}$. Finally, the desired result will follow since, for any $p' \in P^{sym}(L)$, we have $p'_X = p'_Y = p'_Z$.

Consider any $p \in P(L)$, and define the distribution $p' \in P^{sym}(L)$ by $p'(T_{ijk}) := (p(T_{ijk}) + p(T_{jki}) + p(T_{kij}))/3$ for each $T_{ijk} \in L$. In order to show that $\min\{p_X, p_Y, p_Z\} \leq p'_X$, we will show that $(p_X p_Y p_Z)^{1/3} \leq p'_X$:

$$\begin{aligned}
(p_X p_Y p_Z)^{1/3} &= \prod_{i \in [k_X]} \left(\frac{|X_i|}{p(X_i)}\right)^{p(X_i)/3} \left(\frac{|Y_i|}{p(Y_i)}\right)^{p(Y_i)/3} \left(\frac{|Z_i|}{p(Z_i)}\right)^{p(Z_i)/3} \\
&= \prod_{i \in [k_X]} \frac{|X_i|^{p'(X_i)}}{(p(X_i)^{p(X_i)} p(Y_i)^{p(Y_i)} p(Z_i)^{p(Z_i)})^{1/3}} \\
&\leq \prod_{i \in [k_X]} \frac{|X_i|^{p'(X_i)}}{p'(X_i)^{p'(X_i)}} \\
&= p'_X,
\end{aligned}$$

where the second-to-last step follows from the fact that for any real numbers $a, b, c \in [0, 1]$, setting $d = (a + b + c)/3$, we have $a^a b^b c^c \geq d^{3d}$. □

Proposition 5.4 will help simplify calculations when we apply Theorem 5.3 to variable-symmetric tensors later in this Chapter.

5.3.3 Bounds from Parts with Low X-Rank

Finally we give our third tool. This tool will be the least important in proving bounds on $\tilde{S}$ below, but in general it can help to extend upper bounds on $\tilde{S}$ from tensors B for which upper bounds on $\tilde{S}(B)$ are known to tensors T which are slight modifications of B.

For a tensor T, let $m(T) := \max\{S_x(T), S_y(T), S_z(T)\}$. Recall from Lemma 5.1 that for any two tensors A, B we have $S(A \otimes B) \leq S(A) \cdot m(B)$.

In general, for two tensors A and B, even if $\tilde{S}(A)$ and $\tilde{S}(B)$ are 'small', it might still be the case that $\tilde{S}(A + B)$ is 'large', much larger than $\tilde{S}(A) + \tilde{S}(B)$. Here we show that if, not only is $\tilde{S}(A)$ small, but even $S_x(A)$ is small, then we can get a decent bound on $\tilde{S}(A + B)$.

Theorem 5.4. *Suppose T, A, B are tensors such that $A + B = T$. Then,*

$$\tilde{S}(T) \leq \left(\frac{m(A)}{(1-p) \cdot S_x(A)}\right)^{1-p} \cdot \frac{1}{p^p},$$

where $p \in [0,1]$ is given by

$$p := \frac{\log\left(\frac{S_x(B)}{\tilde{S}(B)}\right)}{\log\left(\frac{m(A)}{S_x(A)}\right) + \log\left(\frac{S_x(B)}{\tilde{S}(B)}\right)}.$$

Proof Sketch. We begin by, for any integers $n \geq k \geq 0$, giving bounds on $S(A^{\otimes k} \otimes B^{\otimes(n-k)})$. First, since S_x is submultiplicative, we have

$$S(A^{\otimes k} \otimes B^{\otimes(n-k)}) \leq S_x(A^{\otimes k} \otimes B^{\otimes(n-k)}) \leq S_x(A)^k \cdot S_x(B)^{n-k}.$$

Second, from the definition of m, we have

$$S(A^{\otimes k} \otimes B^{\otimes(n-k)}) \leq m(A^{\otimes k}) \cdot S(B^{\otimes(n-k)}) \leq m(A)^k \cdot \tilde{S}(B)^{n-k}.$$

It follows that for any positive integer n we have

$$\begin{aligned} S(T^{\otimes n}) &\leq \sum_{k=0}^{n} \binom{n}{k} \cdot S(A^{\otimes k} \otimes B^{\otimes(n-k)}) \\ &\leq \sum_{k=0}^{n} \binom{n}{k} \cdot \min\{S_x(A)^k \cdot S_x(B)^{n-k}, m(A)^k \cdot \tilde{S}(B)^{n-k}\}. \end{aligned}$$

We can see that the quantity $\binom{n}{k} \cdot \min\{S_x(A)^k \cdot S_x(B)^{n-k}, m(A)^k \cdot \tilde{S}(B)^{n-k}\}$ is maximized at $k = pn$, and the result follows. □

5.4 Slice Rank Lower Bounds via the Laser Method

In the previous section, we gave three general tools for proving upper bounds on $\tilde{S}(T)$. Before applying them to particular tensors of interest, we begin in this section by giving a general tool for proving *lower bounds* on $\tilde{S}(T)$. Our main tool for proving lower bounds will be the *Laser Method*, the same technique we described in Subsection 4.6.1 which Strassen, Coppersmith, and Winograd developed for proving upper bounds on ω. The Laser Method only applies to tensors T with certain block structure, but we will show that when it applies to T, then not only does it give a lower bound on $\tilde{S}(T)$, but that the resulting bound *matches* the upper bound on $\tilde{S}(T)$ from one of our upper bounding tools, Theorem 5.3.

Consider any tensor T which is minimal over X, Y, Z, and let $X = X_1 \cup \cdots \cup X_{k_X}$, $Y = Y_1 \cup \cdots \cup Y_{k_Y}$, $Z = Z_1 \cup \cdots \cup Z_{k_Z}$ be partitions of the three variable sets. Define T_{ijk}, L, and p_X for a probability distribution p on L, as in the top of Subsection 5.3.2. Recall in particular that T_{ijk} is T restricted to the variable sets X_i, Y_j, and Z_k.

Definition 5.1. *We say that T, along with partitions of X, Y, Z, is a* laser-ready tensor partition *if the following three conditions are satisfied:*

(1) For every $(i,j,k) \in [k_X] \times [k_Y] \times [k_Z]$, either $T_{ijk} = 0$, or else T_{ijk} has a degener-

ation to a tensor $\langle a, b, c\rangle$ with $ab = |X_i|$, $bc = |Y_j|$, and $ca = |Z_k|$ (i.e. a matrix multiplication tensor which is as big as possible given $|X_i|$, $|Y_j|$, and $|Z_k|$).

(2) There is an integer α such that $T_{ijk} \neq 0$ only if $i + j + k = \alpha$.

(3) T is variable-symmetric, and the partitions are T-symmetric.

These conditions match those discussed in Subsection 4.6.1, and are exactly those required for the original Laser Method used by Coppersmith and Winograd [CW90] applies to T. We note that condition (3) is a simplifying assumption rather than a real condition on T: similar to the discussion in Section 4.5, for any tensor T and partitions satisfying conditions (1) and (2), the tensor $sym(T)$ along with the corresponding partitions on its variables (which arise from taking the products of the partitions of the variables of T), satisfies all three conditions, gives at least as good a bound on ω using the Laser Method as T and the original partitions, and more generally has $\omega_u(sym(T)) \leq \omega_u(T)$.

The precise way in which the Laser Method applies to a laser-ready tensor partition is as follows.

Theorem 5.5 ([CW90, DS13, Wil12])**.** *Suppose T, along with the partitions of X, Y, Z, is a laser-ready tensor partition. Then, for any distribution $p \in P^{sym}(L)$, and any positive integer n, the tensor $T^{\otimes n}$ has a degeneration into*

$$\left(\prod_{i \in [k_X]} p(X_i)^{-p(X_i)} \right)^{n - o(n)} \odot \langle a, a, a\rangle, \tag{5.1}$$

where

$$a = \left(\prod_{T_{ijk} \in L} |X_i|^{p(T_{ijk})} \right)^{n/2 - o(n)}. \tag{5.2}$$

Proof. This proof is relatively detailed and technical; a reader willing to take the Theorem statement for granted may wish to skip reading it and instead read the overview in Subsection 4.6.1.

We begin by assuming that $p(T_{ijk})$ is an integer multiple of $1/n$ for all $i, j, k \in [k_X]$. If this is not the case, it can be achieved by slightly modifying p to a new probability distribution which changes $p(T_{ijk})$ by at most $1/n$ for each i, j, k. This in particular changes $p(X_i)$ by at most k_X/n for all $i \in [k_X]$, and so the changes in the quantities (5.1) and (5.2) are subsumed by the '$o(n)$'s in the exponents.

We now proceed to describe the degeneration. We will zero out variables in three phases, leaving a tensor which easily degenerates to the desired tensor after the third phase. We will use the partition notation from Subsection 5.3.2 and Theorem 5.3.

Phase One. We say a variable $x = (x_1, \ldots, x_n) \in X^n$ is *p-satisfying* if, for all $i \in [k_X]$, the number of $j \in [n]$ such that $x_j \in X_i$ is $n \cdot p(X_i)$, and similarly for a

$y \in Y^n$ or a $z \in Z^n$. In phase one, we zero out all $x \in X^n$, $y \in Y^n$, and $z \in Z^n$ which are *not* p-satisfying.

Call an $I \in \{X_1, \ldots, X_{k_X}\}^n$ (or similar for y and z variables) a *block* of variables. Notice that that for each block of variables I, either all its elements are p-satisfying, or none are; call I p-satisfying if all its elements are. The number of p-satisfying blocks of variables is

$$C := \binom{n}{n \cdot p(X_1), n \cdot p(X_2), \ldots, n \cdot p(X_{k_X})} = \left(\prod_{i \in [k_X]} p(X_i)^{-p(X_i)} \right)^{n - o(n)},$$

which is the left-hand quantity in (5.1). For $I = (X_{i_1}, \ldots, X_{i_n}) \in \{X_1, \ldots, X_{k_X}\}^n$, $J = (Y_{j_1}, \ldots, Y_{j_n}) \in \{Y_1, \ldots, Y_{k_X}\}^n$, and $K = (Z_{k_1}, \ldots, Z_{k_n}) \in \{Z_1, \ldots, Z_{k_X}\}^n$, if I, J, K are p-satisfying, and are such that $T_{I_\ell j_\ell k_\ell} \in L$ for all $\ell \in [n]$, then we say (I, J, K) is a *surviving triple*. In particular, if (I, J, K) is a surviving triple, then $T_{IJK} := \bigotimes_{\ell=1}^{n} T_{i_\ell j_\ell k_\ell}$ remains after phase one of zeroing outs. From condition (1) in the definition of a laser-ready tensor partition, we have that $T_{IJK} \trianglerighteq \langle a, a, a \rangle$. Thus, if it were the case that, for each p-satisfying I, there were exactly one surviving triple involving I, and similarly for each p-satisfying J and K, then the result of phase one would be a desired tensor!

This is unfortunately not yet the case, but in the remaining steps we will zero out more blocks of variables so that this will become the case. Let A be the number of surviving triples, and B be the number of surviving triples involving a fixed p-satisfying block I. Note by symmetry that B is independent of which I we pick, and whether I is a block of x, y, or z variables, and moreover that $B = A/C$. Although we could count A and B exactly using multinomial coefficients, it turns out we will only need the simple bound $B \leq 2^{O(n)}$.

Phase Two. Set $M = 4B + 1$. In this phase, we will use a result of Salem and Spencer [SS42]: There is a subset $H \subseteq [M]$ of size $|H| \geq M^{1-o(1)} \geq M/2^{o(n)}$ which does not contain any nontrivial three-term arithmetic progressions mod M; in other words, if $a, b, c \in H$ such that $a + b = 2c \pmod{M}$, then $a = b = c$. Let $M' := |H|$.

Recall from condition (2) in the definition of a laser-ready tensor partition that there is a $\alpha \in \mathbb{N}$ such that $T_{ijk} \in L$ only if $i + j + k = \alpha$. In particular, if (I, J, K) is a surviving triple, then (using the notation above) $i_\ell + j + \ell + k_\ell = \alpha$ for all $\ell \in [n]$.

Pick independently and uniformly random $w_0, w_1, \ldots, w_n \in \mathbb{Z}_M$, and using them define three hash functions (which map variable blocks to integers mod M) $h_X : \{X_1, \ldots, X_{k_X}\}^n \to \mathbb{Z}_M$, $h_Y : \{Y_1, \ldots, Y_{k_X}\}^n \to \mathbb{Z}_M$, $h_Z : \{Z_1, \ldots, Z_{k_X}\}^n \to \mathbb{Z}_M$ by:

$$h_X(X_{i_1}, \dots, X_{i_n}) := 2\sum_{\ell=1}^{n} w_\ell \cdot i_\ell \pmod{M},$$

$$h_Y(Y_{j_1}, \dots, Y_{j_n}) := 2w_0 + 2\sum_{\ell=1}^{n} w_\ell \cdot j_\ell \pmod{M},$$

$$h_Z(Z_{k_1}, \dots, Z_{k_n}) := w_0 + \sum_{\ell=1}^{n} w_\ell \cdot (\alpha - k_\ell) \pmod{M}.$$

Notice that in any surviving triple (I, J, K):

- $h_X(I) + h_Y(J) = 2h_Z(K) \pmod{M}$, and
- not only is $h_X(I)$ a uniformly random value in $[M]$, but it remains a uniformly random value even conditioned on the value $h_Y(J)$ (and similarly, and one of $h_X(I)$, $h_Y(J)$, and $h_Z(K)$ is uniformly random conditioned on any other one).

In phase two, we zero out all the x-variables in any block I such that $h_X(I) \notin H$, all the y-variables in any block J such that $h_Y(J) \notin H$, and all the z-variables in any block K such that $h_Z(K) \notin H$. It follows from the definition of H that any surviving triple (I, J, K) which also survives phase two satisfies $h_X(I) = h_Y(J) = h_Z(K)$. For each $\beta \in H$, let S_β be the set of surviving triples (I, J, K) with $h_X(I) = h_Y(J) = h_Z(K) = \beta$.

Phase Three. Now, in phase three, for every pair of distinct surviving triples (which survived phases one and two) (I, J, K) and (I, J', K') which share the block I, we will zero out all variables in the block I (and then similarly for y and z variables). Hence, for every surviving triple (I, J, K) which survives phase three, there are no other surviving triples which survive phase three which share any of I, J, or K. We will next show that there is a choice of the randomness above so that the number of surviving triples which survive phase three is $\geq C/n^{o(1)}$. This (along with the discussion earlier in phase one) will complete the proof.

We begin by computing some simple expected values. First, for a fixed $\beta \in H$, we compute the expected size of S_β. The probability that a given surviving triple (from phase one) (I, J, K) also survived phase two and is in S_β is M^{-2}. Indeed, each of $h_X(I)$ and $h_Y(J)$ equals β independently with probability M^{-1}, and then given those two events, it follows that $h_Z(K) = \beta$ as well. Hence, by linearity of expectation, the expected size of S_β is A/M^2.

Second, for fixed $\beta \in H$, we compute an upper bound on the expected number of unordered pairs of distinct surviving triples (from phase one) (I, J, K) and (I, J', K') which share the block I, and which are both in S_β. There are A choices for a surviving triple (I, J, K), then $B-1$ choices for the surviving triple (I, J', K'). Both triples will be in S_β if and only if $h_X(I) = h_Y(J) = h_Y(J') = \beta$. Since each of those three hash values is independent, this happens with probability M^{-3}. In all, an upper bound on the expected value is $\frac{1}{2}AB/M^3$ (the $\frac{1}{2}$ is because we are counting the same pair twice). It follows that the expected number of surviving triples which are zeroed out in phase three is at most $3AB/M^3$ (since each surviving triple which is zeroed out is in at least one pair which share a block, and we need to count each of the three types of blocks).

From the calculations above, we know that for a fixed β, the expected number of surviving triples in S_β which survive past phase three is at least $A/M^2 - 3AB/M^3 \geq \frac{1}{4}A/M^2$. Hence, summing over all $\beta \in H$, the expected number of surviving triples which survive past phase three is at least $\frac{1}{4}M'A/M^2 \geq A/(M \cdot n^{o(1)}) \geq A/(B \cdot n^{o(1)}) = C/n^{o(1)}$. Hence, there is a choice of the randomness which achieves at least this many, as desired! □

Applying the Laser Method, we get the following key new result about the asymptotic slice rank of laser-ready tensor partitions.

Theorem 5.6. *Suppose tensor T, along with the partitions of X, Y, Z, is a laser-ready tensor partition. Then,*

$$\tilde{S}(T) = \limsup_{p \in P^{sym}(L)} p_X.$$

Proof. The upper bound, $\tilde{S}(T) \leq \limsup_{p \in P^{sym}(L)} p_X$, is given by Proposition 5.4.

For the lower bound, we know from Theorem 5.5 that for all $p \in P^{sym}(L)$, and all positive integers n, the tensor $T^{\otimes n}$ has a degeneration into

$$\left(\prod_{i \in [k_X]} p(X_i)^{-p(X_i)} \right)^{n-o(n)} \odot \langle a, a, a \rangle,$$

where

$$a = \left(\prod_{T_{ijk} \in L} |X_i|^{p(T_{ijk})} \right)^{n/2-o(n)}.$$

By Proposition 5.3, this means $T^{\otimes n}$ has a degeneration to an independent tensor of size

$$\left(\prod_{i \in [k_X]} p(X_i)^{-p(X_i)} \right)^{n-o(n)} \cdot a^2 = p_X^{n-o(n)}.$$

Applying Propositions 5.1 and 5.2 implies that $\tilde{S}(T) \geq p_X$ for all $p \in P^{sym}(L)$, as desired. □

Corollary 5.3. *If T is a tensor with a laser-ready tensor partition, and applying the Laser Method to T with this partition yields an upper bound on ω of $\omega_u(T) \leq c$ for some $c > 2$, then $\omega_u(T) > 2$.*

Proof. When the Laser Method for a given $p \in P^{sym}(L)$ shows, as in Theorem 5.5, that $T^{\otimes n}$ has a degeneration into

$$\left(\prod_{i\in[k_X]} p(X_i)^{-p(X_i)}\right)^{n-o(n)} \odot \langle a,a,a\rangle,$$

the resulting upper bound on $\omega_u(T)$ is that

$$\left(\prod_{i\in[k_X]} p(X_i)^{-p(X_i)}\right)^{n-o(n)} \cdot a^{\omega_u(T)} \geq \tilde{R}(T)^n.$$

In particular, since the left-hand side equals p_X when $\omega_u(T) = 2$, this yields $\omega_u(T) = 2$ if and only if, for every $\varepsilon > 0$, there is a $p \in P^{sym}(L)$ such that $p_X \geq \tilde{R}(T)^{1-\varepsilon}$. In particular, if it does *not* yield $\omega_u(T) = 2$, then there is a $\delta > 0$ such that all $p \in P^{sym}(L)$ have $p_X \leq \tilde{R}(T)^{1-\delta}$. It follows from Theorem 5.6 that $\tilde{S}(T) \leq \tilde{R}(T)^{1-\delta}$. Combined with Theorem 5.1, this means that $\omega_u(T) \geq 2/(1-\delta) > 2$. □

5.4.1 Slice Rank Versus Asymptotic Subrank

For a tensor T, let $Q'(T)$ denote the largest integer q such that there is a degeneration $T \trianglerighteq \langle q\rangle$. The *asymptotic subrank* of T is defined as $\tilde{Q}(T) := \limsup_{n\in\mathbb{N}} Q'(T^{\otimes n})^{1/n}$. Asymptotic Subrank is an important notion in Strassen's theory of the Asymptotic Spectrum of Tensors [Str86, Str91]. It can be thought of as 'dual' to asymptotic rank: while $\tilde{R}(T)$ measures the 'cost' of T to convert from an independent tensor, $\tilde{Q}(T)$ is a measure of the 'value' of T in converting back to an independent tensor.

Propositions 5.1 and 5.2 above imply that $\tilde{Q}(T) \leq \tilde{S}(T)$ for all tensors T. Similarly, it is not hard to see that Theorem 5.1, our general bound on $\omega_u(T)$, holds with $\tilde{S}$ replaced by $\tilde{Q}$. One could thus conceivably hope to prove stronger lower bounds than those which we will prove in the next section by bounding $\tilde{Q}$ instead of $\tilde{S}$.

However, one interesting corollary of Theorem 5.6 above is that $\tilde{Q}(T) = \tilde{S}(T)$ for every laser-ready tensor T. In particular, every tensor T we study in the next section will be laser-ready, so so such an improvement on our lower bounds on $\omega_u(T)$ using $\tilde{Q}(T)$ is impossible for these tensors T

More generally, there are currently no known tensors T for which the best known upper bound on $\tilde{Q}(T)$ is smaller than the best known upper bound on $\tilde{S}(T)$ (including the new bounds of [CVZ18, CVZ19]). Hence, novel tools for upper bounding $\tilde{Q}$ would be required for such an approach to proving better lower bounds on ω_u.

Corollary 5.4. *Every tensor T with a laser-ready tensor partition has $\tilde{S}(T) = \tilde{Q}(T)$.*

Proof. All tensors satisfy $\tilde{S}(T) \geq \tilde{Q}(T)$. In Theorem 5.6, the upper bound on $\tilde{S}(T)$ showed that $T^{\otimes n}$ has a degeneration to an independent tensor of size $\tilde{S}(T)^{n-o(n)}$, which implies that $\tilde{Q}(T) \geq \tilde{S}(T)$. □

5.5 Computing the Slice Ranks for Tensors of Interest

In this section, we give slice rank upper bounds for a number of tensors of interest. It follows from Theorem 5.6 above that *all of the bounds we prove in this Section are tight*.

5.5.1 Generalized Coppersmith-Winograd Tensors

We begin with the generalized CW tensors defined in Subsection 4.6.1 above, which for a positive integer q and a permutation $\sigma \in S_q$ are given by

$$CW_{q,\sigma} := x_0y_0z_{q+1} + x_0y_{q+1}z_0 + x_{q+1}y_0z_0 + \sum_{i=1}^{q}(x_iy_{\sigma(i)}z_0 + x_iy_0z_i + x_0y_iz_i).$$

The usual Coppersmith-Winograd tensor CW_q results by picking σ to be the identity permutation. We can see that Theorems 5.2, 5.3, and 5.4 can all apply to $CW_{q,\sigma}$ to prove nontrivial upper bounds on $\tilde{S}(T)$.

That said, we will now use Theorem 5.3 to prove a *tight* bound on $\tilde{S}(CW_{q,\sigma})$. Our bound will imply that $\omega_u(CW_{q,\sigma}) \geq 2.16805$ for all $q \in \mathbb{N}$ and all $\sigma \in S_q$. Because of Theorem 5.6, no better lower bound on $\omega_u(CW_{q,\sigma})$ is possible by arguing about $\tilde{S}(CW_{q,\sigma})$ or even $\tilde{Q}(CW_{q,\sigma})$.

We begin by partitioning the variable sets of $CW_{q,\sigma}$, using the notation of Theorem 5.3. Let $X_0 = \{x_0\}$, $X_1 = \{x_1, \ldots, x_q\}$, and $X_2 = \{x_{q+1}\}$, so that $X_0 \cup X_1 \cup X_2$ is a partition of the x-variables of $CW_{q,\sigma}$.[2] Similarly, let $Y_0 = \{y_0\}$, $Y_1 = \{y_1, \ldots, y_q\}$, $Y_2 = \{y_{q+1}\}$, $Z_0 = \{z_0\}$, $Z_1 = \{z_1, \ldots, z_q\}$, and $Z_2 = \{z_{q+1}\}$. We can see this is a $CW_{q,\sigma}$-symmetric partition with $L = \{T_{002}, T_{020}, T_{200}, T_{011}, T_{101}, T_{110}\}$.

Consider any probability distribution $p \in P^{sym}(L)$. By symmetry, we know that $p(T_{002}) = p(T_{020}) = p(T_{200}) = v$ and $p(T_{011}) = p(T_{101}) = p(T_{110}) = 1/3 - v$ for some value $v \in [0, 1/3]$. Applying Theorem 5.3, and in particular Proposition 5.4, combined with Theorem 5.6, yields:

$$\tilde{S}(CW_{q,\sigma}) = \sup_{v \in [0,1/3]} \frac{q^{2(1/3-v)}}{v^v(2/3-2v)^{2/3-2v}(1/3+v)^{1/3+v}}.$$

The values for the first few q can be computed using optimization software as follows:

[2] The sets of partitions were 1-indexed before, but we 0-index here for notational consistency with past work.

q	$\tilde{S}(CW_{q,\sigma})$
1	$2.7551\cdots$
2	$3.57165\cdots$
3	$4.34413\cdots$
4	$5.07744\cdots$
5	$5.77629\cdots$
6	$6.44493\cdots$
7	$7.08706\cdots$
8	$7.70581\cdots$

Finally, using the lower bound $\tilde{R}(CW_{q,\sigma}) \geq q+2$ (in fact, it is known that $\tilde{R}(CW_{q,\sigma}) = q+2$), and the upper bound on $\tilde{S}(CW_{q,\sigma})$ we just proved, we can apply Theorem 5.1 to give lower bounds $\omega_u(CW_{q,\sigma}) \geq 2\log(\tilde{R}(CW_{q,\sigma}))/\log(\tilde{S}(CW_{q,\sigma})) \geq 2\log(q+2)/\log(\tilde{S}(CW_{q,\sigma}))$ as follows:

q	Lower Bound on $\omega_u(CW_{q,\sigma})$
1	$2.16805\cdots$
2	$2.17794\cdots$
3	$2.19146\cdots$
4	$2.20550\cdots$
5	$2.21912\cdots$
6	$2.23200\cdots$
7	$2.24404\cdots$
8	$2.25525\cdots$

It seems clear numerically that the resulting lower bound on $\omega_u(CW_{q,\sigma})$ is increasing with q and is always at least $2.16805\ldots$; below we give a simple proof of this, concluding our main result about $CW_{q,\sigma}$.

Theorem 5.7. *$\omega_u(CW_{q,\sigma}) \geq 2.16805$ for all $q \in \mathbb{N}$ and $\sigma \in S_q$.*

Proof. Define the function $f : [0, 1/3] \to \mathbb{R}$ by

$$f(v) := \frac{1}{v^v(2/3-2v)^{2/3-2v}(1/3+v)^{1/3+v}}.$$

We already showed that

$$\omega_u(CW_{q,\sigma}) \geq \min_{v\in[0,1/3]} 2\frac{\log(q+2)}{\log(q^{2/3-2v}\cdot f(v))}.$$

Moreover, we saw above that $\omega_u(CW_{q,\sigma}) \geq 2.16805$ for all $q \leq 8$.

Let v_q denote the argmin for the optimization problem. In particular, for $q = 8$, the argmin is $v_8 = 0.017732422\ldots$. From the $q^{2/3-2v}$ term in the optimization problem, we see that $v_{q+1} \leq v_q$ for all q, and in particular, $v_q \leq v_8$ for all $q > 8$. It follows that $f(v_q) \leq f(v_8) = 2.07389\ldots$ for all $q > 8$. Thus, for all $q > 8$ we have:

$$\omega_u(CW_{q,\sigma}) \geq \min_{v\in[0,1/3]} 2\frac{\log(q+2)}{\log(q^{2/3-2v}\cdot f(v_8))} = 2\frac{\log(q+2)}{\log(q^{2/3}\cdot f(v_8))}.$$

This expression equals $2.18562\ldots$ at $q = 9$, and is easily seen to be increasing with q for $q > 9$, which implies as desired that $\omega_u(CW_{q,\sigma}) \geq 2.16805$ for all $q \geq 9$ and hence all q. □

5.5.2 Generalized Simple Coppersmith-Winograd Tensors

Similar to $CW_{q,\sigma}$, we can define for a positive integer q and a permutation $\sigma : [q] \to [q]$ the simple Coppersmith-Winograd tensor $cw_{q,\sigma}$ given by:

$$cw_{q,\sigma} := \sum_{i=1}^{q} (x_i y_{\sigma(i)} z_0 + x_i y_0 z_i + x_0 y_i z_i).$$

These tensors, when σ is the identity permutation, are well-studied in the literature on MM algorithms. For instance, Coppersmith and Winograd [CW90] showed that if $\tilde{R}(cw_{2,id}) = 2$ then $\omega = 2$.

We will again give a tight bound on $\tilde{S}(cw_{q,\sigma})$ using Theorem 5.3 combined Theorem 5.6. To apply Theorem 5.3, and in particular Proposition 5.4, we again pick a partition of the variables. Let $X_0 = \{x_0\}$, $X_1 = \{x_1, \ldots, x_q\}$, $Y_0 = \{y_0\}$, $Y_1 = \{y_1, \ldots, y_q\}$, $Z_0 = \{z_0\}$, and $Z_1 = \{z_1, \ldots, z_q\}$. This is a $cw_{q,\sigma}$-symmetric partition with $L = \{T_{011}, T_{101}, T_{110}\}$. There is a unique $p \in P^{sym}(L)$, which assigns probability $1/3$ to each part. It follows that

$$\tilde{S}(cw_{q,\sigma}) = (1/3)^{-1/3}(2/3)^{-2/3} \cdot q^{2/3} = \frac{3}{2^{2/3}} \cdot q^{2/3}.$$

Again, we will see in the next section that this bound is tight. Using the lower bound $\tilde{R}(cw_{q,\sigma}) \geq q + 1$ from ‘flattening’, we get the lower bound

$$\omega_u(cw_{q,\sigma}) \geq 2 \frac{\log(q+1)}{\log\left(\frac{3}{2^{2/3}} \cdot q^{2/3}\right)}.$$

The first few values are as follows; note that we cannot get a bound better than 2 when $q = 2$ because of Coppersmith and Winograd’s remark: if $\tilde{R}(cw_2) = 2$ then $\omega_u(cw_2) = 2$, but the best known bound is only $\tilde{R}(cw_2) \leq 3$.

q	Lower Bound on $\omega_u(cw_{q,\sigma})$
1	$2.17795\cdots$
2	2
3	$2.02538\cdots$
4	$2.06244\cdots$
5	$2.09627\cdots$
6	$2.12549\cdots$
7	$2.15064\cdots$

5.5.3 Cyclic Group Tensors

We next look at the group tensor T_q of the cyclic group C_q for $q \in \mathbb{N}$:

$$T_q = \sum_{i=0}^{q-1} \sum_{j=0}^{q-1} x_i y_j z_{i+j \bmod q}.$$

T_q is one of the first tensors which was studied in the Group-theoretic Method [CU03]. Define also the 'lower triangular' version of T_q, called T_q^{lower}, as:

$$T_q^{lower} = \sum_{i=0}^{q-1} \sum_{j=0}^{q-1-i} x_i y_j z_{i+j}.$$

It is known that $\tilde{R}(T_q) = \tilde{R}(T_q^{lower}) = q$, and a Coppersmith-Winograd-like analysis is possible to yield the best known bound $\omega_u(T_7), \omega_u(T_7^{lower}) \leq 2.373$ (see Section 5.6 below for a proof).

While Theorem 5.3 does not give any nontrivial upper bounds on $\tilde{S}(T_q)$, it does give nontrivial upper bounds on $\tilde{S}(T_q^{lower})$, by using the 'trivial' partition of the variables into parts of size 1. Using computer optimization software, we can compute $\tilde{S}(T_q^{lower})$, using Theorem 5.3 where each partition contains exactly one variable, combined with Theorem 5.6, for the first few values of q:

q	$\tilde{S}(T_q^{lower})$
2	$1.88988\cdots$
3	$2.75510\cdots$
4	$3.61071\cdots$
5	$4.46157\cdots$

We thus get the following lower bounds on $\omega_u(T_q^{lower}) \geq 2\log(q)/\log(\tilde{S}(T_q^{lower}))$:

q	Lower Bound on $\omega_u(T_q^{lower})$
2	$2.17795\cdots$
3	$2.16805\cdots$
4	$2.15949\cdots$
5	$2.15237\cdots$

These numbers match the lower bounds obtained by [AW18a, BCC$^+$17a] in their study of T_q; our Theorem 5.3 can be viewed as an alternate tool to achieve those lower bounds. The bound approaches 2 as $q \to \infty$, as it is known that $\log(S(T_q))/\log(q) = 1 - o(1)$ as $q \to \infty$ [BCC$^+$17a].

There is a simple monomial degeneration $T_q \trianglerighteq_{md} T_q^{lower}$, and it is shown in [CVZ18, Theorem 4.16] that there is a restriction $T_q^{lower} \geq T_q$ over the field $\mathbb{F}_q$, which implies that our bounds above also hold for T_q over $\mathbb{F}_q$.

5.5.4 Lower Triangular Tensors

More generally, we can give a strong characterization of lower triangular tensors T for which Theorem 5.1 can prove $\omega_u(T) > 2$.

Definition 5.2. *For $X = \{x_0, \ldots, x_{q-1}\}$, $Y = \{y_0, \ldots, y_{q-1}\}$ and $Z = \{z_0, \ldots, z_{q-1}\}$, a tensor T over X, Y, Z is* lower triangular *if*

- *For every $i, j \in \{0, \ldots, q-1\}$, there is at most one $k \in \{0, \ldots, q-1\}$ such that*[3] *$x_i y_j z_k \in T$, and*
- *For every $i, j \in \{0, \ldots, q-1\}$ with $i+j \geq q$, $x_i y_j z_k \notin T$ for any $k \in \{1, \ldots, q\}$.*

Terms $x_i y_j z_k$ with $i + j = q - 1$ are called diagonal terms*.*

Theorem 5.8. *For $X = \{x_0, \ldots, x_{q-1}\}$, $Y = \{y_0, \ldots, y_{q-1}\}$ and $Z = \{z_0, \ldots, z_{q-1}\}$, a lower triangular tensor T over X, Y, Z has $\tilde{S}(T) = q$ if and only if it has q diagonal terms, no two of which share a z-variables.*

Proof. First, consider any lower diagonal tensor T whose q diagonal terms do not share z-variables. Evidently $\tilde{S}(T) \leq |X| = q$. There is a simple monomial degeneration from T to the tensor consisting only of its diagonal terms, given by $m(x_i) = m(y_i) = \lambda^{q-i}$ and $m(z_i) = 1$ for all i. Since no two of the diagonal terms share z-variables, this is a monomial degeneration from T to an independent tensor of size q, which implies that $\tilde{S}(T) = q$.

Second, consider any lower diagonal tensor T with $\tilde{S}(T) = q$. Let $f : \{0, \ldots, q-1\}^2 \to \{0, \ldots, q-1\}$ be the map defining which z-variable appears in each term, i.e. such that $x_i y_j z_{f(i,j)}$ is the only term containing $x_i y_j$ for each i, j (we assume that such a term exists for each i, j; if T is missing any such terms, then the proof is even simpler). By Theorem 5.3 (with each part in the partitions of the variables having size 1), we know that for every $\kappa > 0$, there is a probability distribution $p : X \otimes Y \otimes Z \to [0, 1]$ whose support is on the terms of T, such that for any fixed i, $p(x_i) := \sum_{x_i y_j z_k} p(x_i y_j z_k) \geq 1/q - \kappa$, and similarly for $p(y_j)$ and $p(z_k)$. Summing this lower bound for all x-variables other than x_i also shows that $p(x_i) \leq 1/q + (q-1)\kappa$ for each i, and similarly for $p(y_j)$ and $p(z_k)$.

We now prove that for each $j \in \{0, \ldots, q-1\}$, we have $p(x_{q-1-j} y_j z_{f(q-1-j,j)}) \geq 1/q - O(\kappa)$, where we are thinking of q as a constant, so the O hides factors of q. We prove this by strong induction on j. For the base case, when $j = 0$, notice that the term $x_{q-1} y_0 z_{f(q-1,0)}$ is the only term containing x_{q-1}, and so $p(x_{q-1} y_0 z_{f(q-1,0)}) = p(x_{q-1}) \geq 1/q - \kappa$, as desired.

For the inductive step, note that for each $j' < j$, we have by assumption that $p(x_{q-1-j'} y_{j'} z_{f(q-1-j',j')}) \geq 1/q - O_q(\kappa)$. Therefore, for each such j',

[3] We write '$x_i y_j z_k \in T$' to denote that the coefficient of $x_i y_j z_k$ in T is nonzero.

$$\begin{aligned}
&p(x_{q-1-j}y_{j'}z_{f(i,j')}) \\
&\le \sum_{i=0}^{q-2-j'} p(x_i y_{j'} z_{f(i,j')}) \\
&= p(y_{j'}) - p(x_{q-1-j'}y_{j'}z_{f(q-1-j',j')}) \le (1/q + (q-1)\kappa) - (1/q - O(\kappa)) = O(\kappa).
\end{aligned}$$

It follows as desired that

$$\begin{aligned}
p(x_{q-1-j}y_j z_{f(q-1-j,j)}) &= p(x_{q-1-j}) - \sum_{j'=0}^{j-1} p(x_{q-1-j}y_{j'}z_{f(i,j')}) \\
&\ge p(x_{q-1-j}) - O(\kappa) \\
&\ge 1/q - O(\kappa).
\end{aligned}$$

Now, assume to the contrary that there is a k such that $k \neq f(q-1-j,j)$ for any j. Thus,

$$p(z_k) \le 1 - \sum_{j=0}^{q-1} p(x_{q-1-j}y_j z_{f(q-1-j,j)}) \le 1 - \sum_{j=0}^{q-1}(1/q - O(\kappa)) = O(\kappa).$$

Picking a sufficiently small $\kappa > 0$ contradicts Theorem 5.3. □

5.6 Upper Lower Bounds for Group Tensors

In the previous section, we mainly focused on applying our slice rank tools to various tensors which have been used in conjunction with the Laser Method to prove bounds on ω. In this section, we conclude the chapter by showing some implications for both upper and lower bounds on $\omega_u(T_G)$ when T_G is a group tensor of a finite group G.

5.6.1 Tri-Colored Sum-Free Sets

A number of recent works (eg. [BCC+17a, BCC+17b, AW18a]) have explored connections between lower bounds on matrix multiplication algorithms, and a notion from extremal combinatorics called a 'tri-colored sum-free set'. We will make use of them here in our study of $\omega_u(T_G)$ as well.

Definition 5.3. *For a group G, a* tri-colored sum-free set *in G is a set $S \subseteq G^3$ of triples of elements of G such that:*

- *for all $(a,b,c) \in S$, we have $ab = c$, and*
- *for all $(a_1,b_1,c_1),(a_2,b_2,c_2),(a_3,b_3,c_3) \in S$ which are not all the same triple, we have $a_1 b_2 \neq c_3$.*

In the literature, tri-colored sum-free sets are sometimes also called multiplicative matchings*.*

In a recent breakthrough, Ellenberg and Gijswijt [EG17] used techniques introduced by Croot, Lev, and Pach [CLP17] to show that there is a constant $c < 3$ such that tri-colored sum-free sets in $\mathbb{F}_3^n$ have size at most $O(c^n)$. Since then, there has been an explosion of work in the area, and this result has been extended by Sawin [Saw18] to hold for *all* nontrivial groups G, even nonabelian groups:

Theorem 5.9 ([Saw18, Theorem 1])**.** *Let G be any nontrivial finite group. There is a constant $\delta_G < 1$ such that for any positive integer n, any tri-colored sum-free set in G^n has size at most $(\delta_G|G|)^n$.*

There are a number of families of groups G where even stronger upper bounds than this are known; we refer the reader to the introduction of [BCC$^+$17b] for an exposition of these bounds.

We begin with the main connection between group tensors and tri-colored sum-free sets; this was essentially remarked in [BCC$^+$17a], but we reprove it here for completeness:

Lemma 5.2. *For any finite group G and $q \in \mathbb{N}$, if $T_G \leq_{zo} \langle q \rangle$, then G has a tri-colored sum-free set of size q.*

Proof. Let D be the subset of the terms of T_G so that $T_G \leq_{zo} D = \langle q \rangle$. Let $S := \{(a,b,c) \in G^3 \mid x_a y_b z_c \in D\}$. We will show that S is a tri-colored sum-free set in G. First, recall that every $x_a y_b z_c \in T_G$ has $ab = c$, and $D \subseteq T_G$, and so every $(a,b,c) \in S$ has $ab = c$ as well. Second, assume to the contrary that there are $(a_1,b_1,c_1),(a_2,b_2,c_2),(a_3,b_3,c_3) \in S$, not all the same triple, such that $a_1 b_2 = c_3$. This means that none of x_{a_1}, y_{b_2}, or z_{c_3} were zeroed out to get from T_G to D. But, $x_{a_1} y_{b_2} z_{c_3} \in T_G$, and so we must have $x_{a_1} y_{b_2} z_{c_3} \in D$. Since D is independent, this means that $x_{a_1} y_{b_1} z_{c_1}, x_{a_2} y_{b_2} z_{c_2}$, and $x_{a_3} y_{b_3} z_{c_3}$ must all be the same triple, contradicting how we picked them. □

In fact, a more technical proof can strengthen Lemma 5.2 to work for *degenerations* instead of just zeroing outs:

Lemma 5.3 ([BCC$^+$17a])**.** *For any finite group G and $q \in \mathbb{N}$, if $T_G \trianglelefteq \langle q \rangle$, then for $n \in \mathbb{N}$, G^n has a tri-colored sum-free set of size $q^{n-o(n)}$.*

We can use this to give lower bounds on $\omega_u(T_G)$ for any finite group G:

Corollary 5.5. *For any tensor T and any nontrivial finite group G such that $T \trianglelefteq T_G$, we have $\tilde{S}(T) < |G|$.*

Proof. Since $T \trianglelefteq T_G$, we have $\tilde{S}(T) \leq \tilde{S}(T_G)$. Letting $\delta_G < 1$ be the constant from Theorem 5.9 for G, we know that for any positive integer n, any tri-colored sum-free set in G^n has size at most $(\delta_G|G|)^n$. Hence, by Lemma 5.3, we have $S(T_G^{\otimes n}) \leq (\delta_G|G|)^{n-o(n)}$. It follows that $\tilde{S}(T_G) \leq \delta_G|G| < |G|$, as desired. □

Theorem 5.10. *For any finite group G, we have $\omega_g(T_G) > 2$.*

Proof. Simply pick $T = T_G$ in Corollary 5.5, which is known to have $\tilde{R}(T_G) \geq |G|$ (see Subsection 4.6.2), combined with Theorem 5.1. □

This shows that no fixed group tensor T_G can be used to show $\omega = 2$ using the Universal Method. That said, it does not rule out showing $\omega = 2$ by using a *sequence* $G_1, G_2, \ldots$ of groups such that $\lim_{i \to \infty} \omega_g(T_{G_i}) = 2$. Prior work has already made a similar remark for showing $\omega = 2$ by finding large 'simultaneous triple product property' constructions in G via the Group-theoretic Method, and some natural sequences of groups have already been ruled out [BCC+17b].

5.6.2 Asymptotic Slice Rank and Tri-Colored Sum-Free Set Constructions for All Finite Groups

One of the key components of our lower bounding framework is Proposition 5.3, in which we showed that matrix multiplication tensors have degenerations to large independent tensors. In this subsection, we will instead use Proposition 5.3 in a different way: to show that some other tensors of interest also have degenerations to nontrivially-large independent tensors. In particular, we will show this for the group tensor T_G of any finite group G, which will imply a nontrivially-large tri-colored sum-free set in G^n for sufficiently large n. We start with the main additional idea needed for this application:

Theorem 5.11. *For every finite group G of order $|G| = q$, there is a monomial degeneration T_G into a tensor T which is a generalized Coppersmith-Winograd tensor with parameter $q - 2$.*

Proof. Let $1 \in G$ be the identity, and let $g \in G$ be any other element. We will give three maps maps $\alpha : X_G \to \mathbb{Z}$, $\beta : Y_G \to \mathbb{Z}$, and $\gamma : Z_G \to \mathbb{Z}$ such that for any $a, b \in G$ we have $\alpha(x_a) + \beta(y_b) + \gamma(z_{ab}) \geq 0$, and then define our monomial degeneration by the map $m : X_G \cup Y_G \cup Z_G \to Mon$ given by $m(x_a) = \lambda^{\alpha(x_a)+d}$, $m(y_b) = \lambda^{\beta(y_b)+d}$, and $m(z_c) = \lambda^{\gamma(z_c)+d}$ for any $a, b, c \in G$ and a sufficiently large constant $d \in \mathbb{N}$, so that $x_a y_b z_{ab}$ will remain in the result of the monomial degeneration if and only if $\alpha(x_a) + \beta(y_b) + \gamma(z_{ab}) = 0$. The maps are given as follows:

- $\alpha(x_1) = \beta(y_1) = \gamma(z_1) = 0$,
- $\alpha(x_g) = \beta(y_g) = -\gamma(z_g) = 2$, and
- $\alpha(x_h) = \beta(y_h) = -\gamma(z_h) = 1$ for all $h \in G \setminus \{1, g\}$.

Let T be the monomial degeneration of T_G defined by α, β, γ. Define the permutation $\sigma : G \setminus \{1, g\} \to G \setminus \{1, g\}$ which sends $h \in G$ to $\sigma(h) := h^{-1}g$. We can see that:

- $x_1 y_1 z_1 \in T$ since $\alpha(x_1) = \beta(y_1) = \gamma(z_1) = 0$.

- $x_1 y_h z_h \in T$ for all $h \in G \setminus \{1\}$ (including $h = g$), since $\alpha(x_1) = 0$ while $\beta(y_h) = -\gamma(z_h) = 1$.

- $x_h y_1 z_h \in T$ for all $h \in G \setminus \{1\}$ similarly.

- $x_h y_{\sigma(h)} z_g \in T$ for all $h \in G \setminus \{1, g\}$, since $\alpha(x_h) = \beta(y_{\sigma(h)}) = 1$, while $\gamma(z_g) = -2$.

Meanwhile,

- $x_{h_1} y_{h_2} z_{h_3} \notin T$ for any $h_1, h_2, h_3 \in G \setminus \{1, g\}$ with $h_1 h_2 = h_3$, since $\alpha(h_1) = \beta(h_2) = 1$ and $\gamma(h_3) = -1$, so the three sum to 1.

- $x_h y_{h^{-1}} z_1 \notin T$ for any $h \in G \setminus \{1, g\}$ since $\alpha(x_h) = \beta(y_{h^{-1}}) = 1$ while $\gamma(z_1) = 0$, so the three sum to 2.

- $x_g y_{h_1} z_{h_2} \notin T$ for any $h_1, h_2 \in G \setminus \{1, g\}$ with $g h_1 = h_2$, since $\alpha(x_g) = 2$, $\beta(y_{h_1}) = 1$, and $\gamma(z_{h_2}) = -1$, so the three sum to 2.

- $x_{h_1} y_g z_{h_2} \notin T$ for any $h_1, h_2 \in G \setminus \{1, g\}$ with $g h_1 = h_2$ similarly.

- $x_g y_{g^{-1}} z_1 \notin T$ since $\alpha(x_g) = 2$, $\beta(y_{g^{-1}}) = 1$, and $\gamma(z_1) = 0$, so the three sum to 3.

- $x_{g^{-1}} y_g z_1 \notin T$ similarly.

- $x_g y_g z_{g^2} \notin T$ since $\alpha(x_g) = \beta(y_g) = 2$, and definitely $\gamma(z_{g^2}) \geq -2$, so the three sum to at least 2.

This covers all the entries of T_G, showing that we have defined a valid monomial degeneration to

$$T = x_1 y_1 z_1 + x_1 y_g z_g + x_g y_1 z_g + \sum_{h \in G \setminus \{1, g\}} (x_1 y_h z_h + x_h y_1 z_h + x_h y_{\sigma(h)} z_g).$$

This is indeed a generalized Coppersmith-Winograd tensor with parameter $|G \setminus \{1, g\}| = q - 2$, as desired. □

An immediate consequence of this monomial degeneration is that applying the Solar, Galactic or Universal method on T_G for *any* finite group G with $\tilde{R}(T_G) = |G|$ yields the same upper bounds on ω as the best known analysis of $CW_{|G|-2}$. Picking an appropriate group G where group operations are known to be efficient in practice could help lead to a more practical matrix multiplication algorithm.

Next, we will use the fact that matrix multiplication tensors, and hence Coppersmith-Winograd tensors, have large asymptotic slice rank, to show that for any finite group G, T_G also has a relatively large slice rank, and hence that G^n has relatively large tri-colored sum-free sets for large enough n.

Theorem 5.12. *Define $f : \mathbb{N} \to \mathbb{R}$ by $f(q) = \log_q \left(\frac{4(q+2)^3}{27} \right)$. For every positive integer q, and every tensor $CW_{q,\sigma}$ which is a generalized Coppersmith-Winograd tensor of parameter q, we have $\tilde{S}(CW_{q,\sigma}) \geq (q+2)^{2/f(q)}$.*

Remark 5.1. *For $q \geq 3$, we have $f(q) < 3$, and so $\tilde{S}(CW_{q,\sigma}) \geq (q+2)^{2/3}$.*

Remark 5.2. *In the proof of Theorem 5.12, we use a simpler lower bound on $\omega_s(CW_q)$ than is known, for ease of reading; it is, of course, possible to use the better known upper bounds on $\omega_s(CW_q)$ from [CW90, Wil12, LG14] in the proof and improve the result.*

Proof of Theorem 5.12. Define $f : \mathbb{N} \to \mathbb{R}$ by $f(q) = \log_q \left(\frac{4(q+2)^3}{27} \right)$. In [CW90, Section 6], Coppersmith and Winograd show that $\omega_s(CW_{q,\sigma}) \geq f(q)$. Hence, by Theorem 5.1, we get

$$\tilde{S}(CW_{q,\sigma}) \geq \tilde{R}(CW_{q,\sigma})^{2/f(q)} \geq (q+2)^{2/f(q)}.$$

□

Theorem 5.13. *For every (not necessarily abelian) finite group G, there is a constant $c_{|G|} > 2/3$, depending only on $|G|$, such that $\tilde{S}(T_G) \geq |G|^{c_{|G|}}$. In particular, for $n \in \mathbb{N}$, G^n has a tri-colored sum-free set of size at least $|G|^{c_{|G|}n - o(n)}$.*

Proof. The only finite groups G of order $|G| < 5$ are C_1, C_2, C_3, C_4, and C_2^2. For each of these groups, the result is shown, eg. by [KSS18]. For $|G| \geq 5$, we know from Theorem 5.11 that T_G has a monomial degeneration to a generalized Coppersmith-Winograd tensor of parameter $|G| - 2$, and so the result follows by Theorem 5.12. □

Part II

Probabilistic Polynomials and Hamming Nearest Neighbors

Chapter 6

Background and Overview

The polynomial method is a powerful tool in circuit complexity. The idea of the method is to transform all circuits of some class into "simple" polynomials which represent the circuit in some way. If the polynomial is always sufficiently simple (e.g. has low degree), and one can prove that a certain Boolean function f cannot be represented so simply, one concludes that the circuit class is unable to compute f.

Recently, these tools have found surprising uses in algorithm design. If a subproblem of an algorithmic problem can be modeled by a simple circuit, and that circuit can be transformed into a "simple" polynomial (or "simple" distribution on polynomials), then fast algebraic algorithms can be applied to evaluate or manipulate the polynomial quickly. This approach has led to advances on problems such as All-Pairs Shortest Paths [Wil14a], Orthogonal Vectors [WY14, AWY15] and Constraint Satisfaction [Wil14d].

In these applications, the key step is to randomly convert simple circuits into *probabilistic* polynomials. If f is a Boolean function on n variables, and R is a commutative ring, a *probabilistic polynomial over R for f with error $1/s$ and degree d* is a distribution $\mathcal{D}$ of degree-d polynomials over R such that for all $x \in \{0,1\}^n$, $\Pr_{p\sim\mathcal{D}}[p(x) = f(x)] \geq 1 - 1/s$. Razborov [Raz87] and Smolensky [Smo87] introduced the notion of a probabilistic polynomial. They showed that AND, OR, and XOR gates of unbounded fan-in have simple constant degree probabilistic polynomials, and hence that any low-depth $\mathsf{AC}^0[\oplus]$ circuit consisting of these gates can be transformed into a low degree probabilistic polynomial. All the prior work on polynomial method algorithms uses this transformation.

In this Part, we develop new probabilistic polynomial constructions in order to solve a variety of algorithmic problems. We focus especially on polynomial representations of threshold functions. The threshold function TH_θ determines whether at least a θ fraction of its input bits are 1s. Threshold functions are among the simplest Boolean functions that do not have constant degree probabilistic polynomials: Razborov and Smolensky showed that the MAJORITY function (a special case of a threshold function) requires degree $\Omega(\sqrt{n \log s})$. Nonetheless, as we will see throughout this Part, there are many important problems which can be reduced to evaluating circuits involving threshold gates on many inputs, and so further study of polynomial representations of threshold functions is warranted. Indeed, threshold functions have been extensively studied in theoretical computer science for many years, and there are numerous applications of linear and polynomial threshold functions to complexity and learning theory (a sample includes [BRS91, BS92, ABFR94, Bei95, KS01, OS10, She14]).

6.1 Our Results

6.1.1 Polynomial Constructions

We begin in Chapter 7 by giving a number of new polynomial constructions. We consider three different notions of polynomials representing TH_θ. Each achieves different trade-offs between polynomial degree, the randomness required, and how accurately the polynomial represents TH_θ. One key type of circuit which will recur in most of our applications is an OR of many thresholds; each of the polynomials we construct can be used to represent such a circuit by summing up to $s/3$ copies of the polynomial, one for each threshold gate (where $1/s$ is the error parameter of the construction). Each construction leads to improved algorithms in our applications.

Probabilistic Polynomials. We begin with probabilistic polynomials. Razborov and Smolensky showed over 30 year ago that TH_θ on n inputs with error $1/s$ requires degree $\Omega(\sqrt{n \log s})$. We show that their lower bound is tight by giving a matching construction. Our probabilistic polynomial construction is efficiently samplable using only polylog(ns) random bits, which will allow us to use it to design *deterministic* algorithms in some cases.

Theorem 6.1. *For any $0 \leq \theta \leq 1$, there is a probabilistic polynomial for the function TH_θ of degree $O(\sqrt{n \log s})$ on n bits with error $1/s$ over any commutative ring R that can be efficiently sampled using only $O(\log n \log(ns))$ random bits.*

Polynomial Threshold Functions. Second, we consider deterministic Polynomial Threshold Functions (PTFs). A PTF for a Boolean function f is a polynomial (*not* a distribution on polynomials) $p : \{0,1\}^n \to \mathbb{R}$ such that $p(x)$ is smaller than a fixed value when $f(x) = 0$, and $p(x)$ is larger than the value when $f(x) = 1$. In our applications, we seek PTFs with "good threshold behavior", such that $|p(x)| \leq 1$ when $f(x) = 0$, and $p(x)$ is very large otherwise. We can achieve almost the same degree as for a probabilistic polynomial, and even better degree when we focus on *ε-approximate* thresholds rather than exact thresholds:

Theorem 6.2. *We can construct a polynomial $P_{s,t,\varepsilon} : \mathbb{R} \to \mathbb{R}$ of degree $O(\sqrt{1/\varepsilon} \log s)$, such that*

- *if $x \in \{0, 1, \ldots, t\}$, then $|P_{s,t,\varepsilon}(x)| \leq 1$;*
- *if $x \in (t, (1+\varepsilon)t)$, then $P_{s,t,\varepsilon}(x) > 1$;*
- *if $x \geq (1+\varepsilon)t$, then $P_{s,t,\varepsilon}(x) \geq s$.*

For the "exact" setting with $\varepsilon = 1/t$, we can alternatively bound the degree by $O(\sqrt{t}\log(st))$.

By summing multiple copies of the polynomial from Theorem 6.2, we immediately obtain a PTF with the same degree for the OR of $O(s)$ threshold functions (needed in our applications). This theorem follows directly from known extremal properties of Chebyshev polynomials, as well as the lesser known *discrete* Chebyshev polynomials. Chebyshev polynomials are well-known to yield good approximate polynomials for computing certain Boolean functions over the reals [NS94, Pat92, KS01, She13, Val15] (see Section 6.2 below for more background).

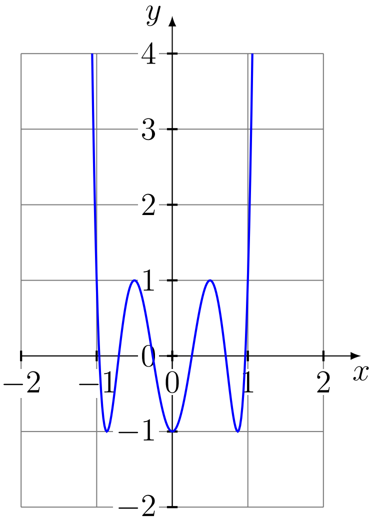

Figure 6-1: A plot of the sixth Chebyshev polynomial, $T_6(x) = 32x^6 - 48x^4 + 18x - 1$. Chebyshev polynomials have the property that, on the interval $[-1, 1]$, they always output a value from $[-1, 1]$. Among all polynomials of a fixed degree with this property, the Chebyshev polynomial takes on the largest possible value at all inputs outside $[-1, 1]$. This makes it useful for applications like Theorem 6.2, where we want a large separation between inputs on either side of a threshold.

Probabilistic PTFs. Third, we introduce a new (natural) notion of a *probabilistic PTF* for a Boolean function f. This is a distribution on PTFs, where for each input x, a PTF drawn from the distribution is highly likely to agree with f on x. Combining the techniques from probabilistic polynomials for TH_θ and the deterministic PTFs in a simple way, we construct a probabilistic PTF with good threshold behavior whose degree is *lower* than both the deterministic PTF and the degree bounds attainable by probabilistic polynomials (surprisingly breaking the "square-root barrier" of the Razborov-Smolensky lower bound):

Theorem 6.3. *We can construct a distribution $\mathcal{L}_{n,s,t,\varepsilon}$ on polynomials $L_{n,s,t,\varepsilon}$: $\{0,1\}^n \to \mathbb{R}$ of degree $O((1/\varepsilon)^{1/3}\log s)$, such that for every $x \in \{0,1\}^n$, when we draw a random $L_{n,s,t,\varepsilon} \sim \mathcal{L}_{n,s,t,\varepsilon}$:*

- *if $\sum_{i=1}^n x_i \leq t$, then $|L_{n,s,t,\varepsilon}(x_1,\ldots,x_n)| \leq 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^n x_i \in (t, t+\varepsilon n)$, then $L_{n,s,t,\varepsilon}(x_1,\ldots,x_n) > 1$ with probability at least $1-1/s$;*
- *if $\sum_{i=1}^n x_i \geq t+\varepsilon n$, then $L_{n,s,t,\varepsilon}(x_1,\ldots,x_n) \geq s$ with probability at least $1-1/s$.*

For the "exact" setting with $\varepsilon = 1/n$, we can alternatively bound the degree by $O(n^{1/3}\log^{2/3}(ns))$.

The PTFs of Theorem 6.3 can be sampled using only $O(\log(n)\cdot\log(ns))$ random bits as well; their lower degree will allow us to design faster randomized algorithms for a variety of problems. For emphasis, we will sometimes refer to PTFs as *deterministic PTFs* to distinguish them from probabilistic PTFs.

6.1.2 Algorithmic Applications

Next, by combining these polynomials for TH_θ with the aforementioned polynomial method in algorithm design (and in particular, making use of fast rectangular matrix multiplication algorithms to quickly evaluate polynomials on many inputs), we design new faster algorithms for many different problems in Chapter 8.

Batch Hamming Nearest Neighbor Search

Recall the *Hamming nearest neighbor problem* (HNN): given a set D of n database points in the d-dimensional hypercube $\{0,1\}^d$, we wish to preprocess D to support queries of the form $q \in \{0,1\}^d$, where a query answer is a point $u \in D$ that differs from q in a minimum number of coordinates. Minsky and Papert [MP69, Chapter 12.7] called this the "Best Match" problem, and it has been widely studied since. Like many situations where one wants to find points that are "most similar" to query points, HNN is fundamental to modern computing, especially in search and error correction [Ind04]. However, known exact solutions to the problem require a data structure of $2^{\Omega(d)}$ size (storing all possible queries) or query time $\Omega(n/\mathrm{poly}(\log n))$ (trying nearly all the points in the database). This is one of many examples of the *curse of dimensionality* phenomenon in search, with corresponding data structure lower bounds. For instance, Barkol and Rabani [BR02] show a size-query tradeoff for HNN in d dimensions in the cell-probe model: if one uses s cells of size b to store the database and probes at most t cells in a query, then either $s = 2^{\Omega(d/t)}$ or $b = n^{\Omega(1)}/t$.

During the late 90's, a new direction opened in the search for better nearest neighbor algorithms. The driving intuition was that it may be easier to find and generally good enough to have *approximate* solutions: points with distance within $(1+\varepsilon)$ of the optimum. Utilizing novel hashing and dimensionality reduction techniques, this beautiful line of work has had enormous impact [Kle97, IM98, KOR00, Pan06, AI06, Val15, AINR14, AR15]. Still, when turning to approximations, the exponential-in-d dependence generally turns into an exponential-in-$1/\varepsilon$ dependence, leading to a "curse of approximation" [Pat08], with lower bounds matching this intuition [CCGL99, CR04, AIP06]. For example, Andoni, Indyk, and Patrascu [AIP06] prove that any data structure for $(1+\varepsilon)$-approximate HNN using $O(1)$ probes requires $n^{\Omega(1/\varepsilon^2)}$ space.

In our first application, we design new algorithms for the natural *off-line* version of HNN. We design faster algorithms for both the exact and the approximate version, in the Hamming metric as well as in other metrics like ℓ_1 and the Jaccard distance.

Offline Hamming Nearest Neighbor Search. We first revisit exact nearest neighbors in the Hamming metric. We study the natural off-line problem of answering n Hamming nearest neighbor queries at once, on a database of size n. We call this the Batch Hamming Nearest Neighbor problem (BHNN). Here the aforementioned data structure lower bounds no longer apply—there is no information bottleneck. Nevertheless, known algorithms for BHNN still run in either about $n^2 d^{\Omega(1)}$ time (try all pairs) [GL01, MKZ09] or about $n 2^{\Omega(d)}$ time (build a table of all possible query answers). Using our probabilistic PTFs, we improve over both these bounds for $\log n \leq d \ll \log^3 n / \log^5 \log n$.

Theorem 6.4. *Given n red and n blue points in $\{0,1\}^d$ for $d = c \log n \ll \log^3 n / \log^5 \log n$, we can find an (exact) Hamming nearest/farthest blue neighbor for every red point in randomized time $n^{2-1/O(\sqrt{c}\log^{3/2} c)}$.*

Using the same ideas, we are also able to derandomize our algorithm, to achieve *deterministic* time $n^{2-1/O(c\log^2 c)}$. When $d = c \log n$ for constant c, these algorithms both have "truly subquadratic" running times. We then apply simple reductions to achieve similar running times for finding closest pairs in ℓ_1 for vectors with small integer entries, and pairs with maximum inner product or Jaccard coefficient, as well as Bichromatic Min Inner Product: given an integer k and a collection of red and blue Boolean vectors, determine if there is a red and blue vector with inner product at most k.

It is important to keep in mind that sufficiently fast off-line Hamming closest pair algorithms would yield a breakthrough in satisfiability algorithms, so there is a potential limit. Indeed, we show:

Theorem 6.5. *Suppose there is $\varepsilon > 0$ such that for all constant c,* Bichromatic Hamming Closest Pair *can be solved in $2^{o(d)} \cdot n^{2-\varepsilon}$ time on a set of n points in $\{0,1\}^{c \log n}$. Then the Strong Exponential Time Hypothesis (SETH) is false.*

The proof is actually a reduction from the (harder-looking) Orthogonal Vectors problem, where it is well-known that $n^{2-\varepsilon}$ time would refute SETH [Wil05]. Our algorithm for Theorem 6.4 shows that for all c, there is a $\delta > 0$ such that Offline Hamming Nearest Neighbor search in dimension $d = c \log n$ takes $O(n^{2-\delta})$ time. Theorem 6.5 says that showing that there is a universal $\delta > 0$ that works for all c would disprove the Strong Exponential Time Hypothesis.

Offline Approximate Nearest Neighbor Search. The problem of finding high-dimensional *approximate* nearest neighbors has received even more attention than the exact variant. Locality-sensitive hashing yields data structures that can find $(1+\varepsilon)$-factor approximate nearest neighbors to any query point in $\widetilde{O}(dn^{1-\Omega(\varepsilon)})$ (randomized) time after preprocessing in $\widetilde{O}(dn+n^{2-\Omega(\varepsilon)})$ time and space, for not only Hamming space but also ℓ_1 and ℓ_2 space [HIM12, AI06]. Thus, a batch of n queries can be answered in $\widetilde{O}(dn^{2-\Omega(\varepsilon)})$ randomized time. Exciting recent work on locality-sensitive hashing [AINR14, AR15] has improved the constant factor in the $\Omega(\varepsilon)$ bound, but not the growth rate in ε. In 2012, Gregory Valiant [Val15] reported a surprising algorithm running in $\widetilde{O}(dn+n^{2-\Omega(\sqrt{\varepsilon})})$ randomized time for the offline version of the problem in ℓ_2. We obtain a still faster algorithm for the offline problem, with $\sqrt{\varepsilon}$ improved to about $\varepsilon^{1/3}$:

Theorem 6.6. *Given n red and n blue points in $[U]^d$ and $\varepsilon \gg \frac{\log^6 \log n}{\log^3 n}$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 nearest/farthest blue neighbor for each red point in $(dn+n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))})\cdot poly(\log(nU))$ randomized time.*

Valiant's algorithm, like the previous polynomial method algorithms, relied on fast matrix multiplication. It also used Chebyshev polynomials but in a seemingly more complicated way. Our new probabilistic PTF construction is inspired by our attempt to unify Valiant's approach with the probabilistic method, which leads to an improvement of Valiant's algorithm. (We also almost succeed in derandomizing Valiant's $n^{2-\tilde{\Omega}(\sqrt{\varepsilon})}$ result in the Hamming case, except for an initial dimension reduction step; see Remark 8.3 in Section 8.3.)

Numerous applications to high-dimensional computational geometry follow; for example, we can approximate the diameter or Euclidean minimum spanning tree of a given set of n points in roughly the same running time.

The Light Bulb Problem. The last problem related to nearest neighbor search we study is the Light Bulb Problem, introduced by Leslie Valiant in 1988 [Val88]: Given as input a set S of n vectors from $\{-1,1\}^d$, which are all independently and uniformly random except for two planted vectors (the correlated pair) which have inner product at least $\rho \cdot d$ for some $0 < \rho \leq 1$, the goal is to find the correlated pair. This is a basic formulation of the problem of finding correlated variables in data analysis, and the best known algorithms for more general problems like finding correlations on the Euclidean sphere [Cha02] and learning sparse parities with noise [Val15, Appendix A] come from reductions to the Light Bulb problem.

The dimension d of the vectors is called the *sample complexity* of the problem, since it corresponds to the number of data points which must be gathered about the variables in order to determine which are correlated. When d is too small (for instance, $d < \log(n)$), then the problem is information-theoretically impossible. By standard concentration inequalities, there is a constant $c > 1$ such that, whenever $d \geq c \log n$, the correlated pair is the closest pair of vectors with high probability. We would like to design algorithms for this $d = O(\log n)$ regime.

It is not hard to see that the Light Bulb Problem is a special case of the $(1+\varepsilon)-$approximate Hamming nearest neighbor problem which we solved in Theorem 6.6 above. However, whereas before we were concentrating on the case when

ε is very small, the Light Bulb problem can be seen as the case when ε is instead a large constant. In other words, the result in Theorem 6.6 was optimizing for a different parameter than is necessary for the Light Bulb problem. Using techniques like Locality-Sensitive Hashing [IM98, PRR95, Dub10], one can solve the Light Bulb Problem in time $n^{2-O(\rho)}$. For constant $\rho > 0$, this gives a truly subquadratic running time, but the running time become quadratic as $\rho \to 0$.

In a breakthrough result, G. Valiant [Val15] gave an algorithm solving the Light Bulb Problem in time $O(n^{(5-\omega)/(4-\omega)+\varepsilon} + nd) < O(n^{1.615} + nd)$, where $\omega < 2.373$ is the exponent of matrix multiplication, for *any* constant $\rho > 0$, no matter how small. Thereafter, Karppa et al. [KKK16] gave an improved algorithm with a running time of $O(n^{2\omega/3+\varepsilon} + nd) < O(n^{1.582} + nd)$. Both of these algorithms work when the sample complexity d matches, up to a constant, the information-theoretically necessary $d = \Theta(\log n)$.

Here, we give a new randomized algorithm with a simple analysis which matches the best known running time $O(n^{2\omega/3+\varepsilon})$ and sample complexity $d = \Theta(\log n)$. Previous algorithms for the problem made use of sophisticated random sampling techniques, but we show that these are unnecessary when approaching the problem using the polynomial method instead.

By leveraging our simpler analysis, we also give new faster deterministic algorithms for the problem. However, as the inputs to the Light Bulb Problem come from a random distribution, we need to be careful about what a deterministic algorithm means. We give algorithms in two different settings:

- an algorithm running in the same time $O(n^{2\omega/3+\varepsilon}) < O(n^{1.582})$ for sample complexity $d = \Theta(\log n)$ which is correct on *almost all instances* (i.e. the probability of drawing an instance where the algorithm fails is $1/\text{poly}(n)$), and

- an algorithm running in time $O(n^{4\omega/5+\varepsilon}) < O(n^{1.899})$ for sample complexity $d = \Theta(\log n)$ which must correctly solve every instance, given the promise that the pairs of vectors other than the correlated pair are not much more correlated than one would expect random vectors to be.

See Subsection 8.12 for more details. In both of these settings, the previous best known running time [KKKÓC16] was at best $O(n^{1.996})$.

Satisfiability Algorithms

Next, we apply our polynomials to design faster *satisfiability algorithms* in a number of different settings which involve threshold functions and counting.

MAX-SAT. We begin with MAX-SAT, the problem of finding an assignment that satisfies the maximum number of clauses in a given CNF formula with n variables. In the sparse case when the number of clauses is cn, a series of papers have given faster exact algorithms, for example, achieving $2^{n-n/O(c\log c)}$ time by Dantsin and Wolpert [DW06a], $2^{n-n/O(c\log c)^{2/3}}$ time by Sakai et al. [SSTT15a], and $2^{n-n/O(\sqrt{c})}$ time by Chen and Santhanam [CS15]. Using the polynomial method and our new probabilistic PTF construction, we obtain the following improved result:

Theorem 6.7. *Given a CNF formula with n variables and $cn \ll n^4/\log^{10} n$ clauses, we can find an assignment that satisfies the maximum number of clauses in randomized $2^{n-n/O(c^{1/3}\log^{7/3} c)}$ time.*

For general dense instances, the problem becomes tougher. Williams [Wil05] gave an $O(2^{0.792n})$-time algorithm for MAX-2-SAT, but an $O(2^{(1-\delta)n})$-time algorithm for MAX-3-SAT (for a universal $\delta > 0$) has remained open; currently the best reported time bound [SSTT15b] is $2^{n-\Omega(n/\log n)^{1/3}}$, which can be slightly improved to $2^{n-\Omega(\sqrt{n/\log n})}$ with more care. We make new progress on not only MAX-3-SAT but also MAX-4-SAT:

Theorem 6.8. *Given a weighted 4-CNF formula F with n variables with positive integer weights bounded by poly(n), we can find an assignment that maximizes the total weight of clauses satisfied in F, in randomized $2^{n-n/O(\log^2 n \log^2 \log n)}$ time. In the sparse case when the clauses have total weight cn, the time bound improves to $2^{n-n/O(\log^2 c \log^2 \log c)}$.*

LTF-LTF Circuit SAT Algorithms and Lower Bounds. Using our small sample space for probabilistic polynomials for threshold functions (Theorem 6.1), we construct a new circuit satisfiability algorithm for circuits with linear threshold functions (LTFs) which improves over several prior results. Let $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[S_1, S_2, S_3]$ be the class of circuits with a layer of S_3 LTFs at the bottom layer (nearest the inputs), a layer of S_2 LTFs above the bottom layer, and a size-S_1 $\mathsf{AC}^0[m]$ circuit of depth d above the two LTF layers.[1]

Theorem 6.9. *For every integer $d > 0$, $m > 1$, and $\delta > 0$, there is an $\varepsilon > 0$ and an algorithm for satisfiability of $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^\varepsilon}, 2^{n^\varepsilon}, n^{2-\delta}]$ circuits that runs in deterministic 2^{n-n^ε} time.*

Williams [Wil14b] gave a comparable SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF}$ circuits of 2^{n^ε} size, where $\varepsilon > 0$ is sufficiently small.[2] Theorem 6.9 strictly generalizes the previous algorithm, allowing another layer of $n^{2-\varepsilon}$ linear threshold functions below the existing LTF layer. Theorem 6.9 also trivially implies deterministic SAT algorithms for $\mathsf{LTF} \circ \mathsf{LTF}$ circuits of up to $n^{2-o(1)}$ gates, improving over the recent SAT algorithms of Chen, Santhanam, and Srinivasan [CSS16] which only work for $n^{1+\varepsilon}$-wire circuits for $\varepsilon \ll 1$, and the SAT algorithms of Impagliazzo, Paturi, and Schneider [IPS13].

Applying the known connection between circuit satisfiability algorithms and circuit lower bounds for E^{NP} problems [Wil13, Wil14c, JMV15], the following is immediate:

Corollary 6.1. *For every $d > 0$, $m > 1$, and $\delta \in (0,1)$, there is an $\varepsilon > 0$ such that the class E^{NP} does not have non-uniform circuits in $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^\varepsilon}, 2^{n^\varepsilon}, n^{2-\delta}]$.*

[1] Recall that for an integer $m \geq 2$, $\mathsf{AC}^0[m]$ refers to constant-depth unbounded fan-in circuits over the basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m\}$, where MOD_m outputs 1 iff the sum of its input bits is divisible by m.

[2] Recall ACC^0 is the infinite union of $\mathsf{AC}^0[m]$ for all integers $m \geq 2$.

In particular, for every $\varepsilon > 0$, E^{NP} does not have $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits where the $\mathsf{ACC}^0 \circ \mathsf{LTF}$ subcircuit has $2^{n^{o(1)}}$ size and the bottom LTF layer has $n^{2-\varepsilon}$ gates.

Most notably, Corollary 6.1 proves lower bounds with $n^{2-\varepsilon}$ LTFs on the bottom layer and *subexponentially many* LTFs on the second layer. This improves upon recent $\mathsf{LTF} \circ \mathsf{LTF}$ gate lower bounds of Kane and Williams [KW16], at the cost of raising the complexity of the hard function from TC^0_3 to E^{NP}. Suguru Tamaki [Tam16] has recently reported similar results for depth-two circuits with both symmetric and threshold gates.

A Powerful Randomized SAT Algorithm. Finally, combining the probabilistic PTF (Theorem 6.3) and probabilistic polynomial (Theorem 6.1) for threshold functions, we give a randomized SAT algorithm for a rather powerful class of circuits. The class $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ denotes the class of circuits with a majority gate at the top, along with two layers of linear threshold gates, and arbitrary $O(1)$-depth AC^0 circuitry between these three layers. This circuit class is arguably much more powerful than TC^0_3 ($\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$), based on known low-depth circuit constructions for arithmetic functions (e.g. [CSV84, MT98, MT99]).

Theorem 6.10. *For all $\varepsilon > 0$ and integers $d \geq 1$, there is a $\delta > 0$ and a randomized satisfiability algorithm for $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of depth d running in $2^{n-\Omega(n^\delta)}$ time, on circuits with the following properties:*

- *the top MAJ gate, along with every LTF on the middle layer, has $O(n^{6/5-\varepsilon})$ fan-in, and*
- *there are $O(2^{n^\delta})$ many AND/OR gates (anywhere) and LTF gates at the bottom layer.*

Theorem 6.10 applies the probabilistic PTF of degree about $n^{1/3}$ (Theorem 6.3) to the top MAJ gate, probabilistic polynomials over $\mathbb{Z}$ of degree about $n^{1/2}$ (Theorem 6.1) to the middle LTFs, and weight reduction to the bottom LTFs; the rest can be represented with poly(n^δ) probabilistic degree.

It would not be surprising if the above circuit class contained strong pseudorandom function candidates; that is, it seems likely that the Natural Proofs barrier applies to this circuit class. Hence from the circuit lower bounds perspective, the problem of derandomizing the SAT algorithm of Theorem 6.10 is extremely interesting.

6.2 Other Related Work

Chebyshev Polynomials in Theoretical Computer Science. New applications of Chebyshev polynomials to algorithm design are a key component of the algorithms in this Part. This is certainly not a new phenomenon in itself; here we briefly survey some prior related usages of Chebyshev polynomials. First, Nisan and Szegedy [NS94] used Chebyshev polynomials to compute the OR function on n Boolean variables with an "approximating" polynomial $p : \mathbb{R}^n \to \mathbb{R}$, such that for all $x \in \{0,1\}^n$ we have $|OR(x) - p(x)| \leq 1/3$, yet $\deg(p) = O(\sqrt{n})$. They also proved the degree bound is tight up to constants in the big-O; Paturi [Pat92] generalized the upper and lower bound to all symmetric functions.

This work has led to several advances in learning theory. Building on the polynomials of Nisan and Szegedy, Klivans and Servedio [KS01] showed how to compute an OR of t ANDs of w variables with a PTF of degree $O(\sqrt{w} \log t)$, similar to our degree bound for computing an OR of t MAJORITYs of w variables of Theorem 7.5 (however, note our bound in the "exact" setting is a bit better, due to our use of discrete Chebyshev polynomials). They also show how to compute an OR of s ANDs on n variables with a *deterministic* PTF of $O(n^{1/3} \log s)$ degree, similar to our cube-root-degree probabilistic PTF for the OR of MAJORITY of Theorem 7.6 in the "exact" setting. However, it looks difficult to generalize Klivans-Servedio's $O(n^{1/3} \log s)$ degree bound to compute an OR of MAJORITY: part of their construction uses a reduction to decision lists which works for conjunctions but not for MAJORITY functions. Klivans, O'Donnell and Servedio [KOS04] show how to compute an AND of k MAJORITY on n variables with a PTF of degree $O(\sqrt{w} \log k)$. By a simple transformation via De Morgan's law, there is a polynomial for OR of MAJORITY with the same degree. Their degree is only slightly worse than ours in terms of k (because we use discrete Chebyshev polynomials).

In streaming algorithms, Harvey, Nelson, and Onak [HNO08] use Chebyshev polynomials to design efficient algorithms for computing various notions of entropy in a stream. As a consequence of a query upper bound in quantum computing, Ambainis et al. [ACR+10] show how to approximate any Boolean formula of size s with a polynomial of degree $s^{1/2+o(1)}$, improving on earlier bounds of O'Donnell and Servedio [OS10] that use Chebyshev polynomials. Sachdeva and Vishnoi [SV13] give applications of Chebyshev polynomials to graph algorithms and matrix algebra. Linial and Nisan [LN90] use Chebyshev polynomials to approximate inclusion-exclusion formulas, and Sherstov [She08] extends this to arbitrary symmetric functions.

Further Applications of our Polynomials for Threshold Functions. Since the publication of a preliminary version of our polynomial constructions [AW15, ACW16], other researchers have applied them to even more problems in matching and computational geometry. Moeller et al. [MPS16] use them to give a subquadratic time algorithm for finding stable matchings in the case when the preference lists are given by a succinct representation rather than a quadratic-size list of lists. Chan [Cha18] applies Chebyshev polynomials to a variety of problems in low-dimensional computational geometry such as approximate nearest neighbor search and constructing ε-kernels.

Hardness of Approximate Nearest Neighbor Search. In our Theorem 6.5, we proved a lower bound on the running time of algorithms for exact batch nearest neighbor search assuming the Strong Exponential Time Hypothesis (SETH). In follow-up work, Rubinstein [Rub18] showed a similar hardness result for *approximate* batch nearest neighbor search: assuming SETH, for every $\delta > 0$, there is an $\varepsilon > 0$ such that the $(1 + \varepsilon)$-approximate batch Hamming nearest neighbor problem on n

input points requires $\Omega(n^{2-\delta})$ time. Assuming SETH, this gives a limit to how much one can improve the running time of our algorithm in Theorem 6.6.

6.3 Bibliographic Details

This Part of the dissertation is based off of the results in three previously published papers:

- 'Probabilistic Polynomials and Hamming Nearest Neighbors' with Ryan Williams [AW15], which appeared in FOCS 2015,
- 'Polynomial Representations of Threshold Functions and Algorithmic Applications' with Timothy M. Chan and Ryan Williams [ACW16], which appeared in FOCS 2016, and
- 'An Illuminating Algorithm for the Light Bulb Problem' [Alm19a], which appeared in SOSA 2019.

Subsection 7.2.1, Section 7.4, Subsection 7.5.1, and Section 8.2 present results from [AW15]. Section 8.4 presents results from [Alm19a]. Subsection 7.5.2, Section 7.6, and the remainder of Chapter 8 present results from [ACW16]. The earlier Sections of Chapter 7 give an introduction to the theory of probabilistic polynomials; we cite the original sources of the results therein when appropriate.

Chapter 7

Probabilistic Polynomials

7.1 Multilinear Polynomials Computing Boolean Functions

The main topic of this chapter is polynomial representations of Boolean functions. We focus in particular on *multilinear* polynomials.

Definition 7.1. *A multilinear polynomial $p : R^n \to R$ over a commutative ring R is an* exact polynomial *for the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if we have $p(x) = f(x)$ for all $x \in \{0,1\}^n$.*

Example 7.1. *The polynomial $p(x_1, x_2, \ldots, x_n) = x_1 \cdot x_2 \cdots x_n$ exactly computes $\mathsf{AND}(x_1, x_2, \ldots, x_n)$, since p outputs 1 when all its inputs are 1, and it outputs 0 when any of its inputs is 0. Similarly, since we can write $\mathsf{OR}(x_1, x_2, \ldots, x_n) = 1 - \mathsf{AND}(1-x_1, 1-x_2, \ldots, 1-x_n)$, it follows that $\mathsf{OR}(x_1, x_2, \ldots, x_n)$ is exactly computed by the polynomial $1 - p(1-x_1, 1-x_2, \ldots, 1-x_n) = 1 - (1-x_1) \cdot (1-x_2) \cdots (1-x_n)$.*

The exact polynomial for a Boolean function f can be seen as a change of basis of f. To be more precise, we need some definitions.

- For a subset $T \subseteq [n]$, let $I(T) \in \{0,1\}^n$ denote the *indicator vector* for T, which has $I(T)_i = 1$ when $i \in T$ and $I(T)_i = 0$ when $i \notin T$.

- For an n-input Boolean function $f : \{0,1\}^n \to \{0,1\}$, let $V(f) \in \{0,1\}^{2^n}$, whose entries are indexed by subsets $T \subseteq [n]$, be the *truth table vector* of f, which has $V(f)_T = f(I(T))$ for all $T \subseteq [n]$.

- For $n \in \mathbb{N}$, define the matrix $M_{\mathsf{SUB,n}} \in \{0,1\}^{2^n \times 2^n}$, whose rows and columns are indexed by subsets of $[n]$, and whose entry $M_{\mathsf{SUB,n}}[T,S]$ for $T, S \subseteq [n]$ is given by

$$M_{\mathsf{SUB,n}}[T,S] = \begin{cases} 1 & \text{if } S \subseteq T, \\ 0 & \text{otherwise.} \end{cases}$$

  Since $M_{\mathsf{SUB,n}}$ is an upper-triangular matrix with all 1s on the diagonal, it has determinant 1, so it has full rank and a unique inverse over any commutative ring.

- Finally, any multilinear polynomial $p : R^n \to R$ over a ring R can be written as

$$p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} \alpha_S \cdot \prod_{i \in S} x_i. \tag{7.1}$$

  Let $\alpha(p) \in R^{2^n}$, whose entries are indexed by subsets of $[n]$, denote the *coefficient vector* of p, given by $\alpha(p)_S = \alpha_S$, the coefficient from (7.1).

Evidently, a Boolean function f is in bijection with its truth table vector $V(f)$, and a multilinear polynomial p is in bijection with its coefficient vector $\alpha(p)$. When p is an exact polynomial for f, then these two vectors are a change of basis of one another:

Proposition 7.1. *The multilinear polynomial $p : R^n \to R$ over a commutative ring R is an exact polynomial for the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if and only if $V(f) = M_{\mathsf{SUB,n}} \cdot \alpha(p)$.*

Proof. For any subset $T \subseteq [n]$, we have

$$p(I(T)) = \sum_{S \subseteq T} \alpha(p)_S = \left[M_{\mathsf{SUB,n}} \cdot \alpha(p)\right]_T.$$

We therefore have $p(I(T)) = f(I(T))$ if and only if $\left[M_{\mathsf{SUB,n}} \cdot \alpha(p)\right]_T = V(f)_T$. This must hold for all $T \subseteq [n]$ for p to exactly compute f. □

Corollary 7.1. *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$ and any commutative ring R, there is a unique multilinear polynomial $p : R^n \to R$ such that $p(x) = f(x)$ for all $x \in \{0,1\}^n$.*

So far, we have written any Boolean function $f : \{0,1\}^n \to \{0,1\}$ in the 'AND basis', i.e. the basis of monomials $\prod_{i \in S} x_i$ for $S \subseteq [n]$ which compute the AND on a subset of the inputs. One can similarly see that over any commutative ring R, every Boolean function f also has a unique representation over some other choices of basis, including:

- The 'NOR basis' of functions $\prod_{i \in S}(1 - x_i)$ for $S \subseteq [n]$.
- The 'XOR basis' of functions $\bigoplus_{i \in S} x_i = \frac{1}{2} - \frac{1}{2} \prod_{i \in S}(1 - 2x_i)$ for $S \subseteq [n]$ (whenever the characteristic of the ring R is not positive and even).

7.2 Typically Correct Polynomials

One of the main goals in the polynomial method is to design low-degree polynomial representations of Boolean functions. As we saw in Example 7.1 above, even very simple Boolean functions like AND and OR on n inputs require degree n to compute exactly, the maximum possible degree of a multilinear polynomial on n inputs. In order to achieve a lower degree, we need to weaken the requirements on our polynomials. One natural way to do so is to require the polynomial be correct only on *most* inputs:

Definition 7.2. *For any $\varepsilon \in [0,1]$, a multilinear polynomial $p : R^n \to R$ over a commutative ring R is a* $(1-\varepsilon)$-correct polynomial *for the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if we have $p(x) = f(x)$ for at least a $(1-\varepsilon)$ fraction of all $x \in \{0,1\}^n$. The* $(1-\varepsilon)$-correct degree *of f over R is the minimum degree of such a polynomial p.*

Many functions have much lower $(1-\varepsilon)$-correct degree than exact degree. For instance, although AND on n inputs has exact degree n, its $(1-\varepsilon)$-correct degree is 0 for any $\varepsilon \geq 2^{-n}$, since the polynomial $p(x) = 0$ computes it correctly on all but one point from $\{0,1\}^n$. Such simple constructions like this are, unfortunately, not particularly useful in applications.

That said, there are interesting Boolean functions whose $(1-\varepsilon)$-correct degree is less trivial. Consider, for instance, the majority function MAJ on n inputs. Although MAJ has exact degree n, we will show in the remainder of this section that it has ε-typically correct degree $\Theta(\sqrt{n \log(1/\varepsilon)})$ for all $\varepsilon > 0$.

7.2.1 Interpolating Polynomials for Symmetric Functions

We begin in this subsection by proving the upper bound, that MAJ on n variables has $(1-\varepsilon)$-correct degree $O(\sqrt{n \log(1/\varepsilon)})$. The key new polynomial construction we will need is an interpolating polynomial for correctly computing symmetric Boolean functions on inputs of certain Hamming weights. Such a polynomial can be derived from prior work (at least over fields [Sri13]), but for completeness, we prove its existence here.

Lemma 7.1. *For any integers n, r, k with $n \geq k + r$ and any integers $c_1, \ldots, c_r$, there is a multivariate polynomial $p : \{0,1\}^n \to \mathbb{Z}$ of degree $r-1$ with integer coefficients such that $p(x) = c_i$ for all $x \in \{0,1\}^n$ with Hamming weight $|x| = k + i$.*

Notice that it is not immediately obvious from univariate polynomial interpolation that the polynomial p exists as described, since the univariate polynomial $q : \mathbb{R} \to \mathbb{R}$ such that $q(k+i) = c_i$ typically has rational (non-integer) coefficients. Lemma 7.1 is more general than a result claimed without proof by Srinivasan ([Sri13], Lemma 14). It also generalizes of a theorem of Bhatnagar et al. ([BGL06], Theorem 2.8).

Proof. Our polynomial p will have the form

$$p(x_1, \ldots, x_n) = \sum_{i=0}^{r-1} a_i \cdot \sum_{\substack{S \subseteq [n] \\ |S| = i}} \left(\prod_{j \in S} x_j \right)$$

for some constants $a_0, \ldots, a_{r-1} \in \mathbb{Z}$. Hence, we will get that for any $x \in \{0,1\}^n$:

$$p(x) = \sum_{i=0}^{r-1} \binom{|x|}{i} a_i.$$

Define the matrix:

$$M = \begin{pmatrix} \binom{k+1}{0} & \binom{k+1}{1} & \cdots & \binom{k+1}{r-1} \\ \binom{k+2}{0} & \binom{k+2}{1} & \cdots & \binom{k+2}{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ \binom{k+r}{0} & \binom{k+r}{1} & \cdots & \binom{k+r}{r-1} \end{pmatrix}.$$

The conditions of the stated lemma are that

$$M \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{r-1} \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{pmatrix}.$$

By Lemma 7.2 (proved below), M always has determinant 1. Because M is a matrix with integer entries and determinant 1, its inverse M^{-1} is also an integer matrix. Multiplying through by M^{-1} above gives integer expressions for the a_i, as desired. □

Lemma 7.2. *For any univariate polynomials $p_1, p_2, \ldots, p_r$ such that p_i has degree $i-1$, and any pairwise distinct $x_1, x_2, \ldots, x_r \in \mathbb{Z}$, the matrix*

$$M = \begin{pmatrix} p_1(x_1) & p_2(x_1) & \cdots & p_r(x_1) \\ p_1(x_2) & p_2(x_2) & \cdots & p_r(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(x_r) & p_2(x_r) & \cdots & p_r(x_r) \end{pmatrix}$$

has determinant

$$det(M) = \left(\prod_{i=1}^{r} c_i\right) \cdot \left(\prod_{1 \le i < j \le r} (x_j - x_i)\right),$$

where c_i is the coefficient of x^{i-1} in p_i.

Proof. For i from 1 up to $r-1$, we can add multiples of column i of M to the subsequent columns in order to make the coefficient of x^{i-1} in all the other columns 0. The resulting matrix is

$$M' = \begin{pmatrix} c_1 & c_2 x_1 & \cdots & c_r x_1^{r-1} \\ c_1 & c_2 x_2 & \cdots & c_r x_2^{r-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 x_r & \cdots & c_r x_r^{r-1} \end{pmatrix}.$$

This is a Vandermonde matrix which has the desired determinant. □

Since MAJ is a symmetric function, we can use Lemma 7.1 to get a $(1-\varepsilon)$-correct degree upper bound for it:

Corollary 7.2. *Over any commutative ring R, MAJ on n inputs has $(1-\varepsilon)$-correct degree $O(\sqrt{n \log(1/\varepsilon)})$ for all $\varepsilon \in (0,1)$.*

Proof. Applying Lemma 7.1, we know there is a polynomial $p : \{0,1\}^n \to \mathbb{Z}$ over $\mathbb{Z}$ of degree $O(\sqrt{n \log(1/\varepsilon)})$ with integer coefficients such that $p(x) = \mathsf{MAJ}(x)$ for all $x \in \{0,1\}^n$ such that $||x| - n/2| \leq \sqrt{n \log(1/\varepsilon)}$. We can view p as a polynomial over R (possibly by taking p mod the characteristic of R when R has positive characteristic) which is correct for all such x as well.

It remains to show that the fraction of $x \in \{0,1\}^n$ with $||x| - n/2| \leq \sqrt{n \log(1/\varepsilon)}$ is at least $1 - \varepsilon$. This is equivalent to showing that, if we draw an $x \sim \{0,1\}^n$ uniformly at random, then the probability that $||x| - n/2| \geq \sqrt{n \log(1/\varepsilon)}$ is at most ε. By Hoeffding's inequality (Lemma 2.1 from the Preliminaries) we see that

$$\Pr\left[|x| \leq \frac{n}{2} - \sqrt{n \log(1/\varepsilon)}\right] \leq \exp\left(-\frac{2(\sqrt{n \log(1/\varepsilon)})^2}{n}\right) = \exp\left(-2\log(1/\varepsilon)\right) < \frac{\varepsilon}{2}.$$

By symmetry, we also have $\Pr\left[|x| \geq \frac{n}{2} + \sqrt{n \log(1/\varepsilon)}\right] < \varepsilon/2$, and so in total, we have that the probability of $||x| - n/2| \geq \sqrt{n \log(1/\varepsilon)}$ is at most ε, as desired. □

7.2.2 The Razborov-Smolensky Lower Bound

We now prove a matching lower bound, that MAJ on n variables requires $(1-\varepsilon)$-correct degree $\Omega(\sqrt{n \log(1/\varepsilon)})$. This is the classic result of Razborov [Raz87] and Smolensky [Smo87]; in this subsection, we present the proof technique of Razborov. We begin with a Lemma showing that low-degree $(1-\varepsilon)$-correct polynomials for MAJ lead to relatively low-degree $(1-\varepsilon)$-correct polynomials for *any* Boolean function.

Lemma 7.3. *For any commutative ring R and set $S \subseteq \{0,1\}^n$, suppose there is a polynomial $p : R^n \to R$ of degree $deg(p) = d$, such that $p(x) = \mathsf{MAJ}(x)$ for all $x \in S$. Then, for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ there is a polynomial $q : R^n \to R$ of degree at most $deg(q) \leq n/2 + d$ such that $q(x) = f(x)$ for all $x \in S$.*

Proof. Let $t : R^n \to R$ be the exact multilinear polynomial for f over R, meaning $t(x) = f(x)$ for all $x \in \{0,1\}^n$. We can write t out in two different ways, first over the 'AND basis' of monomials:

$$t(x) = \sum_{T \subseteq [n]} a_T \prod_{i \in T} x_i,$$

and second over the 'NOR basis':

$$t(x) = \sum_{T \subseteq [n]} b_T \prod_{i \in T} (1 - x_i),$$

where the $a_T, b_T \in R$ are the appropriate coefficients.

Notice that if $x \in \{0,1\}^n$ is such that $\mathsf{MAJ}(x) = 0$, then for any $T \subseteq [n]$ with

$|T| > n/2$ we have $\prod_{i \in T} x_i = 0$. Hence,

$$\text{If } \mathsf{MAJ}(x) = 0, \text{ then } f(x) = \sum_{T \subseteq [n] \text{ s.t. } |T| \leq n/2} a_T \prod_{i \in T} x_i.$$

Similarly,

$$\text{If } \mathsf{MAJ}(x) = 1, \text{ then } f(x) = \sum_{T \subseteq [n] \text{ s.t. } |T| \leq n/2} b_T \prod_{i \in T} (1 - x_i).$$

Combining, we see that for all $x \in \{0,1\}^n$,

$$\begin{aligned} f(x) = \mathsf{MAJ}(x) \cdot & \left[\sum_{T \subseteq [n] \text{ s.t. } |T| \leq n/2} b_T \prod_{i \in T} (1 - x_i) \right] \\ &+ (1 - \mathsf{MAJ}(x)) \cdot \left[\sum_{T \subseteq [n] \text{ s.t. } |T| \leq n/2} a_T \prod_{i \in T} x_i \right]. \end{aligned}$$

Substituting p for MAJ above gives the desired polynomial q. □

We can now prove our lower bound:

Theorem 7.1 ([Raz87, Smo87]). *There is a constant $c > 0$ such that, for every $\varepsilon \in (0, 1/2)$, every commutative ring R (other than the trivial ring), and every $(1-\varepsilon)$-correct polynomial p for MAJ over R, the degree of p is at least $c \cdot \sqrt{n \log(1/\varepsilon)}$.*

Proof. Assume to the contrary that there is such a polynomial $p : R^n \to R$ of degree $d < c\sqrt{n \log(1/\varepsilon)}$. Let $S \subseteq \{0,1\}^n$ be the set of x such that $p(x) = \mathsf{MAJ}(x)$; by assumption we have $|S| \geq (1 - \varepsilon) \cdot 2^n$. In particular, by Lemma 7.3, for every $s \in S$, there is a multilinear polynomial $p_s : R^n \to R$ of degree at most $n/2 + d$ such that $p_s(s) = 1$ and $p_s(x) = 0$ for all $x \in S \setminus \{s\}$.

Consider the vector space V of R-linear combinations of the p_s polynomials, i.e. $V = \{\sum_{s \in S} a_s \cdot p_s(x) \mid a_s \in R\}$. The p_s polynomials are linearly independent, since any linear combination p' of the $p_{s'}$ for $s' \neq s$ will have $p'(s) = 0$, whereas $p_s(s) = 1$. Hence, the dimension of V is at least $|S| \geq (1 - \varepsilon) \cdot 2^n$.

Meanwhile, every polynomial in V has degree at most $n/2 + d$, and so V is a subspace of the space V' of multilinear polynomials over R of degree at most $n/2 + d$. This space V' is spanned by the multilinear monomials of degree at most $n/2 + d$. The number of such monomials is

$$\sum_{i=0}^{n/2+d} \binom{n}{i} = 2^n - \sum_{i=0}^{n/2-d} \binom{n}{i} \leq 2^n - \binom{n}{n/2 - d}.$$

Applying Corollary 2.1 from the Preliminaries, we can further upper bound this by:

$$\leq 2^n - 2^{n - \Theta(d^2/n)} = 2^n \cdot (1 - 2^{-\Theta(d^2/n)}) < 2^n \cdot (1 - 2^{-\Theta(c^2)\log(1/\varepsilon)}) = 2^n \cdot (1 - \varepsilon^{\Theta(c^2)}).$$

If we pick a sufficiently small $c > 0$ then this is less than $2^n \cdot (1-\varepsilon)$. Then V', a vector space of dimension less than $2^n \cdot (1-\varepsilon)$, contains as a subspace V, a vector space of dimension at least $2^n \cdot (1-\varepsilon)$, a contradiction. □

7.3 Probabilistic Polynomials

In the previous section, we showed that MAJ on n bits has $(1-\varepsilon)$-correct degree $\Theta(\sqrt{n \log(1/\varepsilon)})$ over any commutative ring R. However, for other Boolean functions like AND and OR, the $(1-\varepsilon)$-correct polynomials were trivial. This is because of a weakness of typically correct polynomials for a Boolean function f: they can concentrate their errors on the 'hard' inputs of f. In the case of OR, the polynomial can simply get the answer wrong on the one point where it should output 0.

In our algorithmic applications below, we will need a stronger polynomial notion than this, in which the polynomial has a good chance of getting any given input correct:

Definition 7.3. *For any $\varepsilon \in [0,1]$, and any commutative ring R, a* probabilistic polynomial with error ε and degree d *for the Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a distribution $\mathcal{P}$ on polynomials $p : \{0,1\}^n \to R$ of degree at most d over R such that, for every $x \in \{0,1\}^n$, we have*

$$\Pr_{p \sim \mathcal{P}}[p(x) = f(x)] \geq 1 - \varepsilon.$$

The ε-probabilistic degree of f over R is the minimum degree of a probabilistic polynomial with error ε for f.

A probabilistic polynomial is required to take all the different values of $f(x)$ into account by enforcing that ε is the worst case probability, among all $x \in \{0,1\}^n$, that $p \sim \mathcal{P}$ incorrectly outputs a value $p(x) \neq f(x)$.

For every $\varepsilon \in (0,1)$, every commutative ring R, and every Boolean function $f : \{0,1\}^n \to \{0,1\}$, we can see that the ε-probabilistic degree of f over R is at least the $(1-\varepsilon)$-correct degree of f over R. We next give two examples where there is a separation between the two.

7.3.1 The Probabilistic Degree of OR

We noted earlier that the $(1-2^{-n})$-correct degree of OR is 0. In this subsection, we present classical polynomial constructions for OR, showing that the ε-probabilistic degree of OR on n inputs is $\Theta(\log 1/\varepsilon)$ over the finite field $\mathbb{F}_q$ for prime q, and $O(\log n \log(1/\varepsilon))$ over $\mathbb{Z}$. The upper bounds extend without much work to any commutative ring R as well.

Proposition 7.2 ([Raz87, Smo87]). *For any prime q and any $\varepsilon \in (0,1)$, the ε-probabilistic degree of OR on n inputs over $\mathbb{F}_q$ is at most $(q-1) \cdot \lceil \log_q(1/\varepsilon) \rceil$.*

Proof. We construct a probabilistic polynomial $\mathcal{P}$ for OR on n inputs. To draw a polynomial from $\mathcal{P}$, pick $k = \lceil \log_q(1/\varepsilon) \rceil$ independent uniformly random hash functions $h_1, \dots, h_k : [n] \to \mathbb{F}_q$, then output the polynomial

$$p(x_1, \dots, x_n) = 1 - \prod_{\ell=1}^{k} \left(1 - \left(\sum_{i=1}^{n} h_\ell(i) \cdot x_i \right)^{q-1} \right).$$

The degree of p is $(q-1) \cdot k$ as desired.

We now prove correctness. When $x \in \{0,1\}$ is such that $\mathsf{OR}(x) = 0$, then $x_i = 0$ for all $i \in [n]$, so p always outputs 0. Otherwise, if $\mathsf{OR}(x) = 1$, then there is at least one $i \in [n]$ such that $x_i = 1$. It follows that, for each $\ell \in [k]$, the sum $\sum_{i=1}^{n} h_\ell(i) \cdot x_i$ is a uniformly random element of $\mathbb{F}_q$. In particular, with probability $1 - 1/q$ we have $\sum_{i=1}^{n} h_\ell(i) \cdot x_i \neq 0$, and hence $(\sum_{i=1}^{n} h_\ell(i) \cdot x_i)^{q-1} = 1$ by Fermat's little theorem. If this is the case for any $\ell \in [k]$ then p outputs 1. Hence, since the k hash functions are independent, the probability that p does not output 1 is $q^{-k} \leq \varepsilon$, as desired. □

We show next that, up to constant factors, the degree upper bound achieved in Proposition 7.2 is tight.

Proposition 7.3. *For any commutative ring R (other than the trivial ring) and any $\varepsilon \in [2^{-n}, 1/4]$, the ε-probabilistic degree of* OR *on n inputs over R is at least* $\log(1/\varepsilon) - 1$.

Proof. Suppose that OR on n inputs has a probabilistic polynomial $\mathcal{P}$ of degree d for error ε over R. Let m be the biggest integer less than $\log(1/\varepsilon)$, so that $\log(1/\varepsilon) - 1 \leq m < \log(1/\varepsilon)$, and note that $m \leq n$.

We construct a probabilistic polynomial $\mathcal{Q}$ for OR on m inputs, by drawing a random $p \sim \mathcal{P}$ and outputting $q(x_1, \dots, x_m) = p(x_1, \dots, x_m, 0, \dots, 0)$, where we have set $x_i = 0$ in p for all $i > m$. Since $\mathcal{Q}$ is just a restriction of the inputs to $\mathcal{P}$, we have that $\mathcal{Q}$ is a probabilistic polynomial for OR on m inputs with error ε and degree d.

There are only 2^m different possible values for the m binary inputs to $\mathcal{Q}$, but $\mathcal{Q}$ has error $\varepsilon < 2^{-m}$. Hence, there must be a polynomial in the support of $\mathcal{Q}$ that makes no errors, and exactly computes OR on m input bits. This polynomial must have degree at least m, so it follows that $d \geq m$. In particular, we get as desired that $d \geq m \geq \log(1/\varepsilon) - 1$. □

Our probabilistic polynomial construction in Proposition 7.2 above critically relied on the finite field we were working over. We next present another construction with slightly higher degree that works over $\mathbb{Z}$, and hence over any commutative ring R.

Proposition 7.4 ([ABFR94, Lemma 5.1])**.** *For any $\varepsilon \in (0,1)$, the ε-probabilistic degree of* OR *on n inputs over $\mathbb{Z}$ is at most $O(\log(1/\varepsilon) \cdot \log n)$.*

Proof. We first construct a distribution $\mathcal{P}'$ on polynomials $p' : \mathbb{Z}^n \to \mathbb{Z}$ of degree $O(\log n)$ over $\mathbb{Z}$ such that

- $p'(0, 0, \dots, 0) = 0$ for all p' in the support of $\mathcal{P}'$, and

- for all $x \in \{0,1\}^n$ with $\mathsf{OR}(x) = 1$, we have $\Pr_{p' \sim \mathcal{P}'}[p'(x) = 1] \geq 1/2$.

Once we have constructed $\mathcal{P}'$, we can then construct our desired probabilistic polynomial for OR with error ε and degree $O(\log(1/\varepsilon) \cdot \log n)$ by drawing $k := \lceil \log(1/\varepsilon) \rceil$ independent $p'_1, \ldots, p'_k \sim \mathcal{P}'$, and outputting the polynomial $p(x) = 1 - \prod_{\ell=1}^{k}(1 - p'_i(x))$.

We now construct $\mathcal{P}'$. To draw a polynomial p' from $\mathcal{P}'$, we first pick $m := \lceil \log n \rceil + 3$ random subsets $S_0, S_1, \ldots, S_{m-1} \subseteq [n]$ as follows: set $S_0 = [n]$, then for each ℓ from 1 up to $m-1$, let S_ℓ be a uniformly random subset of $S_{\ell-1}$, which includes each element independently with probability 1/2. We then output the polynomial

$$p'(x) = 1 - \prod_{\ell=0}^{m-1} \left(1 - \sum_{i \in S_\ell} x_i\right),$$

which has degree $m \leq O(\log n)$.

For a given $x \in \{0,1\}^n$, if $\mathsf{OR}(x) = 0$, then $x_i = 0$ for all $i \in [n]$ and we will always have $p'(x) = 0$. Otherwise, if $\mathsf{OR}(x) = 1$, then for each $\ell \in \{0, 1, \ldots, m-1\}$, define the random variable $s_\ell(x) := \sum_{i \in S_\ell} x_i$. Since $\mathsf{OR}(x) = 1$, we have that $s_0(x) \geq 1$. The $s_\ell(x)$ form a sequence of nonnegative integers with $s_\ell(x) \leq s_{\ell-1}(x)$ for all $\ell \in [m-1]$. Moreover, each $s_\ell(x)$ is distributed as the sum of $s_{\ell-1}(x)$ independent random values from $\{0,1\}$. If there is any $\ell \in \{0, 1, \ldots, m-1\}$ such that $s_\ell(x) = 1$, then p' will output 1. Notice that exactly one of the following must be the case:

- $s_\ell(x) > 1$ for all $\ell \in \{0, 1, \ldots, m-1\}$. Since $s_0(x) \leq n$, we can apply a union bound to see that this occurs with probability at most $n \cdot 2^{-(m-1)} \leq n \cdot 2^{-\log(n)-2} \leq 1/4$.

- $s_0(x) = 1$.

- There is some $\ell \in [m-1]$ such that $s_{\ell-1}(x) > 1$ and $s_\ell(x) \leq 1$. Note that, given the value $s_{\ell-1}(x)$, we have that $s_\ell(x) = 0$ with probability $2^{-s_{\ell-1}(x)}$, and $s_\ell(x) = 1$ with probability $s_{\ell-1}(x) \cdot 2^{-s_{\ell-1}(x)}$. Hence, conditioned on $s_\ell(x) \leq 1$, we have that $s_\ell(x) = 1$ with probability $s_{\ell-1}(x)/(s_{\ell-1}(x)+1) \geq 2/3$.

Since one of the latter two cases occurs with probability at least 3/4, and in each of those cases, there is an $\ell \in \{0, 1, \ldots, m-1\}$ such that $s_\ell(x) = 1$ with probability at least 2/3, it follows that p' will output 1 with probability at least $\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$, as desired. □

7.3.2 The Probabilistic Degree of Biased Threshold Functions

In the previous subsection, we saw that the OR function has $(1-\varepsilon)$-correct degree 0, and ε-probabilistic degree $O(\log 1/\varepsilon)$ over, say, $\mathbb{F}_2$. The difference between these two degrees can grow unboundedly as $\varepsilon \to 0$, but it is only a constant when ε is a constant. In this subsection, we present a different Boolean function for which the two degrees differ by an unbounded amount even in the constant ε regime.

Let $\mathrm{TH}_{3/4} : \{0,1\}^n \to \{0,1\}$ be the Boolean function

$$\mathrm{TH}_{3/4}(x) = \begin{cases} 1 & \text{if } |x| \geq 3n/4, \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 7.5. *For sufficiently large n, the function $\mathrm{TH}_{3/4}$ has $(2/3)$-correct degree 0.*

Proof. The polynomial $p(x) = 0$ is correct on every input $x \in \{0,1\}^n$ with $|x| \leq 3n/4$. The number of inputs it gets wrong is at most (using the bound from Proposition 2.3 from the Preliminaries): $\sum_{i=0}^{n/4} \binom{n}{i} \leq 2^{H(1/4)\cdot n - o(n)} \leq 2^{0.82\cdot n + o(n)}$, which is less than $\frac{1}{3}2^n$ for big enough n. □

Proposition 7.6. *The function $\mathrm{TH}_{3/4}$ on n inputs has $(1/3)$-probabilistic degree $\Omega(\sqrt{n})$.*

Proof. When half of its inputs are restricted to 0, the function $\mathrm{TH}_{3/4}$ on n inputs becomes the function MAJ on $n/2$ inputs. Similar to the proof of Proposition 7.3, it follows that the $(1/3)$-probabilistic degree of $\mathrm{TH}_{3/4}$ on n is lower bounded by the $(1/3)$-probabilistic degree of MAJ on $n/2$ inputs. But, by Theorem 7.1 (and the fact that the $(1-\varepsilon)$-correct degree is a lower bound on the ε-probabilistic degree) this is $\Omega(\sqrt{n})$, as desired. □

7.4 Probabilistic Polynomials for Majority

In the previous section, we saw examples of Boolean functions whose $(1-\varepsilon)$-correct degree was much lower than their ε-probabilistic degree. We saw in Section 7.2 that MAJ on n inputs has ε-typically correct degree $\Theta(\sqrt{n \log(1/\varepsilon)})$ over any commutative ring R. This raises the question: does MAJ on n inputs have ε-probabilistic degree $O(\sqrt{n \log(1/\varepsilon)})$ as well?

In this section, we show that the answer is *yes*, by giving a new probabilistic polynomial construction for MAJ. As we will see in the next Chapter, this probabilistic polynomial construction will be crucial in a number of our applications, including to nearest neighbor search algorithms. The previous best construction, by Srinivasan [Sri13], achieved degree $O(\sqrt{n \log(1/\varepsilon)} \cdot \mathrm{polylog}(n))$; we will see that the extra $\mathrm{polylog}(n)$ factor would have been prohibitive in most of our applications (see Remark 8.1 in the next Chapter for more details).

Theorem 7.2. *There is a probabilistic polynomial over $\mathbb{Z}$ for MAJ on n variables with error ε and degree $d(n,\varepsilon) = O(\sqrt{n \log(1/\varepsilon)})$. Furthermore, a polynomial can be sampled from the probabilistic polynomial distribution in $\tilde{O}(\sum_{i=0}^{d(n,\varepsilon)} \binom{n}{i})$ time.*

Notation. For $\theta \in [0,1]$, define $\mathrm{TH}_\theta : \{0,1\}^n \to \{0,1\}$ to be the *threshold function* $\mathrm{TH}_\theta(x_1, \ldots, x_n) := [|x|/n \geq \theta]$. In particular, $\mathrm{TH}_{1/2} = \mathsf{MAJ}$. We also define

$\mathrm{NEAR}_{\theta,\delta} : \{0,1\}^n \to \{0,1\}$, such that $\mathrm{NEAR}_{\theta,\delta}(x) := [|x|/n \in [\theta-\delta, \theta+\delta]]$. Intuitively, $\mathrm{NEAR}_{\theta,\delta}$ checks whether $|x|/n$ is "near" θ, with error δ.

In the remainder of this Section, we prove Theorem 7.2. To do so, we construct a probabilistic polynomial for TH_θ over $\mathbb{Z}[x_1, \ldots, x_n]$ which has degree $O(\sqrt{n \log(1/\epsilon)})$ and on each input is correct with probability at least $1-\epsilon$.

Intuition for the construction. First, let us suppose $|x|/n$ is not too close to θ: in particular $|x|/n$ is not within $\delta = O(\sqrt{\log(1/\epsilon)/n})$ of θ. Then, if we construct a new smaller vector $\tilde{x}$ by sampling $1/10$ of the entries of x, it is likely that $|\tilde{x}|/(n/10)$ lies on the same side of θ as $|x|/n$. This suggests a *recursive* strategy: we can use our polynomial construction on the sample $\tilde{x}$. Second, if $|x|/n$ is close to θ, then by interpolating, we can use an exact polynomial of degree $O(\sqrt{n \log(1/\epsilon)})$ (which we call $A_{n,\theta,g}$) that is guaranteed to give the correct answer. To decide which of the two cases we are in, we will use a probabilistic polynomial for NEAR (on a smaller number of variables), which can itself be written as the product of two probabilistic polynomials for TH. The degree incurred by recursive calls can be adjusted to have tiny overhead, with the right parameters.

In comparison, Srinivasan [Sri13] takes a number theoretic approach. For $\Omega(\log n)$ different primes p, his polynomial uses $p-1$ probabilistic polynomials in order to determine the Hamming weight of the input $\pmod p$. Then, it uses an exact polynomial inspired by the Chinese Remainder Theorem to determine the true Hamming weight of the input, and whether it is at least $n/2$. This approach works on a more general class of functions than ours, called W-sum determined, which are determined by a weighted sum of the input coordinates. However, the number of primes being considered inherently means that this type of approach will incur extra logarithmic degree increases. In fact, we also give a better probabilistic degree for every symmetric function.

Probabilistic Polynomial Definition. Let n be an integer for which we want to compute TH_θ. Let $A_{n,\theta,g} : \{0,1\}^n \to \mathbb{Z}$ be an exact polynomial with integer coefficients of degree at most $2g\sqrt{n}+1$ which gives the correct answer to TH_θ for any vector x with $|x| \in [\theta n - g\sqrt{n}, \theta n + g\sqrt{n}]$, and can give arbitrary answers to other vectors. Such a polynomial $A_{n,\theta,g}$ exists by Lemma 7.1 above.

Let $M_{m,\theta,\epsilon} : \{0,1\}^m \to \mathbb{Z}$ denote the probabilistic polynomial for TH_θ with error $\leq \epsilon$ degree as described above for all $m < n$. We can assume as a base case that when m is constant, we simply use the exact polynomial for TH_θ.

Define

$$S_{m,\theta,\delta,\epsilon}(x) := (1 - M_{m,\theta+\delta,\epsilon}(x)) \cdot M_{m,\theta-\delta,\epsilon}(x).$$

Assuming $M_{n,\theta,\epsilon}$ works as prescribed (with $\leq \epsilon$ error), this is a probabilistic polynomial for $\mathrm{NEAR}_{\theta,\delta}$ with error at most 2ϵ. For $x \in \{0,1\}^n$, let $\tilde{x} \in \{0,1\}^{n/10}$ be a vector of length $n/10$, where each entry is an independent and uniformly random entry of x. Hence, each entry of $\tilde{x}$ is a probabilistic polynomial in x of degree 1. Let $a = \sqrt{10} \cdot \sqrt{\ln(1/\epsilon)}$. Our probabilistic polynomial for TH_θ on n variables is defined to be:

$$M_{n,\theta,\epsilon}(x) := A_{n,\theta,2a}(x) \cdot S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) + M_{n/10,\theta,\epsilon/4}(\tilde{x}) \cdot (1 - S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x})).$$

Note that $\tilde{x}$ denotes the *same* randomly chosen vector in each of its appearances, and $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}$ denotes the same draw from the random polynomial distribution in both of its appearances.

Degree of $M_{n,\theta,\varepsilon}$. First we show by induction on n that $M_{n,\theta,\varepsilon}$ has degree $\leq 41\sqrt{n \ln(1/\epsilon)}$. Assume that $M_{m,\theta,\epsilon}$ has degree $\leq 41\sqrt{m \ln(1/\epsilon)}$ for all $m < n$. The degree of $M_{n,\theta,\epsilon}$ is thus equal to

$$\begin{aligned}
&\max\left\{\deg\left[A_{n,\theta,2a}(x) \cdot S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x})\right], \deg\left[M_{n/10,\theta,\epsilon/4}(\tilde{x}) \cdot (1 - S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}))\right]\right\} \\
&= \deg(S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x})) + \max\{\deg(A_{n,\theta,2a}(x)), \deg(M_{n/10,\theta,\epsilon/4}(\tilde{x}))\} \\
&\leq 2 \cdot 41\sqrt{\frac{n}{10}\ln(4/\epsilon)} + \max\left\{4a\sqrt{n}, 41\sqrt{\frac{n}{10}\ln(4/\epsilon)}\right\} \\
&= 2 \cdot 41\sqrt{\frac{n}{10}\ln(4/\epsilon)} + \max\left\{4 \cdot (\sqrt{10}\sqrt{\ln(1/\epsilon)}) \cdot \sqrt{n}, 41\sqrt{\frac{n}{10}\ln(4/\epsilon)}\right\} \\
&= 3 \cdot 41\sqrt{\frac{n}{10}\ln(4/\epsilon)} \leq 41\sqrt{n \ln(1/\epsilon)}.
\end{aligned}$$

Time to compute $M_{n,\theta,\varepsilon}$. Computing $A_{n,\theta,2a}$ can be done in poly(n) time as described in Lemma 7.1, as can sampling $\tilde{x}$ from x. Given the three recursive $M_{n/10,\theta',\varepsilon/4}$ polynomials, we can then compute $M_{n,\theta,\varepsilon}$ in three multiplications. Each recursive polynomial has degree at most $d(n/10, \varepsilon/4)$, and hence at most $\sum_{i=0}^{d(n/10,\varepsilon/4)} \binom{n}{i}$ monomials. Since the time for these multiplications dominates the time for the recursive computations, the total time is $\tilde{O}(\sum_{i=0}^{d(n,\varepsilon)} \binom{n}{i})$ using the fast Fourier transform[1], as desired.

Correctness. Now we prove that $M_{n,\theta,\varepsilon}$ correctly simulates TH_θ with probability at least $1 - \varepsilon$, on all possible inputs. We begin by citing a lemma explaining our choice of the parameter a.

Lemma 7.4. *If $x \in \{0,1\}^n$ with $|x|/n = w$, and $\tilde{x} \in \{0,1\}^{n/10}$ is a vector each of whose entries is an independent and uniformly random entry of x, with $|\tilde{x}|/(n/10) = v$, then for every $\varepsilon < 1/4$,*

$$\Pr\left[v \leq w - a/\sqrt{n}\right] \leq \frac{\varepsilon}{4},$$

where $a = \sqrt{10} \cdot \sqrt{\ln(1/\epsilon)}$.

[1] By replacing each variable with increasing powers of a single variable, we can reduce multivariate polynomial multiplication to single variable polynomial multiplication.

Proof. Each entry of $\tilde{x}$ is drawn from a binomial distribution with probability w of giving a 1. Hence, applying Hoeffding's inequality, Lemma 2.1 from the Preliminaries, with $p = w$, $m = n/10$, and $k = \frac{n}{10}(w - a/\sqrt{n}) = \frac{nw}{10} - \frac{a\sqrt{n}}{10}$ yields:

$$\Pr[v \leq w - a/\sqrt{n}] = \Pr\left[|\tilde{x}| \leq \frac{nw}{10} - \frac{a\sqrt{n}}{10}\right] \leq \exp\left(-2\frac{\left(\frac{a\sqrt{n}}{10}\right)^2}{\frac{n}{10}}\right),$$

which simplifies to $\exp\left(-\frac{a^2}{5}\right) = \exp(-2\ln(1/\epsilon)) = \epsilon^2 < \frac{\varepsilon}{4}$. □

We now move on to the main proof of correctness, which proceeds by induction on n. By symmetry, we may assume we have an input vector $x \in \{0,1\}^n$ with $|x|/n \geq \theta$, and we want to show that $M_{n,\theta,\epsilon}(x)$ outputs 1 with probability at least $1-\epsilon$. We assume $\epsilon < 1/4$ so that we may apply Lemma 7.4.

For notational convenience, define the intervals:

$$\alpha_0 = [\theta - a/\sqrt{n}, \theta],\ \alpha_1 = [\theta, \theta + a/\sqrt{n}],\ \beta = [\theta + a/\sqrt{n}, \theta + 2a/\sqrt{n}],\ \gamma = [\theta + 2a/\sqrt{n}, 1].$$

Note that depending on the values of θ and a, some of these intervals may be empty; this is not a problem for our proof.

Let $w = |x|/n$. Let $\tilde{x}$ be the random "subvector" of x selected in $M_{n,\theta,\epsilon}$ (recall we use the same $\tilde{x}$ in each of the three locations it appears in the definition of M). Let $v = |\tilde{x}|/(n/10)$. Our proof strategy is to consider different cases depending on the value of w. For each case, we show there are at most four events such that, if all events hold then $M_{n,\theta,\epsilon}$ outputs the correct answer, and each event does not hold with probability at most $\frac{\varepsilon}{4}$. By the union bound, this implies that $M_{n,\theta,\epsilon}$ gives the correct answer with probability at least $1-\epsilon$. The cases are as follows:

1. $w \in \alpha_1$ **(**$|x|/n$ **is "very close" to** θ**).** By Lemma 7.4, we know that with probability at least $1 - \frac{\epsilon}{4}$, we have $v \geq \theta - a/\sqrt{n}$. In other words, $v \in \alpha_0 \cup \alpha_1 \cup \beta \cup \gamma$.

   - $v \in \alpha_0 \cup \alpha_1$, then with probability at least $1 - \frac{2\epsilon}{4}$, we have $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) = 1$, by our inductive assumption that $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}$ is a probabilistic polynomial for $\mathrm{NEAR}_{\theta,a/\sqrt{n}}$ with error probability at most $\frac{2\epsilon}{4}$. In this case, $M_{n,\theta,\epsilon}(x) = A_{n,\theta,2a}(x)$, which is 1 by definition of A.
   - $v \in \beta \cup \gamma$, then with probability at least $1 - \frac{2\epsilon}{4}$, we have $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) = 0$, in which case $M_{n,\theta,\epsilon}(x) = M_{n/10,\theta,\epsilon/4}(\tilde{x})$. But, by the inductive hypothesis, this is 1 with probability at least $1 - \frac{\epsilon}{4}$, since $v > \theta$ in this case.

   Since we are in one of these two cases with probability $\geq 1 - \frac{1}{4}\epsilon$, and each gives the correct answer with probability $\geq 1 - \frac{3\epsilon}{4}$, the correct answer is given in this case with probability $\geq 1 - \epsilon$.

2. $w \in \beta$ **(**$|x|/n$ **is "close" to** θ**).** In this case we have $w - \theta \leq 2a/\sqrt{n}$, therefore $A_{n,\theta,2a}(x) = 1$. Hence, if $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) = 1$ then $M_{n,\theta,\epsilon}(x)$ returns the correct

answer. If $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) = 0$, then we return $M_{n/10,\theta,\epsilon/4}(\tilde{x})$. By Lemma 7.4, we have $v \geq \theta$ with probability at least $1 - \frac{\epsilon}{4}$, and in this case, $M_{n/10,\theta,\epsilon/4}(\tilde{x}) = 1$ with probability $\geq 1 - \frac{\epsilon}{4}$. Hence, M returns the correct value with probability at least $1 - \frac{2\epsilon}{4}$, no matter what the value of $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(y)$ happens to be.

3. $w \in \gamma$ **($|x|/n$ is "far" from θ).** By Lemma 7.4, we have $v \in \beta \cup \gamma$ with probability at least $1 - \frac{\epsilon}{4}$. In this case, $v \geq \theta$, and so $M_{n/10,\theta,\epsilon/4}(\tilde{x}) = 1$ with probability $\geq 1 - \frac{\epsilon}{4}$. Moreover, since $v \notin \alpha_0 \cup \alpha_1$, it follows that $S_{n/10,\theta,a/\sqrt{n},\epsilon/4}(\tilde{x}) = 0$ with probability $\geq 1 - \frac{2}{4}\epsilon$, in which case $M_{n,\theta,\epsilon}(x) = M_{n/10,\theta,\epsilon/4}(\tilde{x})$. Overall, $M_{n,\theta,\epsilon}(x) = M_{n/10,\theta,\epsilon/4}(\tilde{x}) = 1$ with probability $\geq 1 - \epsilon$.

This completes the proof of correctness, and the proof of Theorem 7.2.

7.5 Further Probabilistic Polynomial Constructions

In this Section, we give two additional constructions of probabilistic polynomials which strengthen Theorem 7.2 from the previous section. Both of them make use of the following observation about the proof of correctness of Theorem 7.2: The only randomness used by the construction is the sampled vector $\tilde{x}$ at each layer of the recursion, and moreover, the polynomial will always give a correct answer as long as $||x|/n - |\tilde{x}|/(n/10)| < a/\sqrt{n}$ at each layer of recursion. This condition is true for the probabilistic polynomial for TH_θ no matter what $\theta \in [0,1]$ is.

7.5.1 Symmetric Functions

Recall that the Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *symmetric* if the value of $f(x)$ depends only on $|x|$, the Hamming weight of x. We now describe how to use the probabilistic polynomial for TH_θ to derive a probabilistic polynomial for any symmetric function with the same degree as TH_θ:

Theorem 7.3. *Every symmetric function $f : \{0,1\}^n \to \{0,1\}$ on n variables has a probabilistic polynomial of $O(\sqrt{n \log(1/\epsilon)})$ degree and error ϵ over $\mathbb{Z}$.*

Proof. For every $0 \leq i \leq n$, let $f_i \in \{0,1\}$ denote the value of $f(x)$ when x has Hamming weight i. Define:

$$A = \{0 < i \leq n \mid f_i = 1 \text{ and } f_{i-1} = 0\},$$

$$B = \{0 < i \leq n \mid f_i = 0 \text{ and } f_{i-1} = 1\}.$$

Then, f can be written exactly as:

$$f(x) = f_0 + \sum_{i \in B} \mathrm{TH}_{i/n}(x) - \sum_{j \in A} \mathrm{TH}_{j/n}(x). \tag{7.2}$$

We replace each TH_θ in (7.2) with a probabilistic polynomial of Theorem 7.2 with error ϵ. However, we make sure that in all of the different probabilistic polynomials for

TH_θ, we make the same choice for the sampled vector $\tilde{x}$ at each layer of recursion. We can then apply the proof of Theorem 7.2, to see that every one of the TH_θ probabilistic polynomials will give the correct answer as long as $||x|/n - |\tilde{x}|/(n/10)| < a/\sqrt{n}$ at each of the $\log_{10}(n)$ layers of recursion (this is a property only of the sampling, and independent of θ). Just as in the original proof, this will happen with error at most ε, as desired. □

7.5.2 Derandomization

The polynomial for TH_θ in Theorem 7.2 used $\Theta(n)$ random bits in order to randomly sample $\tilde{x}$ from x at each layer of recursion. In this subsection, we show it can be implemented using only $\mathrm{polylog}(n, s)$ random bits. The key will be to sample $\tilde{x}$ with *limited independence*, combined with a Chernoff bound for bits with limited independence (Lemma 2.2 from the Preliminaries). In particular, we use the following strengthening of Lemma 7.4 from the previous Section:

Lemma 7.5. *If $x \in \{0,1\}^n$ with $|x|/n = w$, and $\tilde{x} \in \{0,1\}^{n/10}$ is a vector each of whose entries is k-wise independently chosen entry of x, where $k = \lfloor 20e^{-1/3}\log(1/\epsilon)\rfloor$, with $|\tilde{x}|/(n/10) = v$, then for every $\varepsilon < 1/4$,*

$$\Pr\left[v \le w - \frac{a}{\sqrt{n}}\right] \le \frac{\varepsilon}{4},$$

where $a = \sqrt{10}\cdot\sqrt{\ln(1/\epsilon)}$.

Proof. Apply Lemma 2.2 with $X = |\tilde{x}|$, $\mu = \mathbb{E}[|\tilde{x}|] = wn$, and $\delta = \sqrt{40\log(1/\epsilon)/n}$. □

Theorem 7.4. *For any $0 \le \theta \le 1$, there is a probabilistic polynomial for the threshold function TH_θ of degree $O(\sqrt{n\log 1/\varepsilon})$ on n bits with error ε that can be randomly sampled using $O(\log(n)\log(n/\varepsilon))$ random bits.*

Proof. We follow the construction of Theorem 7.2 exactly, with only one modification. In the original proof, $\tilde{x}$ was a sample of $n/10$ bits of x, chosen independently at random. Here, we instead pick $\tilde{x}$ to be a sample of $n/10$ bits chosen k-wise independently, for $k = \lfloor 20e^{-1/3}\log(1/\epsilon)\rfloor$.

The only requirement of the randomness in the proof of Theorem 7.2 is that it satisfies Lemma 7.4, a concentration inequality for sampling $\tilde{x}$ from x. Our new Lemma 7.5 is identical to Lemma 7.4, except that it replaces the old method of sampling $\tilde{x}$ with new k-wise sampling; the remainder of the proof of correctness is exactly as before.

Recall that in our recursive polynomial construction, we divide n by 10 and divide ϵ by 4 each time we move from one recursive layer to the next. At the jth recursive level of our construction, for $1 \le j < \log_{10}(n)$, we thus need to $O(\log(4^j/\epsilon))$-wise independently sample $n/10^j$ entries from a vector of length $n/10^{j-1}$. Summing across all of the layers, we need a total of $O(n)$ samples from a k-wise independent space, where k is never more than $O(\log(n/\epsilon))$. This can be done all together using $O(n)$ samples from $\{1, 2, \ldots, n\}$ which are $O(\log(n/\epsilon))$-wise independent. Using standard constructions[2], this requires $O(\log(n)\log(n/\epsilon))$ random bits. $\square$

7.6 PTFs for ORs of Threshold Functions

The primary way we will apply our probabilistic polynomial for the threshold function TH_θ in the next Chapter is to efficiently compute ORs of thresholds. Suppose any Boolean function $f : \{0,1\}^n \to \{0,1\}$ has a probabilistic polynomial $\mathcal{P}$ of degree d and error ε over any commutative ring R. Then, for any $s \in \mathbb{N}$, we can construct a probabilistic polynomial for $\mathsf{OR}_s \circ f$, the OR of s independent copies of f, of degree $s \cdot d$ and error at most $s \cdot \varepsilon$: On input $x^{(1)}, x^{(2)}, \ldots, x^{(s)} \in \{0,1\}^n$, we compute $\bigvee_{i \in [s]} f(x^{(i)})$ by drawing a $p \sim \mathcal{P}$, letting $q : R^s \to R$ be the exact polynomial for OR_s of degree s, and outputting $q(p(x^{(1)}), p(x^{(2)}), \ldots, p(x^{(s)}))$. Indeed, by a union bound, the polynomial $p(x^{(i)})$ for each $i \in [s]$ will give the correct answer with error at most $s \cdot \varepsilon$, and so q will exactly compute the OR.

When R has characteristic 0 (say, for instance, $R = \mathbb{R}$), we can do even better: we can construct a 'probabilistic polynomial' for $\mathsf{OR}_s \circ f$ of degree only d and error at most $s \cdot \varepsilon$, by outputting $p(x^{(1)}) + p(x^{(2)}) + \cdots + p(x^{(s)}) - 1$. With error at most $s \cdot \varepsilon$, this polynomial will output -1 when the $\mathsf{OR}_s \circ f$ is false, and a nonnegative value when it is true. This is not a probabilistic polynomial for $\mathsf{OR}_s \circ f$ as we defined it earlier, since it only outputs a nonnegative value in the 'true' case, rather than necessarily outputting 1. However, this 'thresholding' behavior between true and false values still allows us to determine whether the $\mathsf{OR}_s \circ f$ was true or false.

Inspired by this remark, in this Section, we show how to construct low-degree polynomial threshold functions (PTFs) representing TH_θ that have good threshold behavior, and consequently obtain low-degree PTFs for an OR of many threshold functions. By only aiming for such thresholding behavior, we will be able to further decrease the degree of our polynomial representations of TH_θ.

Definition 7.4. *A* polynomial threshold function *(PTF) for the Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a polynomial $p : \{0,1\}^n \to \mathbb{R}$ such that, for every $x \in \{0,1\}^n$, we have $p(x) \geq 0$ if $f(x) = 1$, and $p(x) < 0$ if $f(x) = 0$. The PTF degree of f is the minimum degree of a PTF for f.*

Example 7.2. *The function TH_θ has PTF degree 1, via the PTF $p(x_1, \ldots, x_n) = x_1 + \cdots + x_n - \theta \cdot n$. However, it is unclear how to use this to construct a PTF for $\mathsf{OR}_s \circ \mathrm{TH}_\theta$; we will need a PTF with better thresholding behavior for TH_θ for this.*

[2] For example, one can pick a uniformly random degree k (single-variable) polynomial over $\mathbb{F}_q$ for some prime power $q \geq n$, and output its values on n distinct points from $\mathbb{F}_q$.

7.6.1 Deterministic Construction

We begin by reviewing some basic facts about Chebyshev polynomials. The *degree-q Chebyshev polynomial of the first kind* is

$$T_q(x) := \sum_{i=0}^{\lfloor q/2 \rfloor} \binom{q}{2i} (x^2-1)^i x^{q-2i}.$$

Fact 7.1. *For any $\varepsilon \in (0,1)$,*

- *if $x \in [-1,1]$, then $|T_q(x)| \leq 1$;*
- *if $x \in (1, 1+\varepsilon)$, then $T_q(x) > 1$;*
- *if $x \geq 1+\varepsilon$, then $T_q(x) \geq \frac{1}{2} e^{q\sqrt{\varepsilon}}$.*

Proof. The first property easily follows from the known formula $T_q(x) = \cos(q \arccos(x))$ for $x \in [-1,1]$. The second and third properties follow from another known formula $T_q(x) = \cosh(q \operatorname{arcosh}(x))$ for $x > 1$, which for $x \geq 1+\varepsilon$ implies $T_q(x) \geq \cosh(q\sqrt{\varepsilon}) = \frac{1}{2}(e^{q\sqrt{\varepsilon}} + e^{-q\sqrt{\varepsilon}})$. □

In certain scenarios, we obtain slightly better results using a (lesser known) family of *discrete Chebyshev polynomials* defined as follows [Hir03, page 59]:

$$D_{q,t}(x) := \sum_{i=0}^{q} (-1)^i \binom{q}{i} \binom{t-x}{q-i} \binom{x}{i}.$$

(See also [Sze75, pages 33–34] or Chebyshev's original paper [Che99] with an essentially equivalent definition up to rescaling.)

Fact 7.2. *Let $c_{q,t} = (t+1)^{q+1}/q!$. For all $t > q \geq \sqrt{8(t+1)\ln(t+1)}$,*

- *if $x \in \{0,1,\ldots,t\}$, then $|D_{q,t}(x)| \leq c_{q,t}$;*
- *if $x \leq -1$, then $D_{q,t}(x) \geq e^{q^2/(8(t+1))} c_{q,t}$.*

Proof. From [Hir03, page 61],

$$\begin{aligned}
\sum_{k=0}^{t} D_{q,t}(k)^2 &= \binom{2q}{q}\binom{t+1+q}{2q+1} \\
&= \frac{2q(2q-1)\cdots q}{q(q-1)\cdots 1} \cdot \frac{(t+1+q)(t+q)\cdots(t+1-q)}{(2q+1)(2q)\cdots 1} \\
&= \frac{(t+1)((t+1)^2-1^2)((t+1)^2-2^2)\cdots((t+1)^2-q^2)}{(2q+1)(q!)^2} \\
&\leq \frac{(t+1)^{2q+2}}{(q!)^2}.
\end{aligned}$$

Thus, for every integer $x \in [0,t]$, we have $|D_{q,t}(x)| \leq (t+1)^{q+1}/q! = c_{q,t}$.

For $x \leq -1$, we have $(-1)^i\binom{x}{i} = \frac{(-x)(-x+1)\cdots(-x+i-1)}{1\cdot 2\cdots i} \geq 1$, and by the Chu–Vandermonde identity,

$$\begin{aligned} D_{q,t}(x) &\geq \sum_{i=0}^{q}\binom{q}{i}\binom{t+1}{q-i} = \binom{t+1+q}{q} \\ &= \frac{(t+1)^q(1+\frac{1}{t+1})(1+\frac{2}{t+1})\cdots(1+\frac{q}{t+1})}{q!} \\ &\geq \frac{c_{q,t}}{t+1}e^{\frac{1+2+\cdots+q}{2(t+1)}} = e^{q(q+1)/(4(t+1))-\ln(t+1)}c_{q,t} \geq e^{q^2/(8(t+1))}c_{q,t}. \end{aligned}$$

□

Using the Chebyshev polynomials and the Discrete Chebyshev polynomials, we can design PTFs for TH_θ with 'nice' thresholding behavior. Our construction also extends to 'approximate threshold functions', where there is a middle range of input values between the 'true' and 'false' inputs where we are allowed to output any value. Our construction involves two parameters: s, which corresponds to the number of different copies of the threshold function whose OR we want to take, and ε, the 'gap' in the approximate threshold.

Theorem 7.5. *We can construct a polynomial $P_{s,t,\varepsilon} : \mathbb{R} \to \mathbb{R}$ of degree $O(\sqrt{1/\varepsilon}\log s)$, such that*

- *if $x \in \{0, 1, \dots, t\}$, then $|P_{s,t,\varepsilon}(x)| \leq 1$;*
- *if $x \in (t, (1+\varepsilon)t)$, then $P_{s,t,\varepsilon}(x) > 1$;*
- *if $x \geq (1+\varepsilon)t$, then $P_{s,t,\varepsilon}(x) \geq s$.*

For the "exact" setting with $\varepsilon = 1/t$, we can alternatively bound the degree by $O(\sqrt{t\log(st)})$.

Proof. Set $P_{s,t,\varepsilon}(x) := T_q(x/t)$, where T_q is the Chebyshev polynomial, for a parameter q to be determined. The first two properties are obvious from Fact 7.1. On the other hand, if $x \geq (1+\varepsilon)t$, then Fact 7.1 shows that $P_{s,t,\varepsilon}(x) \geq \frac{1}{2}e^{q\sqrt{\varepsilon}} \geq s$, provided we set $q = \left\lceil\sqrt{1/\varepsilon}\ln(2s)\right\rceil$. This achieves $O(\sqrt{1/\varepsilon}\log s)$ degree.

When $\varepsilon = 1/t$ the above yields $O(\sqrt{t}\log s)$ degree; we can reduce the $\log s$ factor by instead defining $P_{s,t,\varepsilon}(x) := D_{q,t}(t-x)/c_{q,t}$. Now, if $x \geq t+1$, then $P_{s,t,\varepsilon}(x) \geq e^{q^2/(8(t+1))} \geq s$ by setting $q = \left\lceil\sqrt{8(t+1)\ln(\max\{s, t+1\})}\right\rceil$. □

Using Theorem 7.5, we can construct a low-degree PTF for computing an OR of s thresholds of n bits:

Corollary 7.3. *Given n, s, t, ε, we can construct a polynomial $P : \{0,1\}^{ns} \to \mathbb{R}$ of degree at most $\Delta := O(\sqrt{1/\varepsilon}\log s)$ and at most $s \cdot \binom{n}{\Delta}$ monomials, such that*

- *if $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{n} x_{ij} > t\right]$ is false, then $|P(x_{11}, \dots, x_{1n}, \dots, x_{s1}, \dots, x_{sn})| \leq s$;*

- *if* $\bigvee_{i=1}^{s}\left[\sum_{j=1}^{n} x_{ij} \geq t + \varepsilon n\right]$ *is true, then* $P(x_{11}, \ldots, x_{1n}, \ldots, x_{s1}, \ldots, x_{sn}) > 2s$.

For the exact setting with $\varepsilon = 1/n$, *we can alternatively bound* Δ *by* $O(\sqrt{n \log(ns)})$.

Proof. Define $P(x_{11}, \ldots, x_{1n}, \ldots, x_{s1}, \ldots, x_{sn}) := \sum_{i=1}^{s} P_{n,3s,t,\varepsilon}\left(\sum_{j=1}^{n} x_{ij}\right)$, where $P_{n,3s,t,\varepsilon}$ is from Theorem 7.5. It is not hard to see that the stated properties hold. (In the second case, the output is at least $3s - (s-1) > 2s$.) □

7.6.2 Probabilistic Construction

Finally, we give our last polynomial construction, by combining the PTF from Theorem 7.5 from the previous Subsection with our probabilistic polynomial for TH_θ from Theorem 7.2 above. We will construct a distribution of PTFs to randomly draw from, which will allow us to achieve noticeably lower degree than either the PTFs from the previous section, or the probabilistic polynomial from before.

Definition 7.5. *For any* $\varepsilon \in [0,1]$, *a* probabilistic PTF with error ε and degree d *for the Boolean function* $f : \{0,1\}^n \to \{0,1\}$ *is a distribution* $\mathcal{P}$ *on polynomials* $p : \{0,1\}^n \to \mathbb{R}$ *of degree at most* d *over* $\mathbb{R}$ *such that, for every* $x \in \{0,1\}^n$, *we have*

- *If* $f(x) = 1$, *then* $\Pr_{p\sim\mathcal{P}}[p(x) \geq 0] \geq 1 - \varepsilon$, *and*
- *If* $f(x) = 0$, *then* $\Pr_{p\sim\mathcal{P}}[p(x) < 0] \geq 1 - \varepsilon$.

The ε*-probabilistic PTF degree of* f *is the minimum degree of a probabilistic PTF with error* ε *for* f.

Our construction will use a 'random sampling' approach similar to Theorem 7.2 from before.

Restatement of Theorem 7.2 *We can construct a distribution* $\mathcal{Q}_{n,s,t}$ *on polynomials* $Q_{n,s,t} : \{0,1\}^n \to \mathbb{R}$ *of degree* $O(\sqrt{n \log s})$, *such that for every* $x \in \{0,1\}^n$, *when we draw a random* $Q_{n,s,t} \sim \mathcal{Q}_{n,s,t}$:

- *if* $\sum_{i=1}^{n} x_i \leq t$, *then* $Q_{n,s,t}(x_1, \ldots, x_n) = 0$ *with probability at least* $1 - 1/s$;
- *if* $\sum_{i=1}^{n} x_i > t$, *then* $Q_{n,s,t}(x_1, \ldots, x_n) = 1$ *with probability at least* $1 - 1/s$.

Theorem 7.6. *We can construct a distribution* $\mathcal{L}_{n,s,t,\varepsilon}$ *on polynomials* $L_{n,s,t,\varepsilon} : \{0,1\}^n \to \mathbb{R}$ *of degree* $O((1/\varepsilon)^{1/3} \log s)$, *such that every* $x \in \{0,1\}^n$, *when we draw a random* $L_{n,s,t,\varepsilon} \sim \mathcal{L}_{n,s,t,\varepsilon}$:

- *if* $\sum_{i=1}^{n} x_i \leq t$, *then* $|L_{n,s,t,\varepsilon}(x_1, \ldots, x_n)| \leq 1$ *with probability at least* $1 - 1/s$;
- *if* $\sum_{i=1}^{n} x_i \in (t, t + \varepsilon n)$, *then* $L_{n,s,t,\varepsilon}(x_1, \ldots, x_n) > 1$ *with probability at least* $1 - 1/s$;
- *if* $\sum_{i=1}^{n} x_i \geq t + \varepsilon n$, *then* $L_{n,s,t,\varepsilon}(x_1, \ldots, x_n) \geq s$ *with probability at least* $1 - 1/s$.

For the "exact" setting with $\varepsilon = 1/n$, we can alternatively bound the degree by $O(n^{1/3}\log^{2/3}(ns))$.

Proof. Let r and q be parameters to be set later. Draw a random sample $R \subseteq \{1, \dots, n\}$ of size r. Let

$$t_R := \frac{tr}{n} - c_0\sqrt{r\log s} \quad \text{and} \quad t^- := t - 2c_0\left(\frac{n}{\sqrt{r}}\right)\sqrt{\log s}$$

for a sufficiently large constant c_0. Define

$$L_{n,s,t,\varepsilon}(x_1,\ldots,x_d) \;:=\; Q_{r,2s,t_R}(\{x_i\}_{i\in R}) \;\cdot\; P_{s,t',\varepsilon'}\left(\sum_{i=1}^{n} x_i - t^-\right),$$

where $P_{s,t',\varepsilon'}$ is the polynomial from Theorem 7.5, with $t' := t - t^- = \Theta((n/\sqrt{r})\sqrt{\log s})$ and $\varepsilon' := \varepsilon n/t' = \Theta(\varepsilon\sqrt{r}/\sqrt{\log s})$, and $Q_{r,2s,t_R}$ is a polynomial drawn from $\mathcal{Q}_{r,2s,t_R}$ from Theorem 7.2.

To verify the stated properties, consider three cases:

- Case 1: $\sum_{i=1}^n x_i < t^-$. By a Chernoff bound, with probability at least $1 - 1/(2s)$, we have $\sum_{i\in R} x_i < t^- r/n + c_0\sqrt{r\log s} \le t_R$ (assuming that $r \ge \log s$). Thus, with probability at least $1 - 1/s$, we have $Q_{n,2s,t_R}(\{x_i\}_{i\in R}) = 0$ and so $L_{n,s,t,\varepsilon}(x_1, \dots, x_n) = 0$.

- Case 2: $\sum_{i=1}^n x_i \in [t^-, t]$. With probability at least $1 - 1/s$, we have $Q_{r,2s,t_R}(\{x_i\}_{i\in R}) \in \{0, 1\}$ and so $|L_{n,s,t,\varepsilon}(x_1, \dots, x_n)| \le 1$.

- Case 3: $\sum_{i=1}^n x_i > t$. By a Chernoff bound, with probability at least $1 - 1/(2s)$, we have $\sum_{i\in R} x_i \ge tr/n + c_0\sqrt{r\log s} = t_R$. Thus, with probability at least $1-1/s$, we have $Q_{r,2s,t_R}(\{x_i\}_{i\in R}) = 1$ and so $L_{n,s,t,\varepsilon}(x_1, \dots, x_n) > 1$ for $\sum_{i=1}^n x_i \in (t, t + \varepsilon n)$, or $L_{n,s,t,\varepsilon}(x_1, \dots, x_n) \ge s$ for $\sum_{i=1}^n x_i \ge t + \varepsilon n$.

The degree of $L_{n,s,t,\varepsilon}$ is

$$O\left(\sqrt{r\log s} + \sqrt{(1/(\varepsilon\sqrt{r}))\sqrt{\log s}\log s}\right)$$

and we can set $r = \left\lceil (1/\varepsilon)^{2/3}\log s \right\rceil$. For the exact setting, the degree is

$$O\left(\sqrt{r\log s} + \sqrt{(n/\sqrt{r})\sqrt{\log s}\cdot\log(ns)}\right)$$

and we can set $r = \left\lceil n^{2/3}\log^{1/3}(ns)\right\rceil$. □

Remark 7.1. Using the same techniques as in Theorem 7.4, we can sample a probabilistic polynomial from Theorem 7.6 with only $O(\log(n)\log(ns))$ random bits.

Finally, we can construct a probabilistic PTF for an OR of thresholds:

Corollary 7.4. *Given n, s, t, ε, we can construct a distribution $\mathcal{L}$ on polynomials $L : \{0,1\}^{ns} \to \mathbb{R}$ of degree at most $\Delta := O((1/\varepsilon)^{1/3} \log s)$ with at most $s \cdot \binom{n}{\Delta}$ monomials, such that*

- *if $\bigvee_{i=1}^{s} \left[\sum_{j=1}^{n} x_{ij} \geq t\right]$ is false, then $|L(x_{11}, \ldots, x_{1n}, \ldots, x_{s1}, \ldots, x_{sn})| \leq s$ with probability at least $2/3$;*
- *if $\bigvee_{i=1}^{s} \left[\sum_{j=1}^{d} x_{ij} \geq t + \varepsilon n\right]$ is true, then $L(x_{11}, \ldots, x_{1n}, \ldots, x_{s1}, \ldots, x_{sn}) > 2s$ with probability at least $2/3$.*

For the exact setting with $\varepsilon = 1/n$, we can alternatively bound Δ by $O(n^{1/3} \log^{2/3}(ns))$.

Proof. Draw $L_{n,3s,t,\varepsilon} \sim \mathcal{L}_{n,3s,t,\varepsilon}$ from the distribution from Theorem 7.6, then define $L(x_{11}, \ldots, x_{1n}, \ldots, x_{s1}, \ldots, x_{sn}) := \sum_{i=1}^{s} L_{n,3s,t,\varepsilon}(x_{i1}, \ldots, x_{in})$. □

Remark 7.2. The coefficients of the polynomials from Theorem 7.2 are poly(n)-bit integers, and it can be checked that the coefficients of all our deterministic and probabilistic PTFs are rational numbers with poly(n)-bit numerators and a common poly(n)-bit denominator, and that the same bound for the number of monomials holds for the construction time, up to poly(n) factors. That is, computations with these polynomials have low computational overhead relative to n.

Chapter 8

Algorithmic Applications

8.1 Sparse Polynomials and Rectangular Matrix Multiplication

In this Chapter, we give new algorithmic applications of the polynomials we constructed in the previous chapter. Because of the prevalence of threshold functions throughout algorithms and complexity, we will be able to apply them in a variety of settings: nearest neighbor search, high-dimensional computational geometry, the Light Bulb problem from data science, MAX-SAT, circuit SAT, and threshold circuit lower bounds.

A key insight in many of the applications is a way to quickly evaluate a sparse polynomial (i.e. a polynomial with few monomials) on a combinatorial rectangle of inputs by using rectangular matrix multiplication. This makes use of the following simple reduction from evaluating a polynomial to computing an inner product:

Proposition 8.1. *For $d \in \mathbb{N}$, and any commutative ring R, let $p : R^{2d} \to R$ be any polynomial with t monomials. Then, there are maps $\phi, \psi : R^d \to R^t$ such that, for any $x, y \in R^d$, we have $p(x, y) = \langle \phi(x), \psi(y) \rangle$. Moreover, if p has degree Δ, then ϕ and ψ can be computed in $O(t \cdot \Delta)$ arithmetic operations over R.*

Proof. Since p has t monomials, there are maps $a : [d] \times [t] \to \mathbb{N} \cup \{0\}$, $b : [d] \times [t] \to \mathbb{N} \cup \{0\}$, and $c : [t] \to R$ such that

$$p(x, y) = \sum_{\ell=1}^{t} c(\ell) \cdot \left(\prod_{i=1}^{d} x_i^{a(i,\ell)} \right) \cdot \left(\prod_{j=1}^{d} y_j^{b(j,\ell)} \right).$$

We define $\phi : R^d \to R^t$, for $\ell \in [t]$, by:

$$\phi(x)_\ell := c(\ell) \cdot \left(\prod_{i=1}^{d} x_i^{a(i,\ell)} \right).$$

Similarly define $\psi : R^d \to R^t$, for $\ell \in [t]$, by:

$$\psi(y)_\ell := \left(\prod_{j=1}^{d} y_j^{b(j,\ell)} \right).$$

We can see that $p(x, y) = \langle \phi(x), \psi(y) \rangle$ as desired. □

By combining Proposition 8.1 with rectangular matrix multiplication algorithms, we can quickly evaluate sparse polynomials on many pairs of inputs; such a technique was first used in [Wil14a], and also implicitly in [Wil14c].

Proposition 8.2. *For $d \in \mathbb{N}$, and any commutative ring R, let $p : R^{2d} \to R$ be any polynomial of degree Δ with t monomials. Given as input two sets $A, B \subseteq R^d$, using $O((|A| + |B|) \cdot t \cdot \Delta)$ arithmetic operations over R, we can reduce the problem of evaluating p on all pairs $(x, y) \in A \times B$ to the problem of $|A| \times t \times |B|$ matrix multiplication over R.*

Proof. Letting $\phi, \psi : R^d \to R^t$ be the maps from Proposition 8.1, we compute $\phi(x)$ for each $x \in A$, and $\psi(y)$ for each $y \in B$, then multiply the resulting two matrices. □

We will typically apply Proposition 8.2 in conjunction with Coppersmith's very efficient rectangular matrix multiplication algorithm:

Lemma 8.1 ([Cop82]). *For all sufficiently large N, multiplication of an $N \times N^{.172}$ matrix with an $N^{.172} \times N$ matrix can be done in $O(N^2 \log^2 N)$ arithmetic operations over any field.*

Such a rectangular matrix multiplication algorithm requires some care; simply applying the idea from Proposition 4.1 above to Coppersmith's rank expression would yield an algorithm which uses $O(N^{2+\varepsilon})$ arithmetic operations for any $\varepsilon > 0$. A proof of how one can perform only $O(N^2 \log^2 N)$ arithmetic operations can be found in the appendix of [Wil14b]. We will typically use this approach when R is either a finite field, or else $\mathbb{R}$ but with relatively small coefficients, so that the arithmetic operations can be performed in $\text{polylog}(n)$ time:

Lemma 8.2 ([Wil14a]). *Given a polynomial $P(x_1, \ldots, x_d, y_1, \ldots, y_d)$ with at most $n^{0.17}$ monomials such that either:*

- *P is over a finite field $\mathbb{F}_q$ with $q \leq \text{polylog}(n)$, or*
- *P is over $\mathbb{R}$, and all its coefficients are $\text{polylog}(n)$-bit numbers,*

then given two sets of n inputs $A = \{a_1, \ldots, a_n\} \subseteq \{0,1\}^d$, $B = \{b_1, \ldots, b_n\} \subseteq \{0,1\}^d$, we can evaluate P on all pairs $(a_i, b_j) \in A \times B$ in $\tilde{O}(n^2 + d \cdot n^{1.17})$ time.

8.2 Exact Batch Nearest Neighbor Search

We now apply our polynomial constructions to solve the exact batch nearest neighbor problem. We begin by focusing on the related *closest pair* problem.

Let M be a metric on $\{0,1\}^d$. We define the Bichromatic M-Metric Closest Pair problem to be: given an integer k and a collection of "red" and "blue" vectors in $\{0,1\}^d$, determine if there is a pair of red and blue vectors with distance at most k under metric M. This problem arises frequently in algorithms on a metric space M. In what follows, we shall assume that the metric M can be computed on two points of d dimensions in time poly(d). Define the Boolean function

$$\begin{aligned} &M\text{-dist}_k(x_{1,1},\ldots,x_{1,d},\ldots,x_{s,1},\ldots,x_{s,d},y_{1,1},\ldots,y_{1,d},\ldots,y_{s,1},\ldots,y_{s,d}) \\ &\qquad := \bigvee_{i,j=1,\ldots,s} [M(x_{i,1},\ldots,x_{i,d},y_{j,1},\ldots,y_{j,d}) \le k]. \end{aligned}$$

That is, M-dist$_k$ takes two collections of s vectors as input, and outputs 1 if and only if there is a pair of vectors (one from each collection) that have distance at most k under metric M. For example, the Hamming-dist$_k$ function tests if there is a pair of vectors with Hamming distance at most k.

We observe that sparse probabilistic polynomials for computing M-dist$_k$ imply subquadratic time algorithms for finding close bichromatic pairs in metric M.

Theorem 8.1. *Suppose for all k, d, and n, there is an $s = s(d,n)$ such that M-dist$_k$ on $2sd$ variables has a probabilistic PTF with at most $n^{0.17}$ monomials,* polylog(n)*-bit coefficients, and error at most $1/3$, where each sample can be produced in $\tilde{O}(n^2/s^2)$ time. Then* Bichromatic M-Metric Closest Pair *on n vectors in d dimensions can be solved in $\tilde{O}(n^2/s^2 + s^2 \cdot poly(d))$ randomized time.*

Proof. We have an integer k and sets $R, B \subseteq \{0,1\}^d$ such that $|R| = |B| = n$, and wish to determine if there is a $u \in R$ and $v \in B$ such that $M(u,v) \le k$. First, partition both R and B into $\lceil n/s \rceil$ groups, with at most s vectors in each group. By assumption, for all k, there is a probabilistic PTF for M-dist$_k$ with $2sd$ variables, $n^{0.17}$ monomials, polylog(n)-bit coefficients, and error at most $1/3$. Let p be a polynomial sampled from this distribution. Our idea is to efficiently evaluate p on all $O(n^2/s^2)$ pairs of groups from R and B, by feeding as input to p all s vectors x_i from a group of R and all s vectors y_i from a group of B.

Since the number of monomials $m \le n^{0.17}$, we can apply Lemma 8.2, evaluating p on all pairs of groups in time $\tilde{O}(n^2/s^2)$. For each pair of groups from R and B, this evaluation determines if the pair of groups contain a bichromatic pair of distance at most k, with probability at least $2/3$.

To obtain a high probability answer, sample $\ell = 10 \log n$ polynomials $p_1, \ldots, p_\ell$ for M-dist$_k$ independently from the distribution, in $\tilde{O}(n^2/s^2)$ time (by assumption). Evaluate each p_i on all pairs of groups from R and B in $\tilde{O}(n^2/s^2)$ time by the above paragraph. Compute the majority value of $p_1, \ldots, p_\ell$ on all pairs of groups, again in $\tilde{O}(n^2/s^2)$ time. By a Chernoff-Hoeffding bound, the majority value reported for a pair of groups is correct with probability at least $1 - n^{-3}$. Therefore with probability at least $1 - n^{-1}$, we correctly determine for all pairs of groups from R and B whether the pair contains a bichromatic pair of vectors with distance at most k.

Given a pair of groups R' and B' which are reported to contain a bichromatic pair of close vectors,we can simply brute force to find the closest pair in A' and B' in $s^2 \cdot \text{poly}(d)$ time. (In principle, we could also perform a recursive call, but this doesn't asymptotically help us in our applications.) □

We will use our probabilistic PTF from the previous chapter:

Lemma 8.3. *For sufficiently large s and d, the Hamming-dist$_k$ function on $2sd$ variables has a probabilistic PTF of degree $O(d^{1/3}\log^{2/3}(ds))$, error at most $1/3$, and at most $O(s^2 \cdot \binom{d}{O(d^{1/3}\log^{2/3}(ds))})$ monomials, whose coefficients are* polylog(n) *bit numbers. Moreover, we can sample from the probabilistic PTF distribution in time polynomial in the number of monomials.*

Proof. Applying Corollary 7.4 ('exact setting') with $n \leftarrow d$, $s \leftarrow s^2$, and $t \leftarrow k$, gives us a probabilistic PTF degree $O(d^{1/3}\log^{2/3}(ds))$, error at most $1/3$, and at most $O(s^2 \cdot \binom{d}{O(d^{1/3}\log^{2/3}(ds))})$ monomials for, on input $z \in \{0,1\}^{2d\cdot s^2}$, testing the predicate $\bigvee_{i,j\in[s]} \left[\sum_{\ell=1}^{d} z_{i,j,\ell} \leq k\right]$. We use this to solve Hamming-dist$_k$ function on inputs $x, y \in \{0,1\}^{2d\cdot s}$ by substituting in $z_{i,j,\ell} \leftarrow (x_{i,\ell}(1 - y_{j,\ell}) + (1 - x_{i,\ell})y_{j,\ell})$ for all $i, j \in [s]$ and $\ell \in [d]$. Hence, for any $i, j \in [s]$, the quantity $\sum_{\ell=1}^{d} z_{i,j,\ell}$ computes exactly the Hamming distance between x_i and y_j, and so the result is a probabilistic PTF for the Hamming-dist$_k$ function.

Since we substituted in a polynomial with 4 monomials for each $z_{i,j,\ell}$, this increases the number of monomials in the resulting probabilistic PTF by a factor of $4^{O(d^{1/3}\log^{2/3}(ds))}$, but this factor is subsumed by the binomial coefficient $\binom{d}{O(d^{1/3}\log^{2/3}(ds))}$. □

Putting it all together, we obtain a faster algorithm for Bichromatic Hamming Closest Pair:

Theorem 8.2. *For n vectors of dimension $d = c(n)\log n$,* Bichromatic Hamming Closest Pair *can be solved in $n^{2-1/O(\sqrt{c(n)}\log^{3/2} c(n))}$ time by a randomized algorithm that is correct with high probability.*

Proof. Let $d = c\log n$ in the following, with the implicit understanding that c is a function of n. We apply the reduction of Theorem 8.1 and the probabilistic PTF for Hamming-dist$_k$ of Lemma 8.3.

The reduction of Theorem 8.1 requires that the number of monomials in our probabilistic polynomial is at most $n^{0.17}$, while the monomial bound for Hamming-dist$_k$ from Theorem 8.3 is $m = O(s^2 \cdot \binom{d}{ad^{1/3}\log^{2/3}(s)})$ for some universal constant a, provided that $s > d$ are sufficiently large. Therefore our primary task is to maximize the value of s such that $m \leq n^{0.17}$. This will minimize the final running time of $\tilde{O}(n^2/s^2)$. With hindsight, let us guess $s = n^{1/(u\sqrt{c}\log^{3/2} c)}$ for a constant u, and focus on the large binomial in the monomial estimate m. Then,

$$\binom{2d}{ad^{1/3}\log^{2/3}(s)} = \binom{2c\log n}{a(c\log n)^{1/3}\cdot(\log n)^{2/3}/(u\sqrt{c}\log^{3/2}c)^{2/3}}$$
$$= \binom{2c\log n}{a\log n/(u^{2/3}\log c)}.$$

For notational convenience, let $\delta = a/(u^{2/3}\log c)$. By Proposition 2.2 from the Preliminaries, we have

$$\binom{2c\log n}{\delta\log n} \leq \left(\frac{2ce}{\delta}\right)^{\delta\log n} = n^{\delta\log(\frac{2ce}{\delta})}.$$

Plugging $\delta = a/(u^{2/3}\log c)$ back into the exponent, we find

$$\delta\log\left(\frac{2ce}{\delta}\right) = \frac{a\log(\frac{2ceu^{2/3}\log c}{a})}{u^{2/3}\log c}. \tag{8.1}$$

The quantity (8.1) can be made arbitrarily small, by setting u sufficiently large. In that case, the number of monomials $m \leq s^2 n^{\delta\log(\frac{2ce}{\delta})}$ can be made less than $n^{0.1}$. □

Remark 8.1. *Observe that we would not have been able to prove Theorem 8.2 if the probabilistic PTF degree we applied from Lemma 8.3 had an additional* polylog(n) *multiplicative factor (which would have been the case if we hadn't succeeded at removing such factors from the degree in Theorem 7.2 earlier). Indeed, propagating this extra factor through the proof of Theorem 8.2 would have resulted in an additional* polylog n *multiplicative factor in expression* (8.1)*. It would then have been impossible to pick a constant u such that expression* (8.1) *is at most* 0.1*.*

Now we show how to solve Batch Hamming Nearest Neighbor (BHNN). In the following theorem, we assume for all pairs of vectors in our instance that the maximum metric distance is at most some value MAX. (For the Hamming distance, $MAX \leq d$.) We reduce the batch nearest neighbor query problem to the bichromatic close pair problem:

Theorem 8.3. *Let E^d be some d-dimensional domain supporting a metric space M. If the* Bichromatic M-Metric Closest Pair *on n vectors in E^d can be solved in $T(n,d)$ time, then* Batch M-Metric Nearest Neighbors *on n vectors in E^d can be solved in $O(n\cdot T(\sqrt{n},d)\cdot MAX)$ time.*

Proof. We give an oracle reduction similar to previous work [AWY15]. Initialize an table T of size n, with the maximum metric value v in each entry. Given n database vectors D and n query vectors Q, color D red and Q blue. Break D into $\lceil n/s\rceil$ groups of size at most s, and do the same for the set Q. For each pair $(R', B') \subset (D\times Q)$ of groups, and for each $k = MAX-1,\ldots,1,0$, we initialize $D_k := D$, $Q_k := Q$, and call Bichromatic M-Metric Closest Pair on $(R', B') \subset (D_k\times Q_k)$ with integer k. While we continue to find a pair $(x_i, y_j) \in (R'\times B')$ with $M(x_i, y_j) \leq k$, set $T[i] := k$

and remove y_j from Q_k and B'. (With a few more recursive calls, we could also find an explicit vector y_j such that $M(x_i, y_j) \leq k$.)

Now for each call that finds a close bichromatic pair, we remove a vector from Q_k; we do this at most MAX times for each vector, so there can be at most $MAX \cdot n$ such calls. For each pair of groups, there are MAX oracle calls that find no bichromatic pair. Therefore the total running time is $O((n + n^2/s^2) \cdot T(s,d) \cdot MAX)$. Setting $s = \sqrt{n}$ to balance the terms, the running time is $O(n \cdot T(\sqrt{n}, d) \cdot MAX)$. □

The following is immediate from Theorem 8.3 and Theorem 8.2:

Theorem 8.4. *For n vectors of dimension $d = c(n) \log n$,* Batch Hamming Nearest Neighbors *can be solved in $n^{2-1/O(\sqrt{c(n)}\log^{3/2} c(n))}$ time by a randomized algorithm, with high probability.*

Remark 8.2. For a deterministic algorithm, we can use the PTF from Corollary 7.3 instead of the probabilistic PTF from Corollary 7.4 when constructing our polynomial for the Hamming-dist$_k$ function in Lemma 8.3. The exact same algorithm then results in a deterministic running time of $n^{2-1/O(c \log^2 c)}$.

8.2.1 Closest Pair in Hamming Space is Hard

The *Strong Exponential Time Hypothesis* (SETH) states that there is no universal $\delta < 1$ such that for all c, CNF-SAT with n variables and cn clauses can be solved in $O(2^{\delta n})$ time. We next show that, assuming SETH, there is a limit to how much one can improve our running time from Theorem 8.4.

Theorem 8.5. *Suppose there is $\varepsilon > 0$ such that for all constant c,* Bichromatic Hamming Closest Pair *can be solved in $2^{o(d)} \cdot n^{2-\varepsilon}$ time on a set of n points in $\{0,1\}^{c \log n}$. Then SETH is false.*

Proof. The proof is a reduction from the Orthogonal Vectors problem with n vectors $S \subset \{0,1\}^d$, which asks whether there are $u, v \in S$ such that $\langle u, v \rangle = 0$. It is well-known that an algorithm for Orthogonal Vectors running in time $2^{o(d)} \cdot n^{2-\varepsilon}$ would refute SETH [Wil05]. Indeed, we show that Bichromatic Minimum Inner Product (finding a pair of vectors with minimum inner product, not just inner product zero) reduces to Bichromatic Hamming Closest Pair, as well as the version for maximum inner product.

First, we observe that Bichromatic Hamming Closest Pair is equivalent to Bichromatic Hamming Furthest Pair: let $\overline{v}$ be the complement of v (the vector obtained by flipping all the bits of v). Then the Hamming distance of u and v is $H(u,v) = d - H(u, \overline{v})$. Thus by flipping all the bits in the components of the blue vectors, we can reduce from the closest pair problem to furthest pair, and vice versa.

Now we reduce Orthogonal Vectors to Bichromatic Hamming Furthest Pair. Our Orthogonal Vectors instance has red vectors S_r and blue vectors S_b, and we wish to find $u \in S_r$ and $v \in S_b$ such that $\langle u, v \rangle = 0$.

For every d^2 possible choice of $I, J = 1, \ldots, d$, construct the subset $S_{r,I}$ of vectors in S_r with exactly I ones, and construct the subset $S_{b,J}$ of vectors in S_b with exactly

J ones. We will look for an orthogonal pair among $S_{r,I}$ and $S_{b,J}$ for all such I, J separately.

Recall that Hamming distance of two vectors equals the ℓ_2^2 norm distance, in $\{0,1\}^d$. The ℓ_2^2 norm of u and v is

$$||u-v||_2^2 = ||u||_2 + ||v||_2 - 2\langle u, v\rangle.$$

However, in $S_{r,I}$ all vectors have the same norm, and all vectors in $S_{b,J}$ have the same norm. Therefore, finding a red-blue pair $u \in S_{r,I}$ and $v \in S_{b,J}$ with minimum inner product is equivalent to finding a pair in $S_r \times S_b$ with smallest Hamming distance. (Similarly, maximum inner product is equivalent to Hamming closest pair.)

The reduction only requires $O(d^2)$ calls to Bichromatic Hamming Furthest Pair, with no changes to the dimension d nor the number of vectors n. □

8.2.2 Metrics Beyond Hamming Distance

We now show how Theorem 8.4 can be extended to find nearest neighbors for a number of other metrics. We state our best randomized running times in this Subsection, but one could also apply Remark 8.2 to get deterministic algorithms with slightly worse running times in all of the applications below.

Recall that the ℓ_1 norm of two vectors x and y is $\sum_i |x_i - y_i|$. We can solve Batch ℓ_1 Nearest Neighbors on vectors with small integer entries by a simple reduction to Batch Hamming Nearest Neighbors, (which is probably folklore):

Theorem 8.6. *For n vectors of dimension $d = c \log n$ in $\{0, 1, \ldots, m\}^d$,* Batch ℓ_1 Nearest Neighbors *can be solved in $n^{2-1/O(\sqrt{mc}\log^{3/2}(mc))}$ time by a randomized algorithm, with high probability.*

Proof. Notice that for any $x, y \in \{0, \ldots, m\}$, the Hamming distance of their unary representations, written as m-dimensional vectors, is equal to $|x - y|$. Hence, for $x \in \{0, \ldots, m\}^d$, we can transform it into a vector $x' \in \{0,1\}^{md}$ by setting $(x'_{m(i-1)+1}, x'_{m(i-1)+2}, \ldots, x'_{m(i-1)+m})$ equal to the unary representation of x_i, for $1 \le i \le d$. It is then equivalent to solve the Hamming nearest neighbors problem on these md-dimensional vectors. □

It is also easy to extend Theorem 8.4 for vectors over $O(1)$-sized alphabets using equidistant binary codes ([MKZ09], Section 5.1). This is useful for applications in biology, such as finding similar DNA sequences. The above algorithms also apply to computing maximum inner products:

Theorem 8.7. *The* Bichromatic Minimum Inner Product *(and* Maximum*) problem with n red and blue Boolean vectors in $c \log n$ dimensions can be solved in $n^{2-1/O(\sqrt{c}\log^{3/2} c)}$ randomized time.*

Proof. In Theorem 8.5 above, we gave a reduction from Bichromatic Minimum Inner Product to Bichromatic Hamming Furthest Pair, *and* showed that Bichromatic Hamming Furthest Pair is equivalent to Bichromatic Hamming

Closest Pair. The same reduction shows that Bichromatic Maximum Inner Product reduces to the closest pair version. Hence Theorem 8.4 applies, to both minimum and maximum inner products. □

As a consequence, we can answer a batch of n minimum inner product queries on a database of size n with the same time estimate, applying a reduction analogous to that of Theorem 8.3. From there, Theorem 8.7 can be extended to other important similarity measures, such as finding a pair of sets A, B with maximum *Jaccard coefficient*, defined as $\frac{|A\cap B|}{|A\cup B|}$ [Bro97].

Corollary 8.1. *Given n red and blue subsets of a universe of size $c \log n$, we can find the pair of red and blue sets with maximum Jaccard coefficient in $n^{2-1/O(\sqrt{c}\log^{3/2} c)}$ randomized time.*

Proof. Let S be a given collection of red and blue sets over $[d]$. We construe the sets in S as vectors, in the natural way. For all possible values $d_1, d_2 = 1, \ldots, d$, we will construct an instance of Bichromatic Maximum Inner Product S'_{d_1,d_2}, and take the best pair found, appealing to Theorem 8.7.

As in the proof of Theorem 8.5, we "filter" sets based on their cardinalities. In the instance S'_{d_1,d_2} of Bichromatic Maximum Inner Product, we only include red sets with cardinality exactly d_1, and blue sets with cardinality exactly d_2. For sets R, B, we have

$$\frac{|R\cap B|}{|R\cup B|} = \frac{|R\cap B|}{d_1 + d_2 - |R\cap B|}. \tag{8.2}$$

Suppose that we choose a red set R and blue set B that maximize $|R\cap B|$. This choice simultaneously maximizes the numerator and minimizes the denominator of (8.2), producing the sets R and B with maximum Jaccard coefficient over the red sets with cardinality d_1 and blue sets with cardinality d_2. Finding the maximum pair R and B over each choice of d_1, d_2, we will find the overall R and B with maximum Jaccard coefficient. □

8.3 Approximate Batch Nearest Neighbor Search

The same approach that we used in Theorem 8.4 for exact nearest neighbor search in Hamming space can be applied to solve for *approximate* nearest neighbor search in Hamming space as well:

Theorem 8.8. *Given n red and n blue points in $\{0,1\}^d$ and $\varepsilon \gg \log^6(d\log n)/\log^3 n$, we can find an approximate Hamming nearest/farthest blue neighbor with additive error at most εd for each red point in randomized time $n^{2-\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon \log n}))}$.*

Proof. We mimic the proof of Theorem 8.4 up to the definition of the polynomial in Lemma 8.3. However, instead of applying the exact polynomial of Corollary 7.4, we insert the *approximate* polynomial construction from the same Corollary. While the exact polynomial had degree $O(d^{1/3}\log^{2/3}(ds))$, the approximate one has degree $O((1/\epsilon)^{1/3}\log s)$. Setting

$$s := n^{\alpha} := n^{\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon \log n}))},$$

the number of monomials in the new polynomial is now

$$\begin{aligned} s^2 \cdot \binom{O(d)}{O((1/\varepsilon)^{1/3}\log s)} &\le n^{2\alpha}\cdot O\left(\frac{d}{(\alpha/\varepsilon^{1/3})\log n}\right)^{O((\alpha/\varepsilon^{1/3})\log n)} \\ &\le n^{2\alpha}\cdot n^{O((\alpha/\varepsilon^{1/3})\log\frac{d}{\alpha\log n})} \ll (n/s)^{0.1}, \end{aligned}$$

for large enough n. The remainder of the algorithm is the same as the proof of Theorem 8.4, and the running time is $\tilde{O}(n^2/s^2) \le n^{2-\Omega(\varepsilon^{1/3}/\log(\frac{d}{\varepsilon\log n}))}$. □

Remark 8.3. For a deterministic algorithm, using Corollary 7.3 instead of Corollary 7.4, the we get a deterministic running time of $n^{2-\Omega(\sqrt{\varepsilon}/\log(\frac{d}{\varepsilon\log n}))}$.

The algorithm of Theorem 8.8 still has three drawbacks: (i) the exponent in the time bound depends on the dimension d, (ii) the result requires additive instead of multiplicative error, and (iii) the result is for Hamming space instead of more generally ℓ_1 or ℓ_2. We sketch how to resolve all three issues at once, by using known dimension reduction techniques:

Theorem 8.9. *Given n red and n blue points in $[U]^d$ and $\varepsilon \gg \frac{\log^6\log n}{\log^3 n}$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 nearest/farthest blue neighbor for each red point in $(dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))})\cdot poly(\log(nU))$ randomized time.*

Proof. **(The ℓ_1 case.)** We first solve the decision problem for a fixed threshold value t. We use a variant of ℓ_1 locality-sensitive hashing (see [And05]) to map points from ℓ_1 into low-dimensional Hamming space (providing an alternative to Kushilevitz, Ostrovsky, and Rabani's dimension reduction technique for Hamming space [KOR00]). For each red/blue point p and each $i \in \{1,\ldots,k\}$, define $h_i(p) = (h_{i1}(p),\ldots,h_{id}(p))$ with $h_{ij}(p) = \lfloor (p_{a_{ij}} + b_{ij})/(2t)\rfloor$ where $a_{ij} \in \{1,\ldots,d\}$ and $b_{ij} \in [0,2t)$ are independent uniformly distributed random variables. For each of the $O(n)$ hashed values of h_i, pick a random bit; let $f_i(p)$ be the random bit associated with $h_i(p)$. Finally, define $f(p) = (f_1(p),\ldots,f_k(p)) \in \{0,1\}^k$. For any fixed p, q,

$$\begin{aligned} \Pr[h_{ij}(p) \ne h_{ij}(q)] &= \frac{1}{d}\sum_{a=1}^{d}\min\left\{\frac{|p_a - q_a|}{2t}, 1\right\}, \text{ and so} \\ \Pr[f_i(p) \ne f_i(q)] &= \frac{1}{2}\Pr[h_i(p)\ne h_i(q)] = \frac{1}{2}\Pr\left[\bigvee_{j=1}^{k}[h_{ij}(p)\ne h_{ij}(q)]\right]. \end{aligned}$$

Hence,

- If $\|p-q\|_1 \le t$, then $\Pr[h_{ij}(p)\ne h_{ij}(q)] \le \frac{\|p-q\|_1}{2dt} \le \frac{1}{2d}$ and $\Pr[f_i(p)\ne f_i(q)] \le \alpha_0 := \frac{1}{2}(1-(1-\frac{1}{2d})^d)$;

- if $\|p-q\|_1 \geq (1+\varepsilon)t$, then $\Pr[h_{ij}(p) \neq h_{ij}(q)] \geq \min\{\frac{\|p-q\|_1}{2dt}, \frac{1}{d}\} \geq \frac{1+\varepsilon}{2d}$ and $\Pr[f_i(p) \neq f_i(q)] \geq \alpha_1 := \frac{1}{2}(1-(1-\frac{1+\varepsilon}{2d})^d)$.

Note that $\alpha_1 - \alpha_0 = \Omega(\varepsilon)$. By a Chernoff bound, it follows (assuming $k \geq \log n$) that

- if $\|p-q\|_1 \leq t$, then $\|f(p)-f(q)\|_1 \leq A_0 := \alpha_0 k + O(\sqrt{k \log n})$ with probability $1 - O(1/n^3)$;
- if $\|p-q\|_1 \geq (1+\varepsilon)t$, then $\|f(p)-f(q)\|_1 \geq A_1 := \alpha_1 k - O(\sqrt{k \log n})$ with probability $1 - O(1/n^3)$.

Note that $A_1 - A_0 = \Omega(\varepsilon k)$ by setting k to be a sufficiently large constant times $(1/\varepsilon)^2 \log n$. We have thus reduced the problem to an approximate problem with additive error $O(\varepsilon k)$ for Hamming space in $k = O((1/\varepsilon^2)\log n)$ dimensions, which by Theorem 8.8 requires $n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}$ time. The initial cost of applying the mapping f is $O(dkn)$.

This solves the decision problem; we can then solve the original problem by calling the decision algorithm $O(\log_{1+\varepsilon} U)$ times for all t's that are powers of $1+\varepsilon$. □

Proof. **(The ℓ_2 case.)** We use a version of the Johnson–Lindenstrauss lemma to map from ℓ_2 to ℓ_1 (see for example [Mat08]). For each red/blue point p, define $f(p) = (f_1(p), \ldots, f_k(p)) \in \mathbb{R}^k$ with $f_i(p) = \sum_{j=1}^k a_{ij} p_j$, where the a_{ij}'s are independent normally distributed random variables with mean 0 and variance 1. For each fixed $p, q \in \mathbb{R}^d$, it is known that after rescaling by a constant, $\|f(p)-f(q)\|_1$ approximates $\|p-q\|_2$ to within $1 \pm O(\varepsilon)$ factor with probability $1-O(1/n^3)$, by setting $k = O((1/\varepsilon)^2 \log n)$. It suffices to keep $O(\log U)$-bit precision of the mapped points. The initial cost of applying the mapping f is $O(dkn)$ (which can be slightly improved by utilizing a sparse Johnson–Lindenstrauss transform [AC09]). □

Numerous applications to high-dimensional computational geometry now follow. We briefly mention just one such application, building on the work of [IM98, HIM12]:

Corollary 8.2. *Given n points in $[U]^d$ and $\varepsilon \gg \log^6 \log n / \log^3 n$, we can find a $(1+\varepsilon)$-approximate ℓ_1 or ℓ_2 minimum spanning tree in $(dn + n^{2-\Omega(\varepsilon^{1/3}/\log(1/\varepsilon))}) \cdot$ poly$(\log(nU))$ randomized time.*

Proof. Let G_r denote the graph where the vertex set is the given point set P and an edge pq is present whenever p and q have distance at most r. Har-Peled, Indyk, and Motwani [HIM12] gave a reduction of the approximate minimum spanning tree problem to the following *approximate connected components* problem:

> Given a value r, compute a partition of P into subsets with the properties that (i) two points in the same subset must be in the same component in $G_{(1+\varepsilon)r}$, and (ii) two points in different subsets must be in different components in G_r.

The reduction is based on Kruskal's algorithm and increases the running time by a logarithmic factor.

To solve the approximate connected components problem, Har-Peled, Indyk, and Motwani gave a further reduction to online dynamic approximate nearest neighbor search. Since we want a reduction to offline static approximate nearest neighbor search, we proceed differently.

We first reduce the approximate connected components problem to the *offline approximate nearest foreign neighbors* problem:

> Given a set P of n colored points with colors from $[n]$, for each point $q \in P$, find a $(1+\varepsilon)$-approximate nearest neighbor NFN_q among all points in P with color different from q's color.

The reduction can be viewed as a variant of Boruvka's algorithm and is as follows: Initially assign each point a unique color and mark all colors as active. At each iteration, solve the offline approximate nearest foreign neighbors problem for points with active colors. For each q, if NFN_q and q have distance at most $(1+\varepsilon)r$ and have different colors, merge the color class of NFN_q and q. If a color class has not been merged to other color classes during the iteration, mark its color as inactive. When all colors are inactive, output the color classes. Otherwise, proceed to the next iteration. The correctness of the algorithm is obvious. Since each iteration decreases the number of active colors by at least a half, the number of iterations is bounded by $O(\log n)$. Thus, the reduction increases the running time by a logarithmic factor.

To finish, we reduce the offline approximate nearest foreign neighbors problem to the standard (red/blue) offline approximate nearest neighbors problem by a standard trick: For each $j = 1, \ldots, \lceil \log n \rceil$, for each point $q \in P$ where the j-th bit of q's color is 0 (resp. 1), compute an approximate nearest neighbor of q among all points $p \in P$ where the j-th bit of p's color is 1 (resp. 0). Record the nearest among all approximate nearest neighbors found for each point q. The final reduction increases the running time by another logarithmic factor. □

8.4 The Light Bulb Problem

In all our applications of the polynomial method for algorithm design so far, we have been focusing on optimizing the *asymptotics* of the exponent in the running time as a parameter of the problem (in particular, the constant factor c in the dimension for the problem) grew. In this Section, we instead show techniques for optimizing the constants in the exponent of the running time. We focus our attention on the Light Bulb Problem.

Problem 8.1 (Light Bulb Problem)**.** *We are given as input a set S of n vectors from $\{-1,1\}^d$, which are all independently and uniformly random except for two planted vectors (the correlated pair) which have inner product at least $\rho \cdot d$ for some $0 < \rho \leq 1$. The goal is to find the correlated pair.*

Theorem 8.10. *For every $\varepsilon, \rho > 0$, there is a $\kappa > 0$ such that the Light Bulb Problem for correlation ρ can be solved in randomized time $O(n^{2\omega/3+\varepsilon})$ whenever $d = \kappa \log n$ with polynomially low error.*

Proof. Our algorithm can be seen as applying the polynomial method in algorithm design for the very simple polynomial $p(x, y) = (\langle x, y\rangle)^r$.

For two constants $\gamma, k > 0$ to be determined, we will pick $\kappa = \gamma k^2/\rho^2$. Let $S \subseteq \{-1, 1\}^d$ be the set of input vectors, and let $x', y' \in S$ denote the correlated pair which we are trying to find. For distinct $x, y \in S$ other than the correlated pair, the inner product $\langle x, y\rangle$ is a sum of d uniform independent $\{-1, 1\}$ values. Let $v := \gamma(k/\delta) \log n$. By a Chernoff bound, for large enough γ, we have $|\langle x, y\rangle| \le v$ with probability at least $1 - 1/n^3$. Hence, by a union bound over all pairs of uncorrelated vectors, we have $|\langle x, y\rangle| \le v$ for all such x, y with probability at least $1 - 1/n$. We assume henceforth that this is the case. Meanwhile, $\langle x', y'\rangle \ge \rho d = kv$.

Arbitrarily partition S into $m := n^{2/3}$ groups $S_1, \ldots, S_m$ of size $g := n/m = n^{1/3}$ each. We can compute the inner product between each pair of vectors which was assigned to the same group in time $O(m \cdot g^2 \cdot d) = \tilde{O}(n^{4/3})$, and if we find the correlated pair, we can return it and end the algorithm. Otherwise, we may assume the correlated vectors are in different groups, and we continue.

For each $x \in S$, our algorithm picks a value $a^x \in \{-1, 1\}$ independently and uniformly at random. For a constant $\tau > 0$ to be determined, let $r = \lceil \log_k(\tau n^{1/3})\rceil$, and define the polynomial $p : \mathbb{R}^d \to \mathbb{R}$ by $p(z_1, \ldots, z_d) = (z_1 + \cdots + z_d)^r$. Our goal is, for each $(i, j) \in [m]^2$, to compute the value

$$C_{i,j} := \sum_{x\in S_i} \sum_{y\in S_j} a^x \cdot a^y \cdot p(x_1y_1, \ldots, x_dy_d).$$

Solving the problem using $C_{i,j}$

Let us first explain why we are interested in computing $C_{i,j}$. Denote $p(x, y) := p(x_1y_1, \ldots, x_dy_d)$. Intuitively, $p(x, y)$ is computing an *amplification* of $\langle x, y\rangle$. $C_{i,j}$ is then summing these amplified inner products for all pairs $(x, y) \in S_i \times S_j$. We will pick our parameters so that the amplified inner product of the correlated pair is large enough to stand out from the sums of inner products of random pairs.

Let us be more precise. Recall that for uncorrelated x, y we have $|\langle x, y\rangle| \le v$, and hence $|p(x, y)| \le v^r$. Similarly, we have $|p(x', y')| \ge (kv)^r \ge \tau n^{1/3}v^r$. For $x, y \in S$, define $a^{(x,y)} := a^x \cdot a^y$. Notice that, for $i \ne j$, $C_{i,j} = \sum_{x\in S_i, y\in S_j} a^{(x,y)}p(\langle x, y\rangle)$, where the $a^{(x,y)}$ are *pairwise independent* random $\{-1, 1\}$ values.

We will now analyze the random variable $C_{i,j}$ where we think of the vectors in S as fixed, and only the values a^x as random.

Consider first when the correlated pair are not in S_i and S_j. Then, $C_{i,j}$ has mean 0, and (since variance is additive for pairwise independent variables) $C_{i,j}$ has variance at most $|S_i| \cdot |S_j| \cdot \max_{x\in S_i, y\in S_j} |p(\langle x, y\rangle)|^2 \le n^{2/3} \cdot v^{2r}$. For sufficiently large constant τ, by the Chebyshev inequality, we have that $|C_{i,j}| \le \tau n^{1/3}v^r/3$ with probability at least 3/4. Let $\theta = \tau n^{1/3}v^r/3$, so $|C_{i,j}| \le \theta$ with probability at least 3/4.

Meanwhile, if $x' \in S_i$ and $y' \in S_j$, then $C_{i,j}$ is the sum of $a^{(x',y')}p(\langle x', y'\rangle)$

and a variable C' distributed as $C_{i,j}$ was in the previous paragraph. Hence, since $|p(\langle x', y' \rangle)| \geq \tau n^{1/3} v^r = 3\theta$, and $|C'| \leq \theta$ with probability at least $3/4$, we get by the triangle inequality that $|C_{i,j}| \geq 2\theta$ with probability at least $3/4$.

Hence, if we repeat the process of selecting the a^x values for each $x \in S$ independently at random $O(\log n)$ times, whichever pair S_i, S_j has $|C_{i,j}| \geq 2\theta$ most frequently will be the pair containing the correlated pair with polynomially low error, and then a brute force within this set of $O(n^{1/3})$ vectors can find the correlated pair in $\tilde{O}(n^{2/3})$ time. In all, by a union bound over all possible errors, this will succeed with polynomially low error.

Computing $C_{i,j}$

It remains to give the algorithm to compute $C_{i,j}$. Our overall approach will be almost identical to Proposition 8.2. However, rather than appeal directly to the statement of Proposition 8.2, we go through the details, since the running time of the reduction stated there is actually too slow for us.

We begin by rearranging the expression for $C_{i,j}$ into one which is easier to compute. Since we are only interested in the values of p when its inputs are all in $\{-1, 1\}$, we can replace p with its multilinearization[1] $\hat{p}$. Let $M_1, \ldots, M_t$ be an enumeration of all subsets of $[d]$ of size at most r, so $t = \sum_{i=0}^{r} \binom{d}{i}$. Then, there are coefficients $c_1, \ldots, c_t \in \mathbb{Z}$ such that $\hat{p}(x) = \sum_{s=1}^{t} c_s x_{M_s}$ (where, for $x \in \{-1, 1\}^d$ and $M \subseteq [d]$ we define $x_M := \prod_{i \in M} x_i$). Rearranging the order of summation, we see that we are trying to compute

$$C_{i,j} = \sum_{s=1}^{t} \sum_{x \in S_i} \sum_{y \in S_j} a^x \cdot a^y \cdot c_s \cdot x_{M_s} \cdot y_{M_s} = \sum_{s=1}^{t} \left[c_s \left(\sum_{x \in S_i} a^x \cdot x_{M_s} \right) \left(\sum_{y \in S_j} a^y \cdot y_{M_s} \right) \right]. \tag{8.3}$$

In order to compute $C_{i,j}$, we first need to compute the coefficients c_s. Notice that c_s depends only on $|M_s|$ and r. We can thus derive a simple combinatorial expression for c_s, and hence compute all of the c_s coefficients in $\mathrm{poly}(r) = \mathrm{polylog}(n)$ time. Alternatively, by starting with the polynomial $(z_1 + \cdots + z_d)$ and then repeatedly squaring then multilinearizing, we can easily compute all the coefficients in $O(t^2 \,\mathrm{polylog}(n))$ time; this slower approach is still fast enough for our purposes.

Define the matrices $A, B \in \mathbb{Z}^{m \times t}$ by $A_{i,s} = \sum_{x \in S_i} a^x \cdot x_{M_s}$ and $B_{i,s} = c_s \cdot A_{i,s}$. Notice from (8.3) that the matrix product $C := AB^T$ is exactly the matrix of the values $C_{i,j}$ we desire. A simple calculation (see Lemma 8.4 below) shows that for any $\varepsilon > 0$, we can pick a sufficiently big constant $k > 0$ such that $t = O(n^{2/3+\varepsilon})$. Since $m = O(n^{2/3})$, if we have the matrices A, B, then we can compute this matrix product by performing n^{ε} instances of $n^{2/3} \times n^{2/3} \times n^{2/3}$ matrix multiplication over $\mathbb{Z}$ with $\mathrm{polylog}(n)$-bit entries, in $\tilde{O}(n^{2\omega/3+\varepsilon})$ time, completing the algorithm[2].

[1] In other words, whenever a variable appears raised to an exponent bigger than 1, we reduce that exponent mod 2 to either 0 or 1, which does not change the value of the polynomial.

[2] One can slightly decrease the constant $\varepsilon > 0$ so that the $\mathrm{polylog}(n)$ factors do not appear in the final running time.

Unfortunately, computing the entries of A and B naively would take $\Omega(m \cdot t \cdot g) = \Omega(n^{5/3})$ time, which is slower than we would like. We will instead use a clever trick due to Lovett [Lov11], which was first applied in this context by Karppa et al. [KKK16]: we will compute those entries using *another* matrix multiplication. Let $N_1, \ldots, N_u$ be an enumeration of all subsets of $[d]$ of size at most $\lceil r/2 \rceil$. For each $i \in [m]$, define the matrices $L^i, \tilde{L}^i \in \mathbb{Z}^{u \times g}$ (whose columns are indexed by elements $x \in S_i$) by $L^i_{s,x} = x_{N_s}$ and $\tilde{L}^i{}_{s,x} = a^x \cdot x_{N_s}$. Then, compute the product $P^i := L^i \tilde{L}^{i^T}$. We can see that $P^i_{s,s'} = \sum_{x \in S_i} a^x \cdot x_{N_s \oplus N_{s'}}$, where $N_s \oplus N_{s'}$ is the symmetric difference of N_s and $N_{s'}$. Since any set of size at most r can be written as the symmetric difference of two sets of size at most $\lceil r/2 \rceil$, each desired entry $A_{i,s}$ can be found as an entry of the computed matrix P^i. Similar to our bound on t from before (see Lemma 8.4 below), we see that for big enough constant k, we have $u = O(n^{1/3+\varepsilon})$. Computing the entries of the L^i matrices naively takes only $O(m \cdot u \cdot g \cdot r) = \tilde{O}(n \cdot u) = \tilde{O}(n^{4/3+\varepsilon})$ time, and then computing the products P^i takes $O(m \cdot \max(u, g)^\omega) = O(n^{(2+\omega)/3+\varepsilon})$ time; both of these are dominated by $O(n^{2\omega/3+\varepsilon})$. This completes the algorithm! Finally, we perform the computations mentioned above in Lemma 8.4 below. □

Lemma 8.4. *For every $\varepsilon > 0$, there is a $k > 0$ such that (with the same notation as in the proof of Theorem 8.10 above) we can bound $t = O(n^{2/3+\varepsilon})$, and $u = O(n^{1/3+\varepsilon})$.*

Proof. Recall that $d = O(k^2 \log(n))$, and $r = \log_k(O(n^{1/3}))$. Hence, by Proposition 2.2 from the Preliminaries,

$$t \leq (r+1) \cdot \binom{d}{r} \leq (r+1) \cdot (ed/r)^r \leq O(k^2 \log(k))^{\log_k(O(n^{1/3}))} = n^{2/3+O(\log\log(k)/\log(k))}.$$

For any $\varepsilon > 0$ we can thus pick a sufficiently large k so that $t \leq O(n^{2/3+\varepsilon})$. We can similarly bound $\binom{d}{r/2} \leq O(n^{1/3+\varepsilon})$ which implies our desired bound on u. □

8.4.1 Deterministic Algorithms

We now present two deterministic algorithms for the Light Bulb Problem. Each is a slight variation on the algorithm from Theorem 8.10 above.

Theorem 8.11. *For every $\varepsilon, \rho > 0$, there is a $\kappa > 0$ such that the Light Bulb Problem for correlation ρ can be solved in deterministic time $O(n^{2\omega/3+\varepsilon})$ on almost all instances whenever $d = \kappa \log n$.*

Recall that our goal when solving the Light Bulb Problem on almost all instances is to design a deterministic algorithm such that the probability of drawing an instance where the algorithm fails is $1/\text{poly}(n)$.

Proof. The only randomness used by our algorithm for Theorem 8.10 was our choice of an independently and uniformly random $a^x \in \{-1, 1\}$ for each $x \in S$. Since this requires $\Theta(n)$ random bits, and we repeat the entire algorithm $\Theta(\log n)$ times to get our desired correctness guarantee, the total number of random bits used is $\Theta(n \log n)$.

However, the only property of the a^x variables which we use in the proof of correctness is that they are *pairwise*-independent. By standard constructions[3], only $O(\log n)$ independent random bits are needed to generate n pairwise-independent random bits. Thus, our entire algorithm actually only needs $O(\log^2 n)$ independent random bits.

Our entirely deterministic algorithm then proceeds as follows. Pick the same κ as in Theorem 8.10. Let $S \subseteq \{-1,1\}^d$ be the input vectors. Arbitrarily pick a subset $S' \subseteq S$ of $|S'| = \Theta(\log n)$ of the input vectors, and let $R = S \setminus S'$ be the remaining vectors.

We begin by testing via brute-force whether either vector of the correlated pair is in S'. This can be done in $O(|S'| \cdot |S| \cdot d) = O(n \log^2(n))$ time. If we find the correlated pair (a pair with inner product at least $\rho \cdot d$), then we output it, and otherwise, we can assume that the vectors in S' are all uniformly random vectors from $\{-1,1\}^d$. In other words, we can use them as $d \cdot |S'| = \Theta(\log^2 n)$ independent uniformly random bits. We thus use them as the required randomness to run the algorithm from Theorem 8.10 on input vectors R. That algorithm has polynomially low error, which implies the desired correctness guarantee. □

Theorem 8.12. *There is a constant $w > 0$ such that, for every $\varepsilon, \rho > 0$, there is a $\kappa > 0$ such that the Promise Light Bulb Problem with parameter w for correlation ρ can be solved in deterministic time $O(n^{4\omega/5+\varepsilon})$ whenever $d = \kappa \log n$.*

Recall that in the Promise Light Bulb Problem with parameter w, we are promised that every pair of vectors other than the correlated pair has inner product at most $w\sqrt{d \log n}$, and our deterministic algorithm needs to solve the problem correctly on every input with this guarantee.

Proof. The guarantee of the Promise Light Bulb Problem is that, when we pick a sufficiently large w, the uncorrelated vectors have as small inner product as we assumed they did in the first paragraph in the proof of Theorem 8.10. In other words, there is a quantity v such that $|\langle x,y\rangle| \leq v$ for all $x, y \in S$ other than the correlated pair, and moreover, $\langle x', y'\rangle \geq kv$ for a constant $k > 0$ with $k \to \infty$ as $w \to \infty$.

The algorithm is then almost identical to Theorem 8.10, except we need to remove the only use of randomness: the randomness used to pick the a^x values. To do this, we will simply pick $a^x = 1$ for all x.

In order to guarantee the correctness of our algorithm, we must now change the parameters slightly. Instead of partitioning the input into $m = n^{2/3}$ groups of size $g = n^{1/3}$, we will instead partition into $m = n^{4/5}$ groups of size $g = n^{1/5}$. Similarly, instead of picking r (the exponent in the polynomial p) to be $\log_k(O(n^{1/3}))$, we will pick $r = \log_k(3n^{2/5})$, so that $p(x', y') \geq (kv)^r = 3n^{2/5}v^r$.

With these choices, for any i and j such that the correlated pair are not in S_i and S_j, we have $|C_{i,j}| \leq |S_i| \cdot |S_j| \cdot n^{2/5} = n^{2/5}v$, whereas if $x' \in S_i$ and $y' \in S_j$ then by the triangle inequality, $|C_{i,j}| \geq p(x', y') - |S_i| \cdot |S_j| \cdot n^{2/5} \geq 2n^{2/5}v^r$. Hence, the correlated pair must be in whichever S_i and S_j with $i \neq j$ has the largest $|C_{i,j}|$.

[3] For one example, to generate $2^\ell - 1$ pairwise-independent bits, pick only ℓ bits $b_1, \ldots, b_\ell \in \{-1,1\}$ independently and uniformly at random, and then output, for each $I \subseteq [\ell]$, the product $\prod_{i \in I} b_i$.

The algorithm to compute the $C_{i,j}$ values is identical to that of Theorem 8.10. We now get that $t = \sum_{i=0}^{r} \binom{d}{i} \leq O(n^{4/5+\varepsilon})$ and similarly, $u \leq O(n^{2/5+\varepsilon})$, which leads to a final running time of $O(n^{4\omega/5+\varepsilon})$, as desired. □

8.5 Faster Algorithms For MAX-SAT

Next, we apply our probabilistic PTFs for threshold functions to obtain faster algorithms for MAX-SAT for sparse instances with cn clauses. We first consider MAX-k-SAT for small k before solving the general problem:

Theorem 8.13. *Given a k-CNF formula F (or k-CSP instance) with n variables and $cn \ll n^4/(k^4 \log^6 n)$ clauses, we can find an assignment that satisfies the maximum number of clauses (constraints) of F in randomized $2^{n-n/O(k^{4/3}c^{1/3}\log(kc))}$ time.*

Proof. We proceed as in the #k-SAT algorithm of Chan and Williams [CW16]. We first solve the decision problem of testing whether there is a variable assignment satisfying more than t clauses for a fixed $t \in [cn]$. Let $s = \alpha n$ for some parameter $\alpha < 1/2$ to be set later.

For $j \in [c_n]$, define the function $C_j(x_1, \ldots, x_n)$ to output 1 if the j-th clause of the given formula is satisfied, and 0 otherwise. Note that each C_j can be expressed as a polynomial of degree at most k.

Say that a variable is *good* if it occurs in at most $2kc$ clauses. By the pigeonhole principle, at least half of the variables are good, so we can find s good variables $x_1, \ldots, x_s$. Let $x_{s+1}, \ldots, x_n$ be the remaining variables, and let $J \subset [cn]$ be the set of indices of all clauses C_j that contain some occurrence of a good variable; note that $|J| = O(kcs)$. Now for every variable assignment $(x_{s+1}, \ldots, x_n) \in \{0,1\}^{n-s}$, we want to compute

$$F(x_{s+1}, \ldots, x_n) \;:=\; \bigvee_{(a_1,\ldots,a_s)\in\{0,1\}^s} \left[\sum_{j=1}^{cn} C_j(a_1, \ldots, a_s, x_{s+1}, \ldots, x_n) > t\right].$$

We will achieve this by computing for every $t' \in [cn]$:

$$G_{t'}(x_{s+1}, \ldots, x_n) \;:=\; \bigvee_{(a_1,\ldots,a_s)\in\{0,1\}^s} \left[\sum_{j\in J} C_j(a_1, \ldots, a_s, x_{s+1}, \ldots, x_n) > t'\right].$$

Let us define $T[x_{s+1}, \ldots, x_n] := t - \sum_{j\notin J} C_j(0, \ldots, 0, x_{s+1}, \ldots, x_n)$. (Observe that it is not a problem to set the good variables $x_1, \ldots, x_s$ to zero here, because we are only summing over clauses that *do not* contain them.) Note that T can be viewed as a polynomial in $n - s$ variables with only poly(n) monomials. Therefore for all $(x_{s+1}, \ldots, x_n) \in \{0,1\}^{n-s}$, these T-values can be precomputed in poly$(n)2^{n-s}$ time. As these T-values are measuring the contribution from the variables $x_{s+1}, \ldots, x_n$ to the number of satisfied clauses, we have

$$F(x_{s+1}, \ldots, x_n) = G_{T[x_{s+1},\ldots,x_n]}(x_{s+1}, \ldots, x_n).$$

Applying Corollary 7.4 (in the exact setting), we can express any $G_{t'}$ as a sum of 2^s probabilistic PTFs of degree $k \cdot O((kcs)^{1/3}(s+\log(kcs))^{2/3})$, where each probabilistic PTF computes an expression of the form $\left[\sum_{j\in J} p_j(x_{s+1},\ldots,x_n)\right]$ with error probability at most $1/(10\cdot 2^s)$, and for all $j \in J$ we have $\deg(p_j(x_{s+1},\ldots,x_n)) \le k$. The number of monomials in our probabilistic PTF for $G_{t'}$ is at most

$$\begin{aligned} 2^s \cdot \binom{n-s}{k\cdot O((kcs)^{1/3}(s+\log(kcs))^{2/3})} &\le 2^{\alpha n}\cdot O\left(\frac{n}{k^{4/3}c^{1/3}\alpha n}\right)^{O(k^{4/3}c^{1/3}\alpha n)} \\ &\le 2^{\alpha n}\cdot 2^{O(k^{4/3}c^{1/3}\alpha\log\frac{1}{\alpha})n} \ll 2^{0.1n} \end{aligned}$$

by setting α to be a sufficiently small constant times $1/(k^{4/3}c^{1/3}\log(kc))$. The same bound holds for the construction time of the polynomial.

For each t', we can evaluate the polynomial for $G_{t'}$ at all 2^{n-s} input values by divide-and-conquer or dynamic programming using $\mathrm{poly}(n)2^{n-s}$ arithmetic operations [Yat37, Wil14c] on $\mathrm{poly}(n)$-bit numbers. The total time is $2^{n-n/O(k^{4/3}c^{1/3}\log(kc))}$. As before, the error probability can be lowered by taking the majority values over $O(n)$ repetitions, and the original problem can be solved by calling the decision algorithm for at most cn times. □

Theorem 8.14. *Given a CNF formula with n variables and $cn \ll n^4/\log^{10} n$ clauses, we can find an assignment that satisfies the maximum number of clauses in randomized $2^{n-n/O(c^{1/3}\log^{7/3} c)}$ time.*

Proof. We use a standard width reduction technique [SST15] originally observed by Schuler [Sch05] and studied closely by Calabro, Impagliazzo, and Paturi [CIP06]. Consider the following recursive algorithm:

- If all clauses have length at most k, then call the algorithm from Theorem 8.13 and return its output.
- Otherwise, pick a clause $(\alpha_1 \vee \cdots \vee \alpha_\ell)$ with $\ell > k$. Return "SAT" if at least one of the two following calls return "SAT":
  - Recursively solve the instance in which $(\alpha_1 \vee \cdots \vee \alpha_\ell)$ is replaced by $(\alpha_1 \vee \cdots \vee \alpha_k)$, and
  - recursively solve the instance in which $\alpha_1, \ldots, \alpha_k$ are all assigned *false*.

Sakai, Seto, and Tamaki's analysis for MAX-SAT [SST15] can be directly modified to show that the total time of this algorithm remains $2^{n-n/O(k^{4/3}c^{1/3}\log(kc))}$, when the parameter k is set to be a sufficiently large constant times $\log c$. □

For MAX-k-SAT with $k \le 4$, we can obtain a much better dependency on the sparsity parameter c; in fact, we obtain significant speedup even for general dense instances. The approach this time requires only our probabilistic polynomials for threshold functions. Naively, the dense case seems to require threshold functions with superlinearly many arguments, but by incorporating a few new ideas, we manage to solve MAX-4-SAT using only $O(n)$-variate threshold functions.

Theorem 8.15. *Given a weighted 4-CNF formula F with n variables with positive integer weights bounded by $\mathrm{poly}(n)$, we can find an assignment that maximizes the* *total weight of clauses satisfied in F, in randomized $2^{n-n/O(\log^2 n \log^2 \log n)}$ time. In the sparse case when the clauses have total weight cn, the time bound improves to $2^{n-n/O(\log^2 c \log^2 \log c)}$.*

Proof. **(Dense case.)** Let $s = \alpha n$ for some parameter α to be set later. Arbitrarily divide the n variables of F into three groups: $x = \{x_1, \ldots, x_{(n-s)/2}\}$, $y = \{y_1, \ldots, y_{(n-s)/2}\}$, and $z = \{z_1, \ldots, z_s\}$. As in Theorem 8.13, it suffices to solve the decision problem of whether there exist $x, y \in \{0,1\}^{(n-s)/2}$ and $z \in \{0,1\}^s$ such that $f(x, y, z) > t$, for a given degree-4 polynomial f and a fixed $t \in [n^{c_0}]$ (for an appropriately large constant c_0). Since f has degree 4, observe that each term has either (a) at most one y variable, (b) at most one x variable, or (c) no z variable. We can thus write

$$f(x, y, z) = \sum_{i=1}^{(n-s)/2} f_i(x, z) y_i + \sum_{i=1}^{(n-s)/2} g_i(y, z) x_i + h(x, y)$$

where the f_i's and g_i's are degree-3 polynomials, and h is a degree-4 polynomial.

For every $x, y \in \{0,1\}^{(n-s)/2}$, it suffices to compute

$$F(x, y) := \sum_{z \in \{0,1\}^s} [f(x, y, z) > t]\,.$$

More generally, we compute for every $t' \in [n^{c_0}]$:

$$G_{t'}(x, y) := \sum_{z \in \{0,1\}^s} H_{z,t'}(x, y),$$

with

$$H_{z,t'}(x, y) := \left[\sum_{i=1}^{(n-s)/2} f_i(x, z) y_i + \sum_{i=1}^{(n-s)/2} g_i(y, z) x_i > t'\right].$$

Then $F(x, y) = G_{t-h(x,y)}(x, y)$; we can precompute all $h(x, y)$ values in $\text{poly}(n)2^{n-s}$ time.

The $H_{z,t'}(x, y)$ predicate can be viewed as a *weighted* threshold function with $O(n)$ arguments. To further complicate matters, these weights are not fixed: they depend on x and y. We resolve the issue by extending the vectors x and y and using a binary representation trick.

For each vector $x \in \{0,1\}^{(n-s)/2}$, define an *extended vector* x^* where $x^*_i = x_i$ for each $i = 1, \ldots, (n-s)/2$ and $x^*_{i,j,z}$ is the j-th least significant bit in the binary representation of $f_i(x, z)$ for each $i = 1, \ldots, (n-s)/2$, $j = 0, \ldots, \ell$ and $z \in \{0,1\}^s$, with $\ell = O(\log n)$. Note that x^* is a vector in $O(n \cdot \log n \cdot 2^s)$ dimensions. Similarly, for each vector $y \in \{0,1\}^{(n-s)/2}$, define an extended vector y^* where $y^*_i = y_i$ for each $i = 1, \ldots, (n-s)/2$ and $y^*_{i,j,z}$ is the j-th least significant bit in the binary representation of $g_i(y, z)$ for each $i = 1, \ldots, (n-s)/2$, $j = 0, \ldots, \ell$ and $z \in \{0,1\}^s$. We can precompute all extended vectors in $2^{(n-s)/2} \cdot \text{poly}(n)2^s$ time.

Then

$$H_{z,t'}(x,y) := \sum_{(t_0,\ldots,t_\ell)} \prod_{j=0}^{\ell} \left[\sum_{i=1}^{(n-s)/2} x^*_{i,j,z} y_i + \sum_{i=1}^{(n-s)/2} y^*_{i,j,z} x_i = t_j \right],$$

where the outer sum is over all tuples $(t_0,\ldots,t_\ell) \in [n^{c_0}]^\ell$ with $\sum_{j=0}^{\ell} 2^j \cdot t_j > t'$.

By Fact 7.6.2, for each $z \in \{0,1\}^s$, $j = 0,\ldots,\ell$, and $t_j \in [n^{c_0}]$, we can construct a probabilistic polynomial (over $\mathbb{R}$ or $\mathbb{F}_2$) for the predicate $\left[\sum_i x^*_{i,j,z} y_i + \sum_i y^*_{i,j,z} x_i = t_j\right]$ with degree $O(\sqrt{n \log S})$ with error probability at most $1/S$. By the union bound, the probability that there is an error for some z, j, t_j is at most $O((1/S) \cdot 2^s \cdot \log n \cdot n^{O(1)})$, which can be made at most $1/4^s$, for example, by setting $S = n^{c_0} 2^s$ for a sufficiently large constant c_0. Thus, the degree for each predicate is $O(\sqrt{ns})$ (assuming $s \geq \log n$).

For each $z \in \{0,1\}^s$ and $t' \in [n^{c_0}]$, by distributing over the product $\prod_{j=0}^{\ell}$ we can then construct a probabilistic polynomial for $H_{z,t'}(x,y)$ with degree $O(\sqrt{ns}\ell) \leq O(\sqrt{ns}\log n)$. For a fixed z and t', such a polynomial is a function of $O(n \log n)$ free variables in x^* and y^*, and therefore has at most $\binom{O(n\log n)}{O(\sqrt{ns}\log n)}$ monomials. The same bound holds for the time needed to construct the probabilistic polynomial (note the number of tuples $(t_0,\ldots,t_\ell)$ is $n^{O(\log n)}$, which is a negligible factor).

For each $t' \in [n^{c_0}]$, we can thus construct a probabilistic polynomial for $G_{t'}(x,y)$ with degree $O(\sqrt{ns}\log n)$ over x^* and y^*, with the following number of monomials:

$$\begin{aligned} 2^s \cdot \binom{O(n\log n)}{O(\sqrt{ns}\log n)} &\leq 2^{\alpha n} \cdot O\left(\frac{n \log n}{\sqrt{\alpha} n \log n}\right)^{O(\sqrt{\alpha} n \log n)} \\ &\leq 2^{\alpha n} \cdot 2^{\sqrt{\alpha} n (\log(n)) \log(1/\alpha)} \ll 2^{0.1(n-s)/2} \end{aligned}$$

by setting α to be a sufficiently small constant times $1/(\log n \cdot \log\log n)^2$. The same bound holds for the construction time.

We can rewrite the polynomial for $G_{t'}(x,y)$ as the dot product of two vectors $\phi(x^*)$ and $\psi(y^*)$ of $2^{0.1(n-s)/2}$ dimensions. The problem of evaluating $G_{t'}(x,y)$ over all $x, y \in \{0,1\}^{(n-s)/2}$ then reduces to multiplying a $2^{(n-s)/2} \times 2^{0.1(n-s)/2}$ with a $2^{0.1(n-s)/2} \times 2^{(n-s)/2}$ matrix (over $\mathbb{R}$ or $\mathbb{F}_2$), which can be done in poly$(n)2^{n-s}$ time (Lemma 8.1). The total time is $2^{n - n/O(\log^2 n \log^2 \log n)}$. □

Proof. **(Sparse case.)** If the clauses have total weight cn, we can refine the analysis above, as follows. Let μ_i and ν_i be the maximum value of $f_i(x,z)$ and $g_i(y,z)$ respectively. We know that $\sum_i(\mu_i + \nu_i) \leq cn$. The variable $x^*_{i,j,z}$ is needed only when $j \leq \log(\mu_i)$, and the variable $y^*_{i,j,z}$ is needed only when $j \leq \log(\nu_i)$. For each z, j, t_j, the probabilistic polynomial for the predicate

$$\left[\sum_i x^*_{i,j,z} y_i + \sum_i y^*_{i,j,z} x_i = t_j\right]$$

has degree $O(\sqrt{n_j s})$, where n_j is the number of i's with $\mu_i \geq 2^j$ or $\nu_i \geq 2^j$.

Observe that $n_j = O(cn/2^j)$. It follows that the degree for the $H_{z,t'}(x,y)$

polynomial is $O(\sum_{j=0}^{\ell}\sqrt{n_j s}) = O(\sqrt{ns}\log c + \sum_{j>\log c}\sqrt{(cn/2^j)s}) = O(\sqrt{ns}\log c)$. The number of variables in $H_{z,t'}(x,y)$ is at most $O(\sum_{j=0}^{\ell} n_j) = O(n\log c + \sum_{j>\log c}(cn/2^j)) = O(n\log c)$.

Thus, the bound on the total number of monomials becomes

$$\begin{aligned} 2^s \cdot \binom{O(n\log c)}{O(\sqrt{ns}\log c)} &\leq 2^{\alpha n}\cdot O\left(\frac{n\log c}{\sqrt{\alpha}n\log c}\right)^{O(\sqrt{\alpha}n\log c)} \\ &\leq 2^{\alpha n}\cdot 2^{\sqrt{\alpha}n\log c\log(1/\alpha)} \ll 2^{0.1(n-s)/2} \end{aligned}$$

by setting α to be a sufficiently small constant times $1/(\log c\log\log c)^2$. □

8.6 Circuit Satisfiability Algorithms

In this Section, we give new algorithms for solving the SAT problem on some rather expressive circuit classes. First, we outline some notions used in all of these algorithms.

8.6.1 Satisfiability on a Cartesian Product

In intermediate stages of our SAT algorithms, we will study the following generalization of SAT, where the task is to find a SAT assignment in a "Cartesian product" of possible assignments.

Definition 8.1. *Let n be even, and let $A, B \subseteq \{0,1\}^{n/2}$ be arbitrary. The* SAT *problem on the set $A\times B$ is to determine if a given n-input circuit has a satisfying assignment contained in the set $A\times B$.*

Recall that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a linear threshold function (LTF) if there are $a_1,\ldots,a_n,t\in\mathbb{R}$ such that for all $x\in\{0,1\}^n$, $f(x)=1 \iff \sum_i a_i x_i \geq t$.

Let $\mathit{Circuit}\circ\mathsf{LTF}[Z,S]$ be the class of circuits with a layer of S LTFs at the bottom (nearest the inputs), with Z additional arbitrary gates above that layer. Let $\mathit{Circuit}\circ\mathsf{SUM}\circ\mathsf{AND}[Z,S]$ be the analogous circuit class, but with S DNFs at the bottom layer with property that each DNF always has at most *one* conjunct true for every variable assignment. (Thus we may think of the DNF as simply an *integer sum.*) We first prove that the SAT problem for $\mathit{Circuit}\circ\mathsf{LTF}$ can be reduced to the SAT problem for $\mathit{Circuit}\circ\mathsf{SUM}\circ\mathsf{AND}$, utilizing a weight reduction trick that can be traced back to Matoušek's algorithm for computing dominances in high dimensions [Mat91, Wil14b]:

Lemma 8.5. *Let $A, B\subseteq\{0,1\}^{n/2}$, with $|A|=|B|=N\leq 2^n$. Let $K\in[1,N]$ be an integer parameter. The SAT problem for $\mathit{Circuit}\circ\mathsf{LTF}[Z,S]$ circuits on the set $A\times B$ can be reduced to the SAT problem for $\mathit{Circuit}\circ\mathsf{SUM}\circ\mathsf{AND}[Z,S]$ where each DNF has at most $O(\log K)$ terms and each* AND *has fan-in at most $2\log K$, on a prescribed set $A'\times B'$ with $|A'|=|B'|=N$ and $A',B'\subseteq\{0,1\}^{2S\log K}$. The reduction has the* *property that if the latter SAT problem can be solved in time T, then the former SAT problem can be solved in time $(T + N^2 \cdot Z^2/K + N \cdot S) \cdot poly(n)$.*

Proof. For a given circuit C of type $Circuit \circ \mathsf{LTF}[Z, S]$, let the jth LTF in the bottom layer have weights $\alpha_{j,1}, \ldots, \alpha_{j,n}, t_j$. Let the assignments in A be $a_1, \ldots, a_N$, and let the assignments in B be $b_1, \ldots, b_N$. Denote the kth bit of a_i and b_i as $a_i[k]$ and $b_i[k]$, respectively.

Make $N \times S$ matrices M_A and M_B, where

$$M_A(i,j) = \sum_{k=1}^{n/2} \alpha_{j,k} \cdot a_i[k]$$

and

$$M_B(i,j) = t_j - \sum_{k=1}^{n/2} \alpha_{j,n/2+k} \cdot b_i[k].$$

The key property of these matrices is that $M_A(i,j) \geq M_B(i',j)$ if and only if the n-variable assignment $(a_i, b_{i'})$ makes the jth LTF output 1.

For each $j = 1, \ldots, S$, let L_j be the list of all $2 \cdot N$ entries in the jth column of M_A and the jth column of M_B, sorted in increasing order. Partition L_j into K contiguous parts of $O(N/K)$ entries each, and think of each part of L_j as containing a set of $O(N/K)$ assignments from $A \cup B$. (So, the partition of L_j is construed as a partition of the assignments in $A \cup B$.) There are two possible cases for a satisfying assignment to the circuit C:

1. *There is a satisfying assignment $(a_i, b_{i'}) \in A \times B$ such that for some $j = 1, \ldots, S$, a_i and $b_{i'}$ are in the same part of L_j.* By enumerating every $a_i \in A$, every $j = 1, \ldots, S$, and all $O(N/K)$ assignments $b_{i'}$ of B which are in the same part of L_j as a_i, then evaluating the circuit C on the assignment $(a_i, b_{i'})$ in $Z^2 \cdot \text{poly}(n)$ time, we can determine satisfiability for this case in $O(N \cdot N/K \cdot Z^2) \cdot \text{poly}(n)$ time. If this does not uncover a SAT assignment, we move to the second case.

2. *There is a satisfying assignment $(a_i, b_{i'}) \in A \times B$ such that for every $j = 1, \ldots, S$, a_i and $b_{i'}$ are different parts of L_j.* Then for every LTF gate $j = 1, \ldots, S$ on the bottom layer of the circuit, we claim that the j-th LTF can be replaced by a sum of $O(\log K)$ ANDs on $2 \log K$ new variables. In particular, for the j-th LTF we define one new set of $\log K$ variables which encodes the index $k = 1, \ldots, K$ such that a_i is in part k of L_j, and another set of $\log K$ variables which encodes the index k' such that $b_{i'}$ is in part k' of L_j. Then, determining $[k \geq k']$ is equivalent to determining whether $(a_i, b_{i'})$ satisfies the j-th LTF gate. Finally, note that the predicate $[k \geq k']$ can be computed by a DNF of $O(\log K)$ conjuncts. (Take an OR over all $\ell = 0, \ldots, \log K$, guessing that the ℓ-th bit is the most significant bit in which k and k' differ; we can verify that guess with a conjunction on $2 \log K$ variables.) On every possible input $(k, k') \in \{0,1\}^{2 \log K}$, the DNF has at most *one* true conjunction. Thus we can construe the OR as

simply an *integer sum* of ANDs, as desired. Preparing these new assignments for this new SAT problem takes time $O(N \cdot S) \cdot \mathrm{poly}(n)$. $\square$

8.6.2 Simulating LTFs with AC0 of MAJORITY

In our SAT algorithms, we will need a way to simulate LTFs with bounded-depth circuits with MAJORITY gates. This was also used in Williams' work on solving ACC-LTF SAT [Wil14b], as a black box. However, here we must pay careful attention to the details of the construction. In fact, we will actually have to modify the construction slightly in order for our circuit conversion to work out. Let us review the construction here, and emphasize the parts that need modification for this algorithm. Recall that MAJ denotes the majority function.

Theorem 8.16 (Follows from [MT98], Theorem 3.3)**.** *Every LTF can be computed by polynomial-size* $\mathsf{AC}^0 \circ \mathsf{MAJ}$ *circuits. Furthermore, the circuits can be constructed in polynomial time given the weights of the LTF, and the fan-in of each* MAJ *gate can be made* $n^{1+\varepsilon}$*, for every desired* $\varepsilon > 0$*, and the circuit has depth* $O(\log(1/\varepsilon))$*.*

It will be crucial for our final results that the fan-in of the MAJ gates can be made arbitrarily close to linear.

Proof. We begin by revisiting the circuit construction of Maciel and Thérien [MT98], which shows that the addition of n distinct n-bit numbers can be performed with polynomial-size $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits. The original construction of Maciel and Thérien yields MAJ gates of fan-in $\tilde{O}(n^2)$, which is too large for our purposes. We can reduce the fan-in of MAJ gates to $O(n^{1+\varepsilon})$ by setting the parameters differently in their construction. Let us sketch their construction in its entirety, then describe how to modify it.

Recall that SYM denotes the class of symmetric functions. First, we show that addition of n n-bit numbers can be done in $\mathsf{AC}^0 \circ \mathsf{SYM}$. Suppose the n-bit numbers to be added are $A_1, \ldots, A_n$, where $A_i = A_{i,n} \cdots A_{i,1}$ for $A_{j,i} \in \{0,1\}$. Maciel and Thérien partition each A_i into m blocks of ℓ bits, where $m \cdot \ell = n$. They compute the sum S_k of the n ℓ-bit numbers in each block $k = 1, \ldots, m$, i.e.

$$S_k = \sum_{i=1}^{n} \sum_{j=1}^{\ell} A_{i,(k-1)\ell+j} \cdot 2^{j-1},$$

and note that the desired sum is

$$z = \sum_{k=1}^{m} S_k \cdot 2^{(k-1)\ell}.$$

Each S_k can be represented in $\ell + \log n$ bits. Maciel and Thérien set $\ell = \log n$, so that each S_k is represented by 2ℓ bits. They then split each S_k into ℓ-bit numbers H_k and L_k such that

$$S_k = H_k \cdot 2^{\ell} + L_k.$$

Note that the "high" part H_k corresponds to the "carry bits" of S_k. They then note that if

$$y_1 := \sum_{k=1}^{m} H_k \cdot 2^{k\ell}, \quad y_2 := \sum_{k=1}^{m} L_k \cdot 2^{(k-1)\ell},$$

we have

(a) $z = y_1 + y_2$, and

(b) each bit of y_i is a function of exactly one H_k or L_k for some k. In turn, each L_k, H_k is a sum of $n \cdot \ell$ $A_{i,j}$'s where each $A_{i,j}$ is multiplied by a power of two in $[0, 2^\ell]$. Therefore, each bit of y_i can be computed by a SYM gate of fan-in at most $n \cdot \ell \cdot 2^\ell \leq n^2$.

We have therefore reduced the addition of n n-bit numbers to adding the two $O(n)$-bit numbers y_1 and y_2, with a layer of SYM gates. Adding two numbers can be easily computed in AC^0 (see for example [CFL85]), so the whole circuit is of the form $\mathsf{AC}^0 \circ \mathsf{SYM}$.

We wish to reduce the fan-in of the SYM gates to $O(n^{1+\varepsilon})$ for arbitrary $\varepsilon > 0$. To reduce the fan-in further, it suffices to find a construction that lets us reduce ℓ. Naturally, we can try to set $\ell = \varepsilon \log n$ for arbitrarily small $\varepsilon \in (0, 1)$. Without loss of generality, let us assume $1/\varepsilon$ is an integer. Then, each S_k is represented in $\ell + \log n \leq (1 + 1/\varepsilon)\ell$ bits. Let $t = 1 + 1/\varepsilon$. If we then split each S_k into t ℓ-bit numbers $T_k^{t-1}, \ldots, T_k^0$, ranging from high-order to low-order bits, we then have

$$S_k = T_k^{t-1} \cdot 2^{(t-1)\ell} + \cdots + T_k^1 \cdot 2^\ell + T_k^0.$$

Defining the t numbers

$$y_i := \sum_{k=1}^{m} T_k^i \cdot 2^{(k+i-1)\ell},$$

the desired sum is $z = \sum_{i=0}^{t-1} y_i$. Just as before, each bit of y_i is a function of exactly one T_k^i for some k, which is a sum of $n \cdot \ell$ $A_{i,j}$'s where each $A_{i,j}$ is multiplied by an integer in $[0, 2^\ell]$. Hence each bit of y_i can be computed by a SYM gate of fan-in at most $n \cdot \ell \cdot 2^\ell \leq \tilde{O}(n^{1+\varepsilon})$. So with one layer of SYM gates, we have reduced the n number n-bit addition problem to the addition of t $O(n)$-bit numbers $y_0, \ldots, y_{t-1}$. But for $t \leq \log n$, addition of t n-bit numbers can be computed by AC^0 circuits of poly(n)-size and *fixed* depth independent of t (see e.g. [Vol99, p.14-15]). This completes the description of our $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit.

Observe that each SYM gate can be easily represented by an $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MAJ}$ circuit. In particular, the OR is over all $j \in \{0, 1, \ldots, n\}$ such that the SYM gate outputs 1 when given j inputs are equal to 1, and the $\mathsf{AND} \circ \mathsf{MAJ}$ part computes $\sum_j x_j = j$. Again, the fan-in of each MAJ here is $\tilde{O}(n^{1+\varepsilon})$.

We now apply the addition circuits to show how every LTF on n variables can be represented by a polynomial-size $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit. Suppose our LTF has weights $w_1, \ldots, w_{n+1}$, computing $\sum_{j=1}^{n} w_j x_j \geq w_{n+1}$. By standard facts about LTFs, we may assume for all j that $|w_j| \leq 2^{bn \log_2 n}$ for some constant $b > 0$. Set $W = bn \log_2 n$.

Let D be a $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit for adding n W-bit numbers as described above, where each MAJ gate has fan-in $\tilde{O}(n^{1+\varepsilon})$. For all $j = 1, \ldots, n$, connect to the jth W-bit input of D a circuit which, given x_j, feeds w_j to D if the input bit $x_{i_j} = 1$, and the all-zero W-bit string if $x_j = 0$. Observe this extra circuitry is only wires, no gates: we simply place a wire from x_j to all bits of the jth W-bit input where the corresponding bit of w_j equals 1.

This new circuit D' clearly computes the linear form $\sum_{j=1}^{n} w_j x_j$. The linear form can then be compared to w_{n+1} with an AC^0 circuit, since the "less-than-or-equal-to" comparison of two integers can be performed in AC^0. Indeed, this function can be represented as a quadratic-size DNF ($\mathsf{SUM} \circ \mathsf{AND}$), as was noticed in Lemma 8.5. We now have an $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit D'' of size $\text{poly}(W, t) \leq n^b$ computing the LTF, where the MAJ gates have fan-in $\tilde{O}(n^{1+\varepsilon})$. □

8.6.3 Satisfiability for ACC of LTF of LTF

Let $\mathsf{AC}^0[d, m] \circ \mathsf{LTF} \circ \mathsf{LTF}[S_1, S_2, S_3]$ be the class of circuits with a layer of S_3 LTFs at the bottom layer (nearest the inputs), a layer of S_2 LTFs above the bottom layer, and a size S_1 $\mathsf{AC}^0[m]$ circuit of depth d above the two LTF layers.

Theorem 8.17. *For every integer $d > 0$, $m > 1$, and $\delta > 0$, there is an $\varepsilon > 0$ and an algorithm for satisfiability of $\mathsf{AC}^0[d, m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^\varepsilon}, 2^{n^\varepsilon}, n^{2-\delta}]$ circuits that runs in deterministic 2^{n-n^ε} time.*

Before giving the proof, we first sketch the ideas in this SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$. Similar to the SAT algorithm for $\mathsf{ACC}^0 \circ \mathsf{LTF}$ circuits [Wil14b], the bottom layer of LTFs can be replaced by a layer of DNFs, via a weight reduction trick. We replace LTFs in the middle layer with $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits (modifying a construction of Maciel and Thérien [MT98] to keep the fan-in of MAJ gates low), then replace these MAJ gates of $n^{2-\Theta(\delta)}$ fan-in with probabilistic $\mathbb{F}_2$-polynomials of degree $n^{1-\Theta(\delta)+\Theta(\varepsilon)}$ over a small sample space, provided by Theorem 7.4. Taking a majority vote over all samples, and observing that an $\mathbb{F}_2$-polynomial is a $\mathsf{MOD}_2 \circ \mathsf{AND}$ circuit, we obtain a $\mathsf{MAJ} \circ \mathsf{ACC}^0$ circuit, but with $2^{n^{1-O(\delta)}}$ size in some of its layers. By carefully applying known depth reduction techniques, we can convert the circuit into a depth-two circuit of size $2^{n^{1-\Omega(\varepsilon)}}$ which can then be evaluated efficiently on many inputs. (This is not obvious: applying the Beigel-Tarui depth reduction to a $2^{O(n^{1-\varepsilon})}$-size circuit would make its new size *quasi-polynomial in* $2^{O(n^{1-\varepsilon})}$, yielding an intractable bound of $2^{n^{O(1)}}$.)

We now move on to the proof of Theorem 8.17. We use the following depth-reduction theorem of Beigel and Tarui (with important constructibility issues clarified by Allender and Gore [AG94], and recent size improvements by Chen and Papakonstantinou [CP19]):

Theorem 8.18 ([BT94, AG94])**.** *Every $\mathsf{SYM} \circ \mathsf{ACC}$ circuit of size s can be simulated by a $\mathsf{SYM} \circ \mathsf{AND}$ circuit of $2^{(\log s)^{c'}}$ size for some constant c' depending only on the depth d and MODm gates of the ACC part. Moreover, the AND gates of the final circuit have only $(\log s)^{c'}$ fan-in, the final circuit can be constructed from the original in $2^{O((\log s)^{c'})}$ time, and the final symmetric function at the output can be computed in $2^{O((\log s)^{c'})}$ time.*

Proof of Theorem 8.17. Let $\varepsilon > 0$ be a parameter to be set later. The plan is to start with a circuit as specified in the theorem statement, and slowly convert into a nice form that can be evaluated efficiently on many inputs.

1. Trade Variables for Circuit Size. Our first step is standard for ACC-SAT algorithms [Wil14b, Wil14c]: given an $\mathsf{AC}^0[d,m] \circ \mathsf{LTF} \circ \mathsf{LTF}[2^{n^\varepsilon}, 2^{n^\varepsilon}, n^{2-\delta}]$ circuit C with n variables, create a copy of the circuit $C_v := C(v,\cdot)$ for all possible assignments $v \in \{0,1\}^{n^\varepsilon}$ to the first n^ε variables of C, and define

$$C'(x_{n^\varepsilon+1}, \ldots, x_n) := \bigvee_v C_v(x_{n^\varepsilon+1}, \ldots, x_n).$$

Observe that C' is satisfiable if and only if C is satisfiable, C' has size at most $2^{O(n^\varepsilon)}$, C' is also an $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuit, and C' has only $n - n^\varepsilon$ variables.

2. Replace the middle LTFs with MAJORITYs (Theorem 8.16). Note that each LTF on the second layer of C' has fan-in at most $n^{2-\delta} + n$, since the number of LTFs on the first layer is $n^{2-\delta}$. Applying the low fan-in transformation of Theorem 8.16, we can replace each of the LTFs on the second layer of C' with poly(n)-size $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits where each MAJ has fan-in at most $n^{2-\delta/2}$. This generates at most 2^{dn^ε} new MAJ gates in the circuit C', for some constant $d > 0$, and produces a circuit of type

$$\mathsf{ACC}^0 \circ \mathsf{MAJ} \circ \mathsf{LTF}.$$

3. Replace those MAJORITYs with (derandomized) probabilistic polynomials over $\mathbb{F}_2$ (Theorem 7.4). We replace each of these new MAJ gates with our low-randomness probabilistic polynomials for the MAJORITY function, as follows. Recall from Theorem 7.4 that we can construct a probabilistic polynomial over $\mathbb{F}_2$ for k-bit MAJORITY with degree $O(\sqrt{k \log(1/\varepsilon')})$ and error at most ε', using a distribution of $k^{O(\log(k/\varepsilon'))}$ uniformly chosen $\mathbb{F}_2$-polynomials. Setting $k := n^{2-\delta/2}$ for the fan-in of the MAJ gates, and the error to be $\varepsilon' := 1/2^{2dn^\varepsilon}$, the degree becomes

$$D := O\left(\sqrt{n^{2-\delta/2} \cdot 2dn^\varepsilon}\right) \le O(n^{1-\delta/4+\varepsilon/2})$$

and the sample space has size $S = n^{O(n^\varepsilon)}$. For $\varepsilon \ll \delta/4$, we have $D := O(n^{1-\delta/8})$, and each polynomial in our sample space has at most $\binom{n^{2-\delta}}{n^{1-\delta/8}} \le 2^{O(n^{1-\delta/8}\log n)}$ monomials. For every choice of the random seed r to the probabilistic polynomial, let C'_r be the circuit C' with the corresponding $\mathbb{F}_2$ polynomial P_r substituted in place of each MAJ gate. That is, each MAJ gate is substituted by an XOR of $2^{O(n^{1-\delta/8}\log n)}$ ANDs of fan-in at most $O(n^{1-\delta/8})$.

We now form a circuit C'' which takes a majority vote over all $2^{O(n^\varepsilon \log n)}$ circuits C'_r. The new circuit C'' therefore has the form

$$\mathsf{MAJ} \circ \mathsf{ACC}^0 \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{LTF},$$

where the $\mathsf{MAJ} \circ \mathsf{ACC}^0$ part has size $2^{O(n^\varepsilon \log n)}$, and each $\mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{LTF}$ subcircuit has size $2^{O(n^{1-\delta/8}\log n)}$. Since our probabilistic polynomial computes MAJORITY with

$1/2^{2dn^{\varepsilon}}$ error and there are at most $2^{dn^{\varepsilon}}$ MAJ gates in C', the new circuit C'' is equivalent to the original circuit C'.

4. Apply Beigel–Tarui to the top of the circuit, and distribute. It is very important to observe that we *cannot* apply Beigel–Tarui (Theorem 8.18) to the *entire* circuit C'', as its total size is $2^{\Omega(n^{1-\delta/8}\log n)}$, and the quasi-polynomial blowup of Beigel–Tarui would generate a huge circuit of size $\Omega(2^n)$, rendering our conversion intractable.

However, the top $\mathsf{MAJ} \circ \mathsf{ACC}^0$ part is still small. Invoking the depth reduction lemma of Beigel and Tarui (Theorem 8.18 above), we can replace the $\mathsf{MAJ} \circ \mathsf{ACC}^0$ part in C'' of size $2^{O(n^{\varepsilon}\log n)}$ (even though it has $2^{O(n^{\varepsilon}\log n)}$ inputs from the XOR layer!) with a $\mathsf{SYM} \circ \mathsf{AND}$ circuit of size $2^{n^{a\cdot\varepsilon}}$ for a constant $a \geq 1$, where each AND has fan-in at most $n^{a\varepsilon}$, and a depends only on the (constant) depth d and (constant) modulus m of the ACC^0 subcircuit.

The resulting circuit C_3 now has the form

$$\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{LTF}.$$

Applying the distributive law to the $\mathsf{AND} \circ \mathsf{XOR}$ parts, where the ANDs have fan-in at most $n^{a\varepsilon}$ and the XORs have fan-in $2^{O(n^{1-\delta/8}\log n)}$, each $\mathsf{AND} \circ \mathsf{XOR}$ parts can be converted into an $\mathsf{XOR} \circ \mathsf{AND}$ circuit of size $2^{O(n^{1-\delta/8+a\varepsilon}\log n)}$, where the fan-in of ANDs is at most $n^{a\varepsilon}$. Letting $\varepsilon \ll \delta/(ca)$ for sufficiently large $c \geq 1$, the fan-in of the new XORs is at most $2^{O(n^{1-\varepsilon})}$. We now have a circuit C_4 of the form

$$\mathsf{SYM} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{LTF}.$$

Note that the fan-in of the SYM gate is at most $2^{n^{a\cdot\varepsilon}}$, and the fan-in of the (merged) ANDs is $O(n^{1-\delta/8+a\varepsilon})$.

5. Apply modulus-amplifying polynomials to eliminate the XOR layer. We'd like to remove the XOR layer, to further reduce the depth of the circuit. But as the gates of this layer have very high fan-in, we must be careful not to blow the circuit size up to $\Omega(2^n)$. The following construction will take advantage of the fact that we have only $\text{poly}(n)$ total gates in the bottom LTF layer.

We apply one step of Beigel-Tarui's transformation [BT94] (from ACC^0 to $\mathsf{SYM} \circ \mathsf{AND}$) to the $\mathsf{SYM} \circ \mathsf{XOR} \circ \mathsf{AND}$ part of our circuit. In particular, we apply a modulus-amplifying polynomial P (over the integers) of degree $2D' = 2n^{a\cdot\varepsilon}$ to each of the $\mathsf{XOR} \circ \mathsf{AND}$ parts. Construing the $\mathsf{XOR} \circ \mathsf{AND}$ as a sum of products $\sum\prod$, the polynomial P has the property:

- If the $\sum\prod = 1 \bmod 2$, then $P(\sum\prod) = 1 \bmod 2^{D'}$.
- If the $\sum\prod = 0 \bmod 2$, then $P(\sum\prod) = 0 \bmod 2^{D'}$.

So, composing P with each $\mathsf{XOR} \circ \mathsf{AND}$ part, each P outputs either 0 or 1 modulo $2^{n^{a\cdot\varepsilon}}$. The key property here is that the modulus exceeds the fan-in of the SYM gate, so the sum of all $P(\sum\prod)$ simply counts the number of $\mathsf{XOR} \circ \mathsf{AND}$s which are true; this is enough to determine the output of the SYM gate. Construing the output of each bottom LTF gate as a variable, there are at most $2^{n^{\varepsilon}} \cdot n^{2-\varepsilon}$ variables ($n^{2-\varepsilon}$ for each of the $2^{n^{\varepsilon}}$ copies of the circuit from step 1 above). Expressing each $P(\sum\prod)$

(expanded as a sum of products) as a multilinear polynomial in these LTF variables, the total number of terms is at most

$$\binom{2^{n^\varepsilon} \cdot n^{2-\varepsilon}}{D' \cdot n^{1-\delta/8+a\varepsilon}} \leq 2^{O(D' \cdot n^{1-\delta/8+a\varepsilon} \cdot n^\varepsilon)} \leq 2^{O(n^{2a\cdot\varepsilon+1-\delta/8+\varepsilon})}.$$

Let $\varepsilon := \delta/(ca)$ for a sufficiently large constant $c > 1$ so that $2a\varepsilon + 1 - \delta/8 + \varepsilon < 1 - \varepsilon$ (it suffices to pick any $c > 16(1+1/a)$). We can then merge the sum of all $P(\sum\prod)$'s into the SYM gate, and obtain a $\mathsf{SYM} \circ \mathsf{AND}$ circuit where the SYM has fan-in

$$2^{O(n^{2a\cdot\varepsilon+1-\delta/8+\varepsilon})} \leq 2^{O(n^{1-\varepsilon})},$$

and the AND gates have fan-in $O(n^{2a\cdot\varepsilon+1-\delta/8+\varepsilon}) \leq O(n^{1-\varepsilon})$. The result is a circuit C_4 of the form

$$\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{LTF}.$$

6. Replace the bottom threshold gates with DNFs (Theorem 8.5), and distribute. Note that the circuit C_4 has $n - n^\varepsilon$ variables, so our SAT algorithm would follow if we could evaluate C_4 on all of its variable assignments in $2^{n-n^\varepsilon} \cdot \text{poly}(n)$ time. We are now in a position to apply Lemma 8.5, which lets us reduce the evaluation problem for $\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{LTF}$ circuits to the evaluation problem for $\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{SUM} \circ \mathsf{AND}$ circuits, with a parameter K that needs setting. Recall the middle AND gates have fan-in $O(n^{1-\varepsilon})$, and the fan-in of the SUM is $O(\log K)$. Therefore by the distributive law, we can rewrite the circuit as a $\mathsf{SYM} \circ \mathsf{SUM} \circ \mathsf{AND}$ circuit, where each SUM gate has $(\log K)^{O(n^{1-\varepsilon})}$ ANDs below it, and at most *one* AND below each SUM is true. Thus we can wire these AND gates directly into the top SYM gate without changing the output.

In more detail, let $A, B = \{0,1\}^{(n-n^\varepsilon)/2}$, and set $N = 2^{(n-n^\varepsilon)/2}$ and the integer parameter $K := 2^{b\cdot n^{1-\varepsilon}}$ for a sufficiently large constant $b > 1$. By Lemma 8.5, we can reduce the SAT problem for $\mathsf{SYM} \circ \mathsf{AND} \circ \mathsf{LTF}$ circuits of size $2^{O(n^{1-\varepsilon})}$ on the set $A \times B = \{0,1\}^{n-n^\varepsilon}$ to the SAT problem for $\mathsf{SYM} \circ \mathsf{SUM} \circ \mathsf{AND}$ circuits of size

$$2^{O(n^{1-\varepsilon})} \cdot 2^{2bn^{1-\varepsilon}} \cdot n^{2-\delta} \leq 2^{O(n^{1-\varepsilon})}$$

on a prescribed set $A' \times B'$ with $|A'| = |B'| = N$ and $A', B' \subseteq \{0,1\}^{2bn^{2-\delta}\cdot n^{1-\varepsilon}}$. By the distributive argument from the previous paragraph, we can convert the $\mathsf{SYM} \circ \mathsf{SUM} \circ \mathsf{AND}$ circuit into a $\mathsf{SYM} \circ \mathsf{AND}$ circuit of size at most

$$2^{O(n^{1-\varepsilon})} \cdot 2^{O(n^{1-\varepsilon}\log\log K)} \leq 2^{O(n^{1-\varepsilon}\log(n))}.$$

By Lemma 8.5, we know that if the $\mathsf{SYM} \circ \mathsf{AND}$ SAT problem is solvable in time T on the set $A' \times B'$, then the SAT problem for C_4 on the set $A \times B$ can be solved in time $O\left(T + N^2 \cdot Z/K + N \cdot S\right) \cdot \text{poly}(n)$.

7. Evaluate the depth-two circuit on many pairs of points. By applying fast rectangular matrix multiplication in a now-standard way [Wil14c, Wil14b], the resulting $\mathsf{SYM} \circ \mathsf{AND}$ circuit of $2^{\tilde{O}(n^{1-\varepsilon})}$ size can be evaluated on all points in $A' \times B'$, in time $\mathrm{poly}(n) \cdot 2^{n-n^{\varepsilon}}$, thus solving its SAT problem. Therefore, the SAT problem for C_4 can be solved in time

$$\mathrm{poly}(n) \cdot 2^{n-n^{\varepsilon}} + \frac{2^{n-n^{\varepsilon}} \cdot 2^{O(n^{1-\varepsilon})}}{2^{b \cdot n^{1-\varepsilon}}} + 2^{\frac{n-n^{\varepsilon}}{2}} \cdot 2^{O(n^{1-\varepsilon} \log(n))}.$$

Setting $b > 1$ to be sufficiently large, we obtain a SAT algorithm for C_4 (and hence the original circuit C) running in $\mathrm{poly}(n) \cdot 2^{n-n^{\varepsilon}}$ time. □

8.6.4 Satisfiability for Three Layers of Majority and AC0

In this section, we give our SAT algorithm for $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits with low-polynomial fan-in at the output gate and the middle LTF layer:

Theorem 8.19. *For all $\varepsilon > 0$ and integers $d \geq 1$, there is a $\delta > 0$ and a randomized satisfiability algorithm for $\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of depth d running in $2^{n-\Omega(n^{\delta})}$ time, on circuits with the following properties:*

- *the top MAJ gate, along with every LTF on the middle layer, has $O(n^{6/5-\varepsilon})$ fan-in, and*
- *there are $O(2^{n^{\delta}})$ many AND/OR gates (anywhere) and LTF gates at the bottom layer.*

We need one more result concerning probabilistic polynomials over the integers:

Theorem 8.20 ([BRS91, Tar93]). *For every AC^0 circuit C with n inputs and size s, there is a distribution of n-variate polynomials $\mathcal{D}$ over $\mathbb{Z}$ such that every p has degree poly$(\log s)$ (depending on the depth of C) and for all $x \in \{0,1\}^n$, $\Pr_{p \sim \mathcal{D}}[C(x) = p(x)] \geq 1 - 1/2^{poly(\log s)}$.*

Proof of Theorem 8.19. The SAT algorithm is somewhat similar in structure to Theorem 8.17, but with a few important changes. Most notably, we work with probabilistic polynomials over $\mathbb{Z}$ instead of $\mathbb{F}_2$.

Start with a circuit C of the required form. Let s be the number of AND/OR gates in C plus the number of LTF gates on the bottom layer. Let $f \leq n^{6/5-\varepsilon}$ be the maximum fan-in of the top MAJ gate and the LTFs on the middle layer, and recall that we're planning to consider C with size at most $2^{n^{\delta}}$ where $\delta > 0$ is a sufficiently small constant (depending on $\varepsilon > 0$ and the circuit depth) in the following. Our SAT algorithm runs as follows:

1. By Theorem 8.16, every LTF of fan-in f can be replaced by an $\mathsf{AC}^0 \circ \mathsf{MAJ}$ of fan-in $f^{1+o(1)}$ and $\mathrm{poly}(f)$ size. Hence we can reduce C to a circuit of similar size, but of the form

   $$\mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{MAJ} \circ \mathsf{AC}^0 \circ \mathsf{MAJ}.$$

   The fan-ins of the majority gates in the middle and bottom layer can be made at most $n^{6/5-\varepsilon'}$, for any $\varepsilon' > 0$ which is smaller than ε. To be concrete, let us set $\varepsilon' := \varepsilon/2$.

2. Replace the "middle" majority gates of fan-in $n^{6/5-\varepsilon/2}$ with probabilistic polynomials (over $\mathbb{Z}$) of degree $n^{3/5-\varepsilon/4}\text{poly}(\log s)$ and error $1/2^{\text{poly}(\log s)}$ [AW15] (Theorem 7.4 from the previous Chapter). Replace all the AC^0 subcircuits of size s by probabilistic polynomials (over $\mathbb{Z}$) of degree $\text{poly}(\log s)$ and error $1/2^{\text{poly}(\log s)}$, via Lemma 8.20. Note that the latter $\text{poly}(\log s)$ factor depends on the depth of the circuit.

3. Replace the majority gate at the output (of fan-in $f \leq n^{6/5-\varepsilon}$) with the *probabilistic* PTF of Corollary 7.4, setting the threshold parameter s' (which is called s in the statement of the corollary) to be $2^{2n^{\delta}}$ and setting the error (called ε in the statement of the corollary) to be $1/f$. The resulting polynomial has degree $n^{2/5-\varepsilon/3} \cdot \text{poly}(n^{\delta})$.

   Applying the distributive law to all the polynomials from steps 2 and 3, the new circuit C' can be viewed as an *integer sum* of at most T $\mathsf{AND} \circ \mathsf{LTF}$ circuits of at most T size, where

   $$T = 2^{n^{3/5-\varepsilon/4} \cdot n^{2/5-\varepsilon/3} \cdot \text{poly}(\log s, n^{\delta})} = 2^{n^{1-7\varepsilon/12} \cdot \text{poly}(\log s, n^{\delta})}$$

   and all AND gates have fan-in at most $n^{1-7\varepsilon/12} \cdot \text{poly}(\log s, n^{\delta})$ (because the resulting polynomial has at most this degree).

   Now is a good time to mention our choice of δ, as it will considerably clean up the exponents in what follows. We will choose $\delta > 0$ to be sufficiently small so that the $\text{poly}(\log s, n^{\delta})$ factor in the exponent of T is less than $n^{\varepsilon/12}$. That is, we take $\delta := \varepsilon/c$ and the size parameter $s < 2^{n^{\delta}} = 2^{n^{\varepsilon/c}}$, for a sufficiently large constant $c \geq 12$. (Note that c depends on the depth of the circuit, since the degree of the poly log factor depends on the depth.) Thus we have the size bound

   $$T = 2^{n^{1-7\varepsilon/12} \cdot \text{poly}(\log s, n^{\delta})} \leq O(2^{n^{1-7\varepsilon/12} \cdot n^{\varepsilon/12}}) \leq O(2^{n^{1-\varepsilon/2}}),$$

   and all AND gates have fan-in at most $n^{1-\varepsilon/2}$.

4. For all assignments a to the first n^{δ} variables of C', plug a into C', creating a copy C'_a. Let C'' be the integer sum of all $2^{n^{\delta}}$ circuits C'_a. By the properties of the polynomial constructed in Theorem 7.6 and the chosen parameter $s' = 2^{2n^{\delta}}$, with probability at least $2/3$ there is a (computable) threshold value $v = 3s/2$ such that

   - $C''(x) > v$ when at least one $C'_a(x)$ outputs 1, and
   - $C''(x) < v$ when all $C'_a(x)$ output 0.

   The circuit C'' is a Sum-of-$\mathsf{AND} \circ \mathsf{LTF}$ circuit; note that C'' has $n - n^{\delta}$ variables.

5. We now want to evaluate C'' on all of its $2^{n-n^{\delta}}$ possible variable assignments. Applying Lemma 8.5 for an integer parameter $K \in [2^n]$ (to be determined), $N = 2^{(n-n^{\delta})/2}$, and $Z, S = 2^{n^{1-\varepsilon/2}}$, we can convert this evaluation problem for C'' into a corresponding evaluation problem for a Sum-of-$\mathsf{AND} \circ \mathsf{SUM} \circ \mathsf{AND}$

circuit C''', on an appropriate combinatorial rectangle $A' \times B'$ of 2^{n-n^δ} variable assignments in total. The relative size of the circuit is unchanged, as each $\mathsf{SUM} \circ \mathsf{AND}$ has size $O(\log^2 K) \leq O(n^2)$. The time for conversion of C'' into C''' is

$$\left(\frac{N^2 Z^2}{K} + N \cdot S\right) \cdot \mathrm{poly}(n) \leq \frac{2^{n-n^\delta} \cdot 2^{2n^{1-\varepsilon/2}} \cdot \mathrm{poly}(n)}{K}.$$

Setting $K := 2^{2n^{1-\varepsilon/2}}$ makes this time bound $2^{n-\Omega(n^\delta)}$.

Recall that in the Sum-of-$\mathsf{AND} \circ \mathsf{SUM} \circ \mathsf{AND}$ circuit C''', the fan-in of the middle ANDs is at most $n^{1-\varepsilon/2}$, and each SUM has $O(n)$ fan-in. We can therefore apply the distributive law to each $\mathsf{AND} \circ \mathsf{SUM}$ part, and obtain a $\mathsf{SUM} \circ \mathsf{AND}$ of size at most $n^{O(n^{1-\varepsilon/2})}$. Merging the SUMs into the SYM gate, we obtain a $\mathsf{SYM} \circ \mathsf{AND}$ circuit of size at most $n^{O(n^{1-\varepsilon/2})}$.

6. Finally, applying rectangular matrix multiplication (Lemma 8.1) we can evaluate the Sum-of-AND C''' of $n^{O(n^{1-\varepsilon/2})}$ size on the combinatorial rectangle $A' \times B'$ in $2^{n-\Omega(n^\delta)}$ time, by preparing matrices of dimensions $2^{n/2-\Omega(n^\delta)} \times n^{O(n^{1-\varepsilon/2})}$ (for A') and $n^{O(n^{1-\varepsilon/2})} \times 2^{n/2-\Omega(n^\delta)}$ (for B'), then multiplying them. Note that preparing these matrices takes time no more than $2^{n/2+O(n^{1-\varepsilon/2}\log n)}$, which is negligible for us.

   After multiplying the matrices, we obtain a value for $C''(x)$ for each assignment x, which is correct with probability at least $2/3$. By repeating steps 2-5 for $100n$ times, we obtain correct values on all 2^{n-n^δ} points with high probability.

This completes the proof. □

Part III

Probabilistic Rank and Matrix Rigidity

Chapter 9

Background and Overview

Let R be any commutative ring. The rank-r rigidity of a matrix $H \in R^{N \times N}$, denoted $\mathscr{R}_H(r)$, is the minimum Hamming distance between H and any matrix of rank at most r. Ever since Leslie Valiant introduced the notion of matrix rigidity [Val77], it has been a major challenge to construct interesting rigid matrices. Valiant and other complexity theorists have shown that explicit rigid matrices would yield new lower bounds for a number of different models of computation. The two most interesting rigidity parameter regimes for a family $\{M_N\}_{N\in\mathbb{N}}$ of matrices, where M_N is a $N \times N$ matrix, are as follows:

- $\{M_N\}_{N\in\mathbb{N}}$ is called *Valiant-rigid* if there is a constant $\varepsilon > 0$ such that

  $$\mathscr{R}_{M_N}(N/\log\log N) \geq \Omega(N^{1+\varepsilon}).$$

  Valiant [Val77] showed that the linear transformations corresponding to Valiant-rigid matrices cannot be computed by $O(N)$-size $O(\log N)$-depth arithmetic circuits. There are currently no known lower bounds showing that such circuits cannot compute any explicit families of matrices.

- $\{M_N\}_{N\in\mathbb{N}}$ is called *Razborov-rigid* if there is any super-constant function $\alpha(N) = \omega(1)$ such that

  $$\mathscr{R}_{M_N}(2^{(\log\log N)^{\alpha(N)}}) \geq \Omega(N^2).$$

  Razborov [Raz89] (see also [Wun12]) showed that if the communication matrix M_f of a Boolean function f is Razborov-rigid, then f is not in $\mathsf{PH}^{\mathsf{cc}}$, the communication analogue of the polynomial hierarchy. There are currently no explicit Boolean functions known to be outside $\mathsf{PH}^{\mathsf{cc}}$.

We say $\{M_N\}_{N\in\mathbb{N}}$ is *explicit* if there is a deterministic algorithm which, on input N, outputs the matrix M_N in poly(N) time. Aiming for a deterministic algorithm is important, since random matrices are known to be very rigid with high probability. Indeed, a random such matrix R_N has $\mathscr{R}_{R_N}(r) \geq \Omega\left(\frac{(N-r)^2}{\log N}\right)$ for all r with high probability[1] [Val77].

Despite decades of work and many known applications of rigid matrices, there has not been much success in actually constructing rigid matrices for almost any interesting rank parameter. There are essentially only three known deterministic constructions:

- For all ranks r, there is a family of $N \times N$ matrices M_N constructible in P with $\mathscr{R}_{M_N}(r) \geq \Omega\left(\frac{N^2}{r}\log(N/r)\right)$ [Fri93, SSS97]. This is proved via a combinatorial argument ("untouched minor argument"), and it is known that this type of approach cannot be further improved [Lok00].

- A counting argument shows there is a family of $N \times N$ matrices R_N over a finite field $\mathbb{F}_q$ with $\mathscr{R}_{R_N}(r) \geq \Omega(N^2)$ for all $r = o(N)$. By combining a brute-force search for such rigid matrices with a padding argument (see Lemma 11.5 below), we can construct an $N \times N$ matrix L_N in $\mathsf{TIME}[\exp(r^2)]$ with $\mathscr{R}_{L_N}(r) \geq \Omega(N^2)$.

- Goldreich and Tal [GT16] show that random $N \times N$ Toeplitz matrices T_N over a finite field $\mathbb{F}_q$ have $\mathscr{R}_{T_N}(r) \geq \Omega\left(\frac{N^3}{r^2 \log N}\right)$ for all $r \geq \sqrt{N}$ with high probability. Their proof is primarily combinatorial and linear algebraic. Since random $N \times N$ Toeplitz matrices over $\mathbb{F}_2$ are defined by $O(N)$ random bits, such rigid matrices can be constructed in E^{NP}.

Over large fields $\mathbb{F}$, there are also approaches to constructing matrices which are rigid by virtue of having very large entries. For instance, an 'algebraic dimension' approach [SS96] can be used to construct rigid matrices over $\mathbb{C}$ with algebraically independent entries [Lok00, Lok06]. In this dissertation, we focus mainly on matrix rigidity over constant-size finite fields $\mathbb{F}_{p^r}$ where such techniques cannot work.

In this Part of the dissertation, we make new progress on the problem of constructing rigid matrices by using new techniques which haven't yet been used in this area. First, in Chapter 10, we define a generalization of the polynomial method involving a new variant on the rank of a matrix called *probabilistic rank*, and use it to give a number of new rigidity upper bounds. We will give rigidity upper bounds for matrices which were previously conjectured to be very rigid, and give new connections between rigidity, circuit complexity, and communication complexity. Our probabilistic rank constructions use our probabilistic polynomial constructions from earlier in Chapter 7, together with the connection between sparse polynomials and low-rank matrices we explored in Chapter 8

Second, in Chapter 11, we give a new construction of rigid matrices. Our construction makes use of ideas from circuit complexity theory which hadn't been used before in this context, and gives the first nontrivial construction of a family of Razborov-rigid matrices (although it is not an 'explicit' family of matrices as we discuss shortly). Interestingly, both our new rigidity upper bounds and lower bounds make use of the polynomial method: our new construction of rigid matrices critically uses a polynomial method algorithm for counting orthogonal vectors.

[1] By comparison, one can see that $\mathscr{R}_{R_N}(r) \leq (N-r)^2$ for all r and all $N \times N$ matrices R_N.

9.1 Our Results

9.1.1 Probabilistic Rank and Matrix Rigidity

Let R be a commutative ring. In analogy with the notion of a probabilistic polynomial, we define a *probabilistic matrix over* R to be a distribution of matrices $\mathcal{M} \subset R^{n \times n}$. A probabilistic matrix $\mathcal{M}$ *computes* a matrix $A \in R^{n \times n}$ with error $\varepsilon > 0$ if for every entry $(i, j) \in [n]^2$,

$$\Pr_{B \sim \mathcal{M}}[A[i,j] = B[i,j]] \geq 1 - \varepsilon.$$

In this way, a probabilistic matrix is a *worst-case randomized representation* of a fixed matrix. A probabilistic matrix $\mathcal{M}$ has rank r if the maximum rank of a $M \sim \mathcal{M}$ is r.

We define the *ε-probabilistic rank* of a matrix $M \in R^{n \times n}$ to be the minimum rank of a probabilistic matrix computing M with error ε. Such probabilistic matrices are of interest and potentially very useful, because some full rank matrices can be represented by probabilistic matrices of rather low rank. For example, every identity matrix has ε-probabilistic rank $O(1/\varepsilon)$ over any field, by simulating a protocol for EQUALITY using $\log(1/\varepsilon) + O(1)$ communication that computes random inner products (cf. Theorem 10.5).

Probabilistic rank is related to the use of probabilistic polynomials in algorithm design. We saw in Chapter 8 how substituting low-degree probabilistic polynomials in place of common subroutines can be very useful for speeding up the best known running times for many core problems. All our algorithmic applications ended up embedding the low-degree polynomial evaluation problem in a *fast multiplication of two low-rank (rectangular) matrices* (see Lemma 8.2 above). That is, this algorithmic work is really using the fact that that various circuits and subroutines from core algorithms have *low probabilistic rank*, and is applying low-rank representations to obtain an algorithmic speedup. Because "low probabilistic rank" is potentially a far broader notion than that of "low-degree probabilistic polynomials", it makes more sense to study probabilistic rank directly, in the hopes of finding stronger algorithmic applications.

Probabilistic rank is also very related to matrix rigidity: it is not hard to see that probabilistic rank upper bounds imply rigidity upper bounds. In Chapter 10, we consider complexity-theoretic aspects of probabilistic rank. We demonstrate how probabilistic rank is a powerful notion for understanding matrix rigidity, and some models of communication complexity where knowledge is still sparse.

Hadamard Ain't So Rigid. Among the many attempts to prove arithmetic circuit lower bounds via rigidity, perhaps the most commonly studied explicit matrix has been the Walsh-Hadamard transform [PS88, Alo90, Gri, Nis, KR98, Cod00, Lok01, LTV03, Mid05, dW06b, Ras16]:

Definition 9.1. *For vectors $x, y \in \mathbb{R}^d$, let $\langle x, y \rangle$ denote their inner product. Let $v_1, \ldots, v_{2^n} \in \{0,1\}^n$ be the enumeration of all n-bit vectors in lexicographical order. The* Walsh-Hadamard matrix *H_n is the $2^n \times 2^n$ matrix defined by $H_n(v_i, v_j) := (-1)^{\langle v_i, v_j \rangle}$.*

It was believed that H_n is rigid because its rows are mutually orthogonal (i.e., H_n is Hadamard), so in several of the above references, only that property was assumed of the matrices. The best rigidity lower bounds known for H_n have the form $\mathcal{R}_{H_n}(r) \geq \Omega(4^n/r)$; for the target rank $r = O(2^n/\log n)$ in Valiant's problem, the lower bound is only $\Omega(2^n \log n)$. It was a folklore theorem that one can modify only $O(n)$ entries of an $n \times n$ Hadamard matrix and make its rank at most $n/2$ [Lok14], but it was believed that for lower rank many more entries would require modification.

We give a good excuse for the weakness of these lower bounds:

Theorem 9.1 (Non-Rigidity of Hadamard Matrices)**.** *For every commutative ring R, for every sufficiently small $\varepsilon > 0$, and for all n, we have $\mathcal{R}_{H_n}\left(2^{n-f(\varepsilon)n}\right) \leq 2^{n(1+\varepsilon)}$ over R, for a function f where $f(\varepsilon) = \Theta(\varepsilon^2/\log(1/\varepsilon))$.*

In fact, we show a strong non-rigidity upper bound: by modifying at most $2^{\varepsilon n}$ entries in each row of H_n, the rank of H_n drops to $2^{n-f(\varepsilon)n}$. That is, the matrix rigidity approach to arithmetic circuit lower bounds *does not* apply to Hadamard matrices such as the Walsh-Hadamard transform, since it is *not* Valiant-rigid. We would have required lower bounds of the form $\mathcal{R}_{H_n}(2^n/(\log n)) \geq 2^{n(1+\varepsilon)}$ for some $\varepsilon > 0$ to obtain circuit lower bounds, but the upper bound of Theorem 9.1 shows this is impossible.

We do not (yet) believe that the Walsh-Hadamard transform has $O(2^n)$-size $O(n)$-depth circuits; a more appropriate conclusion is that rigidity is too coarse to adequately capture the lower bound problem in this case. Having said that, Theorem 9.1 does imply new circuit constructions: it follows that there is a depth-two unbounded fan-in arithmetic circuit for the Walsh-Hadamard transform with $2^{n+O(\varepsilon\log(1/\varepsilon))n} + 2^{2n-\Omega(\varepsilon^2 n)}$ gates; setting $\varepsilon > 0$ appropriately, we have a $4^{\delta n}$-size circuit for some $\delta < 1$.

We also show non-trivial rigidity upper bounds for H_n in the Razborov-rigidity regime that would be useful for communication complexity, where the rigidity is much closer to 4^n.

Theorem 9.2 (Non-Rigidity of Hadamard Matrices, Part II)**.** *For every integer $r \in [2^{2n}]$, one can modify at most $2^{2n}/r$ entries of H_n and obtain a matrix of rank $(n/\ln(r))^{O(\sqrt{n\log(r)})}$.*

While the product of rank and rigidity (a natural measure) of H_n is only known to be at least $\Omega(4^n)$, Theorem 9.2 provides an upper bound of $4^n \cdot n^{O\left(\sqrt{n\log(r)}\right)}/r$. This is not small enough to refute the conjectured rigidity lower bounds required for communication complexity applications, but as we show later, these upper bounds still have non-trivial consequences for the communication complexity of IP2 (the Inner Product Modulo 2 function).

New Applications of Explicit Rigid Matrices. Rigidity has been studied primarily for its connections to communication complexity and to lower bounds on arithmetic circuits computing linear transformations. We show new implications of constructing explicit rigid matrices for Boolean circuit complexity.

First, we show how explicit rigidity lower bounds would yield Boolean circuit lower bounds where only somewhat weak results are known:

Theorem 9.3. *Let R be an arbitrary commutative ring, and $\{M_n\}$ be a family of Boolean matrices such that (a) M_n is $n \times n$, (b) there is a poly$(\log n)$ time algorithm A such that $A(n,i,j)$ prints $M_n(i,j)$, and (c) there is a $\delta > 0$ such that for infinitely many n,*

$$\mathcal{R}_{M_n}\left(2^{(\log n)^{1-\delta}}\right) \geq \frac{n^2}{2^{(\log n)^{\delta/2}}} \text{ over } R.$$

Then the language $\{(n,i,j) \mid M_n(i,j) = 1\} \in \mathsf{P}$ does not have $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of $n^{2-\varepsilon}$-size and $o(\log n / \log\log n)$-depth, for all $\varepsilon > 0$.

The theorem is obtained by giving non-trivial probabilistic rank bounds for such circuits, building on Lokam [Lok01]. Therefore, proving rigidity (or probabilistic rank) lower bounds for explicit $0/1$ matrices over a commutative ring R would imply nearly-quadratic size lower bounds for $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of unbounded depth, a powerful class of Boolean circuits. The best known lower bounds, which we proved above in Corollary 6.1 and Theorem 8.17, are that functions in the huge class E^{NP} do not have such circuits.

Sign-Rank Rigidity. The sign rank of a $-1/1$ matrix M is the lowest rank of a matrix N over $\mathbb{R}$ such that $\text{sign}(M[i,j]) = \text{sign}(N[i,j])$, for all (i,j). Lower bounds on the sign-rank of matrices were used 15 years ago to prove exponential lower bounds against $\mathsf{LTF} \circ \mathsf{MAJ}$ and $\mathsf{LTF} \circ \mathsf{SYM}$ circuits [For02, FKL$^+$01], i.e. restricted versions of depth-two threshold circuits. We extend the sign-rank connection to a circuit class for which strong lower bounds have long been open: explicit matrices with high rigidity under sign-rank would imply strong depth-two threshold circuit lower bounds. (Here, sign-rank rigidity is defined in the natural way, with "rank" replaced with "sign-rank" in the rigidity definition.)

A corollary of a theorem of Razborov and Sherstov [RS10] (see Theorem 10.13 below) is that for all n, H_n has sign-rank r-rigidity at least $\Omega(4^n/r)$, just as in the case of normal rank rigidity. We show that even a somewhat minor improvement would already imply exponential-size lower bounds for depth-two linear threshold circuits with unbounded weights on both layers, a problem open for decades [HMP$^+$93, KW16]:

Theorem 9.4. *Suppose the sign rank r-rigidity of H_n is $\Omega(4^n/r^{.999})$ for some rank bound $r \geq 2^{\alpha n}$ and some $\alpha > 0$. Then the Inner Product Modulo 2 requires $2^{\Omega(n)}$-size $\mathsf{LTF} \circ \mathsf{LTF}$ circuits.*

Theorem 10.14 below gives a more general statement. Under the hood is an upper bound: matrices defined by small $\mathsf{LTF} \circ \mathsf{LTF}$ circuits have low *probabilistic sign-rank*: for every such circuit of s gates, viewing its truth table as a $2^{n/2} \times 2^{n/2}$ matrix, there is a distribution of $O(s^2 n^2/\varepsilon)$-rank matrices which sign-represent the truth table in a worst-case probabilistic sense with error ε.

Rigidity, Communication, and Probabilistic Rank: An Equivalence. Probabilistic rank arises very naturally in studying generalized models of communication complexity. For a Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, let M_f be the $2^n \times 2^n$ *truth table matrix of* f with $M_f[x,y] = f(x,y)$ for all x, y. The following correspondence between probabilistic rank and communication complexity is immediate (one could even take the proposition as a *definition* of $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ communication complexity).

Proposition 9.1. *Let $m > 1$ be an integer, let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and let M_f be its truth table matrix. The $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ communication complexity of f with error ε equals the (base-2) logarithm of the ε-probabilistic rank of M_f over $\mathbb{Z}_m$ (within additive constants).*

Similarly, AM (Arthur-Merlin communication complexity) is equivalent to probabilistic Boolean rank.

It's easy to see that if a matrix has ε-probabilistic rank r, then its rank-r rigidity is at most $\varepsilon 2^{2n}$; thus rigidity lower bounds imply communication lower bounds. But conversely, it seems easier to prove lower bounds on probabilistic rank compared to rigidity: with probabilistic rank, we need to rule out a "distribution" of erroneous matrix entries which are required to "spread the errors" around; with rigidity, we have to rule out *any* adversarial choice of bad entries.

We show that for every randomly self-reducible function $f : \{0,1\}^{2n} \to R$ in which the self-reduction makes k non-adaptive queries, low rigidity implies low probabilistic rank: the ε-probabilistic rank of its corresponding matrix is at most $(kr)^k$ if its rank-r rigidity is at most $\varepsilon \cdot 4^n$. Thus there is a strong relationship between ε-probabilistic rank (and communication complexity, by Proposition 9.1) and the rank for which the rigidity is an ε-fraction of the matrix. For the Walsh-Hadamard transform, we prove that the probabilistic rank of H_n and the rigidity of H_n are equivalent concepts:

Theorem 9.5. *For every commutative ring R and for every n, $\mathcal{R}_{H_n}(r) \le \varepsilon \cdot 4^n$ over R if and only if H_n has ε-probabilistic rank r over R.*

The matrices H_n represent the communication matrices of the widely-studied Inner Product Modulo 2 (IP2) function. By Proposition 9.1, the $\mathsf{BP} \cdot \mathsf{MOD}_p\mathsf{P}$ communication complexity of IP2 and the rigidity of H_n over $\mathbb{F}_p$ are really equivalent concepts. Applying this theorem, our earlier rigidity upper bounds also imply some modest but interesting improvements on communication complexity protocols. From the rigidity upper bound of Theorem 10.2, we obtain a communication protocol for IP2 with $O(\sqrt{n \log(1/\varepsilon)} \log(\frac{n}{\log(1/\varepsilon)}))$ bits and error ε in the $\mathsf{BP} \cdot \mathsf{MOD}_p\mathsf{P}$ communication model, for every prime p. (Aaronson and Wigderson gave an MA protocol for IP with $O(\sqrt{n} \log(n/\varepsilon))$ communication complexity and error ε [AW09]; ours is more efficient for $\varepsilon \ll 1/2^{\sqrt{\log n}}$.) Applying Theorem 10.1 yields an IP2 protocol with $n(1 - \Omega(\varepsilon^2/\log(1/\varepsilon)))$ communication and only $1/2^{n-\varepsilon n}$ error. We are skeptical that our rigidity upper bounds for H_n are tight; we hope these results will aid future work (to prove rigidity upper bounds, one only has to think about communication protocols for IP2).

9.1.2 Efficient Construction of Rigid Matrices Using an NP Oracle

In Chapter 11, we give a new construction of rigid matrices. Unlike the previous constructions (which we enumerated near the beginning of this Chapter), which primarily use combinatorial and algebraic techniques, our construction primarily uses *complexity-theoretic* ideas. Our matrices are Razborov-rigid, and we can prove new lower bounds in communication complexity, Boolean circuit complexity, and arithmetic circuit complexity.

Constructions of Rigid Matrices in P^{NP}

Our main result is a construction of a rigid matrix in P^{NP}:

Theorem 9.6 (An Infinitely Often Rigid Matrix Construction in P^{NP})**.** *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

By comparison, applying previously known techniques to construct rigid $N \times N$ matrices M_N for this rank $r = 2^{(\log N)^{1/4-\varepsilon}}$, one either obtains:

- M_N constructible in P with only $\mathscr{R}_{M_N}(r) \geq \Omega\left(\frac{N^2}{2^{(\log N)^{1/4-\varepsilon}}}\right)$, or
- M_N only constructible in $\mathsf{TIME}[\exp(\exp((\log N)^{1/4-\varepsilon}))]$ with $\mathscr{R}_{M_N}(r) \geq \Omega\left(N^2\right)$. Note that the time bound here is larger than any quasi-polynomial in N, which can be written as $\exp(\exp(\log\log N))$.

Our construction in Theorem 9.6 is in P^{NP}, and achieves $\mathscr{R}_{M_N}(r) \geq \Omega\left(N^2\right)$.

It is natural to ask whether one can improve the constant $1/4 - \varepsilon$ in the rank in Theorem 9.6. We show an interesting "win-win" theorem: either the constant can be improved from $1/4 - \varepsilon$ to $1 - \varepsilon$, or $\mathsf{NQP} \not\subset \mathsf{P}_{/\mathrm{poly}}$ follows.

Theorem 9.7 (Either a Better Construction in P^{NP} or $\mathsf{NQP} \not\subset \mathsf{P}_{/\mathrm{poly}}$)**.** *There is an absolute constant $\delta > 0$ such that for all prime powers $q = p^r$ and all constants $\varepsilon > 0$, at least one of the following holds:*

- *$\mathsf{NQP} \not\subset \mathsf{P}_{/poly}$.*
- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Theorem 9.7 is interesting from the perspective of proving circuit lower bounds. Recall that a main motivation for constructing rigid matrices is to construct an explicit function which cannot be computed by $O(n)$-size $O(\log n)$-depth circuits [Val77]. If we aim to show that NE (or E^{NP}) does not admit such circuits (which is still open), then we can safely assume $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$ before constructing the required rigid matrices. Therefore, if one could further improve the construction in the second bullet of the above Theorem 9.7 to match the Valiant-rigid parameters (which would require $N \times N$ matrices H_N with $\mathscr{R}_{H_N}(N/\log\log N) \geq N^{1+\varepsilon}$ for any $\varepsilon > 0$, i.e. an improved rank parameter in exchange for a worsened rigidity parameter), it would imply that E^{NP} does not have $O(n)$-size $O(\log n)$-depth circuits.

Applications

Using the many different connections between rigid matrices and different areas of complexity theory, including a new connection we prove in Chapter 10, we derive from our construction a number of new lower bounds.

$\mathsf{PH}^{\mathsf{cc}}$ Lower Bound for $\mathsf{NTIME}[2^{(\log n)^{\omega(1)}}]^{\mathsf{NP}}$. A longstanding open problem in communication complexity is to prove a $\mathsf{PH}^{\mathsf{cc}}$ (the communication complexity analogue of the polynomial hierarchy) lower bound for an explicit function [BFS86] (see [GPW18] for a recent reference). In fact, even for the much weaker subclass $\mathsf{AM}^{\mathsf{cc}}$, it is a notoriously open question to prove an $\omega(\log n)$ lower bound for any explicit function [GPW16, CW19a]. Prior to this, it was even open whether $\mathsf{E}^{\mathsf{NP}} \subset \mathsf{AM}^{\mathsf{cc}}$, i.e., whether every function in E^{NP} has an efficient AM communication protocol.

Recall that Razborov-rigid matrices are known to yield lower bounds against $\mathsf{PH}^{\mathsf{cc}}$:

Lemma 9.1 ([Raz89], see also [Wun12])**.** *Letting f be a function in $\mathsf{PH}^{\mathsf{cc}}$, the $2^n \times 2^n$ communication matrix M_f of f has $\mathscr{R}_{M_f}(2^{(\log n/\varepsilon)^c}) \leq \epsilon \cdot 4^n$, where $\varepsilon > 0$ is arbitrary and $c > 0$ is a constant depending only on f, but not n.*

Using this, our construction of rigid matrices in Theorem 9.6 immediately shows that $\mathsf{E}^{\mathsf{NP}} \not\subset \mathsf{PH}^{\mathsf{cc}}$, giving the first non-trivial lower bound against $\mathsf{PH}^{\mathsf{cc}}$. In fact, our rigidity bound is for a much higher rank than is necessary for applying Lemma 9.1 (setting $n = \log N$ in Theorem 9.6, we give a $2^n \times 2^n$ matrix M with $\mathscr{R}_M(2^{n^{1/4-\varepsilon}}) \geq \delta \cdot 4^n$ for infinitely many n). By a simple modification of our construction, we prove an even stronger lower bound:

Theorem 9.8. *For all functions $\alpha(n) = \omega(1)$ such that $n^{\alpha(n)}$ is time-constructible, there is a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ which is not in $\mathsf{PH}^{\mathsf{cc}}$.*

Of the three previously-known deterministic constructions of rigid matrices mentioned near the beginning of this Chapter, only the second constructs rigid enough matrices to apply Lemma 9.1. However, it only yields a $2^n \times 2^n$ matrix M with $\mathscr{R}_M(2^{(\log n)^{\omega(1)}}) \geq \Omega(4^n)$ in $\mathsf{TIME}[\exp(\exp((\log n)^{\omega(1)}))]$. We obtain exponential time savings using an NP oracle.

Depth-2 Arithmetic Circuit Lower Bounds Although the rank parameters in our rigidity lower bounds from Theorem 9.6 are not high enough to give log-depth arithmetic circuit lower bounds via Valiant's approach, the *rigidity* parameters are high enough that we can prove lower bounds against constant-depth arithmetic circuits. We consider a variant on rigidity which is useful for studying depth-2 arithmetic circuits:

Definition 9.2. *For a field $\mathbb{F}$ and a matrix $A \in \mathbb{F}^{N \times N}$, let*

$$w_2(A) := \min\{\mathrm{nnz}(B) + \mathrm{nnz}(C) \mid A = BC\},$$

where the min is over all pairs B, C of matrices of any dimensions over $\mathbb{F}$ whose product is A, and $\mathrm{nnz}(X)$ denotes the number of nonzero entries in the matrix X.

It is not hard to see that $w_2(A)$ equals, up to an additive[2] n, the minimum size (number of wires) of a depth-2 linear circuit over $\mathbb{F}$ which computes A, i.e. a depth-2 circuit which takes as input the N entries of a vector $x \in \mathbb{F}^N$ and outputs the N entries of the vector Ax, and whose gates compute $\mathbb{F}$-linear combinations of their inputs.

Every matrix $M \in \{0,1\}^{N \times N}$ has $w_2(M) \leq O(N^2/\log N)$ over any field, and similar to the situation for rigidity, for any fixed prime power $q = p^r$, a random matrix $A \in \mathbb{F}_q^{N \times N}$ has $w_2(A) \geq \Omega(N^2/\log N)$ with high probability [Lup56]. However, the best known lower bounds on w_2 for explicit families of $N \times N$ matrices are only:

- $\Omega(N \log N)$ for Boolean Hadamard matrices [AKW90]
- $\Omega(N \log^2 N/(\log\log N)^2)$ for asymptotically good error-correcting codes [GHK+12]
- $\Omega(N \log^2 N/\log\log N)$ for matrices based on super-concentrator graphs [RTS00]

Connections between rigidity lower bounds and w_2 lower bounds for a number of different parameter settings are known [Pud94]. We apply our rigidity lower bounds using a similar connection in the high rigidity setting to show higher w_2 lower bounds for matrices constructible in P^{NP}:

Theorem 9.9. *For all prime powers $q = p^r$ and constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $w_2(H_N) \geq \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$ over $\mathbb{F}_q$.*

Threshold Circuit Lower Bounds. We next give an application of our construction to Boolean circuit complexity. Using the probabilistic rank upper bound we gave for threshold circuits in Theorem 9.3 above, we give a new lower bound:

Theorem 9.10. *For every $\delta > 0$ and prime p, there is an $a > 0$ such that the class E^{NP} does not have non-uniform $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ circuits of depth $o(\log n/\log\log n)$ where the bottom LTF layer has $2^{O(n^a)}$ gates, the rest of the circuit has polynomial size, and the middle layer LTF gates have fan-in $O(n^{1/2-\delta})$.*

[2] $w_2(A)$ equals the minimum size of a depth-2 linear circuit for A when wires are not allowed to go directly from inputs to outputs. We can convert a circuit where wires do go from inputs to outputs to one where they do not, by adding in n middle-level gates which take the values of the n inputs. Hence, the minimum size of a depth-2 linear circuit for A differs from $w_2(A)$ by a negligible additive $\leq n$.

We briefly compare with prior lower bounds for threshold circuits:

- We showed above in Corollary 6.1 and Theorem 8.17 that E^{NP} does not have non-uniform $\mathsf{ACC}^0 \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits where the bottom LTF layer has $n^{2-\varepsilon}$ gates and the remaining $\mathsf{ACC}^0 \circ \mathsf{LTF}$ subcircuit has $2^{n^{o(1)}}$ size. Tamaki [Tam16] also showed similar results for depth-2 circuits with symmetric and threshold gates. Our new lower bound is incomparable to these: we allow for many more LTF gates in the bottom layer, and unbounded depth, but the prior result allowed for larger size above the bottom layer, as well as ACC^0 circuitry rather than just $\mathsf{AC}^0[p]$ circuitry.
- Kane and Williams [KW16] previously showed there is a function in P which requires $\mathsf{MAJ} \circ \mathsf{LTF} \circ \mathsf{LTF}$ circuits of size $\Omega(n^{3/2}/\log^3 n)$. Our lower bound is for much larger circuits than this, but without a MAJ gate on top, and for a function in E^{NP} instead of P.

9.2 Other Related Work

Toggle Rank. By Yao's minimax principle [Yao83], $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ communication complexity (randomized communication with "counting modulo m" power) equals worst-case distributional $\mathsf{MOD}_m\mathsf{P}$ communication complexity. In matrix terms, putting an arbitrary distribution $\mathcal{P}$ on the pairs $\{0,1\}^n \times \{0,1\}^n$, the worst-case ε-distributional complexity of M is the lowest rank (over $\mathbb{Z}_m$) of a $2^n \times 2^n$ matrix N with error $||M - N|| \leq \varepsilon$ over $\mathcal{P}$. Wunderlich [Wun12] calls this rank notion the *approximate toggle rank*. Proposition 9.1 shows that probabilistic rank and approximate toggle rank are very closely related, but they are ***not*** the same as the usual rigidity concept, which corresponds to the uniform distribution on pairs. For structured functions like IP2, we prove (in Theorem 9.5) that the uniform distribution *is* the worst case.

Sign-rank Rigidity and AC0-MOD2 circuits. A tantalizing open problem that has gained popularity in recent years [SV12, ABG+14, CGJ+16] is whether IP2 has polynomial-size $\mathsf{AC}^0 \circ \mathsf{MOD}_2$ circuits: i.e., circuits of $O(1)$-depth over AND/OR/NOT, but with a layer of gates computing PARITY at the bottom nearest the inputs. Servedio and Viola [SV12] propose an interesting attack: in our terminology, they note that $\mathsf{AC}^0 \circ \mathsf{MOD}_2$ circuits of size s have $n^{O(\log^{d-1} s)\log(1/\varepsilon)}$ sign-rank rigidity at most $\varepsilon 2^{2n}$ over $\mathbb{R}$, and prove a lower bound on the correlation of signs of sparse polynomials (taken as a proxy for low-rank sign-matrices) with IP2. That is, they prove a weak sign-rank rigidity lower bound (note Razborov and Sherstov prove an analogous lower bound for sign-rank rigidity of IP2; see Theorem 10.13). Our results have two consequences for this sort of approach. First, Theorem 10.12 shows that a sign-rank rigidity lower bound would prove something much stronger: a lower bound for *depth-two threshold circuits* computing IP2, a longstanding open problem. Second, our non-trivial upper bounds on the rank rigidity of the IP2 matrix (which is H_n) suggest that IP2 may have much lower sign-rank rigidity than expected.

Sign-Rank Rigidity and Margin Complexity. Linial and Shraibman [LS09] prove (in our terminology) that the sign-rank rigidity of an $n \times n$ matrix A is at most εn^2 for target rank $O(mc(A)^2 \log(1/\varepsilon))$, where $mc(A)$ is the "margin complexity" of A. Thus the margin complexity of a matrix can be used to upper bound sign-rank rigidity. They also study rigidity notion based on mc, conjecture that high mc implies high margin-complexity rigidity, and show that high margin-complexity also implies communication complexity lower bounds (for similar parameters as the standard rank-rigidity setting).

Approximate Rank. A different "approximating" rank notion has been studied in [BdW01, KS10, ALSV13], with connections to quantum computing and approximation algorithms. The ε-approximate rank of $M \in \mathbb{R}^{n \times n}$ is the lowest rank of a matrix A such that $||M - A||_\infty \leq \varepsilon$. That is, we can obtain one matrix from the other by perturbing each entry by at most $\varepsilon > 0$. The appropriate analogy here seems to be that probabilistic polynomials are to probabilistic rank, as ℓ_∞-approximate polynomials are to approximate rank: both are natural generalizations of polynomial representations to matrix representations, with different properties.

Non-rigidity of conjectured-to-be-rigid matrices. Are there other conjectured-to-be-rigid matrices which are not? One candidate would be the generating matrix of a good linear code over $\mathbb{F}_2$. Recently, Goldreich [Dvi16] reported a distribution of matrices in which most of them are the generating matrix of a good linear code that is not rigid, found by Dvir. It would be very interesting to find an explicit code with this property. Since a preliminary version of our proof of the non-rigidity of the Walsh-Hadamard transform appeared [AW17], other researchers have extended our results to even more families of matrices: Dvir and Edelman [DE17] showed the non-rigidity of matrices arising from functions of inner products over finite fields, and Dvir and Liu [DL19] showed the non-rigidity of Fourier and circulant matrices.

Explicit Construction Based on Complexity-Theoretical Ideas. In a recent breakthrough work, Oliveira and Santhanam [OS17] gave an infinite often *pseudodeterministic* construction of primes in *sub-exponential time*. (That is, given an input 1^N, the (randomized) algorithm outputs a fixed prime P_N of N bits with high probability, for infinite number of N's, and it runs in sub-exponential time). Their results are similar to our construction of rigid matrices in Theorem 9.6 in that they construct algebraic objects by building on complexity-theoretic ideas.

Our approach differs from theirs in several ways. First, [OS17] make crucial use of the fact that primes can be recognized in polynomial-time [AKS04], while in contrast, testing whether a matrix is rigid is coNP-complete (cf, Proposition 29 of [Des07]). Second, their results build on *hardness vs randomness*, and a crucial component of their arguments is to use special pseudo-random generators to hit the set of all N-bit primes, while our results build on Williams' algorithmic approach to lower bounds [Wil13, Wil14c]: we show one can contradict the non-deterministic time hierarchy theorem, assuming there is no P^{NP} construction of rigid matrices.

Conditional Explicit Construction of Rigid Matrices. There are several works achieving deterministic polynomial-time construction of rigid matrices under strong complexity assumptions. They are all based on the hardness-vs-randomness paradigm [NW94]. The observation is that since checking rigidity is in coNP, the ability to fool a non-deterministic algorithm implies the ability to construct rigid matrices.

In [KvM02], it is shown that under the assumption that E has no $2^{o(n)}$-size SAT-oracle circuits, there is a P-time construction of matrices M_N over $\mathbb{Z}_{p(N)}$ such that $\mathscr{R}(M_N)(r) \geq \Omega((n-r)^2/\log n)$, where $p(N)$ is a prime bounded by a polynomial of N. [MV05] give the same construction under the weaker assumption that E has no $2^{o(n)}$-size non-deterministic circuits[3]. In [GST03], the same construction is achieved with a uniform assumption that E has no $2^{o(n)}$-time Arther-Merlin protocols.

Lower Bounds on w_2. Recall the variant on rigidity, w_2, from Definition 9.2 above. Recently, Kumar and Volk [KV19] gave a construction of matrices with high w_2, which is incomparable with our Theorem 9.9. Among other results, they show that there are constants $a, b, c > 0$ and a family $\{A_N\}_{N\in\mathbb{N}}$ such that A_N is an $N \times N$ matrix over an extension of $\mathbb{F}_2$ of degree $\exp(N^{1-a})$ which can be computed in time $\exp(N^{1-b})$ and with $w_2(A_N) > N^{1+c}$. By comparison, our Theorem 9.9 constructs $N \times N$ matrices H_N in P^{NP} with the worse lower bound $w_2(H_N) \geq \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$, but our matrices are over $\mathbb{F}_2$ instead of a large extension field of $\mathbb{F}_2$. Their techniques seem very different from ours, although they also use a padding trick, similar to our Lemma 11.5, of taking the Kronecker product of a rigid matrix with a large simple matrix to decrease its computational complexity in terms of the matrix size.

Circuit Lower Bounds via PCPP. In recent work, Chen and Williams [CW19b] applied PCPPs to show that, in order to prove $\mathscr{C}$ lower bounds for various non-deterministic time classes such as NEXP or NQP, it suffices to solve CAPP on $\oplus_2 \circ \mathscr{C}$ circuits (an XOR of two $\mathscr{C}$ circuits) in better-than-2^n time. The proof crucially combines the assumed CAPP algorithm and PCPPs to obtain a non-trivial CAPP algorithm for *general circuits*. Here, our proof for Theorem 9.7 makes similar, but more sophisticated use of PCPPs. In particular, we actually require the PCPP to be *smooth*, which is not required in [CW19b]. Our proof for Theorem 9.6 also relies on a completely different bootstrapping argument, which is specific for our task of constructing rigid matrices.

Rigidity and Data Structure Lower Bounds. Recent work by Dvir, Golovnev, and Weinstein [DGW19] showed connections between rigidity and static data structure lower bounds. In particular, they posed the challenge of constructing rigid ma-

[3] Indeed, the requirement is E has no $2^{o(n)}$-size SV-nondeterministic circuits, which is the non-uniform analogue of $\mathsf{NP} \cap \mathsf{coNP}$; see [MV05] for details.

trices in P^{NP} or E^{NP} as an avenue toward proving new data structure lower bounds. Unfortunately, the parameters of our new P^{NP} construction do not seem to yield any new data structure bounds using their approach.

9.3 Bibliographic Details

This Part of the dissertation is based off of the results in two previously published papers:

- ‘Probabilistic Rank and Matrix Rigidity’ with Ryan Williams [AW17], which appeared in STOC 2017, and
- ‘Efficient Construction of Rigid Matrices Using an NP Oracle’ with Lijie Chen [AC19], which will appear in FOCS 2019.

Chapter 10 presents results from [AW17], and Chapter 11 presents results from [AC19].

Chapter 10

Probabilistic Rank and Matrix Rigidity

10.1 Polynomials, Rank, and Rigidity

In this Chapter, we study the interplay between the notions of probabilistic rank and matrix rigidity. We begin in this Section with their basic properties.

Definition 10.1. *For any $\varepsilon \in [0, 1]$, and any commutative ring R, a* probabilistic matrix with error ε and rank r *for the matrix $A \in R^{N \times N}$ is a distribution $\mathcal{M}$ on matrices $B \in R^{N \times N}$ of rank at most r over R such that, for every $i, j \in [N]$, we have*

$$\Pr_{B \sim \mathcal{M}}[A(i, j) = B(i, j)] \geq 1 - \varepsilon.$$

The ε-probabilistic rank of A over R is the minimum rank of a probabilistic matrix with error ε for M.

Definition 10.2. *For any commutative ring R, matrix $A \in R^{N \times N}$, and $r \in \mathbb{N}$, the* rank-r rigidity *of A, denoted by $\mathcal{R}_A(r)$, is the minimum Hamming distance from A to an $N \times N$ matrix of rank r over R. That is, $\mathcal{R}_A(r)$ is the number of entries of A that must be modified in order for the rank to drop to r. (The ring R should be clear from context.)*

By drawing a 'typical' matrix from the probabilistic rank distribution, we can always obtain a matrix rigidity upper bound:

Proposition 10.1. *For any commutative ring R, matrix $M \in R^{N \times N}$, and $\varepsilon \in [0, 1]$, if the ε-probabilistic rank of R is r, then $\mathcal{R}_A(r) \leq \varepsilon \cdot N^2$.*

We now describe a basic connection between probabilistic rank, matrix rigidity, and probabilistic polynomials (recall the definition from Definition 7.3 in Section 7.3 above). We show that probabilistic polynomials for a Boolean function f can be used to give upper bounds on the probabilistic rank of the *truth table matrix* of f:

Definition 10.3. *Let R be any commutative ring, and $f : \{0,1\}^{2n} \to R$ be any function on $2n$ Boolean variables. The* truth table matrix M_f *of f is the $2^n \times 2^n$ matrix given by*

$$M_f(v_i, v_j) = f(v_i, v_j),$$

where $v_1, \ldots, v_{2^n} \in \{0,1\}^n$ is the enumeration of all n-bit vectors in lexicographical order.

Given the above definition, it is natural to define the probabilistic rank of a function:

Definition 10.4. *The ε-probabilistic rank of a function $f : \{0,1\}^{2n} \to R$ is the ε-probabilistic rank of its truth table matrix M_f. The rank of f and the rigidity of f are defined similarly.*

We will make use of the following simple mapping from sparse polynomials to low-rank matrices; this is the same connection we used in Section 8.1 above to design algorithms using probabilistic polynomials.

Lemma 10.1. *Let R be any commutative ring, and $f : \{0,1\}^{2n} \to R$. Let $p : R^{2n} \to R$ be a polynomial with m monomials such that $p(x,y) = f(x,y)$ for any $x, y \in \{0,1\}^n$. Then the rank of f is at most m.*

Proof. Let $a_1, \ldots, a_m, b_1, \ldots, b_m : R^n \to R$ be monomials such that $p(x,y) = \sum_{i=1}^m a_i(x) \cdot b_i(y)$ is the monomial expansion of p. For $1 \leq i \leq m$, define vectors $\vec{\alpha_i}, \vec{\beta_i} \in R^{2^n}$ by $\vec{\alpha_i}(x) = a_i(x)$ and $\vec{\beta_i}(y) = b_i(y)$ for each $x, y \in \{0,1\}^n$. Then $M_f = \sum_{i=1}^m \vec{\alpha_i} \otimes \vec{\beta_i}$, where $\otimes$ denotes the outer product of vectors. Thus $\text{rank}(M_f) \leq m$. □

As a corollary, the probabilistic rank of f is at most the sparsity of a probabilistic polynomial for f:

Corollary 10.1. *Let R be any commutative ring, and $f : \{0,1\}^{2n} \to R$. If f has a probabilistic polynomial $\mathcal{P}$ with at most m monomials and error ε, then the ε-probabilistic rank of f is at most m.*

Proof. Let p be a polynomial in the support of the distribution $\mathcal{P}$. Since p has at most m monomials, by Lemma 10.1 the truth table matrix M_p of p (restricted to the domain $\{0,1\}^{2n}$) has rank at most m. The distribution of M_p over p drawn from $\mathcal{P}$ is therefore an ε-probabilistic rank-m distribution for M_f, since $M_f(x,y) = M_p(x,y)$ if and only if $f(x,y) = p(x,y)$. □

It follows that we can obtain a matrix rigidity upper bound from a sparse probabilistic polynomial.

Corollary 10.2. *Let R be any commutative ring, and $f : \{0,1\}^{2n} \to R$ be any function on $2n$ Boolean variables. If f has a probabilistic polynomial P with at most m monomials and error ε, then one can modify $\varepsilon 2^{2n}$ entries of the truth table matrix M_f and obtain a matrix of rank at most m.*

10.2 Non-Rigidity of Walsh-Hadamard

In this Section, we present our new rigidity upper bounds for the Walsh-Hadamard Transform.

Definition 10.5. *Let $v_1, \dots, v_{2^n} \in \{0,1\}^n$ be the enumeration of all n-bit vectors in lexicographical order. The* Walsh-Hadamard matrix *H_n is the $2^n \times 2^n$ matrix defined by $H_n(v_i, v_j) := (-1)^{\langle v_i, v_j \rangle}$.*

10.2.1 Rigidity Upper Bound for Low Error

We first prove that the Walsh-Hadamard matrices are not rigid enough for Valiant's program:

Theorem 10.1. *For every field K, for every sufficiently small $\varepsilon > 0$, and for all n, we have $\mathcal{R}_{H_n}\left(2^{n-f(\varepsilon)n}\right) \leq 2^{n(1+\varepsilon)}$ over K, for a function f where $f(\varepsilon) = \Theta(\varepsilon^2/\log(1/\varepsilon))$.*

We recall some notation from the Preliminaries. For a vector $v \in \{0,1\}^n$, let $|v|$ be the number of ones in v. Let $H : [0,1] \to [0,1]$ denote the binary entropy function

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p).$$

We also gave the following estimates on binomial coefficients (in Proposition 2.3 and Corollary 2.1). For $\varepsilon \in (0, 1/2)$:

$$\binom{n}{\varepsilon n} \leq n \cdot 2^{H(\varepsilon)n}, \text{ and} \tag{10.1}$$

$$2^{n-O(\varepsilon^2 n)} \leq \binom{n}{(1/2-\varepsilon)n} \leq 2^{n-\Omega(\varepsilon^2 n)}. \tag{10.2}$$

Our first (simple) lemma uses a polynomial to compute a large fraction of H_n's entries with a low-rank matrix. However, this fraction won't be high enough; we'll need another idea to "correct" many entries later.

Lemma 10.2. *For every commutative ring R, and for every $\varepsilon \in (0, 1/2)$, there is a multilinear polynomial $p(x_1, \dots, x_n, y_1, \dots, y_n)$ over R with at most $2^{n-\Omega(\varepsilon^2 n)}$ monomials, such that for all $\vec{x}, \vec{y} \in \{0,1\}^n$ with $\langle \vec{x}, \vec{y} \rangle \in [2\varepsilon n, (1/2+\varepsilon)n]$,*

$$p(\vec{x}, \vec{y}) = (-1)^{\langle \vec{x}, \vec{y} \rangle}.$$

The proof uses properties of multivariate polynomial interpolation over the integers. To be concrete, we will apply the following Lemma from Chapter 7:

Reminder of Lemma 7.1 *For any integers n, r, k with $n \geq r + k$ and any integers $c_1, \dots, c_r$, there is a multivariate polynomial $p : \{0,1\}^n \to \mathbb{Z}$ of degree $r - 1$ with integer coefficients such that $p(z) = c_i$ for all $\vec{z} \in \{0,1\}^n$ with Hamming weight $|\vec{z}| = k + i$.*

Proof of Lemma 10.2. By Lemma 7.1 with $k = 2\varepsilon n - 1$, $r = (1/2 - \varepsilon)n + 1$, and $c_i = (-1)^{k+i}$, one can construct a multivariate polynomial $q : \{0,1\}^n \to \mathbb{Z}$ with integer coefficients, of degree $(1/2 - \varepsilon)n$, such that for all $\vec{z} \in \{0,1\}^n$ with $|\vec{z}| \in [2\varepsilon n, (1/2+\varepsilon)n]$, we have $q(\vec{z}) = (-1)^{|\vec{z}|}$. As discussed in Section 2.1, q can be viewed as a polynomial over R. Then our desired polynomial is

$$p(x_1, \dots, x_n, y_1, \dots, y_n) = q\left(x_1y_1, x_2y_2, \dots, x_ny_n\right).$$

We can upper-bound the number of monomials in p as follows. First, since we only care about the value of p on $\{0,1\}^{2n}$, we can make p multilinear by applying the equation $v^2 = v$ to all variables. Second, observe that for all $i = 1, \dots, n$, x_i and y_i appear in exactly the same monomials. So if we introduce a variable z_i in place of each $x_i \cdot y_i$ in p, the number of monomials in our new n-variate polynomial p' equals the number of monomials in p.

Since p' is multilinear and degree $(1/2 - \varepsilon)n + 1$, the number of monomials is at most $n\binom{n}{(1/2-\varepsilon)n+1}$, which by (10.2) is at most $2^{n - c_2\varepsilon^2 n}$ for some constant $c_2 > 0$. □

Our second lemma says: fixing a vector x with about 1/2 ones, there is a strong upper bound the number of vectors which has about 1/2 ones but has small (integer) inner product with x; we'll use this to upper bound the number of erroneous entries at the very end.

Lemma 10.3. *For every vector $x \in \{0,1\}^n$ with $|x| \in [(1/2-a)n, (1/2+a)n]$, and any parameters $a, b \in (0, 1/5)$, the probability that a uniformly random vector y from $\{0,1\}^n$ satisfies both*

- $|y| \in [(1/2-a)n, (1/2+a)n]$, *and*

- $\sum_{k=1}^{n} x_k y_k \leq bn$

is at most $(2an+1)(bn+1) \cdot 2^{(f(a,b)-1)n}$, where f is a function such that $f(a,b) \to 0$ as $a, b \to 0$.

The usual toolbox of small-deviation estimates does not seem to yield the lemma; we give a direct proof.

Proof. For all x of the above form, every $k \in [(1/2-a)n, (1/2+a)n]$, and every $s \leq bn$, we count the number of $y \in \{0,1\}^n$ with $|y| = k$ and $\sum_{k=1}^{n} x_k y_k = s$. A vector y satisfies these properties if and only if:

- there are exactly s integers i with $y[i] = 1$ and $x[i] = 1$, and

- there are exactly $k - s$ integers i with $y[i] = 1$ and $x[i] = 0$.

So there are $\binom{|x|}{s}\binom{n-|x|}{k-s}$ such choices of y. The total probability is hence

$$\begin{aligned}
&\frac{1}{2^n}\sum_{k=(1/2-a)n}^{(1/2+a)n}\ \sum_{s=0}^{bn}\binom{|x|}{s}\binom{n-|x|}{k-s}\\
=&\frac{1}{2^n}\sum_{k=(1/2-a)n}^{(1/2+a)n}\ \sum_{s=0}^{bn}\binom{|x|}{s}\binom{n-|x|}{k-s}\\
\le&\frac{1}{2^n}\sum_{k=(1/2-a)n}^{(1/2+a)n}\ \sum_{s=0}^{bn}\binom{(1/2+a)n}{s}\binom{(1/2+a)n}{k-s}\\
\le&\frac{1}{2^n}\sum_{k=(1/2-a)n}^{(1/2+a)n}(bn+1)\cdot\binom{(1/2+a)n}{bn}\binom{(1/2+a)n}{(1/2-a-b)n}.\ (*)
\end{aligned}$$

Recall that if $k_1 < k_2 < n/2$ then $\binom{n}{k_1} < \binom{n}{k_2}$, and if $k_3 > k_4 > n/2$, then $\binom{n}{k_3} < \binom{n}{k_4}$. Step (*) therefore follows since $s \le bn < \frac{1}{2}(1/2+a)n$ and $k-s \ge (1/2-a-b)n > \frac{1}{2}(1/2+a)n$ whenever $0 < a, b < 1/5$. Let $g(n) = (2an+1)\cdot(bn+1)$. Simplifying further, the above expression is at most

$$\begin{aligned}
&\frac{g(n)}{2^n}\binom{(1/2+a)n}{bn}\binom{(1/2+a)n}{(2a+b)n}\\
\le\ &\frac{g(n)}{2^n}\cdot 2^{(1/2+a)n\cdot H(b/(1/2+a))}2^{(1/2+a)n\cdot H((2a+b)/(1/2+a))} && \text{(by (10.1))}\\
\le\ &\frac{g(n)}{2^n}\cdot 2^{(1/2+a)n\cdot H(2b)}2^{(1/2+a)n\cdot H(4a+2b)}\\
\le\ &\frac{g(n)}{4^n}\cdot 2^{(1/2+a)n\cdot 2\cdot 2b\cdot\log(1/2b)}2^{(1/2+a)n\cdot 2\cdot(4a+2b)\cdot\log(1/(4a+2b))}\\
& && (H(\varepsilon)\le 2\varepsilon\log_2(1/\varepsilon)\text{ for }\varepsilon<1/2)\\
\le\ &\frac{g(n)}{2^n}\cdot 2^{f(a,b)n},
\end{aligned}$$

where $f(a,b) = (1/2+a)(4b\log(1/2b) + (8a+4b)\log(1/(4a+2b)))$. □

Our third lemma is a simple linear-algebraic observation: given a low-rank matrix M that computes another matrix N on all but a small number of rows and columns, N must also have relatively low rank.

Lemma 10.4. *Let M' be a matrix of rank r which is equal to M except in at most k columns and ℓ rows. Then the rank of M is at most $r+k+\ell$.*

Proof. We will start with M', and add at most $k+\ell$ rank-one matrices to M' so that it equals M.

Consider a column c on which M does not equal M'. We can add to M' a correction matrix C_c given by

$$C_c(i,j) = \begin{cases} M(i,v) - M'(i,v) & \text{if } j = v, \\ 0 & \text{otherwise.} \end{cases}$$

Then, $M' + C_c$ equals M on column c, and is unchanged in any other column. Moreover, since C_c is only nonzero on a single column, it has rank one. So all we have to do is add the correction matrix C_c for each column c on which M and M' differ. The rows of M' can be corrected analogously. □

Corollary 10.3. *Let T be any $2^n \times 2^n$ matrix. Let $a \in (0, 1/2)$, and let M be a $2^n \times 2^n$ matrix of rank r, indexed by n-bit vectors. There is a $2^n \times 2^n$ matrix M' of rank at most $r + 4 \cdot n \cdot 2^{n-\Omega(a^2 n)}$ such that $M'(v_i, v_j) = T(v_i, v_j)$ on all $v_i, v_j \in \{0,1\}^n$ where at least one of the following holds:*

- $|v_i| \notin [(1/2 - a)n, (1/2 + a)n]$,
- $|v_j| \notin [(1/2 - a)n, (1/2 + a)n]$, *or,*
- $M(v_i, v_j) = T(v_i, v_j)$.

Proof. The number of $v_i \in \{0,1\}^n$ with $|v_i| \notin [(1/2 - a)n, (1/2 + a)n]$ is at most

$$\sum_{i=0}^{(1/2-a)n} \binom{n}{i} + \sum_{i=(1/2+a)n}^{n} \binom{n}{i} = 2 \sum_{i=0}^{(1/2-a)n} \binom{n}{i} \leq n \cdot 2^{n-\Omega(a^2 n)},$$

by (10.2). Applying Lemma 10.4 to M and M' with k and ℓ set to $2 \cdot n \cdot 2^{n-\Omega(a^2 n)}$, the result follows. □

Let us outline how we'll use all of the above. First, we construct a matrix M of rank about $2^{n-\Omega(\varepsilon^2 n)}$ approximating H_n, using the polynomial from Lemma 10.2 in a straightforward way. This matrix M has far more erroneous entries than what we desire. But by Lemma 10.3, we can infer that the errors in M are highly concentrated on a relatively small number of rows and columns. Applying Corollary 10.3, the rows and columns can be "corrected" in a way that increases the rank of M by only $2^{n-\Omega(\varepsilon^2 n)}$. By Lemma 10.3, each row of the matrix left over will have $2^{O(\varepsilon \log(1/\varepsilon) n)}$ erroneous entries.

Proof of Theorem 10.1. In fact, we prove that one only has to modify $2^{O(\varepsilon \log(1/\varepsilon) n)}$ entries in each row of H_n, to obtain the desired rank.

Let $\varepsilon > 0$ be given. By Lemma 10.2, there is a polynomial $p(x,y)$ in $2n$ variables with $m = 2^{n-\Omega(\varepsilon^2 n)}$ monomials which computes $(-1)^{\langle x,y \rangle}$ correctly, on all $(x,y) \in \{0,1\}^{2n}$ such that $\langle x,y \rangle \in [2\varepsilon n, (1/2 + \varepsilon)n]$.

Construct a $2^n \times 2^n$ matrix M of rank m as in Corollary 10.1, so that $M(x,y) = p(x,y)$. By definition, M equals H_n on all $(x,y) \in \{0,1\}^{2n}$ satisfying $\langle x,y \rangle \in [2\varepsilon n, (1/2 + \varepsilon)n]$.

Applying Corollary 10.3 to M with $T = H_n$ and $a = \varepsilon$, we obtain a matrix M' of rank $m + 4 \cdot n \cdot 2^{n-\Omega(\varepsilon^2 n)}$ which is correct on all (x, y) where either $|x| \notin [(1/2 - \varepsilon)n, (1/2 + \varepsilon)n]$, $|t| \notin [(1/2 - \varepsilon)n, (1/2 + \varepsilon)n]$, or $\langle x, y\rangle \in [2\varepsilon n, (1/2 + \varepsilon)n]$.

Fix a row of H_n indexed by $x \in \{0,1\}^n$ with $|x| \in [(1/2-\varepsilon)n, (1/2+\varepsilon)n]$ (note the other rows are already correct). To show that M' differs from H_n on a small number of entries, we need to bound the number of y such that none of the above conditions hold, i.e.,

1. $|y| \in [(1/2 - \varepsilon)n, (1/2 + \varepsilon)n]$ and
2. $\langle x, y\rangle \notin [2\varepsilon n, (1/2 + \varepsilon)n]$.

Note for our given x, it is never true that $\langle x, y\rangle > (1/2 + \varepsilon)n$. Therefore we only need to bound the number N of y such that $|y| \in [(1/2 - a)n, (1/2 + a)n]$ and yet $\langle x, y\rangle < 2\varepsilon n$. By Lemma 10.3 with $a = \varepsilon$ and $b = \varepsilon$, the probability that a random y satisfies $\langle x, y\rangle < 2\varepsilon n$ and $|y| \in [(1/2 - \varepsilon)n, (1/2 + \varepsilon)n]$, is at most $O(n^2) \cdot 2^{(f(\varepsilon,\varepsilon)-1)n}$, where $f \to 0$ as $\varepsilon \to 0$. Therefore $N \leq 2^n \cdot O(n^2) \cdot 2^{(f(\varepsilon,\varepsilon)-1)n} \leq O(n^2) \cdot 2^{f(\varepsilon,\varepsilon)n}$.

Now for sufficiently large n and $\varepsilon \in (0, 1/2)$, M' has rank at most $m + 4 \cdot n \cdot 2^{n-\Omega(\varepsilon^2 n)} \leq 5n \cdot 2^{n-\Omega(\varepsilon^2 n)}$. Furthermore, on every row, M' differs from H_n in at most $n^2 \cdot 2^{f(\varepsilon,\varepsilon)} \leq 2^{O(\varepsilon \log(1/\varepsilon)n)}$ entries. □

10.2.2 Rigidity Upper Bound for High Error

In this section, we prove upper bounds on the rigidity of the Walsh-Hadamard transform in the regime where the error is constant, or much larger than $1/2^n$; this setting is of interest for communication complexity lower bounds.

Theorem 10.2. *For every integer $r \in [2^{2n}]$, over any commutative ring R, one can modify at most $2^{2n}/r$ entries of H_n and obtain a matrix of rank at most $(n/\log(r))^{O(\sqrt{n \log(r)})}$.*

The proof follows from applying our optimal-degree probabilistic polynomial for symmetric functions from Chapter 7:

Reminder of Theorem 7.3 *There is a probabilistic polynomial over any commutative ring for any symmetric Boolean function on n variables, with error ε and degree $O(\sqrt{n \log(1/\varepsilon)})$.*

Proof of Theorem 10.2. Set $\varepsilon = 1/r$, and define the Boolean function $IP2 : \{0,1\}^{2n} \to \{-1, 1\}$ by $IP2(x, y) = (-1)^{\langle x,y\rangle}$ for all $x, y \in \{0,1\}^n$. We can see that H_n is the truth table matrix M_{IP2}. By Corollary 10.2, it is sufficient to construct a probabilistic polynomial for $IP2$ with error ε and $(n/\ln(1/\varepsilon))^{O(\sqrt{n \log(1/\varepsilon)})}$ monomials. Consider the Boolean function $PARITY(z_1, \ldots, z_n) = (-1)^{z_1+\cdots+z_n}$ for all $z \in \{0,1\}^n$, and note that $IP2(x_1, \ldots, x_n, y_1, \ldots, y_n) = PARITY(x_1y_1, x_2y_2, \ldots, x_ny_n)$. Since $PARITY$ is symmetric, by Theorem 7.3 it has a probabilistic polynomial P of error ε and degree $d = O(\sqrt{n \log(1/\varepsilon)})$. Hence, the distribution of $p(x_1y_1, \ldots, x_ny_n)$ over p drawn from P is a probabilistic polynomial for $IP2$. Since we are only interested in the value of $p(z)$ when $z \in \{0,1\}^n$, we can make p multilinear by applying the equation $v^2 = v$ to all variables. Then the number of monomials of p is at most $\sum_{i=0}^{O(\sqrt{n\log(1/\varepsilon)})} \binom{n}{i} \leq (n/\ln(1/\varepsilon))^{O(\sqrt{n\log(1/\varepsilon)})}$. Since in $p(x_1y_1, \ldots, x_ny_n)$ we are substituting in a monomial for each variable, its expansion has the same number of monomials as p, as desired. □

10.2.3 Generalization To SYM-AND circuits

Here we generalize Theorems 10.1 and 10.2 to $\mathsf{SYM} \circ \mathsf{AND}$ circuits. In the proof of Theorem 10.1, the key property of the $IP2$ function required is that has the form

$$IP2(x_1, \ldots, x_n, y_1, \ldots, y_n) = f(x_1 \wedge y_1, \ldots, x_n \wedge y_n),$$

where f is a symmetric Boolean function (in our case, f computes parity). The same proof yields the following generalization:

Theorem 10.3. *For every symmetric function $f : \{0,1\}^n \to R$, define the function $IP_f : \{0,1\}^{2n} \to R$ by $IP_f(x,y) = f(x_1 \wedge y_1, \ldots, x_n \wedge y_n)$. For all sufficiently small ε, there is a $\delta < 1$ and a matrix of rank $2^{\delta n}$ which differs from the truth table matrix M_{IP_f} in at most $2^{(1+\varepsilon)n}$ entries.*

The proof of Theorem 10.2 only requires a probabilistic polynomial construction in Corollary 10.2. Our probabilistic matrix distribution simply substitutes monomials into the probabilistic polynomial of Theorem 7.3 for any symmetric function. Since each monomial can be viewed as an AND, the same argument will work for any $\mathsf{SYM} \circ \mathsf{AND}$ circuit.

Theorem 10.4. *For any Boolean function $f : \{0,1\}^{2n} \to R$ which can be written as a $\mathsf{SYM} \circ \mathsf{AND}$ circuit with s AND gates, and for every integer $r \in [2^{2n}]$, one can modify $2^{2n}/r$ entries of the truth table matrix M_f and obtain a matrix of rank at most $(s/\log r)^{O(\sqrt{s\log r})}$.*

10.3 Equivalence Between Probabilistic Rank and Rigidity

In this section, we show that the probabilistic rank of H_n and the rigidity of H_n are the *same* concept over fields. It is easy to see that if ε-probabilistic rank of H_n is k over a field K, then the rank-k rigidity of H_n is at most $\varepsilon 2^{2n}$ over K. Exploiting the random self-reducibility of the H_n function, we can show a converse: lower bounds on probabilistic rank imply proportionate rigidity lower bounds. This is of interest because probabilistic rank lower bounds appear to be fundamentally easier to prove than rigidity lower bounds.

Theorem 10.5. *For every commutative ring R and for every n, $\mathcal{R}_{H_n}(r) \leq \varepsilon 2^{2n}$ over R if and only if H_n has ε-probabilistic rank r over R.*

First let us give some definitions. Let $\otimes$ denote the outer product of vectors. For vectors $a \in K^{2^n}$ whose entries are indexed by $v_1, \ldots, v_{2^n} \in \{0,1\}^n$, and $x, y \in \{0,1\}^n$, let $a^{(x,y)}$ denote the vector in K^{2^n} given by

$$a^{(x,y)}[v_i] = (-1)^{\langle v_i, y\rangle} a[v_i \oplus x].$$

This permutes the entries of a, then negates half of the entries.

Proof. One direction is straightforward: low probabilistic rank implies low rigidity, by simply drawing a "typical" matrix from the distribution. For the other direction, suppose $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ are vectors in R^{2^n} such that the $2^n \times 2^n$ matrix

$$M := \sum_{k=1}^{r} a_k \otimes b_k \tag{10.3}$$

differs from H_n in at most $\varepsilon 2^{2n}$ entries. Pick vectors $x, y \in \{0,1\}^n$ uniformly at random, and consider the $2^n \times 2^n$ matrix

$$M' = (-1)^{\langle x,y\rangle} \sum_{k=1}^{r} a_k^{(x,y)} \otimes b_k^{(y,x)}. \tag{10.4}$$

In this form it is clear that M' has rank at most r. We claim that each entry of M' is equal to the corresponding entry of H_n with probability at least $1-\varepsilon$, over the choice of x and y, which will complete the proof.

Consider a given entry $M'(v_i, v_j)$. It is sufficient to show that if $M(v_i \oplus x, v_j \oplus y) = H_n(v_i \oplus x, v_j \oplus y)$ then $M'(v_i, v_j) = H_n(v_i, v_j)$, since $(v_i \oplus x, v_j \oplus y)$ is a uniformly random pair of vectors in $\{0,1\}^n$. Suppose this is the case, meaning $M(v_i \oplus x, v_j \oplus y) = (-1)^{\langle v_i \oplus x, v_j \oplus y\rangle}$. Applying definition (10.3) and then (10.4) we see that

$$\begin{aligned}
(-1)^{\langle v_i \oplus x, v_j \oplus y\rangle} &= \sum_{k=1}^{r} a_k[v_i \oplus x] \cdot b_k[v_j \oplus y] \\
&= (-1)^{\langle v_i, y\rangle + \langle v_j, x\rangle} \sum_{k=1}^{r} (-1)^{\langle v_i, y\rangle} a_k[v_i \oplus x] \cdot (-1)^{\langle v_j, x\rangle} b_k[v_j \oplus y] \\
&= (-1)^{\langle v_i, y\rangle + \langle v_j, x\rangle} \sum_{k=1}^{r} a_k^{(x,y)}[v_i] \cdot b_k^{(y,x)}[v_j] \\
&= (-1)^{\langle v_i, y\rangle + \langle v_j, x\rangle} \cdot (-1)^{\langle x,y\rangle} \cdot M'(v_i, v_j).
\end{aligned}$$

Rearranging, we see as desired that

$$M'(v_i, v_j) = (-1)^{\langle v_i \oplus x, v_j \oplus y\rangle + \langle v_i, y\rangle + \langle v_j, x\rangle + \langle x, y\rangle} = (-1)^{\langle v_i, v_j\rangle},$$

where the last step follows from the bilinearity of the inner product $\langle \cdot, \cdot \rangle$. □

Therefore, proving communication lower bounds for the IP2 function against (for example) the class $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ is *equivalent* to proving rigidity lower bounds for H_n over the ring $\mathbb{Z}_m$. Applying our rigidity upper bounds for H_n (Theorems 10.1 and 10.2), we obtain surprisingly low probabilistic rank bounds for H_n (and therefore communication-efficient protocols as well):

Corollary 10.4. *For every commutative ring R, for every sufficiently small $\varepsilon > 0$, and for all n, H_n has $1/2^{n(1-\varepsilon)}$-probabilistic rank at most $2^{n-\Omega(\varepsilon^2/\log(1/\varepsilon))n}$, and ε-probabilistic rank at most $(1/\varepsilon)^{O(\sqrt{n}\log n)}$, over R.*

10.3.1 Generalization to Random Self-Reducibile Functions

In fact, our reduction from rigidity to probabilistic rank works for any (non-adaptive) random self-reducible function [FF93] that makes a small number of oracle calls. Our notion of random self-reducibility is adapted for the communication complexity setting (for example, we do not care about the feasibility of the reduction).

Definition 10.6. *A function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is k-*random self-reducible *if there are random sampling procedures S_1, S_2 and a function $g : \{0,1\}^k \to \{0,1\}$ such that:*

(a) S_1 takes $x \in \{0,1\}^n$ and a random string r and outputs $x_1, \ldots, x_k \in \{0,1\}^n$ such that for all n-bit strings z, $\Pr_r[x_i = z] = 1/2^n$ for all i,

(b) S_2 takes $y \in \{0,1\}^n$ and a random string s and outputs $y_1, \ldots, y_k \in \{0,1\}^n$ such that for all n-bit strings z, $\Pr_s[y_i = z] = 1/2^n$ for all i, and

(c) $f(x,y) = g(f(x_1,y_1), \ldots, f(x_k,y_k))$.

Requirements (a) and (b) in the definition ensures that each x_i and y_i are uniform random variables; requirement (c) says that we can reconstruct $f(x,y)$ from the values $f(x_1,y_1), \ldots, f(x_k,y_k)$.

Theorem 10.6. *Let $r, n \in \mathbb{N}$ and $\varepsilon \in (0,1)$, and let $f : \{0,1\}^{2n} \to \{0,1\}$ be k-random self-reducible. Suppose M_f has rank-r rigidity at most $\varepsilon 4^n$ over a commutative ring R. Then the $(k\varepsilon)$-probabilistic rank of M_f over R is at most $O(\binom{kr}{k})$.*

Proof. Suppose there is an $2^n \times r$ matrix A and $r \times 2^n$ matrix B, such that M_f and $A \cdot B$ differ in at most $\varepsilon 4^n$ entries. We construct a distribution of low-rank matrices for M_f as follows.

Let $P(z_1, \ldots, z_k)$ be the unique multilinear polynomial over R that represents the function g from the random self-reduction for f. Given k rows $X_1, \ldots, X_k \in K^r$ of A, and k columns $Y_1, \ldots, Y_k \in K^r$ of B, define a polynomial in $2kr$ variables:

$$Q(X_1, \ldots, X_k, Y_1, \ldots, Y_k) = P(\langle X_1, Y_1 \rangle, \ldots, \langle X_k, Y_k \rangle).$$

Treating each term of the form $X_i[j] \cdot Y_i[j]$ as a variable, Q can be written as a sum of $t \leq \binom{kr}{k}$ total terms. Call the terms $m_1, \ldots, m_t$, each of which are over $2kr$ variables.

Let r be a random string for S_1 and s be a random string for S_2. For $x \in \{0,1\}^n$, let $x_1, \ldots, x_k \in \{0,1\}^n$ be the outputs of $S_1(x)$ with randomness r. We define the xth row of a new $2^n \times t$ matrix A_r to be

$$\Big(m_1(A[x_1,:], \ldots, A[x_k,:], \vec{1}, \ldots, \vec{1}), \ldots, m_t(A[x_1,:], \ldots, A[x_k,:], \vec{1}, \ldots, \vec{1})\Big).$$

For $y \in \{0,1\}^n$, let $y_1, \ldots, y_k$ be the outputs of $S_2(x)$ with randomness s. Define the yth column of a new $t \times 2^n$ matrix B_s to be

$$\Big(m_1(\vec{1}, \ldots, \vec{1}, B[:,y_1], \ldots, B[:,y_k]), \ldots, m_t(\vec{1}, \ldots, \vec{1}, B[:,y_1], \ldots, B[:,y_k])\Big)^T.$$

Then, for all $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, the inner product of the xth row of A_r and the yth column of B_s is

$$\begin{aligned}
&\sum_i m_i(A[x_1,:], \ldots, A[x_k,:], B[:,y_1], \ldots, B[:,y_k]) \\
&= \; Q(A[x_1,:], \ldots, A[x_k,:], B[:,y_1], \ldots, B[:,y_k]) \\
&= \; P(\langle A[x_1,:], B[:,y_1]\rangle, \ldots, \langle A[x_k,:], B[:,y_k]\rangle).
\end{aligned}$$

Since $(A \cdot B)$ differs from M_f on an ε-fraction of entries, for uniform random $x_i, y_j \in \{0,1\}^n$ we have $\langle A[x_i,:], B[:,y_i]\rangle \neq f(x_i, y_i)$ with probability at most ε. So with probability at least $1 - k\varepsilon$, $f(x_i,y_i) = \langle A[x_i,:], B[:,y_i]\rangle$ for all $i = 1, \ldots, k$. Thus the polynomial $P(\langle A[x_1,:], B[:,y_1]\rangle, \ldots, \langle A[x_k,:], B[:,y_k]\rangle)$ being implemented by $A_r \cdot B_s$ outputs $f(x,y)$ with probability at least $1-k\varepsilon$. Hence all matrices $C_{r,s} = A_r \cdot B_s$ in our defined distribution have rank at most $O(\binom{kr}{k})$, and for every $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, $\Pr_{r,s}[C_{r,s}[x,y] = f(x,y)] \geq 1 - k\varepsilon$. □

10.4 Explicit Rigid Matrices and Threshold Circuits

In this section, we show how explicit rigidity lower bounds would also imply circuit lower bounds where we currently only know weak results (e.g., we know that some functions in $\mathbb{E}^{\mathsf{NP}}$ do not have such circuits).

Theorem 10.7. *For every constant $\delta > 0$ and every $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuit C of size-$s = n^{2-\delta}$ and depth-$d = o(\log(n)/\log\log(n/\varepsilon))$, there exists a $\gamma > 0$ such that the truth table of C as a $2^{n/2} \times 2^{n/2}$ matrix M_C has rigidity $\mathcal{R}_{M_C}\left(2^{n^{1-\gamma}\log(1/\varepsilon)}\right) \leq \varepsilon 2^n$, for all $\varepsilon \in (1/2^n, 1)$, over any commutative ring.*

Our proof will use a technique by Maciel and Therien for converting each middle layer LTF gate into an equivalent $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit:

Theorem 10.8 ([MT98] Theorem 3.3, [ACW16] Theorem 7.1)**.** *For every $\alpha > 0$, every LTF on n inputs can be computed by a polynomial-size $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit where the fan-in of each MAJ gate is $n^{1+\alpha}$ and the circuit has depth $O(\log(1/\alpha))$.*

We will also use Tarui's probabilistic polynomial for AC^0:

Theorem 10.9 ([Tar93] Theorem 3.6)**.** *Every circuit in AC^0 with depth d has a probabilistic polynomial over $\mathbb{Z}$ (and hence, any commutative ring) of degree $O(\log^d(n))$ and error $1/2^{\log^{O(1)}(n)}$.*

Proof of Theorem 10.7. By Lemma 10.7, each LTF gate in the bottom layer has ε/s-probabilistic rank $O(n^2 s/\varepsilon)$. We will design a probabilistic polynomial for the upper $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0$ circuitry, which will give the desired result when composed with this probabilistic rank expression.

First, each LTF gate in the middle layer has fan-in at most $s = n^{2-\delta}$. Applying Theorem 10.8 with $\alpha = \delta/2$ to each, the upper $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0$ circuit becomes a $\mathsf{AC}^0 \circ \mathsf{MAJ} \circ \mathsf{AC}^0$ where each MAJ gate has fan-in at most $n^{(2-\delta)(1+\delta/2)} = n^{2-\delta^2/2}$, and the depth is still $O(d)$.

We can now apply the probabilistic polynomial for AC^0 from Theorem 10.9 with degree $O(\log^d(n))$ error $1/2^{\log^{O(1)}(n)}$ to the AC^0 circuits, and the probabilistic probabilistic polynomial for symmetric functions on $n^{2-\delta^2/2}$ bits from Theorem 7.3 with error ε/s and degree $O(n^{1-\delta^2/4}\log(s/\varepsilon))$ to the MAJ gates in the middle. This results in a probabilistic polynomial of degree $O(n^{1-\delta^2/4}\log^{O(d)}(n/\varepsilon))$. For $d = o(\log(n)/\log\log(n/\varepsilon))$, this is $O(n^{1-\beta})$ for any $\beta \in (0, \delta^2/4)$.

We can view the terms in the probabilistic rank expression for the LTF gates in the bottom layer as variables that we substitute into this probabilistic polynomial; the number of monomials in this expansion will upper bound the rank, as in Lemma 10.1. Since there are at most s such gates, and each probabilistic rank expression has $O(n^2 s/\varepsilon)$ terms, we are substituting $O(n^2 s^2/\varepsilon)$ terms into our polynomial. Hence, the number of monomials will be upper bounded by

$$(n^2 s^2/\varepsilon)^{O(n^{1-\beta})} = 2^{O(n^{1-\gamma})\log(1/\varepsilon)},$$

for any $\gamma < \beta$. This is of the desired form, where we can pick any positive value $\gamma < \delta^2/4$. The correctness follows by union bounding over all $\leq s$ probabilistic substitutions we make, each of which has error probability at most ε/s. □

From the above theorem, setting the error ε appropriately, we infer a new consequence of explicit rigid matrices:

Theorem 10.10. *Let R be any commutative ring, and $\{M_n\}$ be a family of Boolean matrices such that (a) M_n is $n \times n$, (b) there is a poly$(\log n)$ time algorithm A such that $A(n,i,j)$ prints $M_n(i,j)$, and (c) there is a $\delta > 0$ such that for infinitely many n,*

$$\mathcal{R}_{M_n}\left(2^{(\log n)^{1-\delta}}\right) \geq \frac{n^2}{2^{(\log n)^{\delta/2}}} \text{ over } R.$$

Then the language $\{(n,i,j) \mid M_n(i,j) = 1\} \in \mathsf{P}$ does not have $\mathsf{AC}^0 \circ \mathsf{LTF} \circ \mathsf{AC}^0 \circ \mathsf{LTF}$ circuits of $n^{2-\varepsilon}$-size and $o(\log n/\log\log n)$-depth, for all $\varepsilon > 0$.

Therefore, proving strong rigidity lower bounds for explicit matrices has consequences for Boolean circuit complexity as well. Indeed, the desired circuit lower bounds could be derived from lower-bounding probabilistic rank.

10.5 Sign-Rank Rigidity and Depth-Two Threshold Circuits

Given a matrix $A \in \mathbb{R}^{n \times n}$, its sign rank is the minimum rank of any $B \in \{-1, 1\}^{n \times n}$ such that $sign(A[i, j]) = sign(B[i, j])$ for all $i, j \in [n]$. The ε-probabilistic sign-rank of A is defined analogously as with probabilistic rank. We say A has *sign rank r-rigidity t* if a minimum of t entries of A need to be modified in order for A to have sign rank at most r.

First, we observe that in the sign-rank setting, random -1/1 matrices are still rigid: for example, with high probability, a random -1/1 matrix has sign-rank-$(n/\log^2 n)$ rigidity at least $\Omega(n^2)$. The proof follows readily from recent work:

Theorem 10.11 (Follows from Alon-Moran-Yehudayoff [AMY16])**.** *Let $r(n) = o(n/\log n)$. For all sufficiently large n, a random $n \times n$ matrix with $-1/1$ entries has sign-rank-$r(n)$ rigidity at least $\Omega(n^2)$, with high probability.*

Proof. There are 2^{n^2} different $n \times n$ matrices over $\{-1, 1\}$. The number of distinct matrices with sign rank at most r is bounded by $2^{O(rn \log n)}$ [AMY16]. For a fixed $\{-1, 1\}$ matrix M, the number of $\{-1, 1\}$ matrices within Hamming distance d of M is at most $O(\binom{n^2}{t})$. Thus the number of matrices for which up to t entries can be changed to obtain a matrix of sign rank at most r, is upper-bounded by

$$2^{O(rn \log n)} \cdot \binom{n^2}{t} \leq n^{O(rn)} \cdot (en^2/t)^t.$$

Suppose we set $t = \varepsilon n^2$. Then the above quantity is at most

$$n^{O(rn)} \cdot (e/\varepsilon)^{\varepsilon n^2}.$$

For $r = o(n/\log n)$ and $\varepsilon \log_2(e/\varepsilon) < 1$, a random matrix is not among these matrices with high probability. Therefore a random matrix has sign rank-$o(n/\log n)$ rigidity $\Omega(n^2)$ with high probability. □

Even though most -1/1 matrices have high sign-rank rigidity, we show that the truth table of a small $\mathsf{LTF} \circ \mathsf{LTF}$ circuit is always close to a matrix of low sign-rank. For even n, we say a function $f : \{0, 1\}^n \to \{0, 1\}$ has *ε-probabilistic sign rank r* if the truth table of C construed as a $2^{n/2} \times 2^{n/2}$ matrix has ε-probabilistic sign-rank r.

Theorem 10.12. *For every function f with a $\mathsf{LTF} \circ \mathsf{LTF}$ circuit of size s, and every $\varepsilon > 0$, the ε-probabilistic sign-rank of f is $O(s^2 n^2/\varepsilon)$. Moreover, we can sample a low-rank matrix from the distribution of matrices in $2^{n/2} \cdot poly(s, n)$ time.*

We will prove this theorem in a few steps. Let $EQ_n : \{0, 1\}^{2n} \to \{0, 1\}$ be the equality function, i.e., $EQ_n(x, x) = [x = y]$ (using Iverson bracket notation). Similarly, let $LEQ_n : \{0, 1\}^{2n} \to \{0, 1\}$ be the function $LEQ_n(x, x) = [x \leq y]$ where x and y are interpreted as integers in $\{0, \ldots, 2^n - 1\}$.

Lemma 10.5. *For every n, EQ_n has ε-probabilistic rank at most $O(1/\varepsilon)$ over any commutative ring.*

Proof. We mimic a well-known randomized communication protocol for EQ_n. Pick $k = \lceil \log_2(1/\varepsilon) \rceil$ uniformly random subsets $S_1, \ldots, S_k \subseteq \{0,1\}^n$, and define the hash functions $h_1, \ldots, h_k : \{0,1\}^n \to \{0,1\}$ by $h_i(x) = \bigoplus_{j \in S_i} x_j$. Note that $h_i(x) \neq h_i(y)$ with $1/2$ chance if $x \neq y$. Hence, the following expression equals $EQ(x,y)$ with error probability at most ε:

$$\prod_{i=1}^{k} (h_i(x)h_i(y) + (1 - h_i(x))(1 - h_i(y))). \tag{10.5}$$

When expanded out, (10.5) is a sum of $2^k = O(1/\varepsilon)$ terms of the form $f(x) \cdot g(y)$ for some functions f and g, each of which has rank one. □

Lemma 10.6. *For every n, LEQ_n has ε-probabilistic rank at most $O(n^2/\varepsilon)$ over any commutative ring.*

Proof. We express LEQ_n in terms of EQ predicates which check for the first bit in which x and y differ, as

$$LEQ_n(x_1, \ldots, x_n, y_1, \ldots, y_n) = \sum_{i=1}^{n} (1 - x_i) \cdot y_i \cdot EQ_{i-1}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}). \tag{10.6}$$

We then get the desired rank bound by replacing each EQ with the probabilistic rank expression from Lemma 10.5 with error ε/n. By the union bound, all n of the EQ predicates will be correct with probability at least $1 - \varepsilon$, and hence we will correctly compute LEQ_n. □

Lemma 10.7. *For every n, every linear threshold function $f : \{0,1\}^{2n} \to \{0,1\}$ has ε-probabilistic rank $O(n^2/\varepsilon)$ over any commutative ring.*

Proof. A linear threshold function f is defined as $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = [\sum_i v_i x_i + \sum_i w_i y_i \geq k]$, where all v_i's, w_i's, and k are reals. We want to show that the $2^n \times 2^n$ matrix indexed by x_i-assignments on the rows and y_i-assignments on the columns has low probabilistic rank. We will exploit the fact that the linear forms on x_i's and y_i's can be preprocessed separately in a rank decomposition.

Define $a : \{0,1\}^n \to \mathbb{R}$ by $a(x_1, \ldots, x_n) = \sum_{i=1}^n v_i x_i$, and $b : \{0,1\}^n \to \mathbb{R}$ by $b(y_1, \ldots, y_n) = k - \sum_{j=1}^n w_i y_i$. Hence

$$f(x,y) = [a(x) \leq b(y)] \, .$$

Let L be the list, sorted in increasing order, of all values of $a(x)$ and $b(y)$, for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$. Then define the function $\alpha : \{0,1\}^n \to \{0,1\}^{n+1}$ where $\alpha(x)$ equals the earliest index of $a(x)$ in the sorted list L, interpreted as a $n+1$ bit number. Define $\beta : \{0,1\}^n \to \{0,1\}^{n+1}$ similarly. Then

$$f(x,y) = LEQ_{n+1}(\alpha(x), \beta(y)).$$

So the ε-probabilistic rank of M_f is at most that of $M_{EQ_{n+1}}$, which we upper-bounded in Lemma 10.6. □

Now we are ready to upper-bound the probabilistic sign-rank of depth-two threshold circuits:

Proof of Theorem 10.12. We interpret our $\mathsf{LTF} \circ \mathsf{LTF}$ circuit C as a function on two groups of $n/2$ bits, $x_1, \ldots, x_{n/2}$, and $y_1, \ldots, y_{n/2}$. Let $w_1, \ldots, w_s, k \in \mathbb{R}$ be the weights of the output gate, so that

$$C(x_1, \ldots, x_n, y_1, \ldots, y_n) = \left[\sum_{i=1}^{s} w_i \cdot f_i(x_1, \ldots, x_n, y_1, \ldots, y_n) \le k \right],$$

for s different LTF functions f_i. By Lemma 10.7, the truth table matrix M_{f_i} of each f_i has (ε/s)-probabilistic rank $r = O(n^2 s/\varepsilon)$. Our probabilistic distribution of matrices for M_C can be constructed as follows: for all $i = 1, \ldots, s$, draw a random rank-r matrix P_i from the distribution for M_{f_i}, and set

$$Q_C = (-k \cdot J) + \sum_i (w_i \cdot P_i),$$

where J is the all-1s matrix. Q_C has rank at most $sr + 1 \le O(n^2 s^2/\varepsilon)$ and for all $(\vec{x}, \vec{y})$, $\Pr[\text{sign}(Q_C[\vec{x}, \vec{y}]) \ne C(\vec{x}, \vec{y})] \le \varepsilon$. □

Are there explicit matrices with non-trivial sign-rank rigidity? We observe that the best-known rank rigidity lower bounds for H_n extend to sign-rank rigidity:

Theorem 10.13 (Follows from Razborov and Sherstov [RS10])**.** *For all n, and $r \in [2^{n/2}, 2^n]$, the sign-rank-r rigidity of H_n is at least $\Omega(4^n/r)$.*

Proof. Theorem 5.1 of [RS10] gives the following lower bound on sign-rank: given any matrix $A \in \{-1,1\}^{n \times n}$, suppose that all but h entries of matrix $\tilde{A}$ have absolute value at least γ. Then

$$\text{sign-rank}(A) \ge \frac{\gamma n^2}{||A||n + \gamma h},$$

where $||A||$ is the spectral norm of A. For the case of H_n, if we modify $h := 4^n/r$ entries arbitrarily, all but h entries have absolute value equal to 1. Thus

$$\text{sign-rank}(H_n) \ge \frac{4^n}{||H_n||n + 4^n/r}.$$

As $||H_n|| \le O(2^{n/2})$ [For02], we have sign-rank$(H_n) \ge \Omega(4^n/(2^{3n/2} + 4^n/r)) \ge \Omega(2^{n/2} + r) \ge \Omega(r)$. □

Can the above lower bound be improved slightly? Combining the previous two theorems, it follows that any minor improvement in the above rank/rigidity trade-off would begin to imply lower bounds for $\mathsf{LTF} \circ \mathsf{LTF}$:

Theorem 10.14. *Suppose there is an $\alpha > 0$ such that for infinitely many n, the sign rank r-rigidity of H_n is $\Omega(4^n/r^{1-\alpha})$, for some $r \geq \omega(n^{2/\alpha}s(n)^{2/\alpha})$. Then the Inner Product Modulo 2 does not have $\mathsf{LTF} \circ \mathsf{LTF}$ circuits of $s(n)$ gates.*

Proof. Suppose the sign rank r-rigidity of H_m is $\Omega(4^n/r^{1-\alpha})$. Let $\varepsilon = 1/r^{1-\alpha}$. It follows that the ε-probabilistic sign-rank of H_n is greater than r. But for a $\mathsf{LTF} \circ \mathsf{LTF}$ function with s gates, its matrix always has $\Omega(\varepsilon)$-probabilistic rank $O(s^2n^2/\varepsilon) = O(s^2n^2r^{1-\alpha})$, by Theorem 10.12. Thus we have a contradiction when $O(s^2n^2r^{1-\alpha})$ is asymptotically less than r, i.e.,

$$r = \omega(n^{2/\alpha}s^{2/\alpha}),$$

corresponding to an s-gate lower bound against $\mathsf{LTF} \circ \mathsf{LTF}$ circuits. Since H_n is just a linear translation of the matrix for Inner Product Modulo 2, the proof is complete. □

For instance, proving the sign-rank $2^{\alpha n}$-rigidity of H_n is at least $4^n/2^{.999\alpha n}$ for some $\alpha > 0$ would imply exponential-gate lower bounds for depth-two threshold circuits computing IP2.

10.6 Equivalence Between Probabilistic Rank Modulo m and BP-MODm Communication Complexity

We conclude this Part by expanding on the connection between probabilistic rank and communication complexity. We sketch how probabilistic rank over $\mathbb{Z}_m$ is equivalent to $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ communication complexity:

Proposition 10.2. *Let $m > 1$ be an integer, let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and let M_f be its truth table matrix. Let $C_\varepsilon(f)$ be the $\mathsf{BP} \cdot \mathsf{MOD}_m\mathsf{P}$ communication complexity of f with error ε, and let ε-rank$_{\mathbb{Z}_m}(M_f)$ be the ε-probabilistic rank of M_f over $\mathbb{Z}_m$. Then $C_\varepsilon(f) \leq \log_2(\varepsilon\text{-rank}_{\mathbb{Z}_m}(M_f) + 1) \leq 2C_\varepsilon(f)$.*

This proposition is different from the one quoted in the introduction (giving constant-factor equivalences between the log of the rank and the communication complexity) because we are assuming a more stringent communication model here. However, the more general model is often taken as the definition, in which case the probabilistic rank and communication complexity truly coincide.

First, given a distribution of low-rank matrices for M_f, it is easy to construct a protocol for f: Alice and Bob publicly randomly sample a matrix from the distribution, which is a product of two matrices A and B. Alice takes the row of A of length r corresponding to her input, Bob takes the column of B of length r corresponding to his, and they then compute the inner product of these two vectors over $\mathbb{Z}_m$ with $\lceil \log_2(r+1) \rceil$ communication in the $\mathsf{MOD}_m\mathsf{P}$ model.

To construct a distribution of matrices from communication protocols, we do a simple modification of the $\mathsf{BPP}^{\oplus\mathsf{P}}$ communication model. In fact, sometimes the literature *defines* the $\mathsf{BPP}^{\oplus\mathsf{P}}$ communication model in this modified way [GPW16]. After the public randomness is chosen, Alice and Bob can, along with their c nondeterministic bits, also sum over all possible *transcripts* of at most c bits between them. For each choice of randomness and nondeterminism there is a unique accepting transcript, so this extra choice does not alter the number of accepting communication patterns. But in this modified version, now Alice and Bob do not even have to communicate: they only have to send a single bit indicating whether they would accept or not, given the transcript and the nondeterminism. From such a protocol, it is straightforward to construct a $2^n \times 2^{2c}$ matrix A representing Alice's protocol and a $2^{2c} \times 2^n$ matrix B representing Bob, for any given string of public randomness.

Chapter 11

Efficient Construction of Rigid Matrices Using an NP Oracle

11.1 Construction Overview

In this Chapter, we give our new P^{NP} construction of rigid matrices:

Theorem 11.1 (An Infinitely Often Rigid Matrix Construction in P^{NP})**.** *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Along the way, we also give the following conditional construction which achieves better parameters:

Theorem 11.2 (Either a Better Construction in P^{NP} or $\mathsf{NQP} \not\subset \mathsf{P}_{/\mathrm{poly}}$)**.** *There is an absolute constant $\delta > 0$ such that for all prime powers $q = p^r$ and all constants $\varepsilon > 0$, at least one of the following holds:*

- *$\mathsf{NQP} \not\subset \mathsf{P}_{/poly}$.*
- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

We begin, in this Section, with an overview of both of these constructions. In the rest of the Chapter we will give the formal proof and all the details, and then discuss some applications. For simplicity, we only consider the field $\mathbb{F}_2$ in this overview.

11.1.1 Either $\mathsf{NE} \not\subset \mathsf{P}_{/\mathrm{poly}}$ or a Construction of Rigid Matrices

We begin with a proof overview of Theorem 11.2. Note that Theorem 11.2 is equivalent to saying that there is a rigid matrix construction in P^{NP} under the assumption

$\mathsf{NQP} \subset \mathsf{P}_{/\mathrm{poly}}$. Here we outline a conditional construction under the stronger assumption $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$ for simplicity. We will then show how to get rid of the assumption using an additional bootstrapping argument.

Theorem 11.3 (Either a Better Construction in P^{NP} or $\mathsf{NE} \not\subset \mathsf{P}_{/\mathrm{poly}}$)**.** *There is an absolute constant $\delta > 0$ such that for all constants $\varepsilon > 0$, at least one of the following holds:*

- $\mathsf{NE} \not\subset \mathsf{P}_{/poly}$.
- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N \times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Low-Rank Matrices as a Circuit Class, and Corresponding Circuit Analysis Algorithms. We begin with the observation that we can view low-rank matrices over $\mathbb{F}_2$ as a special type of 'circuit' defined by a pair of matrices. That is, supposing $M \in \mathbb{F}_2^{N \times N}$ is a matrix with rank r (think of $r \ll N$), then there are matrices $A \in \mathbb{F}_2^{N \times r}$ and $B \in \mathbb{F}_2^{r \times N}$ such that $M = A \cdot B$. Assuming N is a power of 2 for simplicity, M can be interpreted as (the truth-table of) a Boolean function $f : \{0,1\}^{2 \log N} \to \{0,1\}$, which has a special type of circuit of size $O(N \cdot r)$ defined by A and B.

In this way, our task of constructing rigid matrices can equivalently be viewed as the task of proving a certain average-case lower bound against this special class of circuits. This is how Williams' algorithmic approach [Wil13, Wil14c], which exploits circuit analysis algorithms to prove such lower bounds, comes into play. When given the matrices A, B, the corresponding circuit analysis questions are:

1. *Satisfiability (SAT)*, which asks whether $A \cdot B$ is the all zero matrix,
2. *Derandomization* (CAPP), which asks for an estimate of the probability that a random entry of $A \cdot B$ is 1, and
3. *Counting* (#SAT), which asks for the exact number of ones in $A \cdot B$.

In fact, we observe that given the pair (A, B), we can solve the hardest of these three problems, #SAT, in better-than-2^n time (note $n = 2 \log N$). More formally, let a_i denote the i-th row of A, and let b_j denote the j-th column of B. The goal of #SAT is to count the number of pairs such that $\langle a_i, b_j \rangle = 0$ (the number of ones is N^2 minus the number of zeros). This is exactly an instance of *Counting* OV *over* $\mathbb{F}_2$ ($\mathbb{F}_2$-#OV), with N vectors of r dimensions; compared to the usual OV problem, our inner product here is over $\mathbb{F}_2$ instead of $\mathbb{Z}$. An algorithm by Chan and Williams [CW16] solves this problem in *deterministic* $N^{2-\Omega(1/\log(r/\log N))}$ time, for all $r \leq N^{o(1)}$ (see Section 11.6 for the details). This algorithm will play a crucial part in our construction.

Williams' Algorithmic Approach to Circuit Lower Bounds, and a First Attempt. In a seminal work [Wil13], Williams demonstrated an algorithmic approach to proving circuit lower bounds. At a high level, the approach works as follows: Assuming a circuit lower bound is false, one combines the resulting small circuits with other algorithmic ideas to get a better-than-2^n non-deterministic algorithm for $\mathsf{NTIME}[2^n]$, therefore contradicting the non-deterministic time hierarchy theorem [Zák83].

A first attempt at using this approach in our situation proceeds as follows. Let L be a unary language in $\mathsf{NTIME}[2^n] \setminus \mathsf{NTIME}[2^n/n]$ [Zák83]. Fix an efficient PCP verifier V for L (such as [BV14]). That is, for a function $\ell := \ell(n) = n + O(\log n)$, $V(1^n)$ takes ℓ random inputs, runs in $\mathrm{poly}(n)$ time, and is given access to an oracle $O : \{0,1\}^\ell \to \{0,1\}$ (O corresponds to the length-2^ℓ proof for V, but we will interpret it as an ℓ-bit Boolean function to help with intuition later on), and satisfies the following conditions:

1. (PCP Completeness) if $1^n \in L$, then there exists an oracle O such that $V(1^n)^O$ always accepts;

2. (PCP Soundness) if $1^n \notin L$, then for all possible oracles O, the probability $V(1^n)^O$ accepts is $\leq 1/3$.

Intuitively, we are going to show that the truth table of the oracle O which makes V always accept (in the PCP Completeness case) has to be a rigid matrix. More precisely, letting $N = 2^{\ell/2}$, we can fix a P^{NP} machine M_{rigid} such that, on input 1^N, $M_{\mathsf{rigid}}(1^N)$ outputs the lexicographically first oracle O_n which makes $V(1^n)$ always accept. M_{rigid} runs in P^{NP} (on input 1^N, which has length $2^{\Omega(n)}$), since it can guess the oracle outputs bit by bit, using its NP oracle to verify its guesses. The output of $M_{\mathsf{rigid}}(1^N)$, and hence O_n itself, can be viewed as a matrix from $\{0,1\}^{N\times N}$ which we want to show is rigid.

Assume toward a contradiction that $\mathscr{R}_{M_{\mathsf{rigid}}(1^N)}(r) \leq \delta \cdot N^2$ for a small constant δ (one can think of $r := 2^{(\log N)^{1-\varepsilon}}$ for a small constant $\varepsilon > 0$) for all N. It follows that O_n can be $(1-\delta)$-approximated by a matrix of rank at most r. We can thus attempt to solve L as follows:

- Given an input 1^n, we guess matrices $A \in \mathbb{F}_2^{N\times r}$ and $B \in \mathbb{F}_2^{r\times N}$ in $\widetilde{O}(r \cdot 2^{n/2})$ time, with the hope that $M := A \cdot B$ approximates O_n.

- We estimate
$$p_{\mathsf{acc}}(M) = \Pr_{\tau\in\{0,1\}^\ell}[V(1^n)^M(\tau) = 1],$$
and accept only if $p_{\mathsf{acc}}(M) \geq 2/3$.

Following Williams' approach, the hope is that we can estimate $p_{\mathsf{acc}}(M)$ in $2^n/n$ time (i.e. faster than iterating over all choices of the randomness τ) by taking advantage of the given low-rank approximation of M, combined with the $\#\mathsf{SAT}$ algorithm for low rank matrices. If this were possible, it would put L in $\mathsf{NTIME}[2^n/n]$, and contradict the non-deterministic time hierarchy theorem, completing our proof.

Two Issues with the First Attempt. Unfortunately, there are two main issues with this attempt. The first issue is that $V(1^n)^M(\cdot)$ can no longer be written as a low-rank matrix, even if M can. Ideally we would like $V(1^n)^M(\cdot)$ to be a low-rank matrix so that our #SAT algorithm applies to estimate $p_{\mathsf{acc}}(M)$; without this condition, it's unclear how the low-rank matrix M is helpful. From [BV14], one can actually take $V(1^n)$ to be a *3-CNF*, but this is still not enough, since a 3-CNF of low-rank matrices is not necessarily a low-rank matrix.

The second issue is more subtle. If we can estimate $p_{\mathsf{acc}}(M)$ with a high enough accuracy, clearly we will always reject when $1^n \notin L$, by the soundness of the PCP. But, in order to accept when $1^n \in L$, even if we have guessed an M which $(1-\delta)$-approximates O_n, it still could be the case that $p_{\mathsf{acc}}(M)$ is small. For instance, what if $V(1^n)$ always queries positions on which M and O_n differ?

We will ultimately resolve the second issue by making the verifier *smooth* (meaning each query is uniformly distributed), which we will explain later. To resolve the first issue, we use a recent idea from Chen and Williams [CW19b], together with easy-witness lemmas [IKW02, MW18].

The Easy Witness Lemma. Assuming $\mathsf{NE} \subset \mathsf{P}_{/\text{poly}}$, by [IKW02], we know that all NE verifiers have polynomial-size witness circuits, including the verifier $V(1^n)$ discussed above. In other words, when $1^n \in L$, before we were only able to assume there is an oracle $O : \{0,1\}^\ell \to \{0,1\}$ such that $V(1^n)^O$ always accepts, but now we can further assume that there is such an oracle which is computed by a circuit $C : \{0,1\}^\ell \to \{0,1\}$ of size n^k for a constant k. Let us set C_{best} to be the lexicographically first circuit having this property. Now we can modify our algorithm from the first attempt by guessing C, and trying to estimate $p_{\mathsf{acc}}(C)$ instead. Notice that with this modification, there are no longer any low-rank matrices involved in our current approach. We will instead use low-rank approximations of the proof for a different PCP, which we describe next.

Smooth PCP of Proximity (PCPP). We are now going to make use of a very recent construction of a smooth PCPP [Par19]. Using PCPPs in conjunction with Williams' algorithmic approach to circuit lower bounds in this way was a key idea from [CW19b]. For a polynomial-size circuit $F : \{0,1\}^n \to \{0,1\}$ (we are eventually going to pick F to be a modification of the circuit C from above), a smooth PCPP verifier $V_{\mathsf{C\text{-}EVAL}}(F)$ for F[1] takes as input a proof π of length $\mathrm{poly}(n)$ and $O(\log n)$ random bits, and makes a constant number of uniformly distributed, non-adaptive queries to the proof and the input (i.e. which bits are queries depend only on the random bits, and each bit has an equal probability of being queried). Moreover, for some small constant $\delta_p > 0$:

- (PCPP Completeness) If $F(\tau) = 1$, then there is a proof π such that $V_{\mathsf{C\text{-}EVAL}}(F)^{\tau\circ\pi}$ always accepts. Moreover, there is a polynomial-time algorithm which computes π given F and τ.

[1] More precisely, $V_{\mathsf{C\text{-}EVAL}}(F)$ is a smooth PCPP for the Circuit-Eval problem, in which we have fixed the circuit to be F.

- (PCPP Soundness) If $F(\tau') = 0$ for every $\tau' \in \{0,1\}^n$ which differs from τ in at most a δ_p fraction of entries, then $\Pr_{u \in \{0,1\}^{O(\log n)}}[V_{\mathsf{C\text{-}EVAL}}(F)^{\tau \circ \pi}(u) = 1] \leq 1/3$ for all possible proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$.

The fact that the queries are both smooth and non-adaptive will be crucial to our construction later on. This 'proximity' aspect of the soundness condition is necessary for these properties to hold. For instance, if one fixed F to be the parity function, then it is not hard to see that such a construction without the 'proximity' aspect (i.e. with $\delta_p = 0$) is impossible. Our goal is to apply such a smooth PCPP to C, but since we don't have any guarantees about which inputs C should reject, we will first need to make some modifications to C to deal with the 'proximity' aspect.

Defining $D_C(\tau) := V(1^n)^C(\tau)$, which is still a polynomial-size circuit, our goal is to design a fast algorithm to estimate

$$p_{\mathsf{acc}}(C) := \Pr_{\tau \in \{0,1\}^\ell}[V(1^n)^C(\tau) = 1] = \Pr_{\tau \in \{0,1\}^\ell}[D_C(\tau) = 1].$$

In preparation for using the PCP of proximity, we next apply an *error correcting code* to τ. Specifically, fix a constant-rate $\mathbb{F}_2$-linear error correcting code ECC with efficient encoder $\mathsf{Enc} : \{0,1\}^\ell \to \{0,1\}^{c_1 \cdot \ell}$ and decoder $\mathsf{Dec} : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}^\ell$ which can recover error up to a δ_{dec} fraction. We define another circuit $E_C : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}$, as $E_C(w) := D_C(\mathsf{Dec}(w))$. That is, E_C treats the input as a codeword of ECC, decodes it, and feeds the result into the circuit D_C. Now our goal is to estimate

$$p_{\mathsf{acc}}(C) = \Pr_{\tau \in \{0,1\}^\ell}[E_C(\mathsf{Enc}(\tau)) = 1].$$

Now we will apply the PCP of Proximity to simplify the estimation of $p_{\mathsf{acc}}(C)$. More precisely, we use a $q = O(1)$ query smooth PCPP, $V_{\mathsf{C\text{-}EVAL}}(E_C)$, for the circuit E_C, which has proximity parameter $< \delta_{\mathsf{dec}}$, proof length $\ell_{\mathsf{proof}} = \mathrm{poly}(\mathsf{SIZE}(E_C)) = \mathrm{poly}(n)$, and number of random bits $m = O(\log \ell_{\mathsf{proof}}) = O(\log n)$. The crucial observation here is that we have dealt with the 'proximity' aspect of the smooth PCPP by using the error correcting code: if $D_C(\tau) = 0$, then $\mathsf{Enc}(\tau)$ is δ_{dec}-far from any yes-inputs to E_C. This is because, for any $w \in \{0,1\}^{c_1 \cdot \ell}$ which is δ_{dec}-close to $\mathsf{Enc}(\tau)$, w decodes to τ and $E_C(w) = D_C(\tau) = 0$.

Summarizing, so far we have the following:

- (PCPP Completeness) If $D_C(\tau) = 1$, then there is a proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$ such that $V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ \pi}$ always accepts. Moreover, given E_C and τ, there is a polynomial-time computable function $\pi(E_C, \tau) \in \{0,1\}^{\ell_{\mathsf{proof}}}$ to compute the proof π.

- (PCPP Soundness) If $D_C(\tau) = 0$, then $\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ \pi}(u) = 1] \leq 1/3$ for all possible proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$.

The P^{NP} Machine M_{rigid}. Finally, we are ready to define our rigid matrix. It will be the concatenation, over all $\tau \in \{0,1\}^\ell$, of the proof $\pi(E_{C_{\mathsf{best}}}, \tau)$ from the

PCPP Completeness condition above. More precisely, let $\pi_{C_{\mathsf{best}}}(\tau, j)$ be the j-th bit of $\pi(E_{C_{\mathsf{best}}}, \tau)$. Note that $\pi_{C_{\mathsf{best}}}$ is a Boolean function on $n_\pi := n + O(\log n)$ bits. Letting $N = 2^{n_\pi/2}$, we define our P^{NP} machine M_{rigid} as the function which, on input 1^N, outputs the truth-table of $\pi_{C_{\mathsf{best}}}$, which we interpret as a matrix in $\{0,1\}^{N \times N}$. M_{rigid} runs in P^{NP} since, similar to before, one can guess C_{best} bit-by-bit and verify with the NP oracle.

Again, assume toward a contradiction that $\mathscr{R}_{M_{\mathsf{rigid}}(1^N)}(r) \leq \delta \cdot N^2$ for a small constant δ (recall that one can think of $r := 2^{(\log N)^{1-\varepsilon}}$ for a small constant $\varepsilon > 0$) for all N. That is, we know $\pi_{C_{\mathsf{best}}}(\cdot, \cdot)$ can be $(1-\delta)$-approximated by a matrix M of rank at most r. We guess a low-rank decomposition of that matrix $M = A \cdot B$, in $O(N \cdot r)$ time, and now we wish to estimate

$$p_{\mathsf{acc}}(M) := \Pr_{u \in \{0,1\}^m, \tau \in \{0,1\}^\ell}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(\tau) \circ M(\tau, \cdot)}(u) = 1].$$

Recall that τ is the randomness to the old PCP verifier V, of length $\ell = n + O(\log n)$, and u is the randomness to the new smooth PCPP verifier $V_{\mathsf{C\text{-}EVAL}}(E_C)$, of length $m = O(\log n)$.

Fast Algorithm for Computing $p_{\mathsf{acc}}(M)$. We now use the fact that the queries made by $V_{\mathsf{C\text{-}EVAL}}(E_C)$ *only depend on* u. Our algorithm will simply iterate over all poly(n) choices of u. Hence, fix $u \in \{0,1\}^m$, and suppose V queries $M(\tau, j_1), M(\tau, j_2), \ldots, M(\tau, j_{q_1})$ in $M(\tau, \cdot)$, and $e_1, e_2, \ldots, e_{q_2}$ in $\mathsf{Enc}(\tau)$. Now we want to estimate

$$\Pr_{\tau \in \{0,1\}^\ell}[F_u(M(\tau, j_1), M(\tau, j_2), \ldots, M(\tau, j_{q_1}), \mathsf{Enc}(\tau)_{e_1}, \mathsf{Enc}(\tau)_{e_2}, \ldots, \mathsf{Enc}(\tau)_{e_{q_2}}) = 1]$$

for a Boolean function F_u on $q = q_1 + q_2 = O(1)$ inputs. Next, using a standard trick from the analysis of Boolean functions, we observe that since we are aiming to compute the expected value of F_u, we can assume that F_u is a *parity* function. In other words, it is sufficient to quickly estimate

$$\Pr_{\tau \in \{0,1\}^\ell}[M(\tau, j_1) + M(\tau, j_2) + \ldots + M(\tau, j_{q_1}) + \mathsf{Enc}(\tau)_{e_1} + \mathsf{Enc}(\tau)_{e_2} + \ldots + \mathsf{Enc}(\tau)_{e_{q_2}} = 1],$$

where the sum is taken mod 2. The parity of $M(\tau, j_1) + M(\tau, j_2) + \ldots + M(\tau, j_{q_1})$, which is a sum of a constant number of low-rank matrices, can itself be written as a low rank matrix. Since Enc is a linear function over $\mathbb{F}_2$, incorporating $\mathsf{Enc}(\tau)_{e_1} + \mathsf{Enc}(\tau)_{e_2} + \ldots + \mathsf{Enc}(\tau)_{e_{q_2}}$, which is a linear function of the *indices* of the matrix, can only increase the rank by an additive constant. Hence, our goal is exactly to compute the number of 1s in a low rank matrix. This is an instance of the previously discussed #SAT problem for low-rank matrices which, as discussed, can can be solved in $N^{2-\Omega(1/\log r)}$ time as described by [CW16].

Notice that:

- If $1^n \in L$, and we guessed the circuit C_{best} and a matrix M which $(1-\delta)$-approximates $\pi_{C_{\mathsf{best}}}$, then $p_{\mathsf{acc}}(M) \geq 1 - q \cdot \delta$ since $V_{\mathsf{C\text{-}EVAL}}(E_C)$'s queries are *smooth* (meaning, uniformly distributed over the proof).

- If $1^n \notin L$, then for all possibles guesses, $p_{\mathsf{acc}}(M) \leq 1/2$, by the soundness of PCPP and PCP.

Putting everything together, it follows that L is in non-deterministic time

$$\mathrm{poly}(n) \cdot N^{2-\Omega(1/\log r)} = 2^{n-\Omega(n/\log r)} = 2^{n-\Omega(n^{\varepsilon})},$$

contradicting the non-deterministic time hierarchy. This completes the proof overview for Theorem 11.3.

11.1.2 Unconditional Construction of Rigid Matrices

Getting Rid of the Easy-Witness Assumption: A Boot-Strapping Scheme. We now move on to a proof overview of Theorem 11.1. Note that in the above argument, the only consequence of $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$ used is the fact that $V(1^n)$ has a succinct witness circuit. In order to get rid of the assumption $\mathsf{NE} \subset \mathsf{P}_{/\mathrm{poly}}$, we next show how to construct a succinct witness for $V(1^n)$ solely based on the assumption that all P^{NP} machines have non-rigid output matrices.

The key idea is based on a *bootstrapping* argument. Observe that an $N^{o(1)}$-rank decomposition of a matrix $M \in \{0,1\}^{N \times N}$ actually compresses the N^2 bits of M into an $N^{1+o(1)}$ bit representation. If we can further treat those bits after the compression as a low-rank matrix, and compress it again, and so on, we can further reduce the number of bits required to represent the matrix.

A key property of low-rank decompositions we will use is that they are *locally decodable*. That is, if A, B are the two matrices of a rank-r expression for M, then one can compute a particular entry $M_{i,j}$, by looking at only $O(r)$ entries of the matrices A and B (the ith row of A and the jth column of B).

High-Level Idea. Recall from the proof above that O_n is the lexicographically first oracle which makes $V(1^n)$ always accept. The high level idea for constructing a succinct witness for O_n is as follows. We first interpret O_n as a matrix $M_1 \in \{0,1\}^{N_1 \times N_1}$. Letting (A_1, B_1) be its low-rank decomposition, we then interpret the concatenation (A_1, B_1) as a matrix $M_2 \in \{0,1\}^{N_2 \times N_2}$. We will show that M_2 also has a low-rank decomposition (A_2, B_2). We then interpret this as a matrix $M_3 \in \{0,1\}^{N_3 \times N_3}$, and repeat until we have a small enough matrix $M_k \in \{0,1\}^{N_k \times N_k}$. Note that for all i, we have $N_i = N_{i-1}^{1/2+o(1)}$; that is, each time we compress the bits by about a square-root.

Why do all these matrices have low-rank approximations? This follows from our assumption that all P^{NP} machines' output matrices are non-rigid, and hence have low-rank approximations. First, similar to before, we know that there is a P^{NP} algorithm M that, on input $1^{\sqrt{|O_n|}}$, outputs $O_n = M_1$. Then, we can recursively show that each of the matrices $M_2, \ldots, M_k$ can be constructed by an NP oracle machine: for each i, to construct M_i, we use the oracle to find the lexicographically first low-rank approximation of M_{i-1}.

Our succinct witness for O_n is M_k for a large constant k. M_k is small enough that we can construct a circuit for it by brute-force. The idea is to then repeatedly use the local decoding scheme we discussed earlier to construct circuits for $M_{k-1}, M_{k-2}, \ldots, M_1$, since each corresponds to a low-rank approximation of the next. However, having a low-rank approximation of M_i is not enough to recover M_i exactly. To circumvent this issue, we apply *locally-decodable codes* to the matrices. Indeed, if our low-rank decomposition $A_i \cdot B_i$ gives a $(1-\delta)$-approximation to the matrix $\mathsf{Enc}(M_i)$ (the encoding of M_i using a suitable locally-decodable code), rather than to M_i, then we can use the local decoder to compute M_i exactly.

Locally-Decodable Codes and the Actual Compression Scheme $f_i(\cdot)$. We now give more details of the construction. We fix a locally-decodable code $\mathsf{ECC}_{\mathsf{local}}$, with message length $n^{1+\varepsilon_{\mathsf{enc}}}$ ($\varepsilon_{\mathsf{enc}}$ can be made an arbitrarily small constant), and a polylog(n)-time local decoder. Let the encoder be $\mathsf{Enc} : \{0,1\}^n \to \{0,1\}^{n^{1+\varepsilon_{\mathsf{enc}}}}$. The local decoder implies that, for $S \in \{0,1\}^n$, if we have a T-size circuit which approximates the string $\mathsf{Enc}(S)$, then there is a $\mathrm{polylog}(n) \cdot T$-size circuit which computes S exactly.

We now define two functions to describe how to go from a matrix to its low-rank decomposition. First define the function $\mathsf{rk}(N) = 2^{(\log N)^b}$ for a constant $b > 0$. Then, for a string S, define $\mathsf{comp}(S)$ as follows: Let $N = \sqrt{|S|}$ (we will pretend here that $|S|$ is the square of an integer; in the real proof we use a slight padding to make sure of this), and let A, B be two matrices in $\{0,1\}^{N \times \mathsf{rk}(N)}$ and $\{0,1\}^{\mathsf{rk}(N) \times N}$, respectively, such that $A \cdot B$ equals S on the most possible positions (viewing S as a matrix in $\{0,1\}^{N \times N}$). If there are multiple equally good options for A, B, then pick the lexicographically first one. We then define $\mathsf{comp}(S) = A \circ B$, as the concatenation of matrices A and B.

Next, we define a series of functions which recursively give compressions of a given string S:

$$f_i(S) := \begin{cases} \mathsf{Enc}(S) & i = 1, \\ \mathsf{Enc}(\mathsf{comp}(f_{i-1}(S))) & i \geq 2. \end{cases}$$

Note that A and B (the outputs of $\mathsf{comp}(S)$) can be computed from S in $\mathsf{TIME}[\mathrm{poly}(|S|)]^{\mathsf{NP}}$. Now, we set $\ell_{n,i} = \sqrt{|f_i(O_n)|}$. We can then pick our NP oracle machine to, on input $1^{\ell_{n,i}}$, output the corresponding matrix for $f_i(O_n)$. (In the full proof below we use some simple tricks to make sure the $\ell_{n,i}$'s are all distinct.) Therefore, by assumption, we know that each $f_i(O_n)$ can be approximated by a $\mathsf{rk}(\ell_{n,i})$-rank matrix.

Finally, we are ready to implement our bootstrapping. We know that, for a parameter j, $f_j(O_n)$ has an $\ell_{n,j}$-size circuit which computes it *exactly*. Suppose we have a T-size circuit C_j which $(1-\delta)$-approximates $f_j(O_n)$. From this we can construct a circuit C_{j-1} which $(1-\delta)$-approximates $f_{j-1}(O_n)$ as follows:

- First, applying the local decoder for $\mathsf{ECC}_{\mathsf{local}}$, we can construct a $\mathrm{polylog}(\ell_{n,j-1}) \cdot T$-size circuit C_{comp} which exactly computes $\mathsf{comp}(f_{j-1}(O_n))$.

- Let A, B be the matrices corresponding to $\mathsf{comp}(f_{j-1}(O_n))$. By our assumption, $A \cdot B$ is a $(1-\delta)$-approximation for $f_{j-1}(O_n)$. We know that $(A \cdot B)_{x,y}$ can be computed in $\mathsf{rk}(\ell_{n,j-1})$ time, given oracle access to C_{comp}. It follows from the locally-decodable property of $\mathsf{ECC}_{\mathsf{local}}$ that we get a circuit of size $\mathsf{rk}(\ell_{n,j-1}) \cdot \mathrm{polylog}(\ell_{n,j-1}) \cdot T$ which approximates $f_{j-1}(O_n)$.

From this construction, we can show that $f_1(O_n) = \mathsf{Enc}(O_n)$ can be approximated by a small circuit, which in turn shows O_n has a small exact circuit. It is not hard to see, in particular, that

$$\mathsf{SIZE}(O_n) \leq \prod_{i=1}^{j-1} \mathsf{rk}(\ell_{n,j-1})^{1+o(1)} \cdot |\ell_{n,j}| = \mathrm{poly}(\mathsf{rk}(|O_n|)) \cdot |\ell_{n,j}| = 2^{O(n^b)} \cdot |\ell_{n,j}|.$$

The Constant $1/4 - \varepsilon$. Supposing that O_n has a 2^{n^a}-size witness, and the rank we consider is $\mathsf{rk}(N) = 2^{(\log N)^b}$, the running time of our algorithm is

$$2^{O(n^a)} \cdot 2^{n-\Omega(n^{1-b})} = 2^{n+O(n^a)-\Omega(n^{1-b})}.$$

In order to make the above faster than 2^n and get a contradiction, we want to pick $b < 1 - a$. From the bound on $\mathsf{SIZE}(O_n)$, we can see that the bootstrapping scheme can only achieve $a > b$. Therefore, we set $a = 1/2 + \varepsilon$ and $b = 1/2 - 2\varepsilon$, for a small constant $\varepsilon > 0$.

We now consider the running time of M_{comp}. Since we only aim to compress O_n to a witness of size $2^{O(n^a)}$, we can stop if we find $\ell_{n,j} \leq 2^{n^a}$, as there is no need to further compress. Let $M := \ell_{n,j}$. On input 1^M, M_{comp} needs $\mathrm{poly}(\ell_{n,1}) = 2^{O(n)} \leq M^{\log M}$ time to compute $f_i(O_n)$. Therefore, M_{comp} runs in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$, and hence yields a rigid matrix constructible in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ for rank $2^{(\log N)^{1/2-2\varepsilon}}$. In other words, the time is slower than we hoped for, but the rank is higher than we hoped for.

Finally, we use a tensor product argument (Lemma 11.5 below) to transform this into a P^{NP} construction, which is rigid for a worse rank of $2^{(\log N)^{1/4-\varepsilon}}$. The idea is to take the tensor product of our rigid matrix with a large all-1s matrix. The resulting matrix is still rigid for the same rank, but has larger dimensions. Equivalently, in terms of the dimension N of the matrix, the complexity to compute the matrix has gone down, but it is also rigid for a lower rank.

11.2 Tools from Complexity Theory

Our construction of rigid matrices makes use of a number of tools from the complexity theory literature; in this Section we precisely define the tools from prior work which we will use.

The *Circuit Evaluation Problem* ($\mathsf{Circuit\text{-}Eval}$) is the language of pairs (C, w) where C is a general fan-in-2 circuit, and w is an input such that $C(w) = 1$. For two strings a, b, we use $a \circ b$ to denote their concatenation.[2]

[2] The symbol $\circ$ is also used for circuit composition; its meaning will always be clear from context.

Probabilistic Checkable Proofs of Proximity

Our proof will make heavy use of probabilistically checkable proofs of proximity.

Definition 11.1 (Probabilistic Checkable Proofs of Proximity (PCP of proximity, or PCPP)). *For $s, \delta : \mathbb{N} \to [0,1]$ and $r, q : \mathbb{N} \to \mathbb{N}$, a verifier V is a PCP of proximity system for a pair language L with proximity parameter δ, soundness parameter s, number of random bits r and query complexity q if the following holds for all x, y:*

- *(Completeness) If $(x, y) \in L$, then there is a proof π such that $V(x)$ accepts oracle $y \circ \pi$ with probability 1.*
- *(Soundness) If y is $\delta(|x|)$-far from $L(x) := \{z : (x, z) \in L\}$, then for all proofs π, $V(x)$ accepts oracle $y \circ \pi$ with probability at most $s(|x|)$.*
- *$V(x)$ tosses $r(|x|)$ random coins, and makes at most $q(|x|)$ non-adaptive queries.*

Lemma 11.1 ([BGH+06, Theorem 3.3]). *For any constants $0 < \delta, s < 1$, there is a PCP of proximity system for Circuit-Eval with proximity δ, soundness s, number of random bits $r = O(\log n)$ and query complexity $q = O(1)$. Moreover, given the pair $(C, w) \in$ Circuit-Eval, a proof π which makes $V(C)$ always accept can be constructed in poly$(|C| + |w|)$ time.*

Remark 11.1. *The last ('Moreover') sentence is not explicitly stated in [BGH+06], but it is evident from their construction.*

In this paper, we need a stronger PCPP construction which is additionally *smooth*, meaning, every position in the proof π is queried with equal probability (assuming without loss of generality that all queries are non-adaptive and distinct). Such a construction can be found in [Par19].[3]

Lemma 11.2 ([Par19]). *For any constants $0 < \delta, s < 1$, there is a smooth PCP of proximity system for Circuit-Eval with proximity δ, soundness s, number of random bits $r = O(\log n)$ and query complexity $q = O(1)$. Moreover, given the pair $(C, w) \in$ Circuit-Eval, a proof π making $V(C)$ always accepts can be constructed in poly$(|C| + |w|)$ time.*

Error Correcting Codes

We also need standard constructions of two different types of codes: constant-rate linear error correcting codes, and n^{ε}-rate codes with polylog(n) time local decoders.

Lemma 11.3 ([Spi96]). *There is a constant-rate linear error correcting code ECC with a linear-time encoder Enc and a linear-time decoder Dec recovering error up to a universal constant δ.*

Lemma 11.4 (cf, Section 2.3 of [Yek12]). *For any constant $\varepsilon > 0$, there is a n^{ε}-rate error correcting code ECC with a poly(n)-time encoder Enc and a* polylog(n)*-time local-decoder Dec which recovers up to a 0.01 fraction of errors.*

[3] [Par19]'s construction actually ensures that this holds for every query position in the second input y as well. This additional property is not required by our proof.

A Simple Fact About Matrix Rigidity

We use $\mathbf{1}_N$ to denote the all-ones matrix of size $N \times N$, and $\otimes$ to denote the Kronecker product of matrices.

Lemma 11.5. *For any field $\mathbb{F}$ and any matrix $A \in \mathbb{F}^{M \times M}$, we have*

$$\mathscr{R}_{\mathbf{1}_N \otimes A}(r) = \mathscr{R}_A(r) \cdot N^2.$$

Proof. We first show $\mathscr{R}_{\mathbf{1}_N \otimes A}(r) \geq \mathscr{R}_A(r) \cdot N^2$. Assume to the contrary that there is a way to change $k < \mathscr{R}_A(r) \cdot N^2$ entries of $\mathbf{1}_N \otimes A$ to make its rank r. The matrix $\mathbf{1}_N \otimes A$ consists of N^2 disjoint copies of A, so by the pigeonhole principle, there were at most $k/N^2 < \mathscr{R}_A(r)$ entries changed in one of those copies of A. Thus, that submatrix still has rank greater than r after the change, a contradiction.

We next show that $\mathscr{R}_{\mathbf{1}_N \otimes A}(r) \leq \mathscr{R}_A(r) \cdot N^2$. Let B a matrix of rank r whose Hamming distance from A is $\mathscr{R}_A(r)$. Thus, $\mathbf{1}_N \otimes B$ has rank $\operatorname{rank}(\mathbf{1}_N) \cdot \operatorname{rank}(B) = r$, and its Hamming distance from $\mathbf{1}_N \otimes A$ is $\mathscr{R}_A(r) \cdot N^2$. □

$\mathbb{F}_{p^r}$-#OV

One crucial component of our construction is the algorithm for $\mathbb{F}_{p^r}$-#OV from [CW16].

Definition 11.2. *For a prime power $q = p^r$, in an $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$ instance, we are given two collections of vectors from $\mathbb{F}_q^d$, $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, and want to compute the number of pairs such that $\langle a_i, b_j \rangle = 0$ over $\mathbb{F}_q$.*

We use the following algorithm for $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$.

Theorem 11.4 ([CW16])**.** *For all fixed prime powers $q = p^r$, there is an $n^{2-\Omega(1/\log(d/\log n))}$ time deterministic algorithm for $\mathbb{F}_q$-#$\mathsf{OV}_{n,d}$, when $d = n^{o(1)}$.*

The original paper [CW16] only states an algorithm for #OV (the problem when A and B are collections of vectors from $\{0,1\}^d$ and the inner product is over $\mathbb{Z}$). We make two small modifications to their algorithm to get the result stated in Theorem 11.4 above; see Section 11.6 for details.

11.3 Construction of Rigid Matrices Assuming an Easy Witness Lemma

We now move on to the formal construction and proof. We begin in this Section by proving Theorem 11.2.

We say an algorithm is a *matrix-constructing algorithm* if on input 1^N, it outputs a matrix in $\{0,1\}^{N \times N}$. We say a function $f : \mathbb{N} \to \mathbb{N}$ is a *typical resource bound function* if it is strictly increasing, and satisfies $f(n) = \omega(f(n+1)/(n+1))$. We first prove the following lemma, which says that if certain non-deterministic time classes have easy witnesses, then there is a P^{NP} construction of rigid matrices.

Lemma 11.6. *There is an absolute constant $\delta > 0$ such that, for all prime powers $q = p^r$, and any three typical resource bound functions $T, S, R : \mathbb{N} \to \mathbb{N}$ with $T(n), S(n) \geq n$ for all n, the following three conditions cannot hold simultaneously.*

(1) All polynomial-time verifiers[4] *for unary $\mathsf{NTIME}[T(n)]$ languages have $S(n)$-size witness circuits.*

(2) For all P^{NP} matrix-constructing algorithms M, $\mathscr{R}_{M(1^N)}(R(N)) \leq \delta \cdot N^2$ for almost all N.

(3) $\log T(n)/\log R(N) = \omega(\log\log T(n) + \log S(n))$, where $N = 2^{n_\pi/2}$, for $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n))$, and $R(N) = N^{o(1)}$.

Remark 11.2. *In the following proof, we will actually only need the first assumption to hold for the special PCP verifier $V(1^n)$ of the language L we consider. This remark will be useful in the proof in the next Section.*

Proof. Let $\delta > 0$ be a constant to be decided later. We first only consider the case when the field is $\mathbb{F}_2$, and then show how to generalize the argument for other finite fields. We will assume that all three items are true, and derive a contradiction.

Unary Language L and PCP. Let L be a unary language in $\mathsf{NTIME}[T(n)] \setminus \mathsf{NTIME}[T(n)/n]$. Using the non-deterministic time hierarchy theorem [Zák83], such an L exists because $T(n)$ is a typical resource function. Let V be an efficient PCP verifier for L from [BV14]. That is, there is a function $\ell = \ell(n) = \log T(n) + O(\log\log T(n))$, such that $V(1^n)$ takes an oracle $O : \{0,1\}^\ell \to \{0,1\}$ and ℓ random bits as input, runs in $\mathrm{poly}(n)$ time, and:

1. (PCP Completeness) If $1^n \in L$, then there exists a circuit $C : \{0,1\}^\ell \to \{0,1\}$ of size $S(n)$ such that $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^C(r) = 1] = 1$. (This follows from the first assumption of the Lemma.)
2. (PCP Soundness) If $1^n \notin L$, then for all oracles $O : \{0,1\}^\ell \to \{0,1\}$, we have $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^O(r) = 1] \leq 1/n$.

We will next show how to put $L \in \mathsf{NTIME}[T(n)/n]$ by using the second and the third assumptions of the Lemma, which will give us the contradiction we want.

The Plan. Let C_{best} be the circuit of size $S(n)$ such that $\Pr_{r\in\{0,1\}^\ell}[V(1^n)^{C_{\mathsf{best}}}(r) = 1] = 1$, and if there are multiple such circuits, we break the tie by choosing the lexicographically first one. Note that such a circuit doesn't exist when $1^n \notin L$, and in that case we set C_{best} to be a trivial circuit which always outputs 0.

In our non-deterministic algorithm to solve L, given an input 1^n, we first guess a circuit C of size at most $S(n)$, and wish to ensure that the following two conditions hold:

[4] That is, for $L \in \mathsf{NTIME}[T(n)]$, the verifier V takes two inputs x, y with $|x| = n$ and $|y| = \mathrm{poly}(T(n))$, runs in $\mathrm{poly}(|x| + |y|)$ time, and has the property that $x \in L$ if and only if there is a y such that $V(x, y) = 1$.

1. When $1^n \in L$ and $C = C_{\mathsf{best}}$, we accept, and

2. When $1^n \notin L$, we always reject.

If our algorithm satisfies these two conditions and runs in $T(n)/n$ non-deterministic time, then we have put $L \in \mathsf{NTIME}[T(n)/n]$ and arrived at the desired contradiction.

Implementation. Now suppose we have guessed a circuit C of size at most $S(n)$. Toward achieving the two conditions above, we want to estimate

$$p_{\mathsf{acc}}(C) := \Pr_{r \in \{0,1\}^\ell}[V(1^n)^C(r) = 1].$$

Define another circuit $D_C : \{0,1\}^\ell \to \{0,1\}$ as $D_C(r) := V(1^n)^C(r)$. We thus equivalently have that

$$p_{\mathsf{acc}}(C) = \Pr_{r \in \{0,1\}^\ell}[(D_C, r) \in \mathsf{Circuit\text{-}Eval}],$$

by the definition of $\mathsf{Circuit\text{-}Eval}$.

Applying Error Correcting Codes. Fix an $\mathbb{F}_2$-linear error correcting code ECC with rate c_1 and recovering error δ_1, whose existence is guaranteed by Lemma 11.3. Let $\mathsf{Enc} : \{0,1\}^\ell \to \{0,1\}^{c_1 \cdot \ell}$ and $\mathsf{Dec} : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}^\ell$ be the corresponding linear-time encoder and decoder.

We now define yet another circuit $E_C : \{0,1\}^{c_1 \cdot \ell} \to \{0,1\}$ as $E_C(w) := D_C(\mathsf{Dec}(w))$. Then it suffices to estimate

$$p_{\mathsf{acc}}(C) = \Pr_{r \in \{0,1\}^\ell}[(E_C, \mathsf{Enc}(r)) \in \mathsf{Circuit\text{-}Eval}].$$

Notice that $\mathsf{SIZE}(E_C) \leq \mathrm{poly}(n) \cdot S(n)$, since the verifier $V(1^n)$ runs in $\mathrm{poly}(n)$ time, and the decoder Dec runs in linear time.

Applying the PCPP. Now we use a $q_{\mathsf{PCPP}} = O(1)$-query smooth PCPP for $\mathsf{Circuit\text{-}Eval}$ from Lemma 11.2 with constant soundness s_{PCPP} and proximity parameter δ_{PCPP} to be specified later. Let $V_{\mathsf{C\text{-}EVAL}}(E_C)$ be the verifier for this smooth PCPP with the circuit fixed to E_C. Hence, $V_{\mathsf{C\text{-}EVAL}}(E_C)$ uses proof length $\ell_{\mathsf{proof}} = \mathrm{poly}(\mathsf{SIZE}(E_C)) = \mathrm{poly}(S(n))$ and $m = O(\log \ell_{\mathsf{proof}})$ random bits.

Claim 11.1. *$V_{\mathsf{C\text{-}EVAL}}(E_C)$ satisfies the following three properties by setting $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{dec}}$, and $s_{\mathsf{PCPP}} = 1/3$.*

1. *(PCPP Completeness) If $(D_C, r) \in \mathsf{Circuit\text{-}Eval}$, there is a proof $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$ that*
$$\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r) \circ \pi}(u)] = 1.$$

2. *(From* PCPP *Smoothness) Suppose $(D_C, r) \in$ Circuit-Eval, and let π be a proof satisfying the previous property. If proof $\widetilde{\pi} \in \{0,1\}^{\ell_{\mathsf{proof}}}$ is a $(1-\delta)$-approximation to π for some $\delta \in [0,1]$, then*

$$\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\widetilde{\pi}}(u)] \geq 1 - q_{\mathsf{PCPP}} \cdot \delta.$$

3. *(*PCPP *Soundness) If $(D_C, r) \notin$ Circuit-Eval, then for all proofs $\pi \in \{0,1\}^{\ell_{\mathsf{proof}}}$, we have*

$$\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\pi}(u)] \leq 1/3.$$

Property (1) of Claim 11.1 follows from the completeness property of the PCPP system, and property (2) follows from the smoothness of the PCPP system combined with a simple union bound.

For property (3), note that if $(D_C, r) \notin$ Circuit-Eval, then $\mathsf{Enc}(r)$ is δ_{dec}-far from the set $\{w \in \{0,1\}^{c_1 \cdot \ell} : (E_C, w) \in \text{Circuit-Eval}\}$. This is because for any string $w \in \{0,1\}^{c_1 \cdot \ell}$ which is $< \delta_{\mathsf{dec}}$-close to $\mathsf{Enc}(r)$, we know $\mathsf{Dec}(w) = r$ and hence $(E_C, w) \notin$ Circuit-Eval. Therefore, by setting $\delta_{\mathsf{PCPP}} < \delta_{\mathsf{dec}}$, and $s_{\mathsf{PCPP}} = 1/3$, property (3) follows from the soundness of the PCPP system.

The Function $\pi_{C_{\mathsf{best}}}(r, j)$. Note that by Lemma 11.2, there is a polynomial time computable function $\pi(E_C, \mathsf{Enc}(r)) \in \{0,1\}^{\ell_{\mathsf{proof}}}$, such that when $(E_C, \mathsf{Enc}(r)) \in$ Circuit-Eval, we have

$$\Pr_{u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r)\circ\pi(E_C,\mathsf{Enc}(r))}(u)] = 1.$$

Define the Boolean function $\pi_C(r, j)$, for $j \in [\ell_{\mathsf{proof}}]$, to be the j-th bit of $\pi(E_C, r)$ (suppose ℓ_{proof} is a power of 2 for simplicity).

The function $\pi_{C_{\mathsf{best}}}(r, j)$ is computable in E^{NP}, by using the following procedure. First we show how to compute the circuit C_{best} in E^{NP}. We are given two inputs r, j with length $|r| = \ell$ and $|j| = \log \ell_{\mathsf{proof}} = O(\log S(n))$. In $O(2^\ell)$ time with an NP oracle, we can first decide whether $V(1^n)$ always accepts a circuit of size at most $S(n)$. If not, then we just output a trivial circuit. If so, then we guess that circuit bit by bit to construct the lexicographically first one, again using the NP oracle to check each guess. In this way, we can compute C_{best} in E^{NP}. We can then construct the circuit $E_{C_{\mathsf{best}}}$ from C_{best}, and then (using the fact that $\pi(E_C, \mathsf{Enc}(r))$ can be computed in polynomial time) compute $\pi_{C_{\mathsf{best}}}(r)$ and output its j-th bit. The whole procedure runs in E^{NP}.

The P^{NP} Machine M_{rigid}. Note that $\pi_{C_{\mathsf{best}}}$ has input length $n_\pi = \ell + O(\log S(n))$. We can thus construct a P^{NP} machine M_{rigid} such that, given an input $1^{2^{n_\pi/2}}$, it outputs the truth-table of $\pi_{C_{\mathsf{best}}}$ as a matrix. Therefore, by the second assumption of the Lemma, $\pi_{C_{\mathsf{best}}}$ as a matrix can be δ-approximated by a matrix of rank $R(2^{n_\pi/2})$.

Putting L in $\mathsf{NTIME}[T(n)/n]$. Finally, consider the following algorithm for solving L. We first guess a circuit C of size $S(n)$, with the hope that it is C_{best}. Then, letting $N = 2^{n_\pi/2}$, we guess a matrix $M : N \times N$ of rank $R(N)$, with the hope that it δ-approximates $\pi_{C_{\mathsf{best}}}$. More specifically, we guess two matrices U, V of size $N \times R(N)$ and $R(N) \times N$, and set (implicitly, without explicitly computing it) $M = UV$.

Now we try to calculate

$$p_{\mathsf{acc}}(M) := \Pr_{r \in \{0,1\}^\ell, u \in \{0,1\}^m}[V_{\mathsf{C\text{-}EVAL}}(E_C)^{\mathsf{Enc}(r) \circ M(r)}(u) = 1].$$

Fix u, and suppose that for randomness u, the verifier $V_{\mathsf{C\text{-}EVAL}}(E_C)$ queries $M(r, j_1), M(r, j_2), \ldots, M(r, j_{q_1})$ in $M(r, \cdot)$, and $e_1, e_2, \ldots, e_{q_2}$ in $\mathsf{Enc}(r)$ (note that $V_{\mathsf{C\text{-}EVAL}}(E_C)$'s query positions only depend on the randomness u). Now we want to estimate

$$\Pr_{r \in \{0,1\}^\ell}[F_u(M(r, j_1), M(r, j_2), \ldots, M(r, j_{q_1}), \mathsf{Enc}(r)_{e_1}, \mathsf{Enc}(r)_{e_2}, \ldots, \mathsf{Enc}(r)_{e_{q_2}}) = 1]$$

for a Boolean function F_u on $q_{\mathsf{PCPP}} = q_1 + q_2$ inputs. First, we can write F_u in the basis of XOR functions:

$$F_u(z_1, z_2, \ldots, z_{q_{\mathsf{PCPP}}}) = \sum_{S \subseteq [q_{\mathsf{PCPP}}]} \alpha_S \cdot \bigoplus_{i \in S} z_i.$$

(Here, we consider the XOR function $\oplus$ to be outputting a $\{0, 1\}$ value, and the coefficients α_S and the sum Σ are over $\mathbb{R}$, not over $\mathbb{F}_2$.) Since our goal is to compute the expected value of F_u, by linearity of expectation, it suffices to separately compute the expected value of each of the (constant number of) parity functions. Therefore, it suffices to consider the case when F_u is just an XOR function.

Also, note that since ECC is a linear code, it follows that $\mathsf{Enc}(r)_k$ is an XOR function on a subset of coordinates of r. Thus, if $r = a \circ b$ where $|a| = n_\pi/2$ and $|b| = |r| - |a|$ (note that $\ell > n_\pi/2$ by the third assumption of the Lemma), we have $\mathsf{Enc}(r)_e = \mathsf{Enc}_L(a)_e \oplus \mathsf{Enc}_R(b)_e$, where $\mathsf{Enc}_L(a)_e$ and $\mathsf{Enc}_R(b)_e$ are the corresponding contributions of a and b to $\mathsf{Enc}(r)_e$.

Next, we define

$$E_L(a)_e := \begin{cases} (1,0) & \text{if } \mathsf{Enc}_L(a)_e = 0, \\ (0,1) & \text{if } \mathsf{Enc}_L(a)_e = 1, \end{cases} \quad \text{and} \quad E_R(b)_e := \begin{cases} (0,1) & \text{if } \mathsf{Enc}_L(b)_e = 0, \\ (1,0) & \text{if } \mathsf{Enc}_L(b)_e = 1. \end{cases}$$

It is easy to verify that $\langle E_L(a)_e, E_R(b)_e \rangle = \mathsf{Enc}_L(a)_e \oplus \mathsf{Enc}_R(b)_e = \mathsf{Enc}(r)_e$.

Constructing $\mathbb{F}_2$-#OV Instance. We can now simplify the quantity we want to compute as

$$\begin{aligned}
&\Pr_{a\in\{0,1\}^{n_\pi/2},b\in\{0,1\}^{\ell-n_\pi/2}}\left[\bigoplus_{i=1}^{q_1}\langle U_a, V_{b\circ j_i}\rangle \oplus \bigoplus_{i=1}^{q_2}\langle E_L(a)_{e_i}, E_R(b)_{e_i}\rangle = 1\right]\\
=&\Pr_{a\in\{0,1\}^{n_\pi/2},b\in\{0,1\}^{\ell-n_\pi/2}}\left[\left\langle \bigcirc_{i=1}^{q_1} U_a \circ \bigcirc_{i=1}^{q_2} E_L(a)_{e_i} \circ 1, \bigcirc_{i=1}^{q_1} V_{b\circ j_i} \circ \bigcirc_{i=1}^{q_2} E_R(b)_{e_i} \circ 1 \right\rangle = 0\right].
\end{aligned}$$

In above, we use U_i and V_j to denote the i-th row of U and j-th column of V respectively, so that $\langle U_i, V_j\rangle = M_{i,j}$. By duplicating each of the 'b's $2^{n_\pi-\ell}$ times, the above can be reduced to a counting $\mathbb{F}_2$-#$\mathsf{OV}_{N,d}$ instance, with $N = 2^{n_\pi/2}$ vectors of $d = O(R(N))$ dimensions. By Theorem 11.4, this can be solved in time

$$\begin{aligned}
N^{2-\Omega(1/\log d)} &= N^{2-\Omega(1/\log R(N))}\\
&= 2^{n_\pi-\Omega(n_\pi/\log R(N))}\\
&\le 2^{\log T(n)+O(\log\log T(n))+O(\log S(n))-\Omega(\log T(n)/\log R(N))}.\\
&\qquad (n_\pi = \ell + O(\log S(n)) = \log T(n) + \log\log T(n) + S(n))
\end{aligned}$$

Since we also need $\mathrm{poly}(S(n))$ time for enumerating all possible $u \in \{0,1\}^m$, the overall running time for calculating $p_{\mathsf{acc}}(M)$ is

$$2^{\log T(n)+O(\log\log T(n))+O(\log S(n))-\Omega(\log T(n)/\log R(N))}.$$

By our third assumption, we know the above running time is $\le 2^{\log T(n)-\omega(\log S(n))} \le T(n)/n$, since $S(n) \ge n$.

Analysis of the Algorithm. Consider first when $1^n \in L$. We know that on the correct guess of $C = C_{\mathsf{best}}$ and the appropriate $M \approx \pi_{C_{\mathsf{best}}}$, we have that M $(1-\delta)$-approximates $\pi_{C_{\mathsf{best}}}$. That is, for a random $r \in \{0,1\}^\ell$, the average relative distance between $M(r,\cdot)$ and $\pi_{C_{\mathsf{best}}}(r,\cdot)$ is at most δ. Hence, by Property (2) of Claim 11.1 and by linearity of expectation, we know that $p_{\mathsf{acc}}(M) > 1 - q_{\mathsf{PCPP}} \cdot \delta$ in this case.

Otherwise, if $1^n \notin L$, then for every guess of C and M, by the soundness property of PCP, we know that

$$\Pr_{r\in\{0,1\}^\ell}[(D_C, r) \in \mathsf{Circuit\text{-}Eval}] \le 1/n.$$

Then by Property (3) of Claim 11.1, we have that

$$p_{\mathsf{acc}}(M) \le 1/n + 1/3 \le 1/2.$$

Therefore, when we set δ to be small enough so that $1 - q_{\mathsf{PCPP}} \cdot \delta > 1/2$, we can distinguish the above two cases. By the above argument, this puts $L \in \mathsf{NTIME}[T(n)/n]$, a contradiction.

Adaptation for the field $\mathbb{F}_q$. Let $q = p^r$ be a prime power. In the following, we sketch the adaptation to deal with $\mathbb{F}_q$. The only thing we need to modify is how to reduce the computation of $p_{\mathsf{acc}}(M)$ to $\mathbb{F}_q$-$\#\mathsf{OV}$. Again, we guess a rank $R(N)$ matrix $M = UV$ over $\mathbb{F}_q$, and we want to calculate

$$\Pr_{r\in\{0,1\}^\ell}[F_u(M(r,j_1)^{q-1},\ldots,M(r,j_{q_1})^{q-1},\mathsf{Enc}(r)_{e_1},\ldots,\mathsf{Enc}(r)_{e_{q_2}})=1]$$

for a Boolean function F_u on $q_{\mathsf{PCPP}} = q_1 + q_2$ inputs. Note that in the above, we raise all the $M(r,j_i)$ inputs to the $(q-1)$-th power to make them Boolean. Now, we can write F_u as a real sum of $2^{q_{\mathsf{PCPP}}}$ AND functions, each one for a subset of the inputs of F_u. Hence, like before, it suffices to consider the case when F_u is an AND function, and in this case we want to calculate

$$\Pr_{r\in\{0,1\}^\ell}\left[\prod_{i=1}^{q_1} M(r,j_i)^{q-1}\cdot\prod_{i=1}^{q_2}\mathsf{Enc}(r)_{e_i}=1\right],$$

which is equivalent to

$$\begin{aligned}&\Pr_{a\in\{0,1\}^{n_\pi/2}}\ \Pr_{b\in\{0,1\}^{\ell-n_\pi/2}}\left[\prod_{i=1}^{q_1}\langle U_a,V_{b\circ j_i}\rangle^{q-1}\cdot\prod_{i=1}^{q_2}\langle E_L(a)_{e_i},E_R(b)_{e_i}\rangle=1\right]\\ =\ &\Pr_{a\in\{0,1\}^{n_\pi/2}}\ \Pr_{b\in\{0,1\}^{\ell-n_\pi/2}}\left[\langle\Phi_a,\Psi_b\rangle=0\right],\end{aligned}$$

where

$$\Phi_a := \left(\bigotimes_{i=1}^{q_1} U_a^{\otimes(q-1)}\otimes\bigotimes_{i=1}^{q_2}E_L(a)_{e_i}\right)\circ 1$$

and

$$\Psi_b := \left(\bigotimes_{i=1}^{q_1} V_{b\circ j_i}\otimes\bigotimes_{i=1}^{q_2}E_R(b)_{e_i}\right)\circ -1$$

The final equality follows from the fact that for vectors a_1, b_1, a_2, b_2, we always have $\langle a_1,b_1\rangle\cdot\langle a_2,b_2\rangle = \langle a_1\otimes a_2, b_1\otimes b_2\rangle$. Finally, the above can be reduced to an $\mathbb{F}_q$-$\#\mathsf{OV}$ instance with $2^{n_\pi/2}$ vectors of $R(N)^{O(1)}$ dimensions. One can see that this polynomial blowup in the dimension is acceptable, and we can still proceed as in the case of $\mathbb{F}_2$. □

Now we are ready to prove Theorem 11.2 (restated below). Notice that here we use the stronger condition $\mathsf{NQP}\not\subset\mathsf{P}_{/\text{poly}}$ instead of $\mathsf{NE}\not\subset\mathsf{P}_{/\text{poly}}$.

Reminder of Theorem 11.2 *There is an absolute constant $\delta > 0$ such that, for all prime powers $q = p^r$ and all $\varepsilon > 0$ at least one of the following holds:*

- *$\mathsf{NQP}\not\subset\mathsf{P}_{/\text{poly}}$.*
- *There is a P^{NP} machine M such that, for infinitely many N's, on input 1^N, M outputs an $N\times N$ matrix $H_N\in\{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1-\varepsilon}})\ge\delta\cdot N^2$ over $\mathbb{F}_q$.*

Proof of Theorem 11.2. Let $\delta > 0$ be a constant to be chosen later.

Assume that $\mathsf{NQP} \subset \mathsf{P}_{/\mathrm{poly}}$. By [MW18], this in particular implies that for a constant b to be specified later and $T(n) := 2^{\log^b n}$, all polynomial-time verifiers for unary languages in $\mathsf{NTIME}[2^{\log^b n}]$ have $S(n) := n^k$-size witness circuits, for a constant $k = k(b)$.

Set $R(N) := 2^{(\log N)^{1-\varepsilon}}$. We will now apply Lemma 11.6 with R, S, T as above. Note that $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n)) = \log^b n + O(\log n)$ and $N = 2^{n_\pi/2} = 2^{\log^b n/2+O(\log n)}$. We thus calculate that

$$\begin{aligned}\log T(n)/\log R(N) &\geq \log^b n/\log^{b(1-\varepsilon)} n\\ &\geq \log^{b\cdot\varepsilon} n\\ &= \omega(\log n)\\ &= \omega(\log\log T(n) + \log S(n)),\end{aligned}$$

if we set $b > 2/\varepsilon$.

Therefore, since Conditions (1) and (3) of Lemma 11.6 hold, we conclude that Condition (2) of Lemma 11.6 does not hold, and this completes the proof. □

11.4 Unconditional Construction of Rigid Matrices

In this Section, we prove Theorem 11.1, giving our main construction of rigid matrices, by using an additional bootstrapping argument.

For an integer $n \in \mathbb{N}$, we write $n_{[k]}$ to denote the smallest integer $m \geq n$ such that $m \equiv 2^k - 1 \pmod{2^{k+1}}$. Notice that $n_{[k]}$ satisfies $|n - n_{[k]}| \leq 2^{k+1}$. Moreover, for all integers $n, m, i, j \in \mathbb{N}$ with $i \neq j$, we have that $n_{[i]} \neq m_{[j]}$.

We first prove the following lemma, which gives an (unconditional) construction of a matrix which is rigid for a higher rank than the construction in Theorem 11.1, but with a slower construction time of $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$.

Lemma 11.7. *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M such that, for infinitely many N's, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Proof. Let δ be a constant to be specified later. For simplicity, we only consider the finite field $\mathbb{F}_2$ in the following. It is not hard to see that our proof also works for all finite fields $\mathbb{F}_{p^r}$ with a straightforward modification.

Assume toward a contradiction that for all $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machines M, and for almost all input lengths N, the output matrix $H_N \in \{0,1\}^{N\times N}$ of M satisfies $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) < \delta \cdot N^2$. (By padding with zeros or only keeping the first N^2 output bits, we can always assume that M outputs exactly N^2 bits on inputs of length N.)

Notation. Throughout the proof, we will often identify a matrix from $\{0,1\}^{N\times N}$ with a string from $\{0,1\}^{N^2}$ (reading the matrix from top row to bottom row, and from leftmost column to rightmost column to construct the corresponding string).

Define the functions $\mathsf{rk}(N) := 2^{(\log N)^{1/2-\varepsilon}}$ and $\ell_{\mathsf{comp}}(N) := 2 \cdot \sqrt{N} \cdot \mathsf{rk}(\sqrt{N})$.

Set $\varepsilon_{\mathsf{enc}} = 0.01$, and $\ell_{\mathsf{enc}}(N) := N^{1+\varepsilon_{\mathsf{enc}}}$. Define the function $\ell_{\mathsf{PCP}}(N) := N \cdot \log^{C_{\mathsf{PCP}}} N$ for a constant C_{PCP} to be specified later.

Applying Lemma 11.4, we fix a locally-decodable error correcting code $\mathsf{ECC}_{\mathsf{local}}$ with a poly(N)-time encoder $\mathsf{Enc} : \{0,1\}^N \to \{0,1\}^{\ell_{\mathsf{enc}}(N)}$, which has a $(\log N)^{C_{\mathsf{enc}}}$-time randomized decoder that decodes any position with probability at least 0.99 when given oracle access to a codeword which is corrupted in less than a 0.01 fraction of its entries.

The Compression Scheme $f_i(\cdot)$. Now, given a string $S \in \{0,1\}^N$, we define the function $\mathsf{comp}(S)$ as follows. Let N' be the smallest square number $\geq N$ and let $A, B \in \{0,1\}^{\sqrt{N'}\times \mathsf{rk}(\sqrt{N'})}$ be the two matrices such that $S \circ 0^{N'-N}$ (interpreted as a $\{0,1\}^{\sqrt{N'}\times\sqrt{N'}}$ matrix) agrees with AB^T (over $\mathbb{F}_2$) on the greatest number of positions. In case of a tie, make the choice resulting in $A \circ B$ being the lexicographically earliest string. We define $\mathsf{comp}(S) := A \circ B$.

Given a string $S \in \{0,1\}^N$, we further define the sequence of functions $f_1(S) := \mathsf{Enc}(S)$ and $f_i(S) := \mathsf{Enc}(\mathsf{comp}(f_{i-1}(S)))$ for $i > 1$.

We now aim to apply Lemma 11.6 from the previous section. Fix a unary language $L \in \mathsf{NTIME}[2^n]$ such that $L \notin \mathsf{NTIME}[2^n/n]$ [Žák83]. Fix an efficient PCP verifier V for L from [BV14], such that $V(1^n)$ takes $\log \ell_{\mathsf{PCP}}(2^n)$ random bits and oracle access to a string of length $\ell_{\mathsf{PCP}}(2^n)$. In order to apply Lemma 11.6, we need to show $V(1^n)$ has small witness circuits.

The Construction of the $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ Machine M_{comp}: A Bootstrapping Argument. Let $O_n \in \{0,1\}^{\ell_{\mathsf{PCP}}(2^n)}$ be the lexicographically first string which makes $V(1^n)$ always accept, if such a string exists, and $0^{\ell_{\mathsf{PCP}}(2^n)}$ otherwise.

Our $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M_{comp} works as follows. For n and $1 \leq i \leq 2/3 \cdot \log n$, let $\ell_{n,i} := \left\lceil \sqrt{|f_i(O_n)|} \right\rceil_{[i]}$. If $\ell_{n,i} \geq 2^{n^{1/2+\varepsilon_1}}$ for a constant ε_1 to be specified later, then M_{comp} on input $1^{\ell_{n,i}}$ computes $f_i(O_n)$, padded with $\ell_{n,i}^2 - |f_i(O_n)|$ zeros. Otherwise it outputs an all-zero matrix.

We first claim that M_{comp} is well-defined, meaning there exists a constant N_0 such that for all $n \geq N_0$ and $1 \leq i \leq 2/3 \log n$, the $\ell_{n,i}$'s are distinct. To prove this, it suffices to show that $\ell_{n,i} < \ell_{m,i}$ whenever $i \leq 2/3 \log n$ and $n < m$, but this follows from the definitions of the function $f_i(\cdot)$'s.

Next, we note that M_{comp} indeed runs in $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$: on input $1^{\ell_{n,i}}$ of length $m = \ell_{n,i} \geq 2^{n^{1/2+\varepsilon_1}}$, the algorithm runs in time poly$(\ell_{n,1}) = 2^{O(n)} \leq m^{\log m}$.

$V(1^n)$ Has Succinct Witness. We first show from our assumption (that $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ does not have rigid matrices) that $V(1^n)$ has a succinct witness circuit if there is an oracle which always satisfies it.

When this is the case, notice that for all $1 \leq i \leq 2/3 \log n$, the output of $M_{\mathsf{comp}}(1^{\ell_{n,i}})$ can be δ-approximated by a matrix of rank $\mathsf{rk}(\ell_{n,i})$. We can calculate that $\ell_{n,2/3\log n} < 2^{n^{1/2}}$; let j be the largest integer such that $\ell_{n,j} \geq 2^{n^{1/2+\varepsilon_1}}$, and note that $\ell_{n,j} \leq 2^{3n^{1/2+\varepsilon_1}}$. Hence, $M_{\mathsf{comp}}(1^{\ell_{n,j}})$ can be implemented as a circuit of size $2^{O(n^{1/2+\varepsilon_1})}$.

Next, if there is a size-S circuit which $(1-\delta)$-approximates $M_{\mathsf{comp}}(1^{\ell_{n,i}})$, then $M_{\mathsf{comp}}(1^{\ell_{n,i-1}})$ can be $(1-\delta)$-approximated by a $\mathsf{rk}(\ell_{n,i-1}) \cdot \mathrm{poly}(n) \cdot S$ size circuit by using the local decoder of the corresponding locally decodable codes. Therefore, $M_{\mathsf{comp}}(1^{\ell_{n,1}})$ can be $(1-\delta)$-approximated by a circuit of size

$$\prod_{i=1}^{j-1} \mathsf{rk}(\ell_{n,i}) \cdot n^{O(\log n)} \cdot 2^{O(n^{1/2+\varepsilon_1})} = 2^{O(n^{1/2+\varepsilon_1})}.$$

Since $M_{\mathsf{comp}}(1^{\ell_{n,1}}) = \mathsf{Enc}(O_n)$, it follows that O_n can be computed *exactly* by a $2^{O(n^{1/2+\varepsilon_1})}$-size circuit.

Applying Lemma 11.6. Toward applying Lemma 11.6, we set $T(n) = 2^n$, $S(n) = 2^{O(n^{1/2+\varepsilon_1})}$ and $R(N) = 2^{(\log N)^{1/2-\varepsilon}}$, where $\varepsilon_1 := \varepsilon/2 > 0$. The two parameters in Condition (3) of Lemma 11.6 are bounded by $n_\pi = \log T(n) + O(\log\log T(n)) + O(\log S(n)) = n + O(n^{1/2+\varepsilon_1})$ and $N = 2^{n/2+O(n^{1/2+\varepsilon_1})}$. We thus calculate that

$$\log T(n)/\log R(N) = \Omega(n/n^{1/2-\varepsilon}) = \omega(n^{1/2+\varepsilon_1}) = \omega(\log\log T(n) + \log S(n)).$$

Therefore, Conditions (1) and (3) of Lemma 11.6 are satisfied, and it follows that Condition (2) must be violated, which completes the proof. □

Finally, we prove Theorem 11.1 (restated below) by using a simple padding argument.

Reminder of Theorem 11.1 *There is an absolute constant $\delta > 0$ such for all prime powers $q = p^r$ and all constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N's, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/4-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_q$.*

Proof of Theorem 11.1. We have shown, from Lemma 11.7, that there is an absolute constant $\delta > 0$ such that for all constants $\varepsilon > 0$:

- There is a $\mathsf{TIME}[n^{\log n}]^{\mathsf{NP}}$ machine M such that, for infinitely many Ns, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $\mathscr{R}_{H_N}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N^2$ over $\mathbb{F}_2$.

Let $N' = N^{\log N}$, and consider the P^{NP} machine M' which, given an input $1^{N'}$, outputs a matrix $H'_{N'} := \mathbf{1}_{N^{\log N - 1}} \otimes H_N$. By Lemma 11.5, we have

$$\mathscr{R}_{H'_{N'}}(2^{(\log N)^{1/2-\varepsilon}}) \geq \delta \cdot N'^2$$

for infinitely many N'. This rigidity bound is equivalent to

$$\mathscr{R}_{H'_{N'}}(2^{(\log N')^{1/4-\varepsilon/2}}) \geq \delta \cdot N'^2,$$

as desired. □

11.5 Applications

Rigid matrices are known to have applications in many areas of complexity theory. In this section, we give three applications of our construction, to communication complexity, arithmetic circuit complexity, and Boolean circuit complexity.

11.5.1 $\mathsf{PH}^{\mathsf{cc}}$ Communication Lower Bound

In this Section we apply our construction of rigid matrices to prove a $\mathsf{PH}^{\mathsf{cc}}$ communication lower bound for functions in $\mathsf{TIME}\left[2^{(\log n)^{\omega(1)}}\right]^{\mathsf{NP}}$. Our main tool will be a known connection between rigid matrices and $\mathsf{PH}^{\mathsf{cc}}$:

Lemma 11.8 ([Raz89], see also [Wun12])**.** *Letting f be a function in $\mathsf{PH}^{\mathsf{cc}}$, the $2^n \times 2^n$ communication matrix M_f of f has $\mathscr{R}_{M_f}(2^{(\log n/\varepsilon)^c}) \leq \epsilon \cdot 4^n$ over $\mathbb{F}_2$, where $\varepsilon > 0$ is arbitrary and $c > 0$ is a constant depending only on f, but not n.*

We will also use the following simple Lemma.

Lemma 11.9. *For any field $\mathbb{F}$ and any matrix $A \in \mathbb{F}^{N\times N}$, and for $M > N$, define $P_{A,M} \in \mathbb{F}^{M\times M}$ to be the matrix such that the top-left $N \times N$ sub-matrix is A, and the rest of entries are all zeros. For all r, we have*

$$\mathscr{R}_{P_{A,M}}(r) \geq \mathscr{R}_A(r).$$

Theorem 11.5. *For all functions $\alpha(n) = \omega(1)$ such that $n^{\alpha(n)}$ is time-constructible, there is a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ which is not in $\mathsf{PH}^{\mathsf{cc}}$.*

Proof. By Theorem 11.1, we know that there is a P^{NP} machine M such that $\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/5}}) \geq \delta \cdot N^2$ over $\mathbb{F}_2$, for a constant $\delta > 0$ and infinitely many N's. For simplicity, we can assume $\alpha(n) \leq \log n$ (e.g., by setting $\alpha'(n) = \min(\alpha(n), \log n)$).

The Definition of f. Now we define a function $f \in \mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$ as follows:

- Given as input $x \in \{0,1\}^n$, the function f outputs zero immediately if 4 does not divide n. Otherwise let $m = n/4$.
- It treats the first $2m$ bits of the input as an integer N in $[2^{2m}]$, and if $N > 2^{(\log m)^{\alpha(n)}}$, it outputs zero.

- Otherwise, it constructs the matrix $H = M(1^N)$. Let $S = 2^m$, and $Q = P_{\mathbf{1}_{\lfloor S/N \rfloor} \otimes H, S}$. It treats the next $2m$ bits of the input as a pair of integers $(i, j) \in [S] \times [S]$, and outputs $Q_{i,j}$.

$Q_{i,j}$ can be computed easily given H, so f can be computed in $\mathsf{TIME}[2^{(\log n)^{\alpha(n)}}]^{\mathsf{NP}}$.

f is not in in $\mathsf{PH}^{\mathsf{cc}}$. We will now show that f, when interpreted as a communication problem, is not in $\mathsf{PH}^{\mathsf{cc}}$. We distribute the input bits of f among the two players as follows: When 4 divides n, setting $m = n/4$, then Alice holds the bits $x_1, x_2, \dots, x_m$ and $x_{2m+1,\dots,3m}$, and Bob holds the bits $x_{m+1}, x_{m+2}, \dots, x_{2m}$ and $x_{3m+1,3m+2,\dots,4m}$.

Assume to the contrary that $f \in \mathsf{PH}^{\mathsf{cc}}$. This means that for all assignments α to $x_1, x_2, \dots, x_{2m}$, the restricted function $f_\alpha : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ is still in $\mathsf{PH}^{\mathsf{cc}}$. That is, there exists a constant c, such that for all $N \le 2^{(\log m)^{\alpha(n)}}$, $S = 2^m$, and $Q = P_{\mathbf{1}_{\lfloor S/N \rfloor} \otimes M(1^N), S}$, we have

$$\mathscr{R}_Q(2^{(\log m)^c}) \le \delta/2 \cdot S^2.$$

By Lemma 11.9, this implies

$$\mathscr{R}_{\mathbf{1}_{\lfloor S/N \rfloor} \otimes M(1^N)}(2^{(\log m)^c}) \le \delta/2 \cdot S^2 \le \delta \cdot 2/3 \cdot (\lfloor S/N \rfloor \cdot N)^2.$$

By Lemma 11.5, this further implies

$$\mathscr{R}_{M(1^N)}(2^{(\log m)^c}) \le 2/3 \cdot \delta \cdot N^2.$$

Now, let N be a sufficiently large integer such that

$$\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/5}}) \ge \delta \cdot N^2.$$

Let m be the smallest integer such that $2^{(\log m)^{\alpha(4m)}} \ge N$. Since $\alpha(n)$ is unbounded, we can pick N to be large enough such that $\alpha(4m-4) \ge 20 \cdot c$. By definition of m, we have $2^{(\log(m-1))^{\alpha(4m-4)}} < N$, meaning $2^{(\log(m-1))^{20c}} < N$, and so $2^{(\log m)^{10c}} < N$. But then by the above discussion, we have

$$\mathscr{R}_{M(1^N)}(2^{(\log N)^{1/10}}) \le 2/3 \cdot \delta \cdot N^2,$$

a contradiction. □

11.5.2 Depth-2 Arithmetic Circuit Lower Bound

In this section we prove Theorem 11.6 (restated below). Recall first the definition of w_2:

Definition 11.3. *For a field $\mathbb{F}$ and a matrix $A \in \mathbb{F}^{N \times N}$, let*

$$w_2(A) := \min\{\mathrm{nnz}(B) + \mathrm{nnz}(C) \mid A = BC\},$$

where the min is over all pairs B, C of matrices of any dimensions over $\mathbb{F}$ whose product is A, and $\mathrm{nnz}(X)$ denotes the number of nonzero entries in the matrix X.

Theorem 11.6. *For all prime powers $q = p^r$ and constants $\varepsilon > 0$:*

- *There is a P^{NP} machine M such that, for infinitely many N, on input 1^N, M outputs an $N \times N$ matrix $H_N \in \{0,1\}^{N\times N}$ such that $w_2(H_N) \geq \Omega(N \cdot 2^{(\log N)^{1/4-\varepsilon}})$ over $\mathbb{F}_q$.*

We first prove the following folklore lemma.

Lemma 11.10. *For any field $\mathbb{F}$, and any matrix $A \in \mathbb{F}^{N\times N}$, let $r = w_2(A)/N$. Then, for any constant $\delta > 0$, we have*

$$\mathscr{R}_A(\rho_\delta \cdot r^2) \leq \delta \cdot N^2,$$

for some constant ρ_δ depending only on δ.

In other words, if $\mathscr{R}_A(\rho_\delta \cdot r^2) > \delta \cdot N^2$, then we have $w_2(A) \geq r \cdot N$.

Proof. For some integer M, let B and C be matrices over $\mathbb{F}$ of dimensions $N \times M$ and $M \times N$, respectively, such that $A = BC$ and $\mathrm{nnz}(B) + \mathrm{nnz}(C) = r \cdot N$. For $i, j \in [N]$, let $b_i \in \mathbb{F}^M$ be the i-th row of B, and $c_j \in F^M$ be the j-th column of C. Hence, $A_{i,j} = \langle b_i, c_j \rangle$.

Now, let ρ_δ be a function of δ to be specified later, and set $m = \rho_\delta \cdot r^2$. Pick a hash function $P : [M] \to [m]$ uniformly at random. Next, for each b_i, we define a vector $\widetilde{b}_i$ by setting, for each $j \in [m]$:

$$(\widetilde{b}_i)_j := \sum_{k \in P^{-1}(j)} (b_i)_k.$$

We similarly define $\widetilde{c}_j$. Now, let $\widetilde{B}$ be the $N \times m$ matrix with the $\widetilde{b}_i$'s as rows, and $\widetilde{C}$ be the $m \times N$ matrix with the $\widetilde{c}_j$'s as columns. We will now argue that $\widetilde{B}\widetilde{C}$ approximates A well.

First, from definition, we have

$$\mathbb{E}_{(i,j)\in[N]\times[N]}\, \mathrm{nnz}(b_i) + \mathrm{nnz}(c_j) = \frac{\mathrm{nnz}(A) + \mathrm{nnz}(B)}{N} = r.$$

Hence, by Markov's inequality, for at least a $1 - \delta/2$ fraction of the pairs $(i, j) \in [N] \times [N]$, we have $\mathrm{nnz}(b_i) + \mathrm{nnz}(c_j) \leq r \cdot \frac{2}{\delta}$.

Fix such a pair of (i, j), and let $I = \{k \in [M] : (b_i)_k \neq 0 \vee (c_j)_k \neq 0\}$, which has size $|I| \leq r \cdot \frac{2}{\delta}$. Note that if all the elements of I have distinct images under the mapping P, then $\langle \widetilde{b}_i, \widetilde{c}_j \rangle = \langle b_i, c_j \rangle = M_{i,j}$. By a union bound, this happens with probability at least $1 - |I|^2/m$ over the random choice of P.

Setting $\rho_\delta = (\frac{2}{\delta})^3$, we have $1 - |I|^2/m \geq 1 - \delta/2$. Thus, by the probabilistic method, there is a fixed P for which $\widetilde{B}\widetilde{C}$ agrees with A on a $1 - \delta$ fraction of inputs, and hence $\mathscr{R}_A(\rho_\delta \cdot r^2) \leq \delta \cdot N^2$. □

Theorem 11.6 then follows by combining Lemma 11.10, Theorem 11.1, and Theorem 11.2.

11.5.3 Threshold Circuit Lower Bound for E^{NP}

We conclude this Section with a new threshold circuit lower bound for E^{NP}.

Theorem 11.7. *For every $\delta > 0$ and prime p, there is an $a > 0$ such that the class E^{NP} does not have non-uniform $\mathsf{AC}^0[p] \circ \mathsf{LTF} \circ \mathsf{AC}^0[p] \circ \mathsf{LTF}$ circuits of depth $o(\log n / \log\log n)$ where the bottom LTF layer has $2^{O(n^a)}$ gates, the rest of the circuit has polynomial size, and the middle layer LTF gates have fan-in $O(n^{1/2-\delta})$.*

Theorem 11.7 follows from the connection between rigid matrices and threshold circuits which we proved in Theorem 10.7 in the previous Chapter.

11.6 Algorithm for Counting Orthogonal Vectors over Finite Fields

Finally, in this Section, we give a sketch of the algorithm for $\mathbb{F}_{p^r}$-$\#\mathsf{OV}$ which we stated in Theorem 11.4 and which is needed by our construction above. The algorithm is a minor modification of the deterministic algorithm for $\#\mathsf{OV}$ by Chan and Williams [CW16]. It makes use of the polynomial method in algorithm design, the same algorithmic technique we used earlier in Chapter 8.

11.6.1 Reduction to Prime Fields

We begin by sketching a reduction from $\mathbb{F}_{p^r}$-$\#\mathsf{OV}$ to $\mathbb{F}_p$-$\#\mathsf{OV}$. More precisely, for a prime power $q = p^r$, we give a reduction from one instance of $\mathbb{F}_q$-$\#\mathsf{OV}_{n,d}$ to a constant number of different instances of $\mathbb{F}_p$-$\#\mathsf{OV}_{n,d\cdot O_r(1)}$. The reduction builds on ideas from [LPT+17] and [Wil18].

We first define an intermediate problem $\mathbb{F}_p$-$\#\mathsf{AND}$-$\mathsf{OV}_{n,d,r}$: given as input two size-n collections $A, B \subseteq (\mathbb{F}_q^d)^r$, with $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ (so, for instance, each a_i is an r-tuple of vectors from $\mathbb{F}_q^d$), the goal is to compute the number of pairs $(i, i') \in [n]^2$ such that $\langle a_{i,j}, b_{i',j}\rangle = 0$ for all $j \in [r]$.

$\mathbb{F}_q$-$\#\mathsf{OV} \Rightarrow \mathbb{F}_p$-$\#\mathsf{AND}$-$\mathsf{OV}$. We first show how to reduce an $\mathbb{F}_q$-$\#\mathsf{OV}_{n,d}$ instance to an $\mathbb{F}_p$-$\#\mathsf{AND}$-$\mathsf{OV}_{n,dr^2,r}$ instance in nearly linear time. Pick a degree-r $\mathbb{F}_p$ irreducible polynomial P; we know that $\mathbb{F}_q$ isomorphic to $F_p[X]/(P)$. In the calculations below, we perform the arithmetic mod P.

Suppose we have two vectors $u, v \in \mathbb{F}_q^d$. Let $u_i = \sum_{j=0}^{r-1} \alpha_{i,j} \cdot X^j$, and $v_i =$

$\sum_{j=0}^{r-1} \beta_{i,j} \cdot X^j$ for coefficients $\alpha_{i,j}, \beta_{i,j} \in \mathbb{F}_p$. We have that

$$\begin{aligned}\sum_{i=1}^{d} u_i \cdot v_i &= \sum_{i=1}^{d} \left(\sum_{j=0}^{r-1} \alpha_{i,j} \cdot X^j\right) \cdot \left(\sum_{j=0}^{r-1} \beta_{i,j} \cdot X^j\right) \\ &= \sum_{i=1}^{d} \sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \alpha_{i,j} \cdot \beta_{i,k} X^{j+k}.\end{aligned}$$

Define the coefficients $\gamma_{j+k,\ell} \in \mathbb{F}_p$ so that $X^{j+k} = \sum_{\ell=0}^{r-1} \gamma_{j+k,\ell} \cdot X^\ell \pmod{P}$. The above simplifies to

$$\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} X^{j+k} \cdot \sum_{i=1}^{d} \alpha_{i,j} \cdot \beta_{i,k} = \sum_{\ell=0}^{r-1} X^\ell \cdot \left(\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \sum_{i=1}^{d} \gamma_{j+k,\ell} \cdot \alpha_{i,j} \cdot \beta_{i,k}\right) \pmod{P}.$$

We therefore see that $\langle u_i, v_i \rangle = 0$ if and only if

$$\sum_{j=0}^{r-1} \sum_{k=0}^{r-1} \sum_{i=1}^{d} \gamma_{j+k,\ell} \cdot \alpha_{i,j} \cdot \beta_{i,k} = 0 \tag{11.1}$$

for all $0 \le \ell \le r-1$. For each ℓ, we can build vectors $u_i^{(\ell)}$ and $v_i^{(\ell)}$ in $\mathbb{F}_p^{r^2 \cdot d}$ so that $\langle u_i^{(\ell)}, v_i^{(\ell)} \rangle$ equals the left hand side of (11.1). This transformation reduces an $\mathbb{F}_q$-$\#\mathsf{OV}_{n,d}$ instance to an $\mathbb{F}_p$-$\#\mathsf{AND\text{-}OV}_{n,dr^2,r}$ instance as desired.

$\mathbb{F}_p$-#AND-OV ⇒ $\mathbb{F}_p$-#OV. Now, given an $\mathbb{F}_p$-$\#\mathsf{AND\text{-}OV}_{n,d,r}$ instance with input collections A, B, we show how to reduce it to p^r different $\mathbb{F}_p$-$\#\mathsf{OV}_{n,dr+1}$ instances, again in nearly linear time.

Let $a, b \in (\mathbb{F}_p^d)^r$. For a random vector $u \in \mathbb{F}_p^r$, observe that:

- If $\langle a_i, b_i \rangle = 0$ for all $i \in [r]$, then $\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle$ is always zero.
- Otherwise, $\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle = 1$ with probability $1/p$.

For our reduction, we iterate over all vectors $u \in \mathbb{F}_p^r$, and sum the number of pairs $(a, b) \in A \times B$ such that

$$\sum_{i=1}^{r} u_i \cdot \langle a_i, b_i \rangle = \left\langle \bigcirc_{i=1}^{r} u_i a_i, \bigcirc_{i=1}^{r} b_i \right\rangle = 1.$$

For each u, this can be written as an $\mathbb{F}_p$-$\#\mathsf{OV}_{n,dr+1}$ instance (via $\langle a, b \rangle = 1 \Leftrightarrow \langle a \circ 1, b \circ -1 \rangle = 0$).

For a pair $(a, b) \in A \times B$, if $\langle a_i, b_i \rangle = 0$ for all $i \in [r]$, then (a, b) is never counted in the above sum. Otherwise, it is counted p^{r-1} times. Therefore, by summing up the results of all these $\mathbb{F}_p$-$\#\mathsf{OV}$ instances after the reduction, dividing the result by p^{r-1}, and then finally subtracting the resulting number from $|A| \cdot |B|$, we can compute the answer to the given $\mathbb{F}_{n,d,r}$-#AND-OV instance.

11.6.2 Algorithm for Prime Fields

In this subsection, we give a self-contained exposition of the $\mathbb{F}_p$-#OV algorithm which is implicit in [CW16]. We will make use of the polynomial method in algorithm design, and in particular, we will use Lemma 8.2 from Chapter 8 for quickly evaluating a sparse polynomials on many inputs by using fast matrix multiplication. In [CW16], the deterministic #OV algorithm works by combining two key technical tools: *small-biased sets*, and *modulus-amplifying polynomials*. We won't need small-biased sets here as we only aim to solve $\mathbb{F}_{p^r}$-#OV. We first recall the definition of modulus-amplifying polynomials.

Lemma 11.11 (Modulus-Amplifying Polynomial [Yao90, BT94])**.** *For all integers $\ell \geq 1$, there is a polynomial F_ℓ over $\mathbb{Z}$ of degree $(2\ell - 1)$ with $O(\ell)$-bit coefficients such that for all integers $m \geq 1$ and all $a \in \mathbb{Z}$:*

(1) if $a \equiv 0 \pmod{m}$, then $F_\ell(a) \equiv 0 \pmod{m^\ell}$, and

(2) if $a \equiv 1 \pmod{m}$, $F_\ell(a) \equiv 1 \pmod{m^\ell}$.

Now we are ready to prove Theorem 11.4 when the modulus q is a prime. The case when q is a prime power then follows using the reduction from Section 11.6.1.

Theorem 11.8. *For all primes p, there is an $n^{2-\Omega(1/\log(d/\log n))}$ time deterministic algorithm for $\mathbb{F}_p$-#OV$_{n,d}$, when $d = n^{o(1)}$.*

Proof. Let ℓ be a parameter to be specified later. Let X, Y be two collections of $p^{\ell/4}$ vectors from $\mathbb{F}_p^d$. We define the polynomial

$$P(X,Y) := \sum_{(x,y)\in X\times Y} (1 - F_\ell(\langle x,y\rangle^{p-1})),$$

where F_ℓ is the modulus-amplifying polynomial from Lemma 11.11. Hence,

$$1 - F_\ell(\langle x,y\rangle^{p-1}) \equiv \begin{cases} 1 \pmod{p^\ell} & \text{when } \langle x,y\rangle \equiv 0 \pmod{p}, \\ 0 \pmod{p^\ell} & \text{when } \langle x,y\rangle \not\equiv 0 \pmod{p}. \end{cases}$$

Let us count the number M of monomials in $F_\ell(\langle x,y\rangle^{p-1}) = F_\ell((x_1y_1 + x_2y_2 + \cdots + x_dy_d)^{p-1})$ when it is expanded and simplified. F_ℓ is a polynomial of degree $(2\ell - 1)\cdot(p-1)$ in $x, y \in \mathbb{F}_p^d$. In particular, since we are working over $\mathbb{F}_p$, we may simplify F_ℓ so that each of the $2d$ input variables has individual degree at most $p-1$ in any given monomial. Thus, using the simple bound that no monomial depends on more variables than the degree of the polynomial, combined with the fact that the power of x_i in a given monomial is always equal to the power of y_i in that monomial, we get the bound

$$M \le (p-1)^{2\ell \cdot p} \cdot \sum_{i=0}^{2\ell \cdot p} \binom{d}{i} \le (p-1)^{2\ell \cdot p} \cdot O\left(\frac{d}{\ell \cdot p}\right)^{2\ell \cdot p} \le O\left(\frac{d}{\ell}\right)^{2\ell \cdot p}.$$

Next, we will construct two mappings $\Phi_X, \Phi_Y : (\mathbb{F}_p^d)^{p^{\ell/4}} \to \mathbb{Z}^M$ such that for any $X, Y \in (\mathbb{F}_p^d)^{p^{\ell/4}}$,

$$P(X,Y) = \langle \Phi_X(X), \Phi_Y(Y) \rangle.$$

We construct Φ_X, Φ_Y as follows. For a set $S \subseteq [d]$, let x_S (resp. y_S) denote $\prod_{i \in S} x_i$ ($\prod_{i \in S} y_i$). Let $S_1, S_2, \ldots, S_M$ be an enumeration of all subsets of $[d]$ of size no greater than $(2\ell - 1) \cdot (p-1)$. There are corresponding coefficients $c_1, c_2, \ldots, c_M \in \mathbb{Z}$ such that

$$1 - F_\ell(\langle x, y \rangle^{p-1}) = \sum_{i=1}^{M} c_i \cdot x_{S_i} \cdot y_{S_i}.$$

We can then define

$$\Phi_X(X) := \left(\sum_{x \in X} c_1 \cdot x_{S_1}, \sum_{x \in X} c_2 \cdot x_{S_2}, \ldots, \sum_{x \in X} c_M \cdot x_{S_M} \right),$$

$$\Phi_Y(Y) := \left(\sum_{y \in Y} y_{S_1}, \sum_{y \in Y} y_{S_2}, \ldots, \sum_{y \in Y} y_{S_M} \right),$$

and it follows that

$$\langle \Phi_X(X), \Phi_Y(Y) \rangle = \sum_{i=1}^{M} \sum_{(x,y) \in X \times Y} c_i \cdot x_{S_i} \cdot y_{S_i} = P(X,Y).$$

Picking $c = d/\log n$ and $\ell = \varepsilon/p \cdot \log n/\log c$ for a small enough constant ε, we have

$$M \le O\left(\frac{c \log n}{\ell}\right)^{2\ell \cdot p} = O\left(\frac{p \cdot c \log c}{\varepsilon}\right)^{2\varepsilon \log n/\log c} \le n^{0.01}.$$

Let $b = p^{\ell/4}$ (and set ε small enough so that $b \le n^{0.01}$ as well). We partition A (B) into n/b blocks $A_1, A_2, \ldots, A_{n/b}$ ($B_1, B_2, \ldots, B_{n/b}$), each of size b. We then apply the algorithm from Lemma 8.2 to evaluate $P(A_i, B_j)$ for each $(i,j) \in [n/b] \times [n/b]$ in $(n/b)^2 \cdot \mathrm{polylog}(n) = n^{2-1/O(\log c)}$ time by multiplying two matrices of dimensions $n/b \times n^{0.01}$ and $n^{0.01} \times n/b$ over $\mathbb{Z}$ whose entries are polylog(n)-bit integers. Since

$$P(A_i, B_j) \equiv \sum_{(x,y) \in A_i \times B_j} [\langle x, y \rangle \equiv 0 \pmod{p}] \pmod{p^\ell},$$

this allows us to solve $\mathbb{F}_p$-#OV in $n^{2-1/O(\log c)}$ time. □

Bibliography

[ABFR94] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994.

[ABG$^+$14] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC0 ∘ MOD2. In *ITCS*, pages 251–260, 2014.

[AC09] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[AC19] Josh Alman and Lijie Chen. Efficient construction of rigid matrices using an np oracle. In *FOCS, to appear*, 2019.

[ACR$^+$10] Andris Ambainis, Andrew M Childs, Ben W Reichardt, Robert Špalek, and Shengyu Zhang. Any and-or formula of size n can be evaluated in time $n^{1/2+o(1)}$ on a quantum computer. *SIAM J. Computing*, 39(6):2513–2530, 2010.

[ACW16] Josh Alman, Timothy Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *FOCS*, pages 467–476, 2016.

[AFLG15] Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: limitations of the Coppersmith-Winograd method. In *STOC*, pages 585–593, 2015.

[AG94] Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM J. Computing*, 23(5):1026–1049, 1994.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *SODA*, pages 1018–1028, 2014.

[AIP06] Alexandr Andoni, Piotr Indyk, and Mihai Patrascu. On the optimality of the dimensionality reduction method. In *FOCS*, pages 449–458, 2006.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of mathematics*, pages 781–793, 2004.

[AKW90] Noga Alon, Mauricio Karchmer, and Avi Wigderson. Linear circuits over gf(2). *SIAM Journal on Computing*, 19(6):1064–1067, 1990.

[Alm19a] Josh Alman. An illuminating algorithm for the light bulb problem. In *SOSA*, pages 2:1–2:11, 2019.

[Alm19b] Josh Alman. Limits on the universal method for matrix multiplication. In *CCC*, pages 12:1–12:24, 2019.

[Alo90] Noga Alon. On the rigidity of an Hadamard matrix. Manuscript. See [Juk01, Section 15.1.2], 1990.

[ALSV13] Noga Alon, Troy Lee, Adi Shraibman, and Santosh Vempala. The approximate rank of a matrix and its algorithmic applications. In *STOC*, pages 675–684, 2013.

[AM17] Josh Alman and Dylan McKay. Theoretical foundations of team matchmaking. In *AAMAS*, pages 1073–1081, 2017.

[AMY16] Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank versus VC dimension. In *COLT*, pages 47–80, 2016.

[And05] Alexandr Andoni. Approximate nearest neighbor problem in high dimensions. Master's thesis, MIT, 2005.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *STOC*, pages 793–801, 2015.

[ASU13] Noga Alon, Amir Shpilka, and Christopher Umans. On sunflowers and matrix multiplication. *Computational Complexity*, 22(2):219–243, 2013.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1):2:1–2:54, 2009.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and Hamming nearest neighbors. In *FOCS*, pages 136–150, 2015.

[AW17] Josh Alman and Ryan Williams. Probabilistic rank and matrix rigidity. In *STOC*, pages 641–652, 2017.

[AW18a] Josh Alman and Virginia Vassilevska Williams. Further limitations of the known approaches for matrix multiplication. In *ITCS*, pages 25:1–25:15, 2018.

[AW18b] Josh Alman and Virginia Vassilevska Williams. Limits on all known (and some unknown) approaches to matrix multiplication. In *FOCS*, pages 580–591, 2018.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *SODA*, pages 218–230, 2015.

[BCC$^+$17a] 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 Analysis*, 2017(3):1–27, 2017.

[BCC$^+$17b] Jonah Blasiak, Thomas Church, Henry Cohn, Joshua A Grochow, and Chris Umans. Which groups are amenable to proving exponent two for matrix multiplication? *arXiv preprint arXiv:1712.02302*, 2017.

[BCS13] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*. Springer Science & Business Media, 2013.

[BdW01] Harry Buhrman and Ronald de Wolf. Communication complexity lower bounds by polynomials. In *CCC*, pages 120–130, 2001.

[Bei95] Richard Beigel. The polynomial method in circuit complexity. In *Structure in Complexity Theory Conference*, pages 82–95, 1995.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *FOCS*, pages 337–347, 1986.

[BGH$^+$06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BGL06] Nayantara Bhatnagar, Parikshit Gopalan, and Richard J. Lipton. Symmetric polynomials over z_m and simultaneous communication protocols. *J. Comput. Syst. Sci.*, 72(2):252–285, 2006.

[BH74] James R Bunch and John E Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

[BI13] Peter Bürgisser and Christian Ikenmeyer. Explicit lower bounds via geometric complexity theory. In *STOC*, pages 141–150, 2013.

[Bin80] Dario Bini. Border rank of a $p \times q \times 2$ tensor and the optimal approximation of a pair of bilinear forms. In *ICALP*, pages 98–108, 1980.

[Blä13] Markus Bläser. Fast matrix multiplication. *Theory of Computing, Graduate Surveys*, 5:1–60, 2013.

[BR02] Omer Barkol and Yuval Rabani. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *JCSS*, 64(4):873–896, 2002.

[Bro97] Andrei Z Broder. On the resemblance and containment of documents. In *SEQUENCES*, pages 21–29, 1997.

[BRS91] Richard Beigel, Nick Reingold, and Daniel Spielman. The perceptron strikes back. In *Structure in Complexity Theory Conference*, pages 286–291, 1991.

[BS92] Jehoshua Bruck and Roman Smolensky. Polynomial threshold functions, AC^0 functions, and spectral norms. *SIAM J. Comput.*, 21(1):33–42, 1992.

[BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[BV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *ICALP*, pages 163–173, 2014.

[CCGL99] Amit Chakrabarti, Bernard Chazelle, Benjamin Gum, and Alexey Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the hamming cube. In *STOC*, pages 305–311, 1999.

[CFL85] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and associative functions. *JCSS*, 30(2):222–234, 1985.

[CGJ$^+$16] Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $\mathrm{AC}^0\circ\mathrm{MOD}_2$ lower bounds for the boolean inner product. In *ICALP*, pages 35:1–35:14, 2016.

[Cha02] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[Cha18] Timothy M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. *Journal of Computational Geometry*, 9(2):3–20, 2018.

[Che99] Pafnuty L. Chebyshev. Sur l'interpolation. In A. Markoff and N. Sonin, editors, *Oeuvres de P. L. Tchebychef*, volume 1, pages 539–560. Commissionaires de L'Académie Impériale des Sciences, 1899.

[CIP06] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A duality between clause width and clause density for SAT. In *CCC*, pages 252–260, 2006.

[CKL13] Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31:1–31:25, 2013.

[CKSU05] Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Christopher Umans. Group-theoretic algorithms for matrix multiplication. In *FOCS*, pages 379–388, 2005.

[CLP17] Ernie Croot, Vsevolod F Lev, and Péter Pál Pach. Progression-free sets in $\mathbb{Z}_4^n$ are exponentially small. *Annals of Mathematics*, 185(1):331–337, 2017.

[CLS18] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*, pages 938–942, 2018.

[Cod00] Bruno Codenotti. Matrix rigidity. *Linear Algebra and its Applications*, 304(1-3):181–192, 2000.

[Cop82] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982.

[CP19] Shiteng Chen and Periklis A Papakonstantinou. Depth reduction for composites. *SIAM Journal on Computing*, 48(2):668–686, 2019.

[CR04] Amit Chakrabarti and Oded Regev. An optimal randomised cell probe lower bound for approximate nearest neighbour searching. In *FOCS*, pages 473–482, 2004.

[CS15] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In *SAT*, pages 33–45, 2015.

[CSS16] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. In *CCC*, pages 1:1–1:35, 2016.

[CSV84] Ashok K. Chandra, Larry Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM J. Computing*, 13(2):423–439, 1984.

[CU03] Henry Cohn and Christopher Umans. A group-theoretic approach to fast matrix multiplication. In *FOCS*, pages 438–449, 2003.

[CU13] Henry Cohn and Christopher Umans. Fast matrix multiplication using coherent configurations. In *SODA*, pages 1074–1086, 2013.

[CVZ18] Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Universal points in the asymptotic spectrum of tensors. In *STOC*, pages 289–296, 2018.

[CVZ19] Matthias Christandl, Péter Vrana, and Jeroen Zuiddam. Barriers for fast matrix multiplication from irreversibility. In *CCC*, pages 26:1–26:17, 2019.

[CW82] Don Coppersmith and Shmuel Winograd. On the asymptotic complexity of matrix multiplication. *SIAM J. Comput.*, 11(3):472–492, 1982.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

[CW16] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov–Smolensky. In *SODA*, pages 1246–1255, 2016.

[CW19a] Lijie Chen and Ruosong Wang. Classical algorithms from quantum and arthur-merlin communication protocols. *10th Innovations in Theoretical Computer Science*, pages 23:1–23:20, 2019.

[CW19b] Lijie Chen and Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. In *CCC*, pages 19:1–19:43, 2019.

[DE17] Zeev Dvir and Benjamin Edelman. Matrix rigidity and the croot-levpach lemma. *arXiv preprint arXiv:1708.01646*, 2017.

[Des07] Amit Jayant Deshpande. *Sampling-based algorithms for dimension reduction*. PhD thesis, Massachusetts Institute of Technology, 2007.

[DGW19] Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static data structure lower bounds imply rigidity. In *STOC*, pages 967–978, 2019.

[DL19] Zeev Dvir and Allen Liu. Fourier and Circulant Matrices Are Not Rigid. In *CCC*, pages 17:1–17:23, 2019.

[DS13] A.M. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh, Section: A Mathematics*, 143:351–369, 4 2013.

[Dub10] Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010.

[Dvi16] Zeev Dvir. On the non-rigidity of generating matrices of good codes. Writeup by Oded Goldreich. Available at http://www.wisdom.weizmann.ac.il/~oded/MC/209.html, October 30 2016.

[DW06a] Evgeny Dantsin and Alexander Wolpert. MAX-SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time. In *SAT*, pages 266–276, 2006.

[dW06b] Ronald de Wolf. Lower bounds on matrix rigidity via a quantum argument. In *ICALP*, volume 4051, pages 62–71, 2006.

[EG17] Jordan S Ellenberg and Dion Gijswijt. On large subsets of $\mathbb{F}_q^n$ with no three-term arithmetic progression. *Annals of Mathematics*, 185(1):339–343, 2017.

[EGOW18] Klim Efremenko, Ankit Garg, Rafael Oliveira, and Avi Wigderson. Barriers for rank methods in arithmetic complexity. In *ITCS*, pages 1:1–1:19, 2018.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[FKL$^+$01] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *FSTTCS*, pages 171–182, 2001.

[For02] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. System. Sci.*, 65(4):612–625, 2002.

[Fri93] Joel Friedman. A note on matrix rigidity. *Combinatorica*, 13(2):235–239, 1993.

[GHK$^+$12] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Kouckỳ, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. In *STOC*, pages 479–494, 2012.

[GL01] Ben Gum and Richard J Lipton. Cheaper by the dozen: Batched algorithms. In *SDM*, pages 1–11, 2001.

[GPW16] Mika Göös, Toniann Pitassi, and Thomas Watson. Zero-information protocols and unambiguity in arthur-merlin communication. *Algorithmica*, 76(3):684–719, 2016.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018.

[GQ19] Joshua A Grochow and Youming Qiao. Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. *arXiv preprint arXiv:1907.00309*, 2019.

[Gri] D. Yu. Grigor'ev. Unpublished work. Cited in [KR98].

[GST03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

[GT16] Oded Goldreich and Avishay Tal. Matrix rigidity of random toeplitz matrices. In *STOC*, pages 91–104, 2016.

[HIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[Hir03] Mika Hirvensalo. *Studies on Boolean Functions Related to Quantum Computing.* PhD thesis, University of Turku, 2003.

[HMP$^+$93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.*, 46(2):129–154, 1993.

[HNO08] Nicholas J. A. Harvey, Jelani Nelson, and Krzysztof Onak. Sketching and streaming entropy via approximation theory. In *FOCS*, pages 489–498, 2008.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HY09] Pavel Hrubeš and Amir Yehudayoff. Arithmetic complexity in algebraic extensions, 2009.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[Ind04] Piotr Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 877–892. Chapman and Hall, 2nd edition, 2004.

[IPS13] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *FOCS*, pages 479–488, 2013.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[JMV15] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *ICALP*, pages 749–760, 2015.

[Juk01] Stasys Jukna. *Extremal Combinatorics, With Applications in Computer Science*. EATCS Series. Springer, 2001.

[KK19] Matti Karppa and Petteri Kaski. Probabilistic tensors and opportunistic boolean matrix multiplication. In *SODA*, pages 496–515, 2019.

[KKK16] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *SODA*, pages 1288–1305, 2016.

[KKKÓC16] Matti Karppa, Petteri Kaski, Jukka Kohonen, and Padraig Ó Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. In *ESA*, pages 52:1–52:17, 2016.

[Kle97] Jon M Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC*, pages 599–608, 1997.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

[KOS04] Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *J. Comput. Syst. Sci.*, 68(4):808–840, 2004.

[KR98] B. S. Kashin and A. A. Razborov. Improved lower bounds on the rigidity of Hadamard matrices. *Matematicheskie Zametki*, 63(4):535–540, 1998. (in Russian).

[KS01] Adam R. Klivans and Rocco Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. In *STOC*, pages 258–265, 2001.

[KS10] Adam R Klivans and Alexander A Sherstov. Lower bounds for agnostic learning via approximate rank. *Computational Complexity*, 19(4):581–604, 2010.

[KSS18] Robert Kleinberg, Will Sawin, and David Speyer. The growth rate of tri-colored sum-free sets. *Discrete Analysis*, 12, 2018.

[KV19] Mrinal Kumar and Ben Lee Volk. Lower bounds for matrix factorization. *arXiv preprint arXiv:1904.01182*, 2019.

[KvM02] 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.

[KW16] Daniel M. Kane and Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *STOC*, pages 633–643, 2016.

[Lan17] Joseph M Landsberg. *Geometry and complexity theory*, volume 169. Cambridge University Press, 2017.

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC*, pages 296–303, 2014.

[LGU17] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *SODA*, pages 1029–1046, 2017.

[LN90] Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

[Lok00] Satyanarayana V. Lokam. On the rigidity of vandermonde matrices. *Theoretical Computer Science*, 237(1-2):477–483, 2000.

[Lok01] Satyanarayana V. Lokam. Spectral methods for matrix rigidity with applications to size-depth tradeoffs and communication complexity. *Journal of Computer and System Sciences*, 63(3):449–473, 2001.

[Lok06] Satyanarayana V. Lokam. Quadratic lower bounds on matrix rigidity. In *TAMC*, volume 3959, pages 295–307. Springer, 2006.

[Lok14] Satyanarayana V. Lokam. Exercises on matrix rigidity. Simons Institute for Theory of Computing. Available at https://simons.berkeley.edu/sites/default/files/docs/1738/exercises.pdf, 2014.

[Lov11] Shachar Lovett. Computing polynomials with few multiplications. *Theory of Computing*, 7(1):185–188, 2011.

[LPT$^+$17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *SODA*, pages 2190–2202, 2017.

[LS09] Nathan Linial and Adi Shraibman. Learning complexity vs communication complexity. *Combinatorics, Probability & Computing*, 18(1-2):227–245, 2009.

[LTV03] JM Landsberg, J. Taylor, and N.K. Vishnoi. The geometry of matrix rigidity. Available at https://smartech.gatech.edu/handle/1853/6514, 2003.

[Lup56] Oleg B Lupanov. On rectifier and switching-and-rectifier schemes. *Dokl. Akad. Nauk SSSR*, 111(6):1171–1174, 1956.

[Mat91] Jirí Matoušek. Computing dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278, 1991.

[Mat08] Jirí Matoušek. On variants of the Johnson–Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2):142–156, 2008.

[Mid05] Gatis Midrijanis. Three lines proof of the lower bound for the matrix rigidity. *arXiv preprint arXiv:cs/0506081*, 2005.

[MKZ09] Kerui Min, Ming-Yang Kao, and Hong Zhu. The closest pair problem under the hamming metric. In *Computing and Combinatorics*, pages 205–214. Springer, 2009.

[MP69] Marvin L Minsky and Seymour A Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT press Boston, MA:, 1969.

[MPS16] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *CSR*, pages 294–308, 2016.

[MS82] Kurt Mehlhorn and Erik M Schmidt. Las vegas is better than determinism in vlsi and distributed computing. In *STOC*, pages 330–337, 1982.

[MT98] Alexis Maciel and Denis Thérien. Threshold circuits of small majority-depth. *Information and Computation*, 146(1):55–83, 1998.

[MT99] Alexis Maciel and Denis Thérien. Efficient threshold circuits for power series. *Information and Computation*, 152(1):62–73, 1999.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[MW18] Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *STOC*, pages 890–901, 2018.

[Nis] Noam Nisan. Unpublished work. Cited in [KR98].

[NS94] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994.

[NS17] Eric Naslund and Will Sawin. Upper bounds for sunflower-free sets. In *Forum of Mathematics, Sigma*, volume 5, 2017.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[OS10] Ryan O'Donnell and Rocco A. Servedio. New degree bounds for polynomial threshold functions. *Combinatorica*, 30(3):327–358, 2010.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *STOC*, pages 665–677, 2017.

[Pan78] V. Y. Pan. Strassen's algorithm is not optimal. In *FOCS*, pages 166–176, 1978.

[Pan06] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA*, pages 1186–1195, 2006.

[Par19] Orr Paradise. Smooth and strong PCPs. *ECCC preprint TR19-023*, 2019.

[Pat92] Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions. In *STOC*, pages 468–474, 1992.

[Pat08] Mihai Patrascu. *Lower bound techniques for data structures*. PhD thesis, Massachusetts Institute of Technology, 2008.

[PRR95] Ramamohan Paturi, Sanguthevar Rajasekaran, and John Reif. The light bulb problem. *Information and Computation*, 117(2):187–192, 1995.

[PS88] P. Pudlak and P. Savicky. Private communication. Cited in [Raz89], 1988.

[Pud94] Pavel Pudlak. Large communication in constant depth circuits. *Combinatorica*, 14(2):203–216, 1994.

[Ras16] Cyrus Rashtchian. Bounded matrix rigidity and John's theorem. *ECCC preprint TR16-093*, 2016.

[Raz87] A. 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(4):333–338, 1987.

[Raz89] A. A. Razborov. On rigid matrices (in Russian). Manuscript can be found at `http://people.cs.uchicago.edu/~razborov/files/rigid.pdf`, 1989.

[RS10] Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of AC^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000.

[Rub18] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *STOC*, 2018.

[Saw18] Will Sawin. Bounds for matchings in nonabelian groups. *The Electronic Journal of Combinatorics*, 25(4):4–23, 2018.

[Sch81] A. Schönhage. Partial and total matrix multiplication. *SIAM J. Comput.*, 10(3):434–455, 1981.

[Sch05] Rainer Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *J. Algorithms*, 54(1):40–44, 2005.

[She08] Alexander A Sherstov. Approximate inclusion-exclusion for arbitrary symmetric functions. In *CCC*, pages 112–123, 2008.

[She13] Alexander A. Sherstov. Making polynomials robust to noise. *Theory of Computing*, 9:593–615, 2013.

[She14] Alexander A. Sherstov. Breaking the Minsky–Papert barrier for constant-depth circuits. In *STOC*, pages 223–232, 2014.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82, 1987.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[Sri13] Srikanth Srinivasan. On improved degree lower bounds for polynomial approximation. In *FSTTCS*, pages 201–212, 2013.

[SS42] Raphaël Salem and Donald C Spencer. On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, 28(12):561–563, 1942.

[SS96] Victor Shoup and Roman Smolensky. Lower bounds for polynomial evaluation and interpolation problems. *Computational Complexity*, 6(4):301–311, 1996.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Mathematics*, 8(2):223–250, 1995.

[SSS97] M.A. Shokrollahi, D.A. Spielman, and V. Stemann. A remark on matrix rigidity. *Information Processing Letters*, 64(6):283–285, 1997.

[SST15] Takayuki Sakai, Kazuhisa Seto, and Suguru Tamaki. Solving sparse instances of Max SAT via width reduction and greedy restriction. *Theory Comput. Syst.*, 57(2):426–443, 2015.

[SSTT15a] Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. Improved exact algorithms for mildly sparse instances of Max SAT. In *IPEC*, pages 90–101, 2015.

[SSTT15b] Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. A satisfiability algorithm for depth-2 circuits with a symmetric gate at the top and AND gates at the bottom. *ECCC preprint TR15-136*, 2015.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[Str73] Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

[Str86] Volker Strassen. The asymptotic spectrum of tensors and the exponent of matrix multiplication. In *FOCS*, pages 49–54, 1986.

[Str87] Volker Strassen. Relative bilinear complexity and matrix multiplication. *J. reine angew. Math. (Crelles Journal)*, 375–376:406–443, 1987.

[Str91] Volker Strassen. Degeneration and complexity of bilinear maps: some asymptotic spectra. *Crelles J. Reine Angew. Math*, 413:127–180, 1991.

[SV12] Rocco A. Servedio and Emanuele Viola. On a special case of rigidity. *ECCC preprint TR12-144*, 2012.

[SV13] Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Theoretical Computer Science*, 9(2):125–210, 2013.

[Sze75] Gabor Szegö. *Orthogonal Polynomials*. American Mathematical Society, 1975.

[Tam16] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *ECCC preprint TR16-100*, 2016.

[Tao16] Terence Tao. A symmetric formulation of the croot-lev-pach-ellenberg-gijswijt capset bound. `https://terrytao.wordpress.com/2016/05/18/a-symmetric-formulation-of-the-croot-lev-pach-ellenberg-gijswijt-capset-bound/`, 2016.

[Tar93] Jun Tarui. Probabilistic polynomials, AC0 functions and the polynomial-time hierarchy. *Theor. Comput. Sci.*, 113(1):167–183, 1993.

[TS16] Terence Tao and Will Sawin. Notes on the "slice rank" of tensors. `https://terrytao.wordpress.com/2016/08/24/notes-on-the-slice-rank-of-tensors/`, 2016.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *MFCS*, pages 162–176, 1977.

[Val88] Leslie G Valiant. Functionality in neural nets. In *AAAI*, 1988.

[Val15] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015.

[Vol99] Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer, 1999.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *STOC*, pages 887–898, 2012.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14a] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673, 2014.

[Wil14b] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *STOC*, pages 194–202, 2014.

[Wil14c] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2, 2014.

[Wil14d] Ryan Williams. The polynomial method in circuit complexity applied to algorithm design (invited talk). In *FSTTCS*, pages 47–60, 2014.

[Wil18] Ryan Williams. Counting solutions to polynomial systems via reductions. In *SOSA*, pages 6:1–6:15, 2018.

[Win71] Shmuel Winograd. On multiplication of 2× 2 matrices. *Linear algebra and its applications*, 4(4):381–388, 1971.

[Wun12] Henning Wunderlich. On a theorem of razborov. *Computational Complexity*, 21(3):431–477, 2012.

[WY14] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *SODA*, pages 1867–1877, 2014.

[Yao83] Andrew C. Yao. Lower bounds by probabilistic arguments. In *FOCS*, pages 420–428, 1983.

[Yao90] Andrew C. Yao. On ACC and threshold circuits. In *FOCS*, pages 619–627, 1990.

[Yat37] F. Yates. The design and analysis of factorial experiments. *Technical Communication No. 35, Commonwealth Bureau of Soil Science, Harpenden, UK*, 1937.

[Yek12] Sergey Yekhanin. Locally decodable codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012.

[Zák83] Stanislav Zák. A Turing machine time hierarchy. *Theor. Comput. Sci.*, 26:327–333, 1983.