Fine-Grained Complexity Meets Communication Complexity

by

Lijie Chen

B.S., Tsinghua University (2017)

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

Masters of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
May 22, 2019

Signature redacted

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

Signature redacted

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

Fine-Grained Complexity Meets Communication Complexity

by

Lijie Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2019, in partial fulfillment of the
requirements for the degree of
Masters of Science in Computer Science and Engineering

Abstract

Fine-grained complexity aims to understand the exact exponent of the running time of fundamental problems in P. Basing on several important conjectures such as Strong Exponential Time Hypothesis (SETH), All-Pair Shortest Path Conjecture, and the 3-Sum Conjecture, tight conditional lower bounds are proved for numerous *exact* problems from all fields of computer science, showing that many text-book algorithms are in fact optimal.

For many natural problems, a fast approximation algorithm would be as important as fast exact algorithms. So it would be interesting to show hardness for approximation algorithms as well. But we had few techniques to prove tight hardness for approximation problems in P—In particular, the celebrated PCP Theorem, which proves similar approximation hardness in the world of NP-completeness, is not fine-grained enough to yield interesting conditional lower bounds for approximation problems in P.

In 2017, a breakthrough work of Abboud, Rubinstein and Williams [12] established a framework called "Distributed PCP", and applied that to show conditional hardness (under SETH) for several fundamental approximation problems in P. The most interesting aspect of their work is a connection between fine-grained complexity and communication complexity, which shows Merlin-Arther communication protocols can be utilized to give fine-grained reductions between exact and approximation problems.

In this thesis, we further explore the connection between fine-grained complexity and communication complexity. More specifically, we have two sets of results.

In the first set of results, we consider communication protocols other than Merlin-Arther protocols in [12] and show that they can be used to construct other fine-grained reductions between problems.

- **Σ_2 Protocols and An Equivalence Class for Orthogonal Vectors (OV).** First, we observe that efficient Σ_2^{cc} protocols for a function imply fine-grained reductions from a certain related problem to OV. Together with other techniques including locality-sensitive hashing, we establish an equivalence class for OV with $O(\log n)$ dimensions, including Max-IP/Min-IP, approximate Max-IP/Min-IP, and approximate bichromatic closest/further pair.

- **NP·UPP Protocols and Hardness for Computational Geometry Problems in $2^{O(\log^* n)}$ Dimensions.** Second, we consider NP · UPP protocols which are the relaxation of Merlin-Arther protocols such that Alice and Bob only need to be convinced with probability $> 1/2$ instead of $> 2/3$. We observe that NP · UPP protocols are closely connected to $\mathbb{Z}$-Max-IP problem in very small dimensions, and show that $\mathbb{Z}$-Max-IP, ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair in $2^{O(\log^* n)}$ dimensions requires $n^{2-o(1)}$ time under SETH, by constructing an efficient NP · UPP protocol for the Set-Disjointness problem. This improves on the previous hardness result for these problems in $\omega(\log^2 \log n)$ dimensions by Williams [172].

- **IP Protocols and Hardness for Approximation Problems Under Stronger Conjectures.** Third, building on the connection between $\mathsf{IP}^{\mathsf{cc}}$ protocols and a certain alternating product problem observed by Abboud and Rubinstein [11] and the classical IP = PSPACE theorem [123, 155]. We show that several fine-grained problems are hard under conjectures much stronger than SETH (e.g., the satisfiability of $n^{o(1)}$-depth circuits requires $2^{(1-o(1))n}$ time).

In the second set of results, we utilize communication protocols to construct new algorithms.

- **$\mathsf{BQP}^{\mathsf{cc}}$ Protocols and Approximate Counting Algorithms.** Our first connection is that a fast $\mathsf{BQP}^{\mathsf{cc}}$ protocol for a function f implies a fast deterministic additive approximate counting algorithm for a related pair counting problem. Applying known $\mathsf{BQP}^{\mathsf{cc}}$ protocols, we get fast deterministic additive approximate counting algorithms for Count-OV (#OV), Sparse Count-OV and Formula of SYM circuits.

- **$\mathsf{AM}^{\mathsf{cc}}$/$\mathsf{PH}^{\mathsf{cc}}$ Protocols and Efficient SAT Algorithms.** Our second connection is that a fast $\mathsf{AM}^{\mathsf{cc}}$ (or $\mathsf{PH}^{\mathsf{cc}}$) protocol for a function f implies a faster-than-bruteforce algorithm for a related problem. In particular, we show that if the Longest Common Subsequence (LCS) problem admits a fast (computationally efficient) $\mathsf{PH}^{\mathsf{cc}}$ protocol (polylog(n) complexity), then polynomial-size Formula-SAT admits a $2^{n-n^{1-\delta}}$ time algorithm for any constant $\delta > 0$, which is conjectured to be unlikely by a recent work of Abboud and Bringmann [6].

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

Acknowledgments

I would like to express my gratitude to my advisor, Richard Ryan Williams, for all the guidance and support in my graduate career so far. You have always been a supportive advisor and sometimes look omnipotent in your insightful observations. I still remember that when I was very young, young enough that I didn't really understand much about circuit complexity. I read several papers by you (including the famous paper showing $\mathsf{NEXP} \not\subset \mathsf{ACC}^0$) and was caught by the elegance of the arguments in it. Since then, complexity became my obsession. It is kind of miracle to me that many years later I am actually working with this man and proving theorems.

This thesis is based on four papers [70, 71, 72, 73] by my coauthors and me. I would like to thank all my coauthors on these projects, without them these works are definitely impossible, including Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, Aviad Rubinstein, Ruosong Wang, and Ryan Williams.

I would particularly like to thank my previous mentor Scott Aaronson, who hosted me while I was visiting MIT during the spring of 2016 (and again during the winter of 2019, in Austin). I have always been a big fan of you and your wonderful blog and book Quantum Computing since Democritus (I have literally read this book more than 10 times and it is the thing I usually turned to when I felt like I lost motivations). Very sadly, I didn't continue my study in quantum complexity theory and switched to classical complexity, but I think the things I learned from you will help me throughout my future career.

I would love to thank my undergraduate supervisor, Jian Li, for introducing me to the wonderful world of theoretical computer science. I still remember that when I was a second-year undergraduate with basically no knowledge in TCS. I took your advanced theoretical computer science course and was overwhelmed by the intense workload of 20 hours a week. But it turned out to be super rewarding, and later I found myself working on Multi-Armed Bandit with you for the following years. Despite that I eventually left the field and switched to complexity, I have enjoyed a lot in working on these questions and these research projects turned out to be much more fruitful than I previously expected.

I would also like to thank the many friends of the theory group of MIT/Harvard, the lower bound/data science program of Simons Institute, and Yao class of Tsinghua University. This acknowledge is probably too short to list all the names here, but you know who you are. I do want to give a special shout-out to the TCS Boys group back in Tsinghua (intended as a mimic of the popular TF Boys group in China), including me, Yuping Luo, Ruosong Wang, and Wei Zhan for all the great memories.

I would like to thank Akamai and NSF for generously funding my graduate studies and Simons Institute for hosting me during the Fall of 2018.

Finally, I would like to thank my family for their continuing and unconditional support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background: Fine-Grained Complexity, from Exact to Approximate

The program of *Fine-Grained Complexity* (a.k.a. *Hardness in P*), is one of the most exciting recent developments in theoretical computer science. The program seeks to answer the following type of questions: what is the *exact exponent* on the running time for a problem in P? These type of questions were mostly ignored by complexity theorists, as in their language, being in P already means *easy*. But from a practical perspective, there is a clear desire to understand whether the running time of (say) Edit Distance can be made close to linear, or it inherently requires roughly quadratic time, as the difference between linear-time and quadratic-time is tremendous in the real life, especially in this modern big data era.

Motivated by this quest of getting a more "fine-grained" understanding of computation, there has been a surge of works in this area, which established tight running time lower bounds on nearly all classical problems, ranging from pattern matching and bioinformatics [15, 41, 40, 55, 57, 6], dynamic data structures [140, 14, 14, 100, 112, 8, 101, 91], graph algorithms [150, 89, 16, 113, 43, 118, 81], computational geometry [54, 172, 83, 70, 107] and machine learning [42] under the SETH[1], the APSP

[1] The Strong Exponential Time Hypothesis (SETH) states that for every $\varepsilon > 0$ there is a k such that k-SAT cannot be solved in $O((2-\varepsilon)^n)$ time [102].

Conjecture, and the k-Sum Conjecture. See [163] for a wonderful recent survey on the whole development.

The first generation of fine-grained complexity results are mostly concerned about *exact problems*: e.g., they showed that computing Edit Distance, Longest Common Subsequence or Maximum Inner Product *exactly* requires $N^{2-o(1)}$ time under plausible conjectures. But it is often the case that a good enough approximation algorithm would be as significant as a good exact algorithm, and these results fail to show any interesting lower bound for approximation algorithms.

The lack of fine-grained approximation lower bounds had been recognized as one of the most important open questions in this field [3]. One crucial difficulty for showing such approximation lower bounds is that the traditional PCP paradigm [37, 36, 99] which proves similar approximation hardness for NP-hard optimization problems cannot be applied directly to fine-grained complexity, due to the super-linear size blow up in the constructed PCP instances [37, 36, 87], which becomes super-polynomial after reducing to problems in P. (When we care about the *exact* exponent of the running time, a super-polynomial blow-up is certainly unacceptable.)

There were basically no non-trivial fine-grained approximation lower bounds before the breakthrough work of Abboud, Rubinstein, and Williams [12] (which is further improved by Rubinstein [151]). They introduced a "Distributed PCP" framework and used it to show tight conditional lower bounds for several fundamental approximation problems, including approximate Bichromatic Max-Inner Product, Subset Query, Bichromatic LCS Closest Pair, Regular Expression Matching and Diameter in Product Metrics, under the SETH assumption.

1.2 Fine-Grained Complexity Meets Communication Complexity

To establish fine-grained lower bounds for approximation problems, the most important question is how to design a reduction *which creates a gap in the optimal value.* As discussed previously, one cannot afford the blowup of the traditional PCP paradigm, so one must try to explore other methods to construct *gap-creating reductions.*

The most interesting aspect of [12] is that it crucially relies on the $\widetilde{O}(\sqrt{N})$-complexity Merlin-Arther protocol for Set-Disjointness of Aaronson and Widgerson [2]. In a Merlin-Arther Protocol, there are two players Alice and Bob holding two inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and they wish to jointly compute a function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. To minimize their communication, they also seek help from an untrusted prover Merlin. Merlin sends them a proof ψ, and Alice and Bob verify this proof with randomized communication. We say a protocol Π correctly computes the function f, if when $f(x, y) = 1$, there exists a proof ψ which makes Alice and Bob accepts with high probability; and when $f(x, y) = 0$, Alice and Bob reject all proofs with high probability.

Satisfying-Pair Problem and $\mathsf{MA}^{\mathsf{cc}}$-Satisfying-Pair

To further discuss their underlying ideas, we introduce the f-Satisfying-Pair problem. For a function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, the f-Satisfying-Pair is the problem that given two sets $A \subseteq X$ and $B \subseteq Y$, and decide whether there exists a pair $(x, y) \in A \times B$ such that $f(x, y) = 1$. In this way, the well-studied Orthogonal Vectors problem (OV)[2] is just f_{OV}-Satisfying-Pair, where f_{OV} checks whether two Boolean vectors are orthogonal.

Now, suppose f admits an $\mathsf{MA}^{\mathsf{cc}}$ protocol Π. Then we can reduce f-Satisfying-Pair to Π-Satisfying-Pair, the latter one is the similar problem which asks whether there is a pair $(x, y) \in A \times B$ which makes the protocol Π accept (in other words, there exists a proof ψ such that Alice and Bob accepts ψ with inputs x and y).

The key observation of [12] is that the $\mathsf{MA}^{\mathsf{cc}}$-Satisfying-Pair problems (the class of Π-Satisfying-Pair problems for a valid MA protocol Π) can be further reduced to simpler problems such as approximate Max-IP. Also, the function f_{OV} is in fact equivalent to the classical function *Set-Disjointness*[3] in communication complexity.

[2] Given two sets $A, B \subset \{0, 1\}^d$, determine the existence of a pair $(a, b) \in A \times B$ such that $\langle a, b \rangle = 0$.

[3] In the Set-Disjointness problem, Alice and Bob hold two sets $A, B \subseteq [d]$, and want to determine

Therefore, combing these observations together with the efficient MA protocol for Set-Disjointness [2], it established a reduction from OV to approximate Max-IP, which implies approximate Max-IP requires $n^{2-o(1)}$ time under SETH.

Potentially Extensions

Some natural questions arise from their intriguing work.

- First, can we apply this to f-Satisfying-Pair problems for f other than f_{OV}? This general approach would clearly work for any f admitting an efficient $\mathsf{MA}^{\mathsf{cc}}$ protocol, and it is a potential way to create more reductions.

- Second, what about other types of communication protocols? They may correspond to some other problems different than approximate Max-IP. In this way, we may obtain interesting fine-grained reductions to other fundamental problems as well.

- Third, can these connections be used *algorithmically*? Suppose that f has an efficient $\mathcal{C}$ protocol Π, and one can solve the corresponding Π-Satisfying-Pair problem with a faster algorithm. Then, one would automatically obtain a faster algorithm for f-Satisfying-Pair as well.

- Fourth, what about problems beyond f-Satisfying-Pair?

1.3 Our Results

Our thesis is motivated by all the questions above. In particular, we have two sets of results: first, we consider communication protocols other than $\mathsf{MA}^{\mathsf{cc}}$ protocols, and obtain various fine-grained reductions between problems; second, we investigate the algorithmic use of communication protocols, and obtain several new algorithms.

whether their sets are disjoint. Let $I_A, I_B \in \{0,1\}^d$ be the indicator vectors of A and B, then A and B are disjoint iff $\langle I_A, I_B \rangle = 0$.

1.3.1 Fine-Grained Reductions from Other Communication Protocols

The first direction we try to extend [12] is noting that if we consider different communication protocols other than $\mathsf{MA}^{\mathsf{cc}}$ protocols, we would obtain other fine-grained reductions between problems.

Σ_2 Protocols and An Equivalence Class for Orthogonal Vectors

We begin with considering Σ_2^{cc} protocols. The key observation here is that Σ_2^{cc} protocols imply fine-grained reductions to OV. This is especially useful if we want to show some fine-grained problems can be reduced back to OV (and therefore potentially form **an equivalence class for OV**).

Formally, we show OV is truly-subquadratic equivalent to several fundamental problems, all of which (a priori) look harder than OV. A partial list is given below:

1. (Min-IP/Max-IP) Find a red-blue pair of vectors with minimum (respectively, maximum) inner product, among n vectors in $\{0,1\}^{O(\log n)}$.

2. (Exact-IP) Find a red-blue pair of vectors with inner product equal to a given target integer, among n vectors in $\{0,1\}^{O(\log n)}$.

3. (Apx-Min-IP/Apx-Max-IP) Find a red-blue pair of vectors that is a 100-approximation to the minimum (resp. maximum) inner product, among n vectors in $\{0,1\}^{O(\log n)}$.

4. (Approximate Bichrom.-ℓ_p-Closest-Pair) Compute a $(1+\Omega(1))$-approximation to the ℓ_p-closest red-blue pair (for a constant $p \in [1,2]$), among n points in $\mathbb{R}^d$, $d \le n^{o(1)}$.

5. (Approximate ℓ_p-Furthest-Pair) Compute a $(1+\Omega(1))$-approximation to the ℓ_p-furthest pair (for a constant $p \in [1,2]$), among n points in $\mathbb{R}^d$, $d \le n^{o(1)}$.

Therefore, quick constant-factor approximations to maximum inner product imply quick *exact* solutions to maximum inner product, in the $O(\log n)$-dimensional setting.

Another consequence is that the ability to find vectors with zero inner product suffices for finding vectors with maximum inner product.

Our equivalence results are robust enough that they continue to hold in the data structure setting. In particular, we show that there is a $\mathrm{poly}(n)$ space, $n^{1-\varepsilon}$ query time data structure for *Partial Match* with vectors from $\{0,1\}^{O(\log n)}$ if and only if such a data structure exists for $1+\Omega(1)$ *Approximate Nearest Neighbor Search* in Euclidean space. These results are discussed in Chapter 3 in details.

$\mathsf{NP}\cdot\mathsf{UPP}^{\mathsf{cc}}$ Protocols and Hardness for Computational Geometry Problems in $2^{O(\log^* n)}$ Dimensions

Next, we consider $\mathsf{NP}\cdot\mathsf{UPP}^{\mathsf{cc}}$ protocols, which is a relaxation of the original Merlin-Arther protocols. In particular, recall that in a Merlin-Arther protocol, Alice and Bob are required to accept the proof with probability $> 2/3$ if the answer is yes and the proof is correct, and reject the proof with probability $> 2/3$ if the answer is no. In an $\mathsf{NP}\cdot\mathsf{UPP}^{\mathsf{cc}}$ protocol, the threshold $2/3$ is relaxed to $1/2$. That is, Alice and Bob only need to have a better-than-half probability of being correct.

We observe that $\mathsf{NP}\cdot\mathsf{UPP}^{\mathsf{cc}}$ protocols imply reductions to the $\mathbb{Z}$-Max-IP problem[4]. Therefore, by constructing efficient $\mathsf{NP}\cdot\mathsf{UPP}^{\mathsf{cc}}$ for the *Set-Disjointness* problem, we obtain fine-grained reductions from OV to $\mathbb{Z}$-Max-IP, and therefore establish OV-hardness for the latter problem.

Formally, we show that $\mathbb{Z}$-Max-IP is hard to solve in $n^{2-\Omega(1)}$ time, even with $2^{O(\log^* n)}$-dimensional vectors. As direct corollaries, using reductions implicit in [172], we also conclude hardness for ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair under SETH (or OVC) in $2^{O(\log^* n)}$ dimensions.

The above lower bounds on ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair are in sharp contrast with the case of ℓ_2-*Closest Pair*, which can be solved in $2^{O(d)}\cdot n\log^{O(1)} n$ time [49, 109, 86].

Our results are proved by constructing an $\mathsf{NP}\cdot\mathsf{UPP}$ protocol for DISJ_n with $o(n)$ proof complexity and $O(\log^* n)$ communication complexity, which is based on a *re-*

[4] Given two sets $A, B \subset \mathbb{Z}^d$, compute $\max_{(a,b)\in A\times B}\langle a,b\rangle$.

cursive application of the *Chinese Remainder Theorem.* If one can further reduce the communication complexity to $\alpha(n)$ while keeping the proof complexity sub-linear, it would automatically imply OV-hardness for $\mathbb{Z}$-Max-IP with $2^{\alpha(n)}$ dimensions. We discuss the details in Chapter 4.

$\mathsf{IP}^{\mathsf{cc}}$ Protocols and Hardness for Approximation Problems Under Stronger Conjectures

Then we consider $\mathsf{IP}^{\mathsf{cc}}$ protocols, which are similar to $\mathsf{MA}^{\mathsf{cc}}$ protocols except for that Alice, Bob, and Merlin are now allowed to interact. The insight here is that $\mathsf{IP}^{\mathsf{cc}}$ communication protocols can be reduced to several interesting combinatorial problems (this is crucially observed in [11]). Our contribution here is to notice that by the classical IP = PSPACE theorem, all low-space computable functions admit efficient $\mathsf{IP}^{\mathsf{cc}}$ protocols. Therefore, combine these two ideas, we can construct reductions from BP-Satisfying-Pair[5] instead of OV, which lets us to show hardness results under conjectures much weaker than SETH.

Basing on the above observation, we are able to construct reductions from an exact to an approximate solution for a host of problems.

As one (notable) example, we show that the Closest-LCS-Pair problem (Given two sets of strings A and B, compute exactly the maximum $\mathsf{LCS}(a, b)$ with $(a, b) \in A \times B$) is equivalent to its approximation version (under near-linear time reductions, and with a constant approximation factor). More generally, we identify a class of problems, which we call BP-Pair-Class, comprising both exact and approximate solutions, and show that they are all equivalent under near-linear time reductions.

Exploring this class and its properties, we also show:

- Under the NC-SETH assumption (a significantly more relaxed assumption than SETH), solving any of the problems in this class requires essentially quadratic time.

[5] Roughly speaking, BP-Satisfying-Pair is the problem that given two sets $A, B \subset \{0,1\}^d$ and a branching program P on $2d$ bits, asks whether there is a pair $(a, b) \in A \times B$ such that $P(a, b) = 1$. See Section 5.2 for formal definitions of BP-Satisfying-Pair and Branching Programs.

- Modest improvements on the running time of known algorithms (shaving log factors) would imply that NEXP is not in non-uniform NC^1.

- Finally, we leverage our techniques to show new barriers for deterministic approximation algorithms for LCS.

A very important consequence of our results is that they continue to hold in the ***data structure setting***. In particular, it shows that a data structure for *approximate* Nearest Neighbor Search for LCS ($\mathsf{NNS}_{\mathsf{LCS}}$) implies a data structure for *exact* $\mathsf{NNS}_{\mathsf{LCS}}$ and a data structure for answering regular expression queries with essentially the same complexity.

Moreover, we show that under the conjecture that the satisfiability of an $n^{o(1)}$-depth circuit cannot be solved in $2^{(1-\Omega(1))\cdot n}$ time, Subtree Isomorphism, approximate Largest Common Subtree, and Regular Expression Membership Testing cannot be solved in $N^{2-o(1)}$. This improves upon several previous works showing similar hardness results under SETH by significantly weaken the hypothesis (CNF is much, much weaker than $n^{o(1)}$-depth circuits). These results are discussed in Chapter 5.

1.3.2 Algorithms from Communication Protocols

Our second set of results discusses a different way of viewing the connection between fine-grained complexity and communication complexity. We show that in some cases these communication protocols may help us develop interesting algorithms.

BQP Protocols and Approximate Counting Algorithms

We first consider $\mathsf{BQP}^{\mathsf{cc}}$ protocols. It is well-known that an efficient $\mathsf{BQP}^{\mathsf{cc}}$ protocol for a function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ implies a low-rank approximation to the corresponding matrix M_f. This connection is often interpreted as an approach to prove $\mathsf{BQP}^{\mathsf{cc}}$ lower bounds. We show in fact they can also be used to construct new algorithms.

For a function f as above, and two sets $A \subset X$ and $B \subset Y$, the f-#Satisfying-Pair problem asks to count the number of pairs $(x, y) \in A \times B$ such that $f(x, y) = 1$. We show that a fast $\mathsf{BQP}^{\mathsf{cc}}$ protocol implies a *deterministic* algorithm for computing an additive approximation to f-#Satisfying-Pair. In particular, using the $O(\sqrt{n})$-complexity $\mathsf{BQP}^{\mathsf{cc}}$ protocol for Set-Disjointness [1], we obtain an $n^{1+o(1)}$ time algorithm for computing an $\varepsilon \cdot n^2$ additive approximation to #OV[6] with n vectors and $o(\log^2 n)$ dimensions. The details can be found in Chapter 6.

$\mathsf{PH}^{\mathsf{cc}}$ Protocols and Efficient SAT Algorithms

Next, we consider $\mathsf{PH}^{\mathsf{cc}}$ protocols. It is a long-standing open question to prove non-trivial $\mathsf{PH}^{\mathsf{cc}}$ lower bounds for an explicit function [38, 93, 94], which is an intermediate step towards the notoriously hard problem of constructing an explicit rigid matrix [145].

We show that, for a function $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, a computational efficient (meaning, Alice and Bob's actions can be computed in polynomial-time given the history of communication) $\mathsf{PH}^{\mathsf{cc}}$ protocol in fact implies a non-trivial algorithm for f-Satisfying-Pair.

An interesting consequence is that, if the natural $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ problem or $\mathsf{LCS}^{\mathsf{cc}}$ problem (in which Alice and Bob each hold an n-bit string, and want to compute the edit distance/longest common subsequence between them) have a computational efficient $\mathsf{PH}^{\mathsf{cc}}$ protocol of polylog(n) complexity, then the satisfiability of polynomial-size formulas has a $2^{n-n^{1-\delta}}$ time algorithm, for any $\delta > 0$. This is much faster than the current state-of-the-art algorithm in [161], which solves $n^{3-\varepsilon}$-size formula-SAT in $2^{n-n^{\Omega(\varepsilon)}}$ time (the algorithm can only handle formulas of sub-cubic size), and is conjectured to be impossible by a recent work [6]. The details can be found in Chapter 7.

1.4 Related Works

In this section we discuss some related works which are generally related to the topics of this thesis. In each chapter we may also discuss some additional related works which are only related to that chapter.

[6] Given two sets $A, B \subset \{0, 1\}^d$, count the number of pairs $(a, b) \in A \times B$ such that $\langle a, b \rangle = 0$.

Communication Complexity and Conditional Hardness

The connection between communication protocols (in various models) for Set-Disjointness and SETH dates back at least to [142], in which it is shown that a sub-linear, computational efficient protocol for 3-party Number-On-Forehead Set-Disjointness problem would refute SETH. It is worth mentioning that [11]'s result builds on the $\widetilde{O}(\log n)$ IP communication protocol for Set-Disjointness in [2].

Distributed PCP

Using Algebraic Geometry codes (AG codes), [151] obtains a better MA protocol, which in turn improves the efficiency of the previous distributed PCP construction of [12]. He then shows the $n^{2-o(1)}$ time hardness for $1+o(1)$-approximation to Bichromatic Closest Pair and $o(d)$-additive approximation to $\mathsf{Max\text{-}IP}_{n,d}$ with this new technique.

[106] use the Distributed PCP framework to derive inapproximability results for k-Dominating Set under various assumptions. In particular, building on the techniques of [151], it is shown that under SETH, k-Dominating Set has no $(\log n)^{1/\operatorname{poly}(k,e(\varepsilon))}$ approximation in $n^{k-\varepsilon}$ time[7].

[107] also utilize AG codes and polynomial method to show hardness results for exact and approximate Monochromatic Closest Pair and approximate Monochromatic Maximum Inner Product.

Hardness of Approximation in P

Making use of Chebyshev embeddings, [18] prove a $2^{\Omega\left(\frac{\sqrt{\log n}}{\log\log n}\right)}$ inapproximability lower bound on $\{-1,1\}$-$\mathsf{Max\text{-}IP}$. [3] take an approach different from Distributed PCP, and shows that under certain complexity assumptions, LCS does not have a *deterministic* $1+o(1)$-approximation in $n^{2-\varepsilon}$ time. They also establish a connection with circuit lower bounds and show that the existence of such a *deterministic* algorithm implies E^{NP} does not have non-uniform linear-size Valiant Series Parallel circuits. In [11], it is

[7] where $e : \mathbb{R}^+ \to \mathbb{N}$ is some function

improved to that any constant factor approximation deterministic algorithm for LCS in $n^{2-\varepsilon}$ time implies that E^{NP} does not have non-uniform linear-size NC^1 circuits.

Using a mostly combinatorial construction, [117] improves the inapproximability ratio under SETH for k-Dominating set to $\widetilde{\Omega}(\sqrt[k]{\log n})$.

See [12] for more related results in hardness of approximation in P.

Equivalence Classes in Fine-Grained Complexity

It is known that the All-Pairs Shortest Paths problem is sub-cubic time equivalent to many other problems [164, 39, 9, 118]. A partial list includes Negative Triangle, Triangle Listing, Shortest Cycle, 2nd Shortest Path, Max Subarray, Graph Median, Graph Radius and Wiener Index (see [163] for more details on the APSP equivalence class).

In [89], it is shown that "moderate-dimensional" OV (i.e., OV with n^{δ} dimensions for some $\delta > 0$) is equivalent to High-dimension Sparse OV, High-dimension 2-Set Cover, and High-dimension Sperner Family. It is also shown that for every $(k+1)$-quantifier first-order property, its model-checking problem can be reduced to Sparse k-OV. In [80], the authors present an equivalence class for $(\min, +)$-convolution, including some variants of the classical knapsack problem and problems related to subadditive sequences. [115] prove several equivalence between one-dimensional dynamic problems and their corresponding core problems. In particular, it is shown that OV is equivalent to finding the longest subset chain (SubsetChain).

Chapter 2

Preliminaries

In this chapter we introduce some common preliminaries which are used throughout the thesis.

2.1 Notations

We begin by introducing some notation. In this thesis, we let $\mathbb{R}^+$ denote the set of all positive reals. For an integer d, we use $[d]$ to denote the set of integers from 1 to d. For a vector u, we use u_i to denote the i-th element of u.

We use $\log(x)$ to denote the logarithm of x with respect to base 2 with ceiling as appropriate, and $\ln(x)$ to denote the natural logarithm of x.

We also need the iterated logarithm function $\log^*(n)$, which is defined recursively as follows:

$$\log^*(n) := \begin{cases} 0 & n \leq 1; \\ \log^*(\log n) + 1 & n > 1. \end{cases}$$

2.2 Problems and Hypothesis in Fine-Grained Complexity

We first define the F-Satisfying-Pair problem for a problem F.[1]

Definition 2.2.1 ([10]). Let $F : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}$, F-$\mathsf{Satisfying\text{-}Pair}_n$ is the problem: given two sets A and B of n vectors from $\{0,1\}^d$, determine whether there is a pair $(a,b) \in A \times B$ such that $F(a,b) = 1$.

Remark 2.2.2. *For example, let F_{OV} be the function checking whether two vectors from $\{0,1\}^d$ are orthogonal. Then, F_{OV}-$\mathsf{Satisfying\text{-}Pair}_n$ is simply $\mathsf{OV}_{n,d}$ (defined below).*

We use $\mathsf{OV}_{n,d}$ to denote the Orthogonal Vectors problem: given two sets of vectors A, B each consisting of n vectors from $\{0,1\}^d$, determine whether there are $a \in A$ and $b \in B$ such that $a \cdot b = 0$.[2] Similarly, we use $\mathbb{Z}$-$\mathsf{OV}_{n,d}$ to denote the same problem except for that A, B consists of vectors from $\mathbb{Z}^d$ (which is also called Hopcroft's problem).

The following widely used conjecture about OV is used multiple times in this thesis.

Conjecture 2.2.3 (Orthogonal Vectors Conjecture (OVC) [168, 15]). *For every $\varepsilon > 0$, there exists a $c \geq 1$ such that $\mathsf{OV}_{n,d}$ requires $n^{2-\varepsilon}$ time when $d = c \log n$.*

OVC is a plausible conjecture as it is implied by the popular Strong Exponential Time Hypothesis [102, 62] on the time complexity of solving k-SAT [168, 174].

2.3 Communication Protocols

Next we introduce these communication protocols which are used in this thesis. See also [94] for a recent reference on them.

[1] This notation is borrowed from [10], which studied the Satisfying Pair problem for Branching Programs.

[2] Here we use the bichromatic version of OV instead of the monochromatic one for convenience, as they are equivalent.

Definition 2.3.1 ($\mathsf{MA}^{\mathsf{cc}}$ Protocols)**.** We say an MA Protocol is (m, r, ℓ, s)-efficient for a communication problem, if in the protocol:

- There are three parties Alice, Bob and Merlin in the protocol, Alice holds input x and Bob holds input y.
- Merlin sends an advice string z of length m to Alice, which is a function of x and y.
- Alice and Bob jointly toss r coins to obtain a random string w of length r.
- Given y and w, Bob sends Alice a message of length ℓ.
- After that, Alice decides whether to accept or not.
  - When the answer is yes, Merlin has a proof such that Alice always accept.
  - When the answer is no, Alice accepts with probability at most s regardless of the proof sent by Merlin.

Definition 2.3.2 ($\mathsf{IP}^{\mathsf{cc}}$ Protocols)**.** Let $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a function. An $\mathsf{IP}^{\mathsf{cc}}$ protocol Π for F is specified as follows:

- There are three parties Alice, Bob and Merlin in the protocol, Alice holds input x and Bob holds input y.
- Alice and Merlin interact for several rounds. After that, Alice communicates with Bob to decide whether they accept or not.
  - When $F(x, y) = 1$, Merlin has a strategy which make Alice and Bob accept with probability at least $2/3$.
  - When $F(x, y) = 0$, Alice and Bob accept with probability at most $1/3$, regardless of Merlin's strategy.

The communication complexity of Π is simply the total number of bits sent by Alice, Bob, and Merlin.

Definition 2.3.3 (Σ_2^{cc} Protocols [38]). Let $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a function. A Σ_2^{cc} protocol Π for F is specified as follows:

- There are two players, Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$.
- There are two provers Merlin and Megan.
- Merlin sends a string $a \in \{0,1\}^{m_1}$ and Megan sends a string $b \in \{0,1\}^{m_2}$ (which are functions of both x and y) to both Alice and Bob. Then Alice and Bob communicate ℓ bits with each other, and Alice decides whether to accept or reject the pair (a, b).
- $F(x, y) = 1$ if and only if there exists a string a from Merlin, such that for all strings b from Megan, Alice accepts (a, b) after communications with Bob.

We say the protocol Π is computationally-efficient, if both Alice and Bob's response functions can be computed in polynomial time with respect to their input length.

Definition 2.3.4 ($\mathsf{PH}^{\mathsf{cc}}$ Protocols [38]). A PH communication protocol ($\mathsf{PH}^{\mathsf{cc}}$) Π for a function $F : \mathcal{X} \times \mathcal{Y} \to \{0,1,\perp\}$ proceeds as follows:

- Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$.
- For a constant $k \in \mathbb{N}$, there are $2k$ provers $P_1, P_2, \ldots, P_{2k}$.
- For each $i \in [2k]$, the prover P_i sends both Alice and Bob a proof $z_i \in \{0,1\}^{m_i}$.
- We use $A(x, z_1, z_2, \ldots, z_{2k})$ (resp. $B(y, z_1, z_2, \ldots, z_{2k})$) to be the indicator function that whether Alice (resp. Bob) accepts the proof sequence $z_1, z_2, \ldots, z_{2k}$, given the input x (resp. y).
- If $F(x, y) = 1$, then

$$\underset{z_1 \in \{0,1\}^{m_1}}{\exists} \ \underset{z_2 \in \{0,1\}^{m_2}}{\forall} \cdots \underset{z_{2k-1} \in \{0,1\}^{m_{2k-1}}}{\exists} \ \underset{z_{2k} \in \{0,1\}^{m_{2k}}}{\forall} \left[A(x, z_1, \ldots, z_{2k}) \wedge B(y, z_1, \ldots, z_{2k})\right].$$

- If $F(x, y) = 0$, then

$$\underset{z_1 \in \{0,1\}^{m_1}}{\forall} \; \underset{z_2 \in \{0,1\}^{m_2}}{\exists} \cdots \underset{z_{2k-1} \in \{0,1\}^{m_{2k-1}}}{\forall} \; \underset{z_{2k} \in \{0,1\}^{m_{2k}}}{\exists} \left[\neg A(x, z_1, \ldots, z_{2k}) \vee \neg B(y, z_1, \ldots, z_{2k})\right].$$

Moreover, we say the protocol is *computationally efficient* if Alice and Bob's decision functions (the functions A and B) can be computed in polynomial-time w.r.t. their input lengths. The communication complexity of Π is simply the total number of proof bits from all provers, i.e. $\sum_{i=1}^{2k} m_i$.

Chapter 3

Σ_2 Protocols and An Equivalence Class for Orthogonal Vectors

3.1 Introduction

3.1.1 Motivation: On the Similarities and Differences between NP-Hardness and Fine-Grained Complexity

Recall that Fine-grained complexity asks: *what is the "correct" exponent in the running time of a given problem?* For a problem known to be solvable in time $t(n)$, can it be solved in time $t(n)^{1-\varepsilon}$, for a constant $\varepsilon > 0$? If not, can we give evidence that such an improvement is impossible?

In a nutshell, results in the *Fine-Grained Complexity* program begin with the conjecture that it is *hard to improve the runtime exponent* of some problem Π_{hard}, and show it is also hard to improve the exponent of another problem Π, by constructing a "fine-grained" reduction from Π_{hard} to Π. This is similar to the situation with NP-completeness, where one shows a problem Π is "hard" by giving a polynomial-time reduction from another NP-complete problem to Π.

A crucial conceptual difference between the Fine-Grained Complexity program and NP-hardness is that *all of the thousands of known NP-complete problems form an equivalence class*: there is either a polynomial-time algorithm for all of them, or no polynomial-time algorithm for any of them. In contrast, with Fine-Grained Complexity, few equivalence classes are known, especially for those numerous problems whose hardnesses are based on the SETH/OVC (a notable exception is the equivalence class for APSP [164, 163]; see the related works section for more details).

To give three (out of many examples), it is known that Edit Distance [41], Frechet Distance [54], and computing the diameter of a sparse graph [150] cannot be done in $n^{2-\delta}$ time for any $\delta > 0$, assuming the Orthogonal Vectors Conjecture.

However, it is not known if Edit Distance, Frechet Distance, or Diameter are *equivalent* to OV, in any interesting sense.

Prior work has established an equivalence class for "moderate-dimensional OV", where the vector dimension $d = n^{\delta}$ for a constant $\delta > 0$ [89]. In particular, this version of OV is equivalent to various sparse graph and hypergraph problems. It seems likely that "moderate-dimensional OV" is much more difficult to solve than the "low-dimensional" setting of $d = O(\log n)$ as defined above, and the SETH already implies that the low-dimensional case is difficult [168, 174]. Thus, the problem of establishing an equivalence class for "low-dimensional" OV is an interesting one.

3.1.2 An Equivalence Class for Sparse Orthogonal Vectors

Our first result is an interesting equivalence class for Orthogonal Vectors in the $O(\log n)$-dimensional setting. To formally state our results, we begin with some notation.

- For a problem Π on Boolean vectors, we say Π *is in truly subquadratic time* if there is an $\varepsilon > 0$ such that for all constant c, Π is solvable in $O(n^{2-\varepsilon})$ time on n vectors in $c \log n$ dimensions. Note the Orthogonal Vectors Conjecture (OVC) is equivalent to saying "OV is not in truly subquadratic time."

- For a problem Π on real-valued points, we say Π *can be approximated in truly subquadratic time*, if there is a $\delta > 0$ such that for all $\varepsilon > 0$, a $(1 + \varepsilon)$ approximation to Π is computable in $O(n^{2-\delta})$ time.

- For a problem Π with output in $[0, L]$ (for a parameter L), we say Π *can be additively approximated in truly subquadratic time*, if there is a $\delta > 0$ such that for all $\varepsilon > 0$, an $\varepsilon \cdot L$ additive approximation to Π is computable in $O(n^{2-\delta})$ time.

Theorem 3.1.1. *The following problems are either all in (or can be approximated in) truly subquadratic time, or none of them are:*[1]

1. *(OV) Finding an orthogonal pair among n vectors.*

2. *(Min-IP/Max-IP) Finding a red-blue pair of vectors with minimum (respectively, maximum) inner product, among n vectors.*

3. *(Exact-IP) Finding a red-blue pair of vectors with inner product exactly equal to a given integer, among n vectors.*

4. *(Apx-Min-IP/Apx-Max-IP) Finding a red-blue pair of vectors that is a 100-approximation to the minimum (resp. maximum) inner product, among n vectors.*[2]

5. *(Approximate Bichrom. ℓ_p-Closest Pair) Approximating the ℓ_p-closest red-blue pair (for a constant $p \in [1, 2]$), among n points.*

6. *(Approximate ℓ_p-Furthest Pair) Approximating the ℓ_p-furthest pair (for a constant $p \in [1, 2]$), among n points.*

7. *(Approximate Additive Max-IP) Additively approximating the maximum inner product of all red-blue pairs, among n vectors.*

8. *(Approximate Jaccard-Index-Pair) Additively approximating the maximum Jaccard index*[3] *between $a \in A$ and $b \in B$, where A and B are two collections of n sets.*

[1] A list of formal definitions of the these problems can be found in Definition 3.3.1.

[2] The constant 100 can be replaced by any fixed constant $\kappa > 1$.

[3] see Theorem 3.3.3 for a formal definition

For approximate additive Max-IP, L (the additive approximation parameter) is simply the dimensions of the vectors, while for approximate Jaccard-Index-Pair, L is 1. For Π among the first four problems listed above, we use the notation $\Pi_{n,d}$ to denote Π with n vectors from $\{0,1\}^d$. [4] For the last four problems, we assume the dimensions (or the size of the sets) and the bit complexity of the points are $n^{o(1)}$ throughout the chapter.

Prior work showed OV is equivalent to Dominating Pair[5] [67] and other simple set problems [52]; our results add several interesting new members into the equivalence class. All problems listed above were already known to be OV-hard [168, 24, 151]. Our main contribution here is to show that ***they can all be reduced back to OV***. For example, detecting an orthogonal *Boolean* pair (OV) is equivalent to approximating the distance between two sets of points in $\mathbb{R}^{n^{o(1)}}$ (Bichrom.-Closest-Pair)!

In previous works [89, 7], several general techniques are given for constructing reductions to OV. These papers focus on the "moderate-dimensional" setting, and their reductions can not be used directly in the "sparse" $O(\log n)$ dimensional setting here.

Our Techniques: Two Reduction Frameworks for OV. In order to construct reductions to $O(\log n)$ dimensional OV, we propose the following two general frameworks.

- Σ_2^{cc} **Protocols.** Inspired by previous works on the connections between communication complexity and fine-grained complexity [12, 151, 106, 11, 70, 71], we draw another connection along this line, showing that an efficient Σ_2^{cc} protocol[6] for a function F implies a reduction from a related problem to OV. We use this technique to establish the equivalences among the first four problems in Theorem 3.1.1.

[4] In this chapter we will consider red-blue version for all the above problems, and $\Pi_{n,d}$ denotes Π with two sets of n vectors from $\{0,1\}^d$.

[5] Given two sets A, B of vectors from $\mathbb{R}^{O(\log n)}$, find $(a,b) \in A \times B$ such that b *dominates* a (that is, $b_i > a_i$ for all i).

[6] see Definition 3.2.1 for a formal definition

- **Locality-sensitive Hashing Families (LSH).** To show equivalences between OV and the last four approximation problems, we apply known tools from *locality-sensitive hashing.* In particular, we show that for any metric admitting an efficient LSH family, finding the closest bichromatic pair or the furthest pair w.r.t. this metric can be reduced to Apx-Max-IP, which can in turn be reduced to OV.

We remark that there are no non-trivial lower bounds known against Σ_2^{cc} protocols [94], which suggests that Σ_2^{cc} protocols ***could be very powerful***, and the first approach (Theorem 3.2.2) may be applicable to many other problems. This is not the case for $\mathsf{MA}^{\mathsf{cc}}$ protocols which were used in several previous works [12, 151, 106, 70]: for example, there is an essentially tight $\Omega(\sqrt{n})$ $\mathsf{MA}^{\mathsf{cc}}$ lower bound for Set-Disjointness [110, 2, 70]. These two frameworks are discussed in Section 3.2 in detail.

Equivalence between Partial Match and Approximate Nearest Neighbor Search. Our reductions are robust enough that they also hold in the data structure setting. In particular, consider the following two fundamental data structure problems:

- **Partial Match:** Preprocess a database $\mathcal{D}$ of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1,\star\}^d$, either report a point $x \in D$ matching all non-$\star$ characters in q or report that no x exists.

- **Approximate Nearest Neighbor Search (NNS) in ℓ_p space:** Preprocess a database $\mathcal{D}$ of n points from $\mathbb{R}^m$ such that, for all query point $x \in \mathbb{R}^m$, one can find a point $y \in \mathcal{D}$ such that $\|x-y\|_p \le (1+\varepsilon)\cdot \min_{z\in\mathcal{D}} \|x-z\|_p$.

Remark 3.1.2. *We remark that Partial Match is known to be equivalent to an online version of OV [13] (see also Section 3.8), and NNS in ℓ_p space is simply the online version of Bichrom.-ℓ_p-Closest-Pair.*

Partial Match has been studied extensively for decades (see e.g. Rivest's PhD thesis [149]). However, the algorithmic progress beyond trivial solutions (building a look-up table of size $2^{\Omega(d)}$, or trying all n points on each single query) have been quite limited. It is generally believed that it is intractable when d is large enough. Many unconditional lower bounds are known in the cell-probe model [130, 53, 103, 139, 141], but the gap between the best data structures [69, 76] and known lower bounds remains very large.

Approximate Nearest Neighbor Search has a wide range of applications in computing, including machine learning, computer vision, databases and others (see [31, 132] for an overview). Tremendous research effort has been devoted to this problem (see e.g. the recent survey of [32] and Razenshteyn's PhD thesis [146]). Yet all known algorithms exhibit a query time of at least $n^{1-O(\varepsilon)}$ when the approximation ratio is $1+\varepsilon$, approaching the brute-force query time n when ε goes to 0.

In general, whether there is a *polynomial* space, $n^{1-\delta}$ query time data structure for Partial Match for all $d = O(\log n)$, or Approximate NNS for all constant approximation ratio > 1 are two long-standing open questions.[7] We show these two questions are ***equivalent***.

Theorem 3.1.3. *The following are equivalent:*

- *There is a $\delta > 0$ such that for all constant c, there is a data structure for Partial Match with string length $d = c \log n$ that uses* $\operatorname{poly}(n)$ *space and allows $n^{1-\delta}$ query time.*
- *There is a $\delta > 0$ such that for all $\varepsilon > 0$, there is an data structure for Approximate NNS in ℓ_p with approximation ratio $(1+\varepsilon)$ that uses* $\operatorname{poly}(n)$ *space and allows $n^{1-\delta}$ query time, for some constant $p \in [1,2]$.*

A Tighter Connection between Max-IP, Bichrom. ℓ_p-Closest Pair and ℓ_p-Furthest Pair. For a subset of problems in Theorem 3.1.1, we can establish even tighter reductions.

[7] Under SETH, it is shown that there is no such data structure with *polynomial pre-processing time* [18, 168, 151].

The state-of-the-art algorithm for $(1+\varepsilon)$ approximation to $\mathsf{Bichrom.}\text{-}\ell_p\text{-}\mathsf{Closest\text{-}Pair}$ runs in $n^{2-\widetilde{O}(\varepsilon^{1/3})}$ time, and for $\mathsf{Max\text{-}IP}_{n,c\log n}$, the best running time $n^{2-\widetilde{O}(1/\sqrt{c})}$. Both algorithms are presented in [22], and relied on probabilistic threshold functions.

Comparing to the $n^{2-1/O(\log c)}$ time algorithm for $\mathsf{OV}_{n,c\log n}$ [13, 68], the dependence on c or ε in these two algorithms are much worse, rendering them useless when ε^{-1} or c are $\log^{\omega(1)} n$. So it is natural to ask whether the dependence can be improved to at least sub-polynomial in ε and c, i.e. $n^{2-1/c^{o(1)}}$ or $n^{2-\varepsilon^{o(1)}}$.

We show that a modest improvement on the running time dependence on ε or c for any of the following problems directly implies similar improvements for other problems as well.

Theorem 3.1.4. *The following are equivalent:*

- *An $\varepsilon \cdot d$ additive approximation to $\mathsf{Max\text{-}IP}_{n,d}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time.*
- *$\mathsf{Max\text{-}IP}_{n,c\log n}$ is solvable in $n^{2-1/c^{o(1)}}$ time.*
- *$\mathsf{Exact\text{-}IP}_{n,c\log n}$ is solvable in $n^{2-1/c^{o(1)}}$ time.*
- *A $(1+\varepsilon)$ approximation to $\mathsf{Bichrom.}\text{-}\ell_p\text{-}\mathsf{Closest\text{-}Pair}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time (for a constant $p \in [1,2]$).*
- *A $(1+\varepsilon)$ approximation to $\ell_p\text{-}\mathsf{Furthest\text{-}Pair}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time (for a constant $p \in [1,2]$).*

In [151] (Theorem 4.1), it is implicitly shown that $\mathsf{Exact\text{-}IP}_{n,c\log n}$ can be reduced to $(1 + 1/\exp(c))$ approximating $\mathsf{Bichrom.}\text{-}\ell_p\text{-}\mathsf{Closest\text{-}Pair}$. This suffices for the case when c is a constant (which is needed for Theorem 3.1.1), but falls short of proving the above tighter connections.

In a nutshell, [151]'s reduction applies a very efficient MA protocol for Set-Disjointness using AG-codes, and it uses "brute-force" gadgets to simulate an inner product between two short vectors in $\mathbb{F}_{q^2}$. We improve [151]'s reduction by carefully modifying its MA protocol, and replacing its brute-force gadgets by a more efficient one. Informally, our theorem shows $\mathsf{Exact\text{-}IP}_{n,c\log n}$ can be reduced to $(1 + 1/\operatorname{poly}(c))$ approximating

Bichrom.-Closest-Pair (see Lemma 3.7.4 and Lemma 3.7.7), which is an exponential improvement over the old reduction.

Equivalence Results in the Moderate Dimensional Setting. Theorem 3.1.1 establishes an equivalence class for the sparse $O(\log n)$ dimensional setting. It is natural to ask whether the equivalence continues to hold in the moderate dimensional case as well.

Unfortunately, an unusual (and interesting) property of our reduction used in Theorem 3.1.1 is that it blows up c (the constant before $\log n$) exponentially, and creates multiple instances. That is, an Exact-IP instance with $c \log n$ dimensions is reduced to *many* OV instances with $\exp(c) \log n$ dimensions (see the proof of Lemma 3.5.2). This renders the reduction useless in the moderate-dimensional setting, where c could be as large as n^{δ}.

Still, using different techniques, we obtain some additional equivalence results in the moderate dimensional setting. For a problem Π on Boolean vectors, we say that *moderate dimensional* Π *is in truly subquadratic time*, if there are two constants $\varepsilon, \delta > 0$ such that Π is solvable in $n^{2-\varepsilon}$ time on n vectors with n^{δ} dimensions.

Theorem 3.1.5. *Moderate dimensional OV is in truly subquadratic time if and only if moderate dimensional Apx-Min-IP is.*

Theorem 3.1.6. *For moderate dimensional Max-IP, Min-IP, and Exact-IP, either all of them are in truly subquadratic time, or none of them are.*

To show moderate dimensional OV and Apx-Min-IP are equivalent, we use a sophisticated reduction which is partially inspired by the classical Goldwasser-Sipser AM protocol for approximate counting [92] (see the proof of Lemma 3.6.1 for details). For Max-IP, Min-IP and Exact-IP, we apply some folklore encoding tricks.

It is an interesting open question that whether these two separate equivalence classes can be merged into one. In particular, *is moderate dimensional OV equivalent to moderate dimensional Max-IP?*

An immediate corollary of Theorem 3.1.5 is that it adds Apx-Min-IP as a new member to the equivalence class of moderate dimensional OV established in [89].

3.2 Techniques: Two General Frameworks for Establishing Equivalence with OV

In the following we discuss two general frameworks for reductions to OV.

3.2.1 Σ_2 Communication Protocols and Reductions to Orthogonal Vectors

Our first framework is based on Σ_2 communication protocols (Σ_2^{cc} protocols). We begin with a formal definition of such protocols.

Definition 3.2.1 (Σ_2^{cc} Protocol [38]). Let $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a function. A Σ_2^{cc} protocol Π for F is specified as follows:

- There are two players, Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$.
- There are two provers Merlin and Megan.
- Merlin sends a string $a \in \{0,1\}^{m_1}$ and Megan sends a string $b \in \{0,1\}^{m_2}$ (which are functions of both x and y) to both Alice and Bob. Then Alice and Bob communicate ℓ bits with each other, and Alice decides whether to accept or reject the pair (a,b).
- $F(x,y) = 1$ if and only if there exists a string a from Merlin, such that for all strings b from Megan, Alice accepts (a,b) after communications with Bob.

We say the protocol Π is computationally-efficient, if both Alice and Bob's response functions can be computed in polynomial time with respect to their input length.

We show that for any function F, if F admits a certain efficient Σ_2^{cc} protocol, then F-Satisfying-Pair can be efficiently reduced to OV. Formally, we have:

Theorem 3.2.2. *Let $F : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}$ and $n \in \mathbb{N}$, suppose F has a computationally-efficient Σ_2^{cc} protocol, in which Merlin sends m_1 bits, Megan sends*

m_2 bits, and Alice and Bob communicate ℓ bits. Then there is a reduction from every F-Satisfying-Pair$_n$ instance I to $\mathsf{OV}_{n,2^{(m_2+\ell)}}$ instances $J_1, J_2, \ldots, J_{2^{m_1}}$, such that I is a yes instance if and only if there is a j such that J_j is a yes instance. The reduction takes $n \cdot 2^{O(m_1+m_2+\ell)} \cdot \mathrm{poly}(d)$ time.

Applications. We use Theorem 3.2.2 to establish the equivalence between OV, Min-IP / Max-IP, Apx-Max-IP / Apx-Min-IP and Exact-IP. Previous works have established that OV can be reduced to all these problems, and that these problems can be reduced to Exact-IP. So it suffices for us to construct a reduction from Exact-IP to OV. Let the $\mathsf{IP}_{d,m} : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}$ be the function that checks whether $\langle x, y \rangle = m$, Exact-IP is $\mathsf{IP}_{d,m}$-Satisfying-Pair, so we can apply Theorem 3.2.2 with an efficient Σ_2^{cc} protocol for $\mathsf{IP}_{d,m}$.

Locality-sensitive Hashing (LSH) Families and Reductions to Additive Approximation to Max-IP

To establish equivalence between OV and other approximation problems, we make use of a connection with LSH families. We begin with a generalized definition of an LSH family for a partial function. In the following, let $\mathcal{X}$ be an arbitrary set.

Definition 3.2.3. Let $f : \mathcal{X} \times \mathcal{X} \to \{0, 1, \perp\}$[8]. We say f admits a (p_1, p_2)-sensitive LSH family, if there is a family $\mathcal{F}$ of functions $h : \mathcal{X} \to \mathcal{S}$, such that for any $x, y \in \mathcal{X}$, a uniformly random function $h \in \mathcal{F}$ satisfies:

- If $f(x, y) = 1$, then $h(x) = h(y)$ with probability at least p_1.
- If $f(x, y) = 0$, then $h(x) = h(y)$ with probability at most p_2.

In addition, we require that h can be efficiently drawn from $\mathcal{F}$, and $h(p)$ can be efficiently computed.[9]

The usual LSH families for a metric space are special cases of the above generalized definition.

[8] $f(x, y) = \perp$ means f is "undefined" on (x, y).

[9] Being efficient here means the running time is polynomial in the bit complexity of the input.

Definition 3.2.4. For a function $\mathsf{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, we say dist admits an LSH family, if for all $\varepsilon > 0$ and real $R > 0$, there are two reals $p_1 = p_1(\varepsilon)$ and $p_2 = p_2(\varepsilon)$ such that the function $f^{\mathsf{dist}}_{R,(1+\varepsilon)R} : \mathcal{X} \times \mathcal{X} \to \{0, 1, \perp\}$ defined as

$$f^{\mathsf{dist}}_{R,(1+\varepsilon)R}(x, y) = \begin{cases} 1 & \mathsf{dist}(x, y) \leq R, \\ 0 & \mathsf{dist}(x, y) \geq (1 + \varepsilon) \cdot R, \\ \perp & \text{otherwise}, \end{cases}$$

admits a (p_1, p_2)-sensitive LSH family and $p_1 > p_2$.

In particular, we show that an LSH family for a function implies a reduction to additively approximating Max-IP, which can in turn be reduced to OV. To formally state our reduction, we need to define $\mathcal{F}$-Satisfying-Pair for a partial function $\mathcal{F}$.

Definition 3.2.5. For a partial function $\mathcal{F} : \mathcal{X} \times \mathcal{X} \to \{0, 1, \perp\}$, $\mathcal{F}$-Satisfying-Pair$_n$ is the problem: given two sets $A, B \subseteq \mathcal{X}$ of size n, distinguish between the two cases:

- There is an $(x, y) \in A \times B$ such that $\mathcal{F}(x, y) = 1$.
- For all $(x, y) \in A \times B$, $\mathcal{F}(x, y) = 0$.

Remark 3.2.6. *Let $\mathcal{X}$ be $\mathbb{R}^d$, and set $\mathcal{F}(x, y) = 1$ for $\|x - y\| \leq R$, $\mathcal{F}(x, y) = 0$ for $\|x - y\| \geq (1 + \varepsilon) \cdot R$ and undefined otherwise. Then $\mathcal{F}$-Satisfying-Pair distinguishes between the cases that the minimum distance between A and B is $\leq R$ and $\geq (1+\varepsilon)\cdot R$, which is the decision version of $(1 + \varepsilon)$-approximation to Bichrom.-Closest-Pair.*

Now we are ready to state our general reduction.

Theorem 3.2.7. *Suppose $f : \mathcal{X} \times \mathcal{X} \to \{0, 1, \perp\}$ admits a (p_1, p_2)-sensitive LSH family. Let $\varepsilon = p_1 - p_2$.*

Then there is a randomized reduction from f-Satisfying-Pair$_n$ to computing an $\varepsilon/8 \cdot d$ additive approximation to Max-IP$_{n,d}$ with $d = O(\varepsilon^{-2} \log n)$, which succeeds with probability at least $1 - 1/n$.

From Theorem 3.2.7, reductions from Bichrom.-ℓ_2-Closest-Pair and FP to OV follows:

Corollary 3.2.8. *For a distance function dist* : $\mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ *which admits an LSH family, Bichrom.-Closest-Pair*$_{n,\mathsf{dist}}$ *and FP*$_{n,\mathsf{dist}}$ *can be approximated in truly subquadratic time if OV is in truly subquadratic time.*

Applications. We use Theorem 3.2.7 and Corollary 3.2.8 to establish the equivalence between OV and all approximation problems listed in Theorem 3.1.1. In particular, the ℓ_p metric and Jaccard Index admit efficient LSH families via p-stable distributions and the minHash method, which implies that they can be reduced to OV by Theorem 3.2.7.

3.3 Preliminaries

For notational convenience, we first give the formal definitions of the problem we study in this section.

3.3.1 Problem List

Definition 3.3.1 (Boolean Vector Problem List). For $n, d \in \mathbb{N}$, we define several problems. For all of them, the input is the same: we are given sets A and B of n vectors from $\{0,1\}^d$.

1. OV$_{n,d}$[10]: *Given* $A, B \subseteq \{0,1\}^d$ *with* $|A| = |B| = n$, *determine whether there exists* $(a,b) \in A \times B$ *such that* $a \cdot b = 0$.

2. Exact-IP$_{n,d}$: *Given* A, B *as before, and an integer* $0 \leq m \leq d$, *determine whether there exists* $(a,b) \in A \times B$ *such that* $a \cdot b = m$.

[10] Note that we consider the red-blue version of OV in this chapter for convenience, and it is equivalent to the original monochromatic version.

3. $\mathsf{Max\text{-}IP}_{n,d}$: *Given A, B as before, compute*

$$\mathsf{Max}(A, B) := \max_{a \in A, b \in B} a \cdot b.$$

4. $\mathsf{Min\text{-}IP}_{n,d}$: *Given A, B as before, compute*

$$\mathsf{Min}(A, B) := \min_{a \in A, b \in B} a \cdot b.$$

5. $\mathsf{Apx\text{-}Max\text{-}IP}_{n,d}$: *Given A, B as before, output a number $\widetilde{\mathsf{Max}}(A, B) \in [\mathsf{Max}(A, B)/2, \mathsf{Max}(A, B)]$.*

6. $\mathsf{Apx\text{-}Min\text{-}IP}_{n,d}$: *Given A, B as before, output a number $\widetilde{\mathsf{Min}}(A, B) \in [\mathsf{Min}(A, B), 2 \cdot \mathsf{Min}(A, B)]$.*

Remark 3.3.2. *The constant factor 2 in the definitions of Apx-Min-IP and Apx-Max-IP is only chosen for convenience, it can be replaced by any constant $\kappa > 1$ (such as 1.001, or 100).*

Definition 3.3.3 (Other Problems). We define the following problems.

1. $\mathsf{Bichrom.\text{-}}\ell_p\mathsf{\text{-}Closest\text{-}Pair}_n$: For a fixed real $p \in [1, 2]$, given two sets A, B of n points in $\mathbb{R}^d$ where $d = n^{o(1)}$, compute $\min_{(a,b) \in A \times B} \|a - b\|_p$.

2. $\ell_p\mathsf{\text{-}Furthest\text{-}Pair}_n$: For a fixed real $p \in [1, 2]$, given a set A of n points in $\mathbb{R}^d$ where $d = n^{o(1)}$, compute $\max_{(a,b) \in A \times A} \|a - b\|_p$.

3. $\mathsf{Jaccard\text{-}Index\text{-}Pair}_n$: Given A, B as two collections of n sets of size $n^{o(1)}$, compute $\max_{(S,T) \in A \times B} J(S, T)$, where $J(S, T) := \frac{|S \cap T|}{|S \cup T|}$.

3.3.2 Locality-sensitive Hashing

In this chapter we apply some well-known results from the theory of *locality-sensitive hashing* (LSH) (See [165, 32] for excellent recent references on LSH families and their applications).

ℓ_p Norm. From the theory of p-stable distributions, LSH families for ℓ_p norm when $p \in [1,2]$ have been constructed.

Lemma 3.3.4 ([82]). *For a constant $p \in [1,2]$, the ℓ_p distance $\mathsf{dist}_p(x,y) := \|x-y\|_p$ admits a LSH family. Moreover, for all real $\varepsilon \in (0, 0.1)$ and real $R > 0$, $f^{\mathsf{dist}_p}_{R,(1+\varepsilon)R}$ admits a (p_1, p_2)-sensitive LSH family, such that $p_1 - p_2 \geq \Omega(\varepsilon)$.*

Jaccard Index. For two sets A, B, recall that their Jaccard index is defined as $J(A,B) := \frac{|A \cap B|}{|A \cup B|}$. It is well-known that this measure admits a LSH family by the MinHash method.

Lemma 3.3.5 ([58]). *Let $0 \leq p_2 < p_1 \leq 1$ be two reals, and f be the function on two sets such that $f(A,B) = 1$ when $J(A,B) \geq p_1$, $f(A,B) = 0$ when $J(A,B) \leq p_2$ and undefined otherwise. f admits a (p_1,p_2)-sensitive LSH family.*

3.4 General Reduction Frameworks with Σ_2 Communication Protocols and LSH Families

In this section we present two general reduction frameworks for showing equivalence to OV.

3.4.1 Σ_2 Communication Protocols and Reductions to OV

We first show that an efficient Σ_2^{cc} protocol for a function f implies a reduction from f-$\mathsf{Satisfying\text{-}Pair}$ to OV.

Reminder of Theorem 3.2.2 *Let $F : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}$ and $n \in \mathbb{N}$, suppose F has a computationally-efficient Σ_2^{cc} protocol, in which Merlin sends m_1 bits, Megan sends m_2 bits, and Alice and Bob communicate ℓ bits. Then there is a reduction from every F-$\mathsf{Satisfying\text{-}Pair}_n$ instance I to $\mathsf{OV}_{n,2^{(m_2+\ell)}}$ instances $J_1, J_2, \ldots, J_{2^{m_1}}$, such that I is a yes instance if and only if there is a j such that J_j is a yes instance. The reduction takes $n \cdot 2^{O(m_1+m_2+\ell)} \cdot \operatorname{poly}(d)$ time.*

Proof of Theorem 3.2.2. Let F be the given function and Π be its Σ_2 protocol. Fix $a \in \{0,1\}^{m_1}$ and $b \in \{0,1\}^{m_2}$ as the proofs from Merlin and Megan. Let $w_1, w_2, \dots, w_{2^\ell}$ be an enumeration of all possible communication transcripts between Alice and Bob (note they communicate ℓ bits). We define two binary vectors $R_x(a,b), R_y(a,b) \in \{0,1\}^{2^\ell}$ as follows: for all a, b, $R_x(a,b)_i = 1$ ($R_y(a,b)_i = 1$) if and only if the transcript w_i is consistent with Alice's input x (Bob's input y), and w_i makes Alice reject. Note that since the transcript is uniquely determined by x, y, a and b, only one w_i is consistent with both x and y given the pair (a,b). It follows that $\langle R_x(a,b), R_y(a,b)\rangle = 0$ if and only if Alice accepts the pair (a,b).

Now, suppose we are given an F-$\mathsf{Satisfying\text{-}Pair}_n$ instance I with sets A and B of n vectors from $\{0,1\}^d$. We first enumerate Merlin's possible string $a \in \{0,1\}^{m_1}$, and use $R_x(a,\cdot)$ to denote the string obtained by concatenating all $R_x(a,b)$'s for $b \in \{0,1\}^{m_2}$. $R_y(a,\cdot)$ is defined similarly. For each a, let A_a be the set of $R_x(a,\cdot) \in \{0,1\}^{m_2+\ell}$ for all $x \in A$, and B_a be the set of $R_y(a,\cdot) \in \{0,1\}^{m_2+\ell}$ for all $y \in B$.

We claim I is a yes instance if and only if some pair (A_a, B_a) is a yes instance for OV.

- Suppose I is a yes instance. Then there is an $(x,y) \in A \times B$ such that $F(x,y) = 1$. By the definition of Σ_2^{cc} protocols and our constructions, there is an $a \in \{0,1\}^{m_1}$ such that for all $b \in \{0,1\}^{m_2}$ we have $\langle R_x(a,b), R_y(a,b)\rangle = 0$. Hence, for such an a, $\langle R_x(a,\cdot), R_y(a,\cdot)\rangle = 0$, and therefore (A_a, B_a) is a yes instance for OV.

- Suppose I is a no instance. Then for all $(x,y) \in A \times B$, $F(x,y) = 0$. Hence, for all $a \in \{0,1\}^{m_1}$ and all $(x,y) \in A \times B$, we have $\langle R_x(a,\cdot), R_y(a,\cdot)\rangle \neq 0$, which means all (A_a, B_a)'s are no instances for OV.

Finally, since Π is computationally-efficient, the above reduction takes $O(n \cdot 2^{O(m_1+m_2+\ell)} \cdot \mathrm{poly}(d))$ time, which completes the proof. □

3.4.2 LSH Families and Reductions to Additive Approximate Max-IP

Next, we show that an efficient LSH family implies a reduction to additively approximating Max-IP.

Reminder of Theorem 3.2.7 *Suppose $f : \mathcal{X} \times \mathcal{X} \to \{0, 1, \perp\}$ admits a (p_1, p_2)-sensitive LSH family. Let $\varepsilon = p_1 - p_2$.*

Then there is a randomized reduction from f-Satisfying-Pair$_n$ to computing an $\varepsilon/8 \cdot d$ additive approximation to Max-IP$_{n,d}$ with $d = O(\varepsilon^{-2} \log n)$, which succeeds with probability at least $1 - 1/n$.

Proof. Let $\mathcal{F}$ be the corresponding (p_1, p_2)-sensitive LSH family, and $\mathcal{S}$ be the co-domain for hash functions from $\mathcal{F}$. Consider the following process: draw h from $\mathcal{F}$ uniformly at random, then map each item in $\mathcal{S}$ independently to the string $(0, 1)$ or $(1, 0)$, each with probability 0.5. Let this map be φ. Composing h and φ, we obtain a function $g(x) = \varphi(h(x))$ such that:

- If $f(x, y) = 1$, then $\langle g(x), g(y) \rangle = 1$ with probability at least $p_1 + (1 - p_1)/2 \geq \frac{1}{2} + \frac{1}{2} \cdot p_1$.
- If $f(x, y) = 0$, then $\langle g(x), g(y) \rangle = 1$ with probability at most $p_2 + (1 - p_2)/2 \leq \frac{1}{2} + \frac{1}{2} \cdot p_2$.

Repeat the above process for $N = c \log n$ times, independently drawing functions $g_1, g_2, \ldots, g_N$, where c is a parameter to be specified later. We set our reduction $w(x)$ to be the concatenation of all $g_i(x)$'s. Let $\tau_1 = \frac{1}{2} + \frac{1}{2} \cdot (p_1 - \varepsilon/4)$ and $\tau_2 = \frac{1}{2} + \frac{1}{2} \cdot (p_2 + \varepsilon/4)$. By a simple Chernoff bound, there is a real $c_1 = \Theta(\varepsilon^2)$ such that

- If $f(x, y) = 1$, then $\langle w(x), w(y) \rangle > \tau_1 \cdot N$ with probability at least $1 - 2^{c_1 \cdot N}$.
- If $f(x, y) = 0$, then $\langle w(x), w(y) \rangle < \tau_2 \cdot N$ with probability at least $1 - 2^{c_1 \cdot N}$.

Set $c := 3/c_1$, and let A_{new} (respectively, B_{new}) be the set of $w(a)$'s for all $a \in A$ (the set of $w(b)$'s for all $b \in B$). It follows that with probability at least $1 - 1/n$, if there is an $(x, y) \in A \times B$ with $f(x, y) = 1$ then $\mathsf{Max}(A_{\mathsf{new}}, B_{\mathsf{new}}) > \tau_1 \cdot N$, and if $f(x, y) = 0$ for all $(x, y) \in A \times B$, then $\mathsf{Max}(A_{\mathsf{new}}, B_{\mathsf{new}}) < \tau_2 \cdot N$. Observe this reduction satisfies the desired approximation property. □

3.5 An Equivalence Class for Orthogonal Vectors

In this section we apply our two general frameworks to prove Theorem 3.1.1.

3.5.1 Equivalence between Boolean Vectors Problems

We first show that all Boolean vectors problems listed in Theorem 3.1.1 can be trivially reduced to Exact-IP, and OV can be reduced to all of them.

Lemma 3.5.1. *The following holds:*

- *If Exact-IP is in truly subquadratic time, then so are OV, Apx-Min-IP (Apx-Max-IP) and Max-IP (Min-IP).*
- *If any of Apx-Min-IP (Apx-Max-IP), Max-IP (Min-IP) and Exact-IP is in truly subquadratic time, then so is OV.*

Proof. For the first item, Apx-Min-IP (Apx-Max-IP) and Max-IP (Min-IP) can all be trivially reduced to Exact-IP, and OV can be reduced to Max-IP by [168].

For the second item, the case of Apx-Max-IP follows from Theorem 4.1 in [151], and it is easy to see that OV can be trivially reduced to Min-IP or Apx-Min-IP (OV is equivalent to asking whether the minimum inner product is zero). □

Therefore, all we need is a reduction from Exact-IP to OV. We provide it by constructing a good Σ_2 communication protocol, and applying Theorem 3.2.2.

Lemma 3.5.2. *If OV is in truly subquadratic time, then so is Exact-IP.*

Proposition 3.5.3. *Let $\mathsf{IP}_{n,k} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the function that checks whether $\langle x, y \rangle = k$. For all $n, k \in \mathbb{Z}^+$, and a parameter $1 \le \ell \le n$, there is a Σ_2^{cc} computationally-efficient protocol for $\mathsf{IP}_{n,k}$ in which Merlin sends $\ell \cdot \lceil \log(\lceil n/\ell \rceil + 1) \rceil$ bits, Megan sends $\lceil \log \ell \rceil$ bits and Alice and Bob communicate $\lceil n/\ell \rceil$ bits.*

Proof. We assume ℓ divides n for simplicity. Let x, y be the inputs of Alice and Bob, respectively. We partition x into ℓ equally-sized groups of length n/ℓ, let them be $x_1, x_2, \ldots, x_\ell$. Similarly, we partition y into groups $y_1, y_2, \ldots, y_\ell$. Clearly, $\langle x, y\rangle = \sum_{i=1}^{\ell}\langle x_i, y_i\rangle$.

Merlin's message is a vector $\psi \in \{0, 1, \ldots, n/\ell\}^\ell$, where ψ_i is intended to be $\langle x_i, y_i\rangle$.

Alice rejects immediately if $\sum_{i=1}^{\ell}\psi_i \neq k$, regardless of Megan's message. Otherwise, Megan's message is an index i in $[\ell]$. Bob sends y_i to Alice, and Alice accepts if and only if $\langle x_i, y_i\rangle = \psi_i$.

We argue the protocol correctly decides $\mathsf{IP}_{n,k}$. If $\langle x, y\rangle = k$, it is easy to see that for the correct ψ, Alice accepts all messages from Megan (and Bob). When $\langle x, y\rangle \neq k$, for all ψ such that $\sum_{i=1}^{\ell}\psi_i = k$ (otherwise Alice always rejects), there must be an i such that $\langle x_i, y_i\rangle \neq \psi_i$, which means Alice rejects on the pair ψ and i. Finally, it is easy to see that the protocol satisfies the requirements of computational efficiency, which completes the proof. □

Now we are ready to prove Lemma 3.5.2.

Proof of Lemma 3.5.2. Suppose there is a universal constant $\delta > 0$ such that for all constants c', $\mathsf{OV}_{n,c'\log n}$ can be solved in $n^{2-\delta}$ time. Let c be an arbitrary constant.

Observe that an $\mathsf{Exact\text{-}IP}_{n,c\log n}$ instance with target integer m, is simply a $\mathsf{IP}_{c\log n,m}$ -$\mathsf{Satisfying\text{-}Pair}_n$ instance. Set $\ell := \varepsilon \cdot \log n$ for an $\varepsilon > 0$ to be specified later. By Proposition 3.5.3, there is a Σ_2^{cc} protocol for $\mathsf{IP}_{c\log n,m}$ such that Merlin sends $\varepsilon \cdot \log(c/\varepsilon) \cdot \log n$ bits, Megan sends $\log(\varepsilon \log n)$ bits and Alice and Bob communicate c/ε bits.

By Theorem 3.2.2, there is a reduction from an $\mathsf{Exact\text{-}IP}_{n,c\log n}$ instance to $2^{\varepsilon\log(c/\varepsilon)\log n} = n^{\varepsilon\log(c/\varepsilon)}$ many $\mathsf{OV}_{n,O(2^{c/\varepsilon}\log n)}$ instances. We can set ε so that $\varepsilon\log(c/\varepsilon) < \delta/2$. Note that ε only depends on c and δ, so it is still a fixed constant, which means (by assumption) that $\mathsf{OV}_{n,O(2^{c/\varepsilon}\log n)}$ can be solved in $n^{2-\delta}$ time. Applying the algorithm for OV, we get an $n^{2-\delta/2}$ time algorithm for $\mathsf{Exact\text{-}IP}_{n,c\log n}$, which completes the proof. □

3.5.2 Equivalence between OV and Approximation Problems

Now we deal with approximation problems in Theorem 3.1.1.

Bichrom.-ℓ_p-Closest-Pair and ℓ_p-Furthest-Pair

We first show OV is equivalent to approximate Bichrom.-ℓ_p-Closest-Pair, ℓ_p-Furthest-Pair and additive approximate Max-IP. One direction is already established in [151].

Lemma 3.5.4 (Theorem 4.1 of [151]). *If Bichrom.-ℓ_p-Closest-Pair or ℓ_p-Furthest-Pair can be approximated in truly subquadratic time for any $p \in [1, 2]$ or Max-IP can be additively approximated in truly subquadratic time, then OV is in truly subquadratic time.*[11]

In the following we show the reverse also holds.

Lemma 3.5.5. *If OV is in truly-subquadratic time, then for all $p \in [1, 2]$, Bichrom.-ℓ_p -Closest-Pair and ℓ_p-Furthest-Pair can be approximated in truly subquadratic time, and Max-IP can be additively approximated in truly subquadratic time.*

We are going to apply Theorem 3.2.7 and will actually prove a much stronger result. We show that for any metric $\mathsf{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ which admits a *Locality-sensitive hashing* (LSH) family, approximate Bichrom.-Closest-Pair and FP with respect to dist can be efficiently reduced to OV.

In the following, we use Bichrom.-Closest-Pair$_{n,\mathsf{dist}}$ and FP$_{n,\mathsf{dist}}$ to denote the corresponding problems with respect to the metric dist. Now we are ready to give the reduction.

Reminder of Corollary 3.2.8 *For a distance function dist : $\mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ which admits an LSH family, Bichrom.-Closest-Pair$_{n,\mathsf{dist}}$ and FP$_{n,\mathsf{dist}}$ can be approximated in truly subquadratic time if OV is in truly subquadratic time.*

Proof. Suppose OV is in truly subquadratic time. By Lemma 3.5.1 and Lemma 3.5.2, Max-IP and Min-IP are also in truly-subquadratic time. In the following we only

[11] [151] only discussed Bichrom.-ℓ_p-Closest-Pair and additive approximation to Max-IP, but it is easy to see that the proof also works for ℓ_p-Furthest-Pair.

discuss Bichrom.-Closest-Pair$_{n,\mathsf{dist}}$; the reduction for FP$_{n,\mathsf{dist}}$ is analogous (with Min-IP in place of Max-IP).

Let $\varepsilon > 0$ be an arbitrary constant. We want to approximate the minimum distance between two sets A and B of n elements from $\mathcal{X}$ within a $(1+\varepsilon)$ multiplicative factor. By a standard (simple) search to decision reduction that incurs only a negligible factor in the running time, we only have to consider the decision version, in which you are given a real R, and want to distinguish the following two cases: (1) $\min_{(a,b)\in A\times B} d(a,b) \le R$; (2) $\min_{(a,b)\in A\times B} d(a,b) \ge (1+\varepsilon)\cdot R$.

By Theorem 3.2.7, this decision problem can be reduced to additive approximation to Max-IP$_{n,O(\log n)}$, which is in truly-subquadratic time by Lemma 3.5.2. This completes the proof. □

Now, from the LSH families for ℓ_p-metric, Lemma 3.5.5 follows directly.

Proof of Lemma 3.5.5. Assume OV is in truly-subquadratic time. It follows directly from Corollary 3.2.8 and Lemma 3.3.4 that for all $p \in [1,2]$, Bichrom.-ℓ_p-Closest-Pair and ℓ_p-Furthest-Pair can be approximated in truly subquadratic time.

Also, by a simple random sampling method and a Chernoff bound (see e.g. Lemma 3.6 of [70]), computing an $\varepsilon \cdot d$ additive approximation to Max-IP$_{n,d}$ can be reduced to Max-IP$_{n,O(\varepsilon^{-2}\log n)}$, which can be solved in truly-subquadratic time by Lemma 3.5.2 and Lemma 3.5.1. □

Jaccard-Index-Pair

Finally, we show the equivalence between OV and approximate Jaccard-Index-Pair.

Lemma 3.5.6. *OV is in truly-subquadratic time if and only if Jaccard-Index-Pair can be additively approximated in truly-subquadratic time.*

Proof. For one direction, suppose OV is in truly subquadratic time. Using a similar argument as in Corollary 3.2.8, from Lemma 3.3.5 and Theorem 3.2.7 it follows that Jaccard-Index-Pair can be additively approximated in truly-subquadratic time.

For the other direction, suppose Jaccard-Index-Pair can be additively approximated in truly subquadratic time. By Lemma 3.5.4, it suffices to show that Max-IP can be additively approximated in truly-subquadratic time. Given a $\mathsf{Max\text{-}IP}_{n,d}$ instance with sets A, B consisting of n vectors from $\{0,1\}^d$, suppose we want to compute an $\varepsilon \cdot d$ approximation to it. In the following we show how to reduce it to a Jaccard-Index-Pair instance.

We begin by setting up some notation. For $t \in [d]$, we use $e^{[t]}$ to denote the Boolean vector $1^t 0^{d-t}$ from $\{0,1\}^d$ (that is, the first t coordinates are 1, and the rest are 0). For two vectors a, b, we use $a \circ b$ to denote their concatenation.

For each $x \in A \subseteq \{0,1\}^d$ and $y \in B \subseteq \{0,1\}^d$, we create two vectors $\widehat{x}, \widehat{y} \in \{0,1\}^{3d}$, as follows:

$$\widehat{x} = x \circ e^{[d-\|x\|_1]} \circ e^{[0]}, \widehat{y} = y \circ e^{[0]} \circ e^{[d-\|y\|_1]}.$$

Interpreting $\widehat{x}$ and $\widehat{y}$ as indicator vectors, we create their corresponding sets $S_x, T_y \subseteq [3d]$. That is, for $i \in [3d]$, $\widehat{x}_i = 1$ if and only if $i \in S_x$ (the same holds for $\widehat{y}$ and T_y). Observe that

$$J(S_x, T_y) = \frac{|S_x \cap T_y|}{|S_x \cup T_y|} = \frac{\langle x, y \rangle}{2d - \langle x, y \rangle}. \tag{3.1}$$

Now we create $\widehat{A}$ and $\widehat{B}$ as the sets of all S_x for $x \in A$ and T_y for $y \in B$. Let $t = \max_{(S,T) \in \widehat{A} \times \widehat{B}} J(S,T)$ and $w = \max_{(a,b) \in A \times B} \langle a, b \rangle$. From Equation (3.1), we can see $t = \frac{w}{2d-w}$ and $w = d \cdot 2 \cdot \frac{t}{t+1}$. Therefore, an $\varepsilon/3$ approximation to t is enough to obtain an $\varepsilon \cdot d$ approximation to w, which completes the reduction. □

And Theorem 3.1.1 follows from Lemma 3.5.1, Lemma 3.5.2, Lemma 3.5.4, Lemma 3.5.5 and Lemma 3.5.6.

3.6 Equivalence between Moderate Dimensional Problems

In this section we prove our equivalence theorems for moderate dimensional Boolean vectors problems.

3.6.1 OV and Apx-Min-IP

We first show moderate dimensional OV and Apx-Min-IP are equivalent.

Reminder of Theorem 3.1.5 *Moderate dimensional OV is in truly subquadratic time if and only if moderate dimensional Apx-Min-IP is.*

To prove Theorem 3.1.5, we construct the following reduction.

Lemma 3.6.1. *For all integers n, d and a parameter $\varepsilon > 0$, an $\mathsf{Apx\text{-}Min\text{-}IP}_{n,d}$ instance can be reduced to $n^{O(\varepsilon)}$ $\mathsf{OV}_{n,d^{O(1/\varepsilon)}\log n}$ instances. The reduction is randomized and succeeds with probability at least $2/3$, and it takes $n^{1+O(\varepsilon)} \cdot d^{O(1/\varepsilon)}$ time.*

Before proving Lemma 3.6.1, we show it implies Theorem 3.1.5.

Proof of Theorem 3.1.5. Recall that $\mathsf{Min}(A, B) := \min_{(a,b)\in A\times B}\langle a, b\rangle$. For the first direction, note that OV with two sets A and B essentially asks whether $\mathsf{Min}(A, B) = 0$, and a 2-approximation to $\mathsf{Min}(A, B)$ is already enough to answer that question. Therefore, if moderate dimensional Apx-Min-IP is in truly subquadratic time, then so is OV.

For the second direction, suppose there are constants $\varepsilon_1, \delta_1 > 0$ such that $\mathsf{OV}_{n,n^{\delta_1}}$ can be solved in $n^{2-\varepsilon_1}$ time. Let ε be a parameter to be set later, by Lemma 3.6.1, there are constants c_1, c_2 such that all $\mathsf{Apx\text{-}Min\text{-}IP}_{n,n^{\delta}}$ instance can be efficiently reduced to $n^{c_1\varepsilon}$ $\mathsf{OV}_{n,n^{\delta c_2/\varepsilon}}$ instances.

We set ε such that $c_1\varepsilon = \varepsilon_1/2$, and δ such that $\delta \cdot c_2/\varepsilon < \delta_1$. Then applying the algorithm for OV, $\mathsf{Apx\text{-}Min\text{-}IP}_{n,n^{\delta}}$ can be solved in $n^{2-\varepsilon_1/2}$ time, which completes the proof. □

The following probability inequality will be useful in the proof of Lemma 3.6.1.

Lemma 3.6.2. *Letting $\varepsilon \in (0, 0.1)$, and $\mathcal{U}$ be a distribution on $\{0,1\}$ such that $\mathbb{E}_{X \sim \mathcal{U}}[X] = \varepsilon$, there is a universal constant c such that for any integer m and any cm independent random variables $X_1, X_2, \ldots, X_{cm}$ from $\mathcal{U}$, we have*

$$\Pr\left[\sum_{i=1}^{cm} X_i \geq \frac{1}{2} \cdot cm\right] \leq \varepsilon^{-m}.$$

The proof of Lemma 3.6.2 can be found in Section 3.9.

Finally, we prove Lemma 3.6.1.

Proof of Lemma 3.6.1. Before presenting the reduction, we first introduce some notation. For a vector $x \in \{0,1\}^d$, and a subset $S \subset [d]$, $x_{|S} \in \{0,1\}^{|S|}$ denotes the projection of x onto the coordinates of S. Similarly, for a sequence T of integers from $[d]$, let $x_{|T} \in \{0,1\}^{|T|}$ denote the projection of x on T, such that $\left(x_{|T}\right)_i := x_{T_i}$ for each $i \in [|T|]$. We also use the Iverson bracket notation: for a predicate P, $[P]$ takes value 1 when P is true, and 0 otherwise.

Reduction to a Decision Problem. Our reduction will focus on a corresponding decision problem: given two sets A, B of n vectors from $\{0,1\}^d$ and an integer $\tau \leq d/2$, we want to distinguish the following two cases: $\mathsf{Min}(A, B) \geq 2\tau$ or $\mathsf{Min}(A, B) \leq \tau$ (the algorithm can output anything when $\tau < \mathsf{Min}(A, B) < 2\tau$). It is easy to see that via a binary search, $\log d$ calls to this decision problem can be used to solve the original Apx-Min-IP problem, and a factor of $\log d \leq \log n$ can be ignored here.

One Step Reduction with $\mathcal{U}_T$. Now, suppose we pick a sequence of d/τ uniform random numbers from $[d]$ and let $\mathcal{U}_T$ be its distribution. Then for $x, y \in \{0,1\}^d$, we have:

- If $\langle x, y \rangle \le \tau$:

$$\Pr_{T \leftarrow \mathcal{U}_T}[\langle x_{|T}, y_{|T} \rangle = 0] \ge (1 - \tau/d)^{d/\tau} \ge \left(1 - \frac{1}{2}\right)^2$$
$$> 0.25.$$

- If $\langle x, y \rangle \ge 2\tau$:

$$\Pr_{T \leftarrow \mathcal{U}_T}[\langle x_{|T}, y_{|T} \rangle = 0] \le (1 - 2\tau/d)^{d/\tau} \le e^{-2} < 0.14.$$

The important observation is that there is a constant probability gap between the above two cases.

A Micro Reduction to OV. Now, let N be an integer and $\mathcal{U}_T^{\otimes N}$ be the joint distribution of N independent samples from $\mathcal{U}_T$. We write $\{T_i\} \leftarrow \mathcal{U}_T^{\otimes N}$ to denote that $(T_1, T_2, \dots, T_N)$ is a random sample from $\mathcal{U}_T^{\otimes N}$. By a standard Chernoff bound, when $\{T_i\} \leftarrow \mathcal{U}_T^{\otimes N}$, there is a constant c_1 such that:

- If $\langle x, y \rangle \le \tau$:

$$\Pr\left[\sum_{i=1}^{N} \left[\langle x_{|T_i}, y_{|T_i} \rangle = 0\right] > 0.2N\right] \ge 1 - 2^{-c_1 N}.$$

- If $\langle x, y \rangle \ge 2\tau$:

$$\Pr\left[\sum_{i=1}^{N} \left[\langle x_{|T_i}, y_{|T_i} \rangle = 0\right] < 0.2N\right] \ge 1 - 2^{-c_1 N}.$$

Now, for a fixed $\{T_i\}$, we can distinguish the above two cases via a reduction to a "micro" OV instance.

Note that $\sum_{i=1}^{N} \left[\langle x_{|T_i}, y_{|T_i} \rangle = 0\right] > 0.2N$ is equivalent to the condition that there is are $t = 0.8N$ pairs $(i_1, j_1), (i_2, j_2), \dots, (i_t, j_t) \in [N] \times [d/\tau]$ such that all i_k's are distinct, and for all $k \in [t]$, $\left(x_{|T_{i_k}}\right)_{j_k} \cdot \left(y_{|T_{i_k}}\right)_{j_k} = 1$.

With this observation, we can construct our reduction. There are

$$L = \binom{N}{t} \cdot (d/\tau)^t = (d/\tau)^{O(N)}$$

possible t-tuples of pairs. We sort them in an arbitrary but consistent order. Now we construct a mapping $\phi_{\{T_i\}} : \{0,1\}^d \to \{0,1\}^L$ as follows:

For each $\ell \in [L]$, let $(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)$ be the ℓ-th t-tuple of pairs. For a vector $z \in \{0,1\}^d$, we set $\phi_{\{T_i\}}(z)_\ell = 1$, iff $\left(z_{|T_{i_k}}\right)_{j_k} = 1$ for all $k \in [t]$.

Then for all $x, y \in \{0,1\}^d$, we have $\sum_{i=1}^{N} \left[\langle x_{|T_i}, y_{|T_i}\rangle = 0\right] > 0.2N$ is further equivalent to $\langle \phi_{\{T_i\}}(x), \phi_{\{T_i\}}(y)\rangle = 0$. For convenience, we let $\mathcal{U}_\phi$ denote the distribution of $\phi_{\{T_i\}}$ when $\{T_i\}$ is drawn from $\mathcal{U}_T^{\otimes N}$ and we set $N = \varepsilon^{-1}/c_1$.

To summarize, we have:

- If $\langle x, y\rangle \le \tau$:

$$\Pr_{\phi \leftarrow \mathcal{U}_\phi}\left[\langle \phi(x), \phi(y)\rangle = 0\right] \ge 1 - 2^{-\varepsilon^{-1}}.$$

- If $\langle x, y\rangle \ge 2\tau$:

$$\Pr_{\phi \leftarrow \mathcal{U}_\phi}\left[\langle \phi(x), \phi(y)\rangle > 0\right] \ge 1 - 2^{-\varepsilon^{-1}}.$$

The Final Reduction. Finally, letting c_2 be the universal constant in Lemma 3.6.2, we pick $m = 3c_2 \cdot \varepsilon \log n$ i.i.d. mappings $\phi_1, \phi_2, \ldots, \phi_m$ from $\mathcal{U}_\phi$. Applying Lemma 3.6.2, we have:

- If $\langle x, y\rangle \le \tau$:

$$\Pr_{\{\phi_i\} \leftarrow \mathcal{U}_\phi^{\otimes m}}\left[\sum_{i=1}^{m} \left[\langle \phi_i(x), \phi_i(y)\rangle = 0\right] > \frac{1}{2} \cdot m\right] \ge 1 - n^{-3}.$$

- If $\langle x, y\rangle \ge 2\tau$:

$$\Pr_{\{\phi_i\} \leftarrow \mathcal{U}_\phi^{\otimes m}}\left[\sum_{i=1}^{m} \left[\langle \phi_i(x), \phi_i(y)\rangle = 0\right] < \frac{1}{2} \cdot m\right] \ge 1 - n^{-3}.$$

Now, we use our final reduction to distinguish the above two cases. Note that $\sum_{i=1}^{m} [\langle \phi_i(x), \phi_i(y) \rangle = 0] > \frac{1}{2} \cdot m$ is equivalent to the condition that there is a subset $S \subseteq [m]$ with $|S| > \frac{1}{2} \cdot m$ such that $\langle \phi_i(x), \phi_i(y) \rangle = 0$ for all $i \in S$.

We enumerate all possible such subsets S. For a vector $z \in \{0,1\}^d$, we define $\phi_S(z)$ to be the concatenation of $\phi_i(z)$'s for all $i \in S$. We set A_S as the set of all $\phi_S(x)$'s for $x \in A$, and B_S as the set of all $\phi_S(y)$'s for $y \in B$.

Then we can see that $\sum_{i=1}^{m} [\langle \phi_i(x), \phi_i(y) \rangle = 0] > \frac{1}{2} \cdot m$ is further equivalent to whether there is a subset S with $|S| > \frac{1}{2} \cdot m$ and (A_S, B_S) is a yes instance for OV.

Summary. Putting everything together, we have a randomized reduction to $T = 2^{O(\varepsilon \log n)} = n^{O(\varepsilon)}$ $OV_{n,(d/\tau)^{O(1/\varepsilon)} \log n}$ instances with set-pairs $(A_1, B_1), (A_2, B_2), \ldots, (A_T, B_T)$ such that, with probability at least $1 - 1/n$:

- If $\mathsf{Min}(A, B) \le \tau$, then one of the (A_i, B_i) is a yes instance for OV.
- If $\mathsf{Min}(A, B) \ge 2\tau$, all (A_i, B_i)'s are no instance for OV.

The above completes the proof. □

3.6.2 Exact-IP, Max-IP, and Min-IP

Now we proceed to show moderate dimensional Exact-IP, Max-IP and Min-IP are equivalent.

Reminder of Theorem 3.1.6 *For moderate dimensional Max-IP, Min-IP and Exact-IP, either all of them are in truly subquadratic time, or none of them are.*

To prove the above theorem, we need the following two simple reductions, whose proofs can be found in Section 3.9.

Lemma 3.6.3. *There are functions $\psi^x_{\mathsf{rev}}, \psi^y_{\mathsf{rev}} : \{0,1\}^* \to \{0,1\}^*$ such that for all integer d and $x, y \in \{0,1\}^d$, we have $\psi^x_{\mathsf{rev}}(x), \psi^y_{\mathsf{rev}}(y) \in \{0,1\}^{2d}$ and $\langle \psi^x_{\mathsf{rev}}(x), \psi^y_{\mathsf{rev}}(y) \rangle = d - \langle x, y \rangle$.*

Lemma 3.6.4. *For all integers d and $0 \le m \le d$, there are mappings $\varphi^x_{d,m}, \varphi^y_{d,m} : \{0,1\}^d \to \{0,1\}^{O(d^2)}$ and an integer M_d, such that for all $x, y \in \{0,1\}^d$:*

- *If $\langle x, y\rangle = m$, then $\langle \varphi^x_{d,m}(x), \varphi^y_{d,m}(y)\rangle = M_d$.*
- *Otherwise, $\langle \varphi^x_{d,m}(x), \varphi^y_{d,m}(y)\rangle > M_d$.*

Proof of Theorem 3.1.6. By Lemma 3.6.3, one can easily reduce a $\textsf{Max-IP}_{n,d}$ instance to a $\textsf{Min-IP}_{n,2d}$ and vice versa. Therefore, moderate dimensional Max-IP and Min-IP are truly-subquadratic equivalent. We only need to show that moderate dimensional Min-IP and Exact-IP are equivalent.

Assuming moderate dimensional Exact-IP is in truly subquadratic time, so there are two constants ε and δ such that $\textsf{Exact-IP}_{n,n^\delta}$ can be solved in $n^{2-\varepsilon}$ time. Let $\delta' = \min(\varepsilon, \delta)/2$. Given a $\textsf{Min-IP}_{n,n^{\delta'}}$ instance, by enumerating all possible inner products between 0 and $n^{\delta'}$, we can reduce the instance to $n^{\delta'}$ instances of $\textsf{Exact-IP}_{n,n^{\delta'}}$. Applying the algorithm for Exact-IP, we then have an $n^{2-\varepsilon+\delta'} \le n^{2-\delta'}$ time algorithm for $\textsf{Min-IP}_{n,n^{\delta'}}$. Hence, moderate dimensional Min-IP is also in truly-subquadratic time.

Finally, assume moderate dimensional Min-IP is in truly subquadratic time. Note that by Lemma 3.6.4, an $\textsf{Exact-IP}_{n,d}$ instance can be reduced to a $\textsf{Min-IP}_{n,O(d^2)}$ instance, which immediately implies that moderate dimensional Exact-IP is also in truly subquadratic time. □

3.7 A Tighter Connection between Max-IP, Bichrom.-ℓ_p-Closest-Pair, and ℓ_p-Furthest-Pair

In this section we establish the tighter connections between Max-IP, Bichrom.-ℓ_p-Closest-Pair and ℓ_p-Furthest-Pair.

In Section 3.7.1, we show tighter connections for Max-IP, Exact-IP and additive approximation to Max-IP. And in Section 3.7.2, we show similar connections for additive approximation to Max-IP, Bichrom.-ℓ_p-Closest-Pair and ℓ_p-Furthest-Pair.

3.7.1 A Tighter Connection between Exact-IP, Max-IP, and Additive Approximation to Max-IP

The following lemma is implicit in [151], which is used to show Bichrom.-ℓ_p-Closest-Pair can not be approximated in truly-subquadratic time under SETH. [151] only states a reduction from OV. However, the MA protocol in [151] works equally well for the Inner Product problem, so it actually gives a reduction from Exact-IP.

Lemma 3.7.1 (Implicit in Theorem 4.1 of [151]). *For all sufficiently large integers n, c and a parameter $\varepsilon > 0$, an Exact-$\mathsf{IP}_{n,c\log n}$ instance can be reduced to $n^{O(\varepsilon \log(c/\varepsilon))}$ instances of computing $\Omega(1/\exp\{\widetilde{O}(c/\varepsilon)\}) \cdot d$ additive approximation to Max-$\mathsf{IP}_{n,d}$ for $d = n^{o(1)}$.*

In order to prove our tighter connection, our goal here is to improve the additive approximation ratio from $\Omega(1/\exp\{\widetilde{O}(c/\varepsilon)\})$ to $\Omega(1/\operatorname{poly}(c/\varepsilon))$.

A New MA Protocol for Inner Product

For that purpose, we need to modify the MA protocol from [151]. In the following, we first describe the MA protocol for Inner Product in [151] based on AG codes. Below we only summarize the relevant properties we need; readers can refer to [151] for the details of the protocol.

Lemma 3.7.2 (Theorem 3.1 [151]). *For every $T \in [2, N]$, there is a computationally-efficient MA protocol for Inner Product such that*

1. *Alice and Bob hold input $x, y \in \{0,1\}^N$ respectively, and want to decide whether $\langle x, y\rangle = m$ for a target integer m.*

2. *Set q to be the first prime larger than T and a universal constant c_1, and set $R = \log(N/T) + O(1)$.*

3. *Merlin sends Alice a vector $z \in \mathbb{F}_{q^2}^{2^R}$, Alice rejects z immediately if it doesn't satisfy some conditions.*

4. *Alice and Bob then toss R coins to get $r \in [2^R]$. Based on x (or y) and r, Alice and Bob generate two vectors in $\mathbb{F}_{q^2}^T$, $\vec{a}(x,r)$ and $\vec{b}(y,r)$ respectively,*

5. *Bob sends Alice $\vec{b}(y,r)$, and Alice calculates $u(x,y,r) = \langle \vec{a}(x,r), \vec{b}(y,r) \rangle$. Alice accepts if and only if $u(x,y,r) = z_r$.*

The protocol satisfies the following conditions:

- *If $\langle x, y \rangle = m$, then there is a proof (the vector z) from Merlin such that Alice always accepts.*

- *If $\langle x, y \rangle \neq m$, then for all proofs from Merlin, Alice accepts with probability at most $1/2$.*

Our Modified Protocol. We make some minor modifications to the above protocol. First, note that an element from $\mathbb{F}_{q^2}$ can be treated as an element in $\mathbb{F}_q[x]/(P_{\mathsf{irred}}(x))$, where $P_{\mathsf{irred}}(x) \in \mathbb{F}_q[x]$ is an irreducible polynomial of degree 2. In this way, we can interpret all elements in $\vec{a}(x,r)$ and $\vec{b}(y,r)$ as degree 1 polynomials in $\mathbb{F}_q[x]$, which can in turn be interpreted as degree 1 polynomials in $\mathbb{Z}[x]$. We denote these vectors of polynomials by $\vec{U}(x,r), \vec{V}(y,r) \in \mathbb{Z}[x]^T$, with coefficients from $\{0, 1, \ldots, q-1\}$.

Next, we set $W(x,y,r) = \langle \vec{U}(x,r), \vec{V}(y,r) \rangle$, which is a degree 2 polynomial in $\mathbb{Z}[x]$. Note that the coefficients of $W(x,y,r)$ are between 0 and $O(q^2 \cdot T) = O(T^3)$.

Now, in the message from Merlin, for all possible $r \in [2^R]$, we also add a claimed description of $W(x,y,r)$. This takes $O\left(\frac{N \log T}{T}\right)$ bits, so it doesn't affect the message complexity from Merlin. Then, after Alice receives $\vec{b}(y,r)$ from Bob (from which she can obtain $\vec{V}(y,r)$), Alice computes $W(x,y,r)$ instead of $u(x,y,r)$, and rejects immediately if this $W(x,y,r)$ does not match the one given by Merlin. After that, she knows that $u(x,y,r) = W(x,y,r)/(P_{\mathsf{irred}}(x))$, and proceeds as in the original protocol.

It is easy to see that, when $\langle x, y \rangle = m$, if Merlin provides the correct $W(x,y,r)$'s, then Alice still always accepts (regardless of r). And when $\langle x, y \rangle \neq m$, since these $W(x,y,r)$'s only provide additional checks, Alice still accepts with probability at most $1/2$ for all proofs.

We let Π_{orig} denote the protocol from [151] (Lemma 3.7.2), and Π_{new} denote our new protocol. In the following we utilize Π_{new} to give an improved reduction from Exact-IP to additive approximation to Max-IP.

Before that, we need the following encoding trick, whose proof can be found in Section 3.9.

Lemma 3.7.3. *For all integers d, r and $0 \le m \le dr^2$, there are mappings $\varphi^x, \varphi^y : \{0, 1, \dots, r\}^d \to \{0, 1\}^{O(dr^2)^2}$ and an integer $0 \le M \le O(dr^2)^2$, such that for all $x, y \in \{0, 1, \dots, r\}^d$:*

- *If $\langle x, y\rangle = m$, then $\langle \varphi^x(x), \varphi^y(y)\rangle = M$.*
- *Otherwise, $\langle \varphi^x(x), \varphi^y(y)\rangle < M$.*
- *Moreover, M only depends on d and r.*

Lemma 3.7.4. *For all sufficiently large integers n, c and a parameter $\varepsilon > 0$, every Exact-$\mathsf{IP}_{n,c\log n}$ instance can be reduced to $n^{O(\varepsilon \log(c/\varepsilon))}$ instances of computing an $\Omega((\varepsilon/c)^6) \cdot d$ additive approximation to Max-$\mathsf{IP}_{n,d}$ for $d = n^{o(1)}$.*

Proof. Consider an Exact-$\mathsf{IP}_{n,c\log n}$ instance with sets A and B, and integer m. Using our protocol Π_{new} for checking whether $\langle x, y\rangle = m$, we only need to figure out whether there is a pair $(x, y) \in A \times B$ and a proof from Merlin such that Alice always accepts.

Let $N = c\log n$, and set $T = c/\varepsilon$. Then the message complexity from Merlin is $O(\varepsilon \log n \log(c/\varepsilon))$ and the total number of random bits is $R = \log(N/T) + O(1) \le \log(\varepsilon \log n) + O(1)$.

We first enumerate all valid proofs ψ, which is a pair of $z \in \mathbb{F}_{q^2}^{2^R}$ and $W \in \mathbb{Z}[x]^{2^R}$ such that for all $r \in [2^R]$, we have $z_r = W_r/P_{\mathsf{irred}}(x)$.

Next, we want to determine whether there is a pair $(x, y) \in A \times B$, such that this proof ψ makes Alice always accepts. Note we only need to distinguish the following two cases:

- For all $r \in [2^R]$, $\langle \vec{U}(x, r), \vec{V}(y, r)\rangle = W_r$.

- For at most half of $r \in [2^R]$, $\langle \vec{U}(x,r), \vec{V}(y,r) \rangle = W_r$.

Recall that $\vec{U}(x,r)$ and $\vec{V}(y,r)$ are vectors of T degree 1 polynomials from $\mathbb{Z}[x]$, with coefficients in $\{0,1,\ldots,q-1\}$, and W_r is a degree 2 polynomial in $\mathbb{Z}[x]$, with coefficients in $\{0,1,\ldots,O(q^3)\}$. For a polynomial $P(x)$ in $\mathbb{Z}[x]$ and an integer t, let $[t]P(x)$ denote the coefficient of x^t in $P(x)$. Then we can see $\langle \vec{U}(x,r), \vec{V}(y,r) \rangle = W_r$ is equivalent to the condition: for all $0 \le t \le 2$,

$$\sum_{i=0}^{t} \sum_{k=1}^{T} [i]\vec{U}(x,r)_k \cdot [t-i]\vec{V}(y,r)_k = [t]W_r. \tag{3.2}$$

Note that the left side of Equation (3.2) is an inner product between two vectors from $\{0,1,\ldots,q-1\}^{3T}$. By Lemma 3.7.3, we can construct three Boolean vectors $u_0, u_1, u_2 \in \{0,1\}^{O(q^6)}$ from $\vec{U}(x,r)$ and also $v_0, v_1, v_2 \in \{0,1\}^{O(q^6)}$ from $\vec{V}(y,r)$ and an integer M (which only depends on T), such that:

- If Equation (3.2) holds for all t, then $\sum_{i=0}^{2} \langle u_i, v_i \rangle = M$.

- Otherwise, $\sum_{i=0}^{2} \langle u_i, v_i \rangle < M$.

Now, we concatenate all these u_0, u_1, u_2 for all possibles r's to form a single vector u_x, and construct v_y similarly. We have:

- If for all $r \in [2^R]$, $\langle \vec{U}(x,r), \vec{V}(y,r) \rangle = W_r$, then $\langle u_x, v_y \rangle \ge 2^R \cdot M$.

- If for at most half of $r \in [2^R]$, $\langle \vec{U}(x,r), \vec{V}(y,r) \rangle = W_r$, then $\langle u_x, v_y \rangle \le 2^R \cdot (M - 1/2)$.

Now, let A_ψ and B_ψ be the collections of u_x and v_y with the proof ψ respectively. Then we want to distinguish between the following two cases:

- There is a ψ such that $\mathsf{Max}(A_\psi, B_\psi) \ge 2^R \cdot M$.

- For all ψ, $\mathsf{Max}(A_\psi, B_\psi) \le 2^R \cdot (M - 1/2)$.

Note that vectors in A_ψ and B_ψ are of dimension $d = O(q^6 \cdot 2^R)$, so the above can be solved by $2^{O(\varepsilon \log n \log(c/\varepsilon))} = n^{O(\varepsilon \log(c/\varepsilon))}$ calls to $\Omega(1/q^6) \cdot d = \Omega((\varepsilon/c)^6) \cdot d$ additive approximation to $\mathsf{Max\text{-}IP}_{n,d}$, which completes the proof. □

Now we are ready to prove Theorem 3.7.5.

Theorem 3.7.5. *The following are equivalent:*

1. *An $\varepsilon \cdot d$ additive approximation to $\mathsf{Max\text{-}IP}_{n,d}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time.*

2. *$\mathsf{Max\text{-}IP}_{n,c\log n}$ is solvable in $n^{2-1/c^{o(1)}}$ time.*

3. *$\mathsf{Exact\text{-}IP}_{n,c\log n}$ is solvable in $n^{2-1/c^{o(1)}}$ time.*

Proof. We only need to show that Item (1) implies Item (3). By Lemma 3.7.4, there are constants c_1, c_2 such that for any constant $\varepsilon_1 > 0$, every $\mathsf{Exact\text{-}IP}_{n,c\log n}$ instance can be reduced to $n^{c_1\varepsilon_1 \log(c/\varepsilon_1)}$ instances of $c_2 \cdot (\varepsilon_1/c)^6 \cdot d$ additive approximations to $\mathsf{Max\text{-}IP}_{n,d}$ for $d = n^{o(1)}$.

Suppose Item (1) holds, we set $\varepsilon_1 = 1/c$, then $\mathsf{Exact\text{-}IP}_{n,c\log n}$ can be solved in

$$n^{c_1 \log(c^2)/c + 2 - (c_2 \cdot c^{-12})^{o(1)}} = n^{2-1/c^{o(1)}}$$

time, which completes the proof. □

3.7.2 A Tighter Connection between Additive Approximation to Max-IP and Some Geometric Problems

Now we are ready to establish a similar connection between additive approximation to Max-IP and some geometric problems.

Theorem 3.7.6. *The following are equivalent:*

1. *An $\varepsilon \cdot d$ additive approximation to $\mathsf{Max\text{-}IP}_{n,d}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time.*

2. *An $\varepsilon \cdot d$ additive approximation to $\mathsf{Min\text{-}IP}_{n,d}$ is computable in $n^{2-\varepsilon^{o(1)}}$ time.*

3. *A $(1+\varepsilon)$ approximation to Bichrom.-ℓ_p-Closest-Pair is computable in $n^{2-\varepsilon^{o(1)}}$ time (for a constant $p \in [1,2]$).*

4. *A $(1+\varepsilon)$ approximation to ℓ_p-Furthest-Pair is computable in $n^{2-\varepsilon^{o(1)}}$ time (for a constant $p \in [1,2]$).*

One direction is simple, and already implicit in previous work.

Lemma 3.7.7 (Theorem 4.1 [151]). *For any $p \in [1,2]$, if Bichrom.-ℓ_p-Closest-Pair or ℓ_p-Furthest-Pair can be approximated in $n^{2-\varepsilon^{o(1)}}$ time, then there is an algorithm computing $\varepsilon \cdot d$ additive approximation to Max-IP in $n^{2-\varepsilon^{o(1)}}$ time.*

So it suffices to prove the other direction, we are going to apply Theorem 3.2.7.

Proof of Theorem 3.7.6. The equivalence between Item (1) and (2) follows directly from Lemma 3.6.3. By Lemma 3.7.7, Item (3) and (4) both imply Item (1). So it suffices to show Item (1) implies Item (3) and Item (4).

We only consider Bichrom.-ℓ_p-Closest-Pair here; the case for ℓ_p-Furthest-Pair are symmetric. Note that by a binary search (which incurs a negligible factor in the running time), we only need to consider the decision version, in which we are given a real R, and want to distinguish the two cases: (1) $\min_{(a,b)\in A\times B} \|a-b\|_p \le R$; (2) $\min_{(a,b)\in A\times B} \|a-b\|_p \ge (1+\varepsilon)\cdot R$.

By Theorem 3.2.7 and Lemma 3.3.4, this decision problem can be reduced to computing an $\Omega(\varepsilon \cdot d)$ approximation to $\mathsf{Max\text{-}IP}_{n,O(\varepsilon^{-2}\log n)}$, which by assumption can be solved in $n^{2-\varepsilon^{o(1)}}$ time. □

Finally, Theorem 3.1.4 is a simple corollary of Theorem 3.7.5 and Theorem 3.7.6.

3.8 Equivalence in the Data Structure Setting

In this section, we generalize our equivalence results to the data structure setting.

We first introduce the data structure versions of OV and Max-IP, which are used as intermediate problems for the reductions.

- **Online OV:** Preprocess a database $\mathcal{D}$ of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$, either report a point $x \in \mathcal{D}$ which is orthogonal to q or report that no x exists.
- **Online Max-IP:** Preprocess a database $\mathcal{D}$ of n points in $\{0,1\}^d$ such that, for all query of the form $q \in \{0,1\}^d$, find a point $x \in \mathcal{D}$ maximizing $\langle x, q \rangle$.

Theorem 3.8.1. *The following are equivalent:*

- *There is a $\delta > 0$ such that for all constant c, there is a data structure for Online OV with $d = c \log n$ uses $\operatorname{poly}(n)$ space and allows $n^{1-\delta}$ query time.*
- *There is a $\delta > 0$ such that for all constant c, there is a data structure for Online Max-IP with $d = c \log n$ uses $\operatorname{poly}(n)$ space and allows $n^{1-\delta}$ query time.*
- *There is a $\delta > 0$ such that for all $\varepsilon > 0$, there is a data structure for approximate NNS in ℓ_p with approximation ratio $(1+\varepsilon)$ uses $\operatorname{poly}(n)$ space and allows $n^{1-\delta}$ query time for a constant $p \in [1,2]$.*

Note that by [13], Online OV is equivalent to Partial Match, so the above theorem implies Theorem 3.1.3.

We also need the following two important observations from the proof of Lemma 3.5.2 and Lemma 3.7.4.

Lemma 3.8.2 (Implicit in Lemma 3.5.2). *Let n be an integer, c be a constant, $\varepsilon > 0$ and $0 \le k \le c \log n$. There are two families of functions $f_1, f_2, \ldots, f_m$ and $g_1, g_2, \ldots, g_m$ from $\{0,1\}^{c \log n}$ to $\{0,1\}^{2^{O(c/\varepsilon)} \log n}$ where $m = n^{O(\varepsilon \log(c/\varepsilon))}$, such that for all $x, y \in \{0,1\}^{c \log n}$, $\langle x, y \rangle = k$ if and only if there is an $i \in [m]$ such that $\langle f_i(x), g_i(y) \rangle = 0$. Moreover, functions f_i's and g_i's can be evaluated in $\operatorname{polylog}(n)$ time.*

Lemma 3.8.3 (Implicit in Lemma 3.7.4 and 3.7.7). *Let $p \in [1,2]$, n be an integer, c be a constant, $\varepsilon > 0$ and $0 \le k \le c \log n$. There are two families of functions $f_1, f_2, \ldots, f_m$ and $g_1, g_2, \ldots, g_m$ from $\{0,1\}^{c \log n}$ to $\mathbb{R}^{n^{o(1)}}$ where $m = n^{O(\varepsilon \log(c/\varepsilon))}$, such that for all $x, y \in \{0,1\}^{c \log n}$,*

- *If $\langle x, y\rangle = k$, then there is an $i \in [m]$ such that $\|f_i(x) - g_i(y)\|_p \le 1 - \Omega((\varepsilon/c)^6)$.*

- *Otherwise, for all $i \in [m]$, $\|f_i(x) - g_i(y)\|_p \ge 1$.*

Moreover, functions f_i's and g_i's can be evaluated in $n^{o(1)}$ time.

Proof of Theorem 3.8.1. In the below we first show the equivalence between Online OV and Online Max-IP, the equivalence between Online Max-IP and NNS is proved similarly, so we only sketch the main ideas.

Online OV $\Leftrightarrow$ Online Max-IP. The reduction from Online OV to Online Max-IP is trivial. For the other direction, suppose there is a $\delta > 0$ such that for all constant c, there is an algorithm for Online OV with $d = c\log n$ such that it uses $\mathrm{poly}(n)$ space and allows $n^{1-\delta}$ query time.

Let $d = c\log n$ for a constant c, and c_1 be the constant hiding in the big-O of $m = 2^{O(\varepsilon\log(c/\varepsilon))}$ in Lemma 3.8.2. Suppose we are given a set $\mathcal{D}$ of n points from $\{0,1\}^d$.

We set ε such that $c_1 \cdot \varepsilon\log(c/\varepsilon) = \delta/2$ and apply Lemma 3.8.2. Now, for each $0 \le k \le d$, we build $n^{c_1\cdot\varepsilon\log(c/\varepsilon)} = n^{\delta/2}$ data structures for Online OV, the i-th data structure consists of the $f_i(x)$'s for all $x \in \mathcal{D}$. Note that the $f_i(x)$'s have length $2^{O(c/\varepsilon)} \cdot \log n$, which is still $O(\log n)$ as ε is a constant.

For each query $q \in \{0,1\}^d$, note that there is an $x \in \mathcal{D}$ such that $\langle x, q\rangle = k$ if and only if there is an i such that the i-th Online OV structure contains an orthogonal point to $g_i(q)$. Therefore, by enumerating k from d down to 0, i from $[n^{\delta/2}]$, and making corresponding queries to the Online OV data structures, one can answer queries for Online Max-IP in $n^{1-\delta/2} \cdot d$ time.

Online Max-IP $\Leftrightarrow$ Approximate NNS (Sketch). Using Lemma 3.8.3, the reduction from Online Max-IP to Approximate NNS can be proved similarly as from Online Max-IP to Online OV.

For the direction from approximate NNS to Online Max-IP: suppose the approximation ratio is $(1+\varepsilon)$. It suffices, for all R of the form $(1+\varepsilon/3)^k$ for an integer k, to construct a data structure which finds a point with distance smaller than $R \cdot (1+\varepsilon/3)$ if the minimum distance is smaller than R, and reports a failure if the minimum distance is greater than $R \cdot (1+\varepsilon/3)$ (its behavior can be arbitrary if neither case holds). Using the reduction implicit in proof of Theorem 3.2.7, this can be reduced to Online Max-IP with $d = O(\log n)$.

□

3.9 Missing Proofs

Here we give some missing proofs in the chapter.

Proof of Lemma 3.6.2. Let $X = \sum_{i=1}^{cm} X_i$ and $\mu = \mathbb{E}[X] = \varepsilon \cdot cm$. Set $\delta = \varepsilon^{-1}/3$. By the multiplicative Chernoff bound, we have

$$\Pr[X > (1+\delta)\cdot\mu] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}.$$

Note that $(1+\delta)\cdot\mu = (1+\varepsilon^{-1}/3)\cdot\varepsilon\cdot cm < \frac{1}{2}\cdot cm$. Also, we have

$$\begin{aligned}\left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} &= e^{-\mu\cdot[(1+\delta)\ln(1+\delta)-\delta]}\\ &\le e^{-\mu\cdot[\delta\ln\delta-\delta]}\\ &\le e^{-\varepsilon\cdot cm\cdot\left[\varepsilon^{-1}/3\ln(\varepsilon^{-1}/3)\right]/2}\\ &\le e^{-\varepsilon\cdot cm\cdot\left[\varepsilon^{-1}\ln(\varepsilon^{-1})\right]/12}\\ &\le e^{-\ln\varepsilon^{-1}\cdot cm/12} = \varepsilon^{-cm/12}.\end{aligned}$$

Therefore, we can set $c = 12$, and the proof is completed. □

Proof of Lemma 3.6.3. We define two functions $\varphi_x, \varphi_y : \{0,1\} \to \{0,1\}^2$ such that:

$$\varphi_x(0) := (1,0), \quad \varphi_x(1) := (0,1), \quad \varphi_y(0) := (1,1), \quad \varphi_y(1) := (1,0).$$

It is easy to check that for $a, b \in \{0,1\}$, $a \cdot b = 1 - \langle\varphi_x(a), \varphi_y(b)\rangle$. Then, for

$x, y \in \{0,1\}^d$, we define $\psi^x_{\mathsf{rev}}(x) \in \{0,1\}^{2d}$ as the concatenation of $\varphi_x(x_i)$ for each $i \in [d]$, and similarly $\psi^y_{\mathsf{rev}}(y) \in \{0,1\}^{2d}$ as the concatenation of $\varphi_y(y_i)$ for each $i \in [d]$.

Then we can see $\langle \psi^x_{\mathsf{rev}}(x), \psi^y_{\mathsf{rev}}(y) \rangle = \sum_{i=1}^{d} \langle \varphi_x(x_i), \varphi_y(y_i) \rangle = d - \langle x, y \rangle$. □

Proof of Lemma 3.6.4. We remark the reduction here is essentially the same as the trick used in [172]. For a vector $v \in \{0,1\}^*$, we use $v^{\otimes k}$ to denote the concatenation of k copies of v.

Consider the following polynomial $P(x,y) := (\langle x, y \rangle - m)^2$, we have

$$P(x,y) = \langle x, y \rangle^2 - 2m\langle x, y \rangle + m^2 = \langle x, y \rangle^2 + 2m(d - \langle x, y \rangle) + m^2 - 2dm.$$

For convenience, for a vector $z \in \{0,1\}^{d^2}$, we use $z_{(i,j)}$ to denote the $(i-1) \cdot d + j$-th coordinate of z. For $x, y \in \{0,1\}^d$, we construct $\widetilde{x}, \widetilde{y} \in \{0,1\}^{d^2}$ such that $\widetilde{x}_{(i,j)} = x_i \cdot x_j$ and $\widetilde{y}_{(i,j)} = y_i \cdot y_j$. Then we can see

$$\langle \widetilde{x}, \widetilde{y} \rangle = \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j \cdot y_i y_j = \left(\sum_{i=1}^{d} x_i y_i \right)^2 = \langle x, y \rangle^2.$$

Let ψ^x_{rev} and ψ^y_{rev} be the two functions from Lemma 3.6.3. For $x, y \in \{0,1\}^d$, we define

$$\varphi^x_{d,m}(x) := (\widetilde{x}, \psi^x_{\mathsf{rev}}(x)^{\otimes(2m)}) \qquad \text{and} \qquad \varphi^y_{d,m}(y) := (\widetilde{y}, \psi^y_{\mathsf{rev}}(y)^{\otimes(2m)}).$$

Then we have $\langle \varphi^x_{d,m}(x), \varphi^y_{d,m}(y) \rangle = \langle x, y \rangle^2 + 2m(d - \langle x, y \rangle) = P(x,y) + 2dm - m^2$. And we set $M_{d,m} = 2dm - m^2$.

Now, if $\langle x, y \rangle = m$, we have $P(x,y) = 0$, and therefore $\langle \varphi^x_{d,m}(x), \varphi^y_{d,m}(y) \rangle = M_{d,m}$. Otherwise, $\langle x, y \rangle \neq m$ and we have $P(x,y) > 0$, and hence $\langle \varphi^x_{d,m}(x), \varphi^y_{d,m}(y) \rangle > M_{d,m}$. Note that $\varphi^x_{d,m}(x), \varphi^y_{d,m}(y) \in \{0,1\}^{d^2+4dm}$, we add $5d^2 - M_{d,m}$ dummy ones to the end of $\varphi^x_{d,m}(x)$ and $\varphi^y_{d,m}(y)$ and set $M_d = 5d^2$, which completes the proof. □

Proof of Lemma 3.7.3. We begin by the construction of two embeddings $\psi_x, \psi_y : \{0,1,\dots,r\} \to \{0,1\}^{r^2}$ such that for any $x, y \in \{0,1,\dots,r\}$, $\langle \psi_x(x), \psi_y(y) \rangle = x \cdot y$.

For convenience, in the following we use $z_{(i,j)}$ to denote the $(i-1) \cdot r + j$-th coordinate of z. Then we define $\psi_x(x)_{(i,j)}$ as 1 when $i \leq x$, and 0 otherwise; similarly, we define $\psi_y(y)_{(i,j)}$ as 1 when $j \le y$, and 0 otherwise. We have

$$\langle \psi_x(x), \psi_y(y) \rangle = \sum_{i=1}^{r} \sum_{j=1}^{r} \psi_x(x)_{(i,j)} \cdot \psi_y(y)_{(i,j)} = \sum_{i=1}^{x} \sum_{j=1}^{y} 1 \cdot 1 = \langle x, y \rangle.$$

Slightly abusing notations, for $x, y \in \{0, 1, \dots, r\}^d$, we define $\psi_x(x)$ and $\psi_y(y)$ as the concatenation of ψ_x or ψ_y applying on all coordinates of x and y. Then we have $\psi_x(x), \psi_y(y) \in \{0,1\}^{dr^2}$, and $\langle \psi_x(x), \psi_y(y) \rangle = \langle x, y \rangle$. Then applying Lemma 3.6.4 and Lemma 3.6.3 completes the proof. □

3.10 More Applications of the Σ_2^{cc} Reduction Framework

To demonstrate the potential power of our Σ_2^{cc} framework. In the following we discuss some of its applications other than establishing our equivalence class. The first one is a very simple reduction from Hopcroft's problem in *constant dimensions* to OV in *polylogarithmic dimensions.* And the second one is a reduction from 3-SUM to 3-OV.

3.10.1 Integer Inner Product and Hopcroft's Problem

The Hopcroft's problem is defined as follows: you are given two sets A, B of n vectors from $\mathbb{Z}^d$, and want to determine whether there is an $(a, b) \in A \times B$ such that $\langle a, b \rangle = 0$. In other words, it is the same as OV except for now vectors consist of integer entries.

We use $\mathbb{Z}\text{-}\mathsf{OV}_{n,d}$ to denote this problem in d dimensions for simplicity, and assume the integers in $\mathbb{Z}\text{-}\mathsf{OV}_{n,d}$ belong to $[-n^c, n^c]$ for a constant c, which is the most interesting case. Now we formally state our reduction.

Theorem 3.10.1. *Let c, d be two constants, a $\mathbb{Z}\text{-}\mathsf{OV}_{n,d}$ instance I with entries in $[-n^c, n^c]$ can be reduced to an $\mathsf{OV}_{n,O(\log n)^{d+1}}$ instance J in $n^{1+o(1)}$ time, such that I is a yes instance if and only if J is a yes instance.*

An immediate corollary is that if moderate dimensional OV is in truly subquadratic time, then $\mathbb{Z}\text{-}\mathsf{OV}_{n,d}$ is also in truly subquadratic time for all constant d.

Let d be an integer, we define $\text{Z-IP}_d : \mathbb{Z}^d \times \mathbb{Z}^d \to \{0, 1\}$ as the function that checks whether two d dimensional vectors in $\mathbb{Z}^d$ are orthogonal. Note that $\mathbb{Z}\text{-OV}_{n,d}$ is equivalent to $\text{Z-IP}_d\text{-Satisfying-Pair}_n$.

Theorem 3.10.1 is just a direct corollary of the following fast Σ_2 communication protocol for Z-IP_d (it is in fact a coNP communication protocol, as Merlin sends nothing) and Theorem 3.2.2.

Lemma 3.10.2. *Let c, d be two constants, there is a Σ_2^{cc} protocol for Z-IP_d with entries in $[-n^c, n^c]$, in which Merlin sends nothing, Megan sends $\log\log(n) + O(1)$ bits and Alice and Bob communicate $d \cdot \log\log(n) + O(d)$ bits.*

Proof. Let x, y be two vectors from $[-n^c, n^c]^d$, we have $|\langle x, y\rangle| \le d \cdot n^{2c}$.

Let t be the smallest number such that the first t primes $p_1, p_2, \ldots, p_t$ satisfy $\prod_{i=1}^t p_i > d \cdot n^{2c}$. We first bound t and the largest prime p_t. Clearly, $t \le \log(d \cdot n^{2c}) = O(\log n)$. Recall that $n\#$ denotes the product of all primes less than or equal to n (the primordial function), and we have $n\# = e^{(1+o(1))n}$. By the definition of t, it follows that $(p_t - 1)\# = e^{(1+o(1))\cdot(p_t-1)} \le d \cdot n^{2c}$ and $p_t = O(\log n)$.

From our choice of t, we have $\langle x, y\rangle = 0$ if and only if $\langle x, y\rangle \equiv 0 \pmod{p_i}$ for all $i \in [t]$. So in the protocol, Merlin sends nothing. Megan sends an index $i \in [t]$, which takes $\log t = \log\log n + O(1)$ bits. After that, for each $j \in [d]$, Bob sends $y_j \bmod p_i$ to Alice, which takes $d \cdot \log p_i \le d \cdot \log\log n + O(d)$ bits, and Alice accepts if and only if $\langle x, y\rangle \equiv 0 \pmod{p_i}$. □

3.10.2 Sum-Check and 3-Sum

Next we discuss a reduction from 3-SUM to 3-OV. 3-OV is a generalized version of OV, in which you are given three sets A, B, C, each of n vectors from $\{0, 1\}^d$, and want to determine whether there is an $(a, b, c) \in A \times B \times C$ such that $\sum_{i=1}^d a_i \cdot b_i \cdot c_i = 0$ (the generalized inner product of a, b, and c is zero). We use $\text{3-OV}_{n,d}$ to denote the 3-OV problem with sets of n vectors of d dimensions.

Theorem 3.10.3. *If 3-OV is in truly-subquadratic time*[12]*, then so is 3-SUM.*

We remark that this reduction is not optimal, as it is conjectured that 3-OV requires $n^{3-o(1)}$ time (also implied by SETH). We include it here only as an illustration of the applicability of our reduction framework. It would be very interesting to improve it to a reduction from 3-SUM to OV[13].

Note that 3-OV is actually a *Satisfying-Triple* problem[14], and Theorem 3.2.2 only works for Satisfying-Pair problems. Still, we can generalize Theorem 3.2.2 easily to get the same connection between a 3-party Σ_2 communication protocol and a reduction from a satisfying-triple problem to 3-OV.

Let $F : \left(\{0,1\}^d\right)^3 \to \{0,1\}$ be a function. F-Satisfying-Triple$_n$ is the problem that you are given sets A, B, C of n vectors from $\{0,1\}^d$, and want to determine whether there is an $(a,b,c) \in A \times B \times C$ such that $F(a,b,c) = 1$. A 3-party Σ_2 communication protocol can be defined similarly as in Definition 3.2.1 with the third player named Charles (we omit it here). We have the following analogous theorem of Theorem 3.2.2.

Theorem 3.10.4. *Let $F : \left(\{0,1\}^d\right)^3 \to \{0,1\}$ and n be an integer, suppose F admits a computationally-efficient Σ_2^{cc} protocol. In which Merlin sends m_1 bits, Megan sends m_2 bits, Alice, Bob, and Charles communicate ℓ bits.*

Then there is a reduction from an F-Satisfying-Triple$_n$ instance I to 2^{m_1} 3-OV$_{n,2^{(m_2+\ell)}}$ instances $J_1, J_2, \dots, J_{2^{m_1}}$, such that I is a yes instance if and only if one of the reduced instances is a yes instance. And the reduction takes $O(n \cdot 2^{O(m_1+m_2+\ell)} \cdot \mathrm{poly}(d))$ time.

We omit its proof here as it is identical to that of Theorem 3.2.2.

Note that we can assume the integers in the 3-SUM instance are in $[-n^4/2, n^4/2)$ without loss of generality. In order to apply Theorem 3.10.4, we need an efficient Σ_2^{cc} protocol for checking whether 3 numbers sum to zero.

[12] This means there is an $\varepsilon > 0$ such that for all constants c, 3-OV$_{n,c\log n}$ can be solved in $n^{2-\varepsilon}$ time.

[13] In [63], it is shown that under the NSETH (which is controversial, see [167]), there is no fine-grained reduction from OV to 3-SUM. But there is no formal evidence against the other direction.

[14] It is also called a *Product Space Problem* in [106].

Theorem 3.10.5. *For an integer n, let $F_{\mathsf{zero}} : \left(\{0,1\}^{4\log n}\right)^3 \to \{0,1\}$ be the function that treats its inputs as three numbers in $[-n^4/2, n^4/2)$ (via a natural encoding), and checks whether they sum to zero. For any $1 \le T \le 4\log n$, F_{zero} admits a Σ_2^{cc} protocol in which Merlin sends $O(\log n/T)$ bits, Megan sends $\lceil \log(n/T) \rceil$ bits and Alice, Bob, and Charles communicate $O(T)$ bits.*

Proof. Let x, y, z be three input numbers. Suppose Alice holds x, Bob holds y, and Charles holds z. We add $n^4/2$ to each of them so that they now belong to $[0, n^4)$, and we want to check whether they sum up to $t = n^4/2 \cdot 3$. Assuming T divides $4\log n$ for simplicity, we treat x, y, z as numbers in 2^T base. Let $\ell = 4\log n/T$, and $x_1, x_2, \dots, x_\ell$ be the digits of x (from the least significant one to the most significant one, y_i's, z_i's, and t_i's are defined similarly).

Suppose we add x, y, z together as numbers in 2^T base. Let $c \in \{0,1,2\}^\ell$ be a sequence of carries. We can see $x + y + z = t$ with respect to the carry sequence c if and only if

$$x_i + y_i + z_i + c_{i-1} = t_i + c_i \cdot 2^T \quad \text{for } i \in [\ell+1].$$

In the above we set $c_0 = x_{\ell+1} = y_{\ell+1} = z_{\ell+1} = 0$.

Therefore, in the protocol, Merlin sends the carry sequence c, which takes $O(\ell) = O(\log n/T)$ bits. Megan sends an index $i \in [\ell+1]$. After that, Bob and Charles send y_i and z_i to Alice, respectively, and Alice accepts if and only if the above equality holds. It is straightforward to verify the protocol works. □

Finally, we are ready to prove Theorem 3.10.3.

Proof of Theorem 3.10.3. Suppose 3-OV is in truly-subquadratic time. That is, there is a constant ε such that for all constant c, $\mathsf{3\text{-}OV}_{n,c\log n}$ can be solved in $n^{2-\varepsilon}$ time.

Given a 3-SUM instance with integer entries in $[-n^4/2, n^4/2)$, it is just an F_{zero} -$\mathsf{Satisfying\text{-}Triple}_n$ instance. Let c_1 be the constant hiding in $O(\log n/T)$ of Theorem 3.10.4, then F_{zero} admits a Σ_2^{cc} protocol in which Merlin sends $c_1 \log n/T$ bits, Megan sends $\lceil \log(n/T) \rceil$ bits and Alice, Bob, and Charles communicate $O(T)$ bits.

Set $T = c_1 \cdot 2/\varepsilon$, Theorem 3.10.4 implies this 3-SUM instance can be reduced to $n^{\varepsilon/2}$ $\textsf{3-OV}_{n,2^{O(1/\varepsilon)}\log n}$ instances. Applying the algorithm for 3-OV, 3-SUM can be solved in $n^{2-\varepsilon/2}$ time, which completes the proof. □

Chapter 4

$\mathsf{NP} \cdot \mathsf{UPP}$ Protocols and Hardness for Furtherest Pair and Hopcroft Problem in $2^{O(\log^* n)}$ Dimensions

4.1 Introduction

We study the following fundamental problem from similarity search and statistics, which asks to find the most correlated pair in a dataset:

Definition 4.1.1 (Bichromatic Maximum Inner Product ($\mathsf{Max\text{-}IP}$)). For $n, d \in \mathbb{N}$, the $\mathsf{Max\text{-}IP}_{n,d}$ problem is defined as: *given two sets A, B of vectors from $\{0,1\}^d$ compute*

$$\mathsf{OPT}(A, B) := \max_{a \in A, b \in B} a \cdot b.$$

We use $\mathbb{Z}$-$\mathsf{Max\text{-}IP}_{n,d}$ ($\mathbb{R}$-$\mathsf{Max\text{-}IP}_{n,d}$) to denote the same problem, but with A, B being sets of vectors from $\mathbb{Z}^d$ ($\mathbb{R}^d$).

Hardness of Exact $\mathbb{Z}$-$\mathsf{Max\text{-}IP}$

Recall that from [168], there is no $n^{2-\Omega(1)}$-time algorithm for exact Boolean $\mathsf{Max\text{-}IP}_{n,\omega(\log n)}$. Since in real life applications of similarity search, one often deals with real-valued data instead of just Boolean data, it is natural to ask about $\mathbb{Z}$-Max-IP (which is certainly a special case of $\mathbb{R}$-Max-IP): what is the maximum d such that $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ can be solved exactly in $n^{2-\Omega(1)}$ time?

Besides being interesting in its own right, there are also reductions from $\mathbb{Z}$-Max-IP to ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair. Hence, lower bounds for $\mathbb{Z}$-Max-IP imply lower bounds for these two famous problems in computational geometry (see [172] for a discussion on this topic).

Prior to our work, it was implicitly shown in [172] that:

Theorem 4.1.2 ([172]). *Assuming SETH, there is no $n^{2-\Omega(1)}$-time algorithm for $\mathbb{Z}$-Max-$\mathsf{IP}_{n,\omega((\log\log n)^2)}$ with vectors of $O(\log n)$-bit entries.*

However, the best known algorithm for $\mathbb{Z}$-Max-IP runs in $n^{2-\Theta(1/d)}$ time [125, 17, 176][1], hence there is still a gap between the lower bound and the best known upper bounds. To confirm these algorithms are in fact optimal, we would like to prove a lower bound with $\omega(1)$ dimensions.

Hardness of Exact $\mathbb{Z}$-Max-IP in $2^{O(\log^* n)}$ Dimensions

In this paper, we significantly strengthen the previous lower bound from $\omega((\log\log n)^2)$ dimensions to $2^{O(\log^* n)}$ dimensions ($2^{O(\log^* n)}$ is an *extremely slow-growing* function, see preliminaries for its formal definition).

Theorem 4.1.3. *Assuming SETH (or OVC), there is a constant c such that any exact algorithm for $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ for $d = c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

As direct corollaries of the above theorem, using reductions implicit in [172], we also conclude hardness for ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair under SETH (or OVC) in $2^{O(\log^* n)}$ dimensions.

[1] [17, 176] are for ℓ_2-Furthest Pair or Bichromatic ℓ_2-Closest Pair. They also work for $\mathbb{Z}$-Max-IP as there are reductions from $\mathbb{Z}$-Max-IP to these two problems, see [172] or Lemma 4.4.5 and Lemma 4.4.6.

Theorem 4.1.4 (Hardness of ℓ_2-Furthest Pair in $c^{\log^* n}$ Dimensions). *Assuming SETH (or OVC), there is a constant c such that ℓ_2-Furthest Pair in $c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

Theorem 4.1.5 (Hardness of Bichromatic ℓ_2-Closest Pair in $c^{\log^* n}$ Dimensions). *Assuming SETH (or OVC), there is a constant c such that Bichromatic ℓ_2-Closest Pair in $c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

The above lower bounds on ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair are in sharp contrast with the case of *ℓ_2-Closest Pair*, which can be solved in $2^{O(d)} \cdot n \log^{O(1)} n$ time [49, 109, 86].

Improved Dimensionality Reduction for OV and Hopcroft's Problem

Our hardness of $\mathbb{Z}$-Max-IP is established by a reduction from Hopcroft's problem, whose hardness is in turn derived from the following significantly improved dimensionality reduction for OV.

Lemma 4.1.6 (Improved Dimensionality Reduction for OV). *Let $1 \le \ell \le d$. There is an*

$$O\left(n \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))} \cdot \operatorname{poly}(d)\right)\text{-time}$$

reduction from $\mathsf{OV}_{n,d}$ to $\ell^{O(6^{\log^* d} \cdot (d/\ell))}$ instances of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$, with vectors of entries with bit-length $O\left(d/\ell \cdot \log \ell \cdot 6^{\log^* d}\right)$.

Comparison with [172]. Comparing to the old construction in [172], our reduction here is more efficient when ℓ is much smaller than d (which is the case we care about). That is, in [172], $\mathsf{OV}_{n,d}$ can be reduced to $d^{d/\ell}$ instances of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$, while we get $\left\{\ell^{6^{\log^* d}}\right\}^{d/\ell}$ instances in our improved one. So, for example, when $\ell = 7^{\log^* d}$, the old reduction yields $d^{d/7^{\log^* d}} = n^{\omega(1)}$ instances (recall that $d = c \log n$ for an arbitrary constant c), while our improved one yields only $n^{o(1)}$ instances, each with $2^{O(\log^* n)}$ dimensions.

From Lemma 4.1.6, the following theorem follows in the same way as in [172].

Theorem 4.1.7 (Hardness of Hopcroft's Problem in $c^{\log^* n}$ Dimensions). *Assuming SETH (or OVC), there is a constant c such that* $\mathbb{Z}$-$\mathsf{OV}_{n,c^{\log^* n}}$ *with vectors of* $O(\log n)$*-bit entries requires* $n^{2-o(1)}$ *time.*

Connection between $\mathbb{Z}$-Max-IP Lower Bounds and NP · UPP Communication Protocols

We also show a new connection between $\mathbb{Z}$-Max-IP and a special type of communication protocol. Let us first recall the Set-Disjointness problem:

Definition 4.1.8 (Set-Disjointness). Let $n \in \mathbb{N}$, in Set-Disjointness (DISJ_n), Alice holds a vector $X \in \{0,1\}^n$, Bob holds a vector $Y \in \{0,1\}^n$, and they want to determine whether $X \cdot Y = 0$.

In [12], the hardness of approximating Max-IP is established via a connection to MA communication protocols (in particular, an MA communication protocol with small communication complexity for Set-Disjointness). Our lower bound for (exact) $\mathbb{Z}$-Max-IP can also be connected to similar NP · UPP protocols (note that MA = NP · promiseBPP).

Formally, we define NP · UPP protocols as follows[2]:

Definition 4.1.9. For a problem Π with inputs x, y of length n (Alice holds x and Bob holds y), we say a communication protocol is an *(m, ℓ)-efficient NP · UPP communication protocol* if the following holds:

- There are three parties Alice, Bob and Merlin in the protocol.

[2] Here are some comments on the name NP · UPP. Roughly speaking, a UPP protocol Π is a private-coin randomized communication protocol in which Alice and Bob accept with probability $> 1/2$ if and only if the answer is yes. For a communication protocol class $\mathcal{D}$, NP · $\mathcal{D}$ denotes a new class of protocol resembling a Merlin-Arthur game: A prover (who knows the inputs of Alice and Bob) sends a proof to both Alice and Bob first; then Alice and Bob run a prescribed $\mathcal{D}$ protocol on their inputs and the proof to decide whether they accept the proof or not. The protocol needs to satisfy the soundness and completeness conditions similar to an original MA protocol.

- Merlin sends Alice and Bob an advice string z of length m, which is a function of x and y.

- Given y and z, Bob sends Alice ℓ bits, and Alice decides to accept or not.[3] They have an unlimited supply of private random coins (not public, which is important) during their conversation. The following conditions hold:

  - If $\Pi(x, y) = 1$, then there is an advice z from Merlin such that Alice accepts with probability $\geq 1/2$.
  - Otherwise, for all possible advice strings from Merlin, Alice accepts with probability $< 1/2$.

Moreover, we say the protocol is (m, ℓ)-computational-efficient, if in addition the probability distributions of both Alice and Bob's behavior can be computed in poly(n) time given their input and the advice.

Our new reduction from OV to Max-IP actually implies an efficient NP $\cdot$ UPP protocol for Set-Disjointness.

Theorem 4.1.10. *For all* $1 \leq \alpha \leq n$, *there is an*

$$\left(\alpha \cdot 6^{\log^* n} \cdot (n/2^{\alpha}), O(\alpha)\right) \textit{-computational-efficient}$$

NP $\cdot$ UPP communication protocol for DISJ_n.

For example, when $\alpha = 3\log^* n$, Theorem 4.1.10 implies there is an $O(o(n), O(\log^* n))$-computational-efficient NP $\cdot$ UPP communication protocol for DISJ_n. Moreover, we show that if the protocol of Theorem 4.1.10 can be improved a little bit (like removing the $6^{\log^* n}$ term), we would obtain the desired hardness for $\mathbb{Z}$-Max-IP in $\omega(1)$-dimensions.

[3] In UPP, actually one-way communication is equivalent to the seemingly more powerful one in which they communicate [143].

Theorem 4.1.11. *Assuming SETH (or OVC), if there is an increasing and unbounded function f such that for all $1 \le \alpha \le n$, there is an*

$$(n/f(\alpha), \alpha)\text{-}\textit{computational-efficient}$$

NP· UPP communication protocol for DISJ_n, then $\mathbb{Z}$-Max-$\mathsf{IP}_{n,\omega(1)}$ requires $n^{2-o(1)}$ time with vectors of polylog(n)*-bit entries. The same holds for ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair.*

4.2 Intuition for Dimensionality Self Reduction for OV

The $2^{O(\log^* n)}$ factor in Lemma 4.1.6 is not common in theoretical computer science[4], and our new reduction for OV is considerably more complicated than the polynomial-based construction from [172]. Hence, it is worth discussing the intuition behind Lemma 4.1.6, and the reason why we get a factor of $2^{O(\log^* n)}$.

A Direct Chinese Remainder Theorem Based Approach. We first discuss a direct reduction based on the *Chinese Remainder Theorem* (CRT) (see Theorem 4.3.3 for a formal definition). CRT says that given a collection of distinct primes $q_1, \dots, q_b$, and a collection of integers $r_1, \dots, r_b$, there exists a unique integer $t = \mathsf{CRR}(\{r_j\}; \{q_j\})$ such that $t \equiv r_j \pmod{q_j}$ for each $j \in [b]$ and $0 \le t < \prod_{j=1}^{b} q_j$ (CRR stands for *Chinese Remainder Representation*).

Now, let $b, \ell \in \mathbb{N}$, suppose we would like to have a dimensionality reduction φ from $\{0,1\}^{b \cdot \ell}$ to $\mathbb{Z}^{\ell}$. We can partition an input $x \in \{0,1\}^{b \cdot \ell}$ into ℓ blocks, each of length b, and represent each block via CRT: that is, for a block $z \in \{0,1\}^{b}$, we map it into a single integer $\varphi_{\mathsf{block}}(z) := \mathsf{CRR}(\{z_j\}; \{q_j\})$, and the concatenations of φ_{block}

[4] Other examples include an $O(2^{O(\log^* n)} n^{4/3})$ time algorithm for $\mathbb{Z}$-$\mathsf{OV}_{n,3}$ [126], $O(2^{O(\log^* n)} n \log n)$ time algorithms (Fürer's algorithm with its modifications) for Fast Integer Multiplication [88, 78, 98] and an old $O(n^{d/2} 2^{O(\log^* n)})$ time algorithm for Klee's measure problem [64].

over all blocks of x is $\varphi(x) \in \mathbb{Z}^{\ell}$.

The key idea here is that, for $z, z' \in \{0,1\}^b$, $\varphi_{\mathsf{block}}(z) \cdot \varphi_{\mathsf{block}}(z') \pmod{q_j}$ is simply $z_j \cdot z'_j$. That is, the multiplication between two *integers* $\varphi_{\mathsf{block}}(z) \cdot \varphi_{\mathsf{block}}(z')$ simulates the coordinate-wise multiplication between two *vectors* z and z'!

Therefore, if we make all primes q_j larger than ℓ, we can in fact determine $x \cdot y$ from $\varphi(x) \cdot \varphi(y)$, by looking at $\varphi(x) \cdot \varphi(y) \pmod{q_j}$ for each j. That is,

$$x \cdot y = 0 \Leftrightarrow \varphi(x) \cdot \varphi(y) \equiv 0 \pmod{q_j} \quad \text{for all } j.$$

Hence, let V be the set of all integer $0 \le v \le \ell \cdot \left(\prod_{j=1}^{b} q_j\right)^2$ that $v \equiv 0 \pmod{q_j}$ for all $j \in [b]$, we have

$$x \cdot y = 0 \Leftrightarrow \varphi(x) \cdot \varphi(y) \in V.$$

The reduction is completed by constructing a $\mathbb{Z}$-OV instance for each $v \in V$: we append corresponding values to make $\varphi_A(x) = [\varphi(x), -1]$ and $\varphi_B(y) = [\varphi(y), v]$ (this step is from [172]).

Note that a nice property for φ is that each $\varphi(x)_i$ only depends on the i-th block of x, and the mapping is the same on each block (φ_{block}); we call this the *block mapping property*.

Analysis of the Direct Reduction. To continue building intuition, let us analyze the above reduction. The size of V is the number of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ instances we create, and $|V| \ge \prod_{j=1}^{b} q_j$. These primes q_j have to be all distinct, and it follows that $\prod_{j=1}^{b} q_j$ is $b^{\Theta(b)}$. Since we want to create at most $n^{o(1)}$ instances (or n^{ε} for arbitrarily small ε), we need to set $b \le \log n / \log\log n$. Moreover, to base our hardness on OVC which deals with $c \log n$-dimensional vectors, we need to set $b \cdot \ell = d = c \cdot \log n$ for an arbitrary constant c. Therefore, we must have $\ell \ge \log\log n$, and the above reduction only obtains the same hardness result as [172].

Key Observation: "Most Space Modulo q_j" is Actually Wasted. To improve the above reduction, we need to make $|V|$ smaller. Our key observation about

φ is that, for the primes q_j's, they are mostly larger than $b \gg \ell$, but $\varphi(x) \cdot \varphi(y) \in \{0, 1, \ldots, \ell\} \pmod{q_j}$ for all these q_j's. Hence, *"most space modulo q_j" is actually wasted.*

Make More "Efficient" Use of the "Space": Recursive Reduction. Based on the previous observation, we want to use the "space modulo q_j" more efficiently. It is natural to consider a *recursive reduction.* We will require all our primes q_j's to be larger than b. Let b_{m} be a very small integer compared to b, and let $\psi : \{0,1\}^{b_{\mathsf{m}} \cdot \ell} \to \mathbb{Z}^{\ell}$ with a set V_{ψ} and a block mapping ψ_{block} be a similar reduction on much smaller inputs: for $x, y \in \{0,1\}^{b_{\mathsf{m}} \cdot \ell}$, $x \cdot y = 0 \Leftrightarrow \psi(x) \cdot \psi(y) \in V_{\psi}$. We also require here that $\psi(x) \cdot \psi(y) \leq b$ for all x and y.

For an input $x \in \{0,1\}^{b \cdot \ell}$ and a block $z \in \{0,1\}^{b}$ of x, our key idea is to partition z again into b/b_{m} "micro" blocks each of size b_{m}. And for a block z in x, let $z^1, \ldots, z^{b/b_{\mathsf{m}}}$ be its b/b_{m} micro blocks, we map z into an integer $\varphi_{\mathsf{block}}(z) := \mathsf{CRR}(\{\psi_{\mathsf{block}}(z^j)\}_{j=1}^{b/b_{\mathsf{m}}}; \{q_j\}_{j=1}^{b/b_{\mathsf{m}}})$.

Now, given two blocks $z, w \in \{0,1\}^{b}$, we can see that

$$\varphi_{\mathsf{block}}(z) \cdot \varphi_{\mathsf{block}}(w) \equiv \psi_{\mathsf{block}}(z^j) \cdot \psi_{\mathsf{block}}(w^j) \pmod{q_j}.$$

For two inputs $x, y \in \{0,1\}^{b \cdot \ell}$, for $(i,j) \in [\ell] \times [b/b_{\mathsf{m}}]$, we use $x^{i,j} \in \{0,1\}^{b_{\mathsf{m}}}$ ($y^{i,j}$) to denote x's (y's) j-th micro block in the i-th block. We also define $x^{[j]} \in \{0,1\}^{b_{\mathsf{m}} \cdot \ell}$ as the concatenation of $x^{1,j}, x^{2,j}, \ldots, x^{\ell,j}$ ($y^{[j]}$ is defined similarly). Then we have

$$\begin{aligned} \varphi(x) \cdot \varphi(y) &\equiv \sum_{i=1}^{\ell} \psi_{\mathsf{block}}(x^{i,j}) \cdot \psi_{\mathsf{block}}(y^{i,j}) \pmod{q_j} \\ &= \psi(x^{[j]}) \cdot \psi(y^{[j]}). \qquad\qquad (\psi(x^{[j]}) \cdot \psi(y^{[j]}) \leq b < q_j) \end{aligned}$$

Hence, for all $j \in [b/b_{\mathsf{m}}]$, we can determine whether $x^{[j]} \cdot y^{[j]} = 0$ from whether $\varphi(x) \cdot \varphi(y) \pmod{q_j} \in V_{\varphi}$, and therefore also determine whether $x \cdot y = 0$ from $\varphi(x) \cdot \varphi(y)$.

We can now observe that $|V| \leq b^{\Theta(b/b_{\mathsf{m}})}$, smaller than before; thus we get an improvement, depending on how large can b_{m} be. Clearly, the reduction ψ can also be constructed from even smaller reductions, and after recursing $\Theta(\log^* n)$ times, we can switch to the direct construction discussed before. By a straightforward (but tedious) calculation, we can derive Lemma 4.1.6.

High-Level Explanation on the $2^{O(\log^* n)}$ Factor. Ideally, we want to have a reduction from OV to $\mathbb{Z}$-OV with only $\ell^{O(b)}$ instances, in other words, we want $|V| = \ell^{O(b)}$. The reason we need to pay an extra $2^{O(\log^* n)}$ factor in the exponent is as follows:

In our reduction, $|V|$ is at least $\prod_{j=1}^{b/b_{\mathsf{m}}} q_j$, which is also the bound on each coordinate of the reduction: $\varphi(x)_i$ equals to a CRR encoding of a vector with $\{q_j\}_{j=1}^{b/b_{\mathsf{m}}}$, whose value can be as large as $\prod_{j=1}^{b/b_{\mathsf{m}}} q_j - 1$. That is, all we want is to control the upper bound on the coordinates of the reduction.

Suppose we are constructing an "outer" reduction $\varphi : \{0,1\}^{b \cdot \ell} \to \mathbb{Z}^{\ell}$ from the "micro" reduction $\psi : \{0,1\}^{b_{\mathsf{m}} \cdot \ell} \to \mathbb{Z}^{\ell}$ with coordinate upper bound L_ψ ($\psi(x)_i \le L_\psi$), and let $L_\psi = \ell^{\kappa \cdot b_{\mathsf{m}}}$ (that is, κ is the extra factor comparing to the ideal case). Recall that we have to ensure $q_j > \psi(x) \cdot \psi(y)$ to make our construction work, and therefore we have to set q_j larger than L_ψ^2.

Then the coordinate upper bound for φ becomes $L_\varphi = \prod_{j=1}^{b/b_{\mathsf{m}}} q_j \ge (L_\psi)^{2 \cdot b/b_{\mathsf{m}}} = \ell^{2\kappa \cdot b}$. Therefore, we can see that after one recursion, the "extra factor" κ at least doubles. Since our recursion proceeds in $\Theta(\log^* n)$ rounds, we have to pay an extra $2^{O(\log^* n)}$ factor on the exponent.

4.3 Preliminaries

4.3.1 Number Theory

Here we recall some facts from number theory. In our reduction from OV to $\mathbb{Z}$-OV, we will apply the famous prime number theorem, which supplies a good estimate of the number of primes smaller than a certain number. See e.g. [33] for a reference on this.

Theorem 4.3.1 (Prime Number Theorem). *Let $\pi(n)$ be the number of primes $\leq n$, then we have*

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

From a simple calculation, we obtain:

Lemma 4.3.2. *There are $10n$ distinct primes in $[n+1, n^2]$ for any sufficiently large n.*

Proof. For a sufficiently large n, from the prime number theorem, the number of primes in $[n+1, n^2]$ is equal to

$$\pi(n^2) - \pi(n) \sim n^2/2\ln n - n/\ln n \gg 10n.$$

□

Next we recall the Chinese remainder theorem, and Chinese remainder representation.

Theorem 4.3.3. *Given d pairwise co-prime integers $q_1, q_2, \ldots, q_d$, and d integers $r_1, r_2, \ldots, r_d$, there is exactly one integer $0 \leq t < \prod_{i=1}^{d} q_i$ such that*

$$t \equiv r_i \pmod{q_j} \quad \textit{for all } i \in [d].$$

We call this t the Chinese remainder representation (or the CRR encoding) of the r_i's (with respect to these q_i's). We also denote

$$t = \mathsf{CRR}(\{r_i\}; \{q_i\})$$

for convenience. We sometimes omit the sequence $\{q_i\}$ for simplicity, when it is clear from the context.

Moreover, t can be computed in polynomial time with respect to the total bits of all the given integers.

4.4 Hardness of Exact $\mathbb{Z}$-Max-IP, Hopcroft's Problem and More

In this section we show hardness of Hopcroft's problem, exact $\mathbb{Z}$-Max-IP, ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair. Essentially our results follow from the framework of [172], in which it is shown that hardness of Hopcroft's problem implies hardness of other three problems, and is implied by dimensionality reduction for OV.

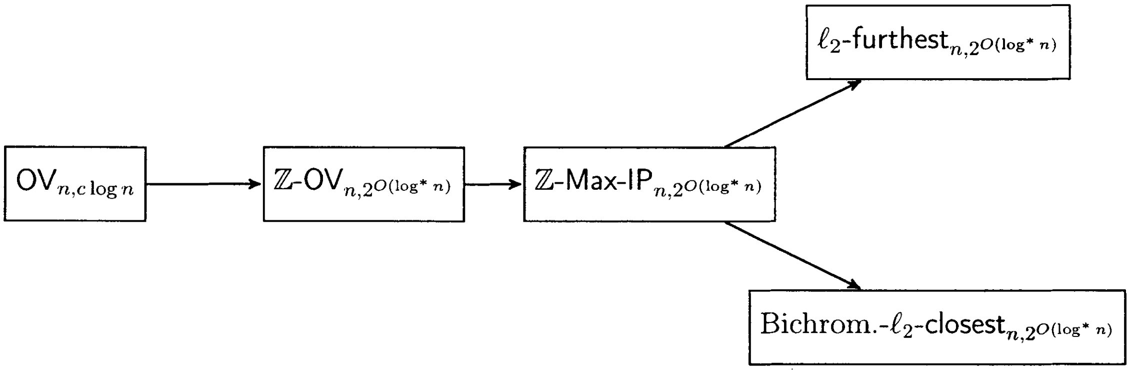
$\ell_2\text{-furthest}_{n,2^{O(\log^* n)}}$
$\mathsf{OV}_{n,c\log n}$
$\mathbb{Z}\text{-}\mathsf{OV}_{n,2^{O(\log^* n)}}$
$\mathbb{Z}\text{-Max-IP}_{n,2^{O(\log^* n)}}$
$\text{Bichrom.-}\ell_2\text{-closest}_{n,2^{O(\log^* n)}}$

Figure 4-1: A diagram for all reductions in this section.

The Organization of this Section

In Section 4.4.1, we prove the improved dimensionality reduction for OV. In Section 4.4.2, we establish the hardness of Hopcroft's problem in $2^{O(\log^* n)}$ dimensions with the improved reduction. In Section 4.4.3, we show Hopcroft's problem can be reduced to $\mathbb{Z}$-Max-IP and thus establish the hardness for the later one. In Section 4.4.4, we show $\mathbb{Z}$-Max-IP can be reduced to ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair, therefore the hardness for the last two problems follow. See Figure 4-1 for a diagram of all reductions covered in this section.

The reductions in last three subsections are all from [172] (either explicit or implicit), we make them explicit here for our ease of exposition and for making the paper self-contained.

4.4.1 Improved Dimensionality Reduction for OV

We begin with the improved dimensionality reduction for OV. The following theorem is one of the technical cores of this paper, which makes use of the CRR encoding (see Theorem 4.3.3) recursively.

Theorem 4.4.1. *Let b, ℓ be two sufficiently large integers. There is a reduction $\psi_{b,\ell} : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^{\ell}$ and a set $V_{b,\ell} \subseteq \mathbb{Z}$, such that for every $x, y \in \{0,1\}^{b\cdot\ell}$,*

$$x \cdot y = 0 \Leftrightarrow \psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y) \in V_{b,\ell}$$

and

$$0 \leq \psi_{b,\ell}(x)_i < \ell^{6^{\log^*(b)} \cdot b}$$

for all possible x and $i \in [\ell]$. Moreover, the computation of $\psi_{b,\ell}(x)$ takes $\operatorname{poly}(b \cdot \ell)$ time, and the set $V_{b,\ell}$ can be constructed in $O\left(\ell^{O(6^{\log^*(b)} \cdot b)} \cdot \operatorname{poly}(b \cdot \ell)\right)$ time.

Remark 4.4.2. *We didn't make much effort to minimize the base 6 above to keep the calculation clean, it can be replaced by any constant > 2 with a tighter calculation.*

Proof. We are going to construct our reduction in a recursive way. ℓ will be the same throughout the proof, hence in the following we use ψ_b (V_b) instead of $\psi_{b,\ell}$ ($V_{b,\ell}$) for simplicity.

Direct CRR for small b: When $b < \ell$, we use a direct Chinese remainder representation of numbers. We pick b distinct primes $q_1, q_2, \ldots, q_b$ in $[\ell+1, \ell^2]$ (they are guaranteed to exist by Lemma 4.3.2), and use them for our CRR encoding.

Let $x \in \{0,1\}^{b\cdot\ell}$, we partition it into ℓ equal size groups, and use x^i to denote the i-th group, which is the sub-vector of x from the $((i-1)\cdot b+1)$-th bit to the $(i \cdot b)$-th bit.

Then we define $\psi_b(x)$ as

$$\psi_b(x) := \left(\mathsf{CRR}\left(\left\{x^1_j\right\}_{j=1}^{b}\right), \mathsf{CRR}\left(\left\{x^2_j\right\}_{j=1}^{b}\right), \ldots, \mathsf{CRR}\left(\left\{x^\ell_j\right\}_{j=1}^{b}\right)\right).$$

That is, the i-th coordinate of $\psi_b(x)$ is the CRR encoding of the i-th sub-vector x^i with respect to the primes q_j's.

Now, for $x, y \in \{0,1\}^{b \cdot \ell}$, note that for $j \in [b]$,

$$\begin{aligned} &\psi_b(x) \cdot \psi_b(y) \pmod{q_j} \\ \equiv &\sum_{i=1}^{\ell} \mathsf{CRR}\left(\{x_j^i\}_{j=1}^{b}\right) \cdot \mathsf{CRR}\left(\{y_j^i\}_{j=1}^{b}\right) \pmod{q_j} \\ \equiv &\sum_{i=1}^{\ell} x_j^i \cdot y_j^i \pmod{q_j}. \end{aligned}$$

Since the sum $\sum_{i=1}^{\ell} x_j^i \cdot y_j^i$ is in $[0, \ell]$, and $q_j > \ell$, we can see

$$\sum_{i=1}^{\ell} x_j^i \cdot y_j^i = 0 \Leftrightarrow \psi_b(x) \cdot \psi_b(y) \equiv 0 \pmod{q_j}.$$

Therefore, $x \cdot y = \sum_{j=1}^{b} \sum_{i=1}^{\ell} x_j^i \cdot y_j^i = 0$ is equivalent to that

$$\psi_b(x) \cdot \psi_b(y) \equiv 0 \pmod{q_j}$$

for every $j \in [b]$.

Finally, we have $0 \le \psi_b(x)_i < \prod_{j=1}^{b} q_j < \ell^{2 \cdot b} \le \ell^{6^{\log^*(b)} \cdot b}$. Therefore

$$\psi_b(x) \cdot \psi_b(y) < \ell^{6^{\log^*(b)} \cdot 2b + 1},$$

and we can set V_b to be the set of all integers in $[0, \ell^{6^{\log^*(b)} \cdot 2b+1}]$ that is 0 modulo all the q_j's, and it is easy to see that

$$x \cdot y \Leftrightarrow \psi_b(x) \cdot \psi_b(y) \in V_b$$

for all $x, y \in \{0,1\}^{b \cdot \ell}$.

Recursive Construction for larger b: When $b \ge \ell$, suppose the theorem holds for all $b' < b$. Let b_{m} be the number such that (we ignore the rounding issue here and pretend that b_{m} is an integer for simplicity),

$$\ell^{6^{\log^*(b_{\mathsf{m}})} \cdot b_{\mathsf{m}}} = b.$$

Then we pick b/b_{m} primes $p_1, p_2, \dots, p_{b/b_{\mathsf{m}}}$ in $[(b^2\ell), (b^2\ell)^2]$, and use them as our reference primes in the CRR encodings.

Let $x \in \{0,1\}^{b \cdot \ell}$, as before, we partition x into ℓ equal size sub-vectors $x^1, x^2, \dots, x^\ell$, where x^i consists of the $((i-1) \cdot b + 1)$-th bit of x to the $(i \cdot b)$-th bit of x. Then we partition each x^i again into b/b_{m} micro groups, each of size b_{m}. We use $x^{i,j}$ to denote the j-th micro group of x^i after the partition.

Now, we use $x^{[j]}$ to denote the concatenation of the vectors $x^{1,j}, x^{2,j}, \dots, x^{\ell,j}$. That is, $x^{[j]}$ is the concatenation of the j-th micro group in each of the ℓ groups. Note that $x^{[j]} \in \{0,1\}^{b_{\mathsf{m}} \cdot \ell}$, and can be seen as a smaller instance, on which we can apply $\psi_{b_{\mathsf{m}}}$.

Our recursive construction then goes in two steps. In the first step, we make use of $\psi_{b_{\mathsf{m}}}$, and transform each b_{m}-size micro group into a single number in $[0, b)$. This step transforms x from a vector in $\{0,1\}^{b \cdot \ell}$ into a vector $S(x)$ in $\mathbb{Z}^{(b/b_{\mathsf{m}}) \cdot \ell}$. And in the second step, we use a similar CRR encoding as in the base case to encode $S(x)$, to get our final reduced vector in $\mathbb{Z}^\ell$.

$S(x)$ is simply

$$\begin{aligned} S(x) := \Big(& \psi_{b_{\mathsf{m}}}(x^{[1]})_1, \psi_{b_{\mathsf{m}}}(x^{[2]})_1, \dots, \psi_{b_{\mathsf{m}}}(x^{[b/b_{\mathsf{m}}]})_1, \\ & \psi_{b_{\mathsf{m}}}(x^{[1]})_2, \psi_{b_{\mathsf{m}}}(x^{[2]})_2, \dots, \psi_{b_{\mathsf{m}}}(x^{[b/b_{\mathsf{m}}]})_2, \\ & \dots, \dots, \dots \\ & \psi_{b_{\mathsf{m}}}(x^{[1]})_\ell, \psi_{b_{\mathsf{m}}}(x^{[2]})_\ell, \dots, \psi_{b_{\mathsf{m}}}(x^{[b/b_{\mathsf{m}}]})_\ell \Big). \end{aligned}$$

That is, we apply $\psi_{b_{\mathsf{m}}}$ on all the $x^{[j]}$'s, and shrink all the corresponding micro-groups in x into integers. Again, we partition $S = S(x)$ into ℓ equal size groups $S^1, S^2, \dots, S^\ell$.

Then we define $\psi_b(x)$ as

$$\psi_b(x) := \left(\mathsf{CRR}\left(\{S_j^1\}_{j=1}^{b/b_{\mathsf{m}}}\right), \mathsf{CRR}\left(\{S_j^2\}_{j=1}^{b/b_{\mathsf{m}}}\right), \dots, \mathsf{CRR}\left(\{S_j^\ell\}_{j=1}^{b/b_{\mathsf{m}}}\right)\right).$$

In other words, the i-th coordinate of $\psi_b(x)$ is the CRR representation of the number sequence S^i, with respect to our primes $\{q_j\}_{j=1}^{b/b_{\mathsf{m}}}$.

Now, note that for $x, y \in \{0,1\}^{b\cdot\ell}$, $x \cdot y = 0$ is equivalent to $x^{[j]} \cdot y^{[j]} = 0$ for every $j \in [b/b_{\mathsf{m}}]$, which is further equivalent to

$$\psi_{b_{\mathsf{m}}}(x^{[j]}) \cdot \psi_{b_{\mathsf{m}}}(y^{[j]}) \in V_{b_{\mathsf{m}}}$$

for all $j \in [b/b_{\mathsf{m}}]$, by our assumption on $\psi_{b_{\mathsf{m}}}$.

Since $0 \le \psi_{b_{\mathsf{m}}}(x^{[j]})_i, \psi_{b_{\mathsf{m}}}(y^{[j]})_i < b$ for all $x, y \in \{0,1\}^{b\cdot\ell}$, $i \in [\ell]$ and $j \in [b/b_{\mathsf{m}}]$, we also have $\psi_{b_{\mathsf{m}}}(x^{[j]}) \cdot \psi_{b_{\mathsf{m}}}(y^{[j]}) < b^2 \cdot \ell$, therefore we can assume that $V_{b_{\mathsf{m}}} \subseteq [0, b^2\ell)$.

For all $x, y \in \{0,1\}^{b\cdot\ell}$ and $j \in [b/b_{\mathsf{m}}]$, we have

$$\begin{aligned}
&\psi_b(x) \cdot \psi_b(y) \\
\equiv&\sum_{i=1}^{\ell} \mathsf{CRR}\left(\{S(x)_j^i\}_{j=1}^{b/b_{\mathsf{m}}}\right) \cdot \mathsf{CRR}\left(\{S(y)_j^i\}_{j=1}^{b/b_{\mathsf{m}}}\right) \pmod{p_j} \\
\equiv&\sum_{i=1}^{\ell} S(x)_j^i \cdot S(y)_j^i \pmod{p_j} \\
\equiv&\sum_{i=1}^{\ell} \psi_{b_{\mathsf{m}}}(x^{[j]})_i \cdot \psi_{b_{\mathsf{m}}}(y^{[j]})_i \pmod{p_j} \\
\equiv&\psi_{b_{\mathsf{m}}}(x^{[j]}) \cdot \psi_{b_{\mathsf{m}}}(y^{[j]}) \pmod{p_j}.
\end{aligned}$$

Since $p_j \ge b^2 \cdot \ell$, we can determine $\psi_{b_{\mathsf{m}}}(x^{[j]}) \cdot \psi_{b_{\mathsf{m}}}(y^{[j]})$ from $\psi_b(x) \cdot \psi_b(y)$ by taking modulo p_j. Therefore,

$$x \cdot y = 0$$

is equivalent to

$$(\psi_b(x) \cdot \psi_b(y) \mod p_j) \in V_{b_{\mathsf{m}}},$$

for every $j \in [b/b_{\mathsf{m}}]$.

Finally, recall that we have

$$\ell^{6^{\log^*(b_{\mathsf{m}})} \cdot b_{\mathsf{m}}} = b.$$

Taking logarithm of both sides, we have

$$6^{\log^*(b_{\mathsf{m}})} \cdot b_{\mathsf{m}} \cdot \log \ell = \log b.$$

Then we can upper bound $\psi_b(x)_i$ by

$$\begin{aligned}
\psi_b(x)_i &< \prod_{j=1}^{b/b_{\mathsf{m}}} p_j \\
&< (b^2\ell)^{2\cdot(b/b_{\mathsf{m}})} && (b \geq \ell.) \\
&\leq 2^{6\cdot b/b_{\mathsf{m}} \cdot \log b} \\
&\leq 2^{6\cdot b/b_{\mathsf{m}} \cdot 6^{\log^*(b_{\mathsf{m}})} \cdot b_{\mathsf{m}} \cdot \log \ell} \\
&\leq \ell^{6\cdot 6^{\log^*(b_{\mathsf{m}})} \cdot b} \\
&\leq \ell^{6^{\log^*(b)} \cdot b} && (b_{\mathsf{m}} \leq \log b, \log^*(b_{\mathsf{m}}) + 1 \leq \log^*(\log b) + 1 = \log^*(b).)
\end{aligned}$$

Therefore, we can set V_b as the set of integer t in $[0, \ell^{6^{\log^*(b)} \cdot 2b+1})$ such that

$$(t \mod p_j) \in V_{b_{\mathsf{m}}}$$

for every $j \in [b/b_{\mathsf{m}}]$. And it is easy to see this V_b satisfies our requirement.

Finally, it is easy to see that the straightforward way of constructing $\psi_b(x)$ takes $O(\mathrm{poly}(b \cdot \ell))$ time, and we can construct V_b by enumerating all possible values of $\psi_b(x) \cdot \psi_b(y)$ and check each of them in $O(\mathrm{poly}(b \cdot \ell))$ time. Since there are at most $\ell^{O(6^{\log^*(b)} \cdot b)}$ such values, V_b can be constructed in

$$O\left(\ell^{O(6^{\log^*(b)} \cdot b)} \cdot \mathrm{poly}(b \cdot \ell)\right)$$

time, which completes the proof.

□

Now we prove Lemma 4.1.6, we recap its statement here for convenience.

Reminder of Lemma 4.1.6 *Let $1 \le \ell \le d$. There is an*

$$O\left(n \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))} \cdot \mathrm{poly}(d)\right)\textit{-time}$$

reduction from $\mathsf{OV}_{n,d}$ to $\ell^{O(6^{\log^* d} \cdot (d/\ell))}$ instances of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$, with vectors of entries with bit-length $O\left(d/\ell \cdot \log \ell \cdot 6^{\log^* d}\right)$.

Proof. The proof is exactly the same as the proof for Lemma 1.1 in [172] with different parameters, we recap it here for convenience.

Given two sets A' and B' of n vectors from $\{0,1\}^d$, we apply $\psi_{d/\ell,\ell}$ to each of the vectors in A' (B') to obtain a set A (B) of vectors from $\mathbb{Z}^\ell$. From Theorem 4.4.1, there is a $(u,v) \in A' \times B'$ such that $u \cdot v = 0$ if and only if there is a $(u,v) \in A \times B$ such that $u \cdot v \in V_{d/\ell,\ell}$.

Now, for each element $t \in V_{d/\ell,\ell}$, we are going to construct two sets A_t and B_t of vectors from $\mathbb{Z}^{\ell+1}$ such that there is a $(u,v) \in A \times B$ with $u \cdot v = t$ if and only if there is a $(u,v) \in A_t \times B_t$ with $u \cdot v = 0$. We construct a set A_t as a collection of all vectors $u_A = [u, 1]$ for $u \in A$, and a set B_t as a collection of all vectors $v_B = [v, -t]$ for $v \in B$. It is easy to verify this reduction has the properties we want.

Note that there are at most $\ell^{O(6^{\log^* d} \cdot (d/\ell))}$ numbers in $V_{d/\ell,\ell}$, so we have such a number of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ instances. And from Theorem 4.4.1, the reduction takes

$$O\left(n \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))} \cdot \mathrm{poly}(d)\right)$$

time.

Finally, the bit-length of reduced vectors is bounded by

$$\log\left(\ell^{O(6^{\log^* d}\cdot(d/\ell))}\right) = O\left(d/\ell \cdot \log \ell \cdot 6^{\log^* d}\right),$$

which completes the proof. □

A Transformation from Nonuniform Construction to Uniform Construction

The proof for Theorem 4.4.1 works recursively. In one recursive step, we reduce the construction of $\psi_{b,\ell}$ to the construction of $\psi_{b_{\mathsf{m}},\ell}$, where $b_{\mathsf{m}} \le \log b$. Applying this reduction $\log^* n$ times, we get a sufficiently small instance that we can switch to a direct CRR construction.

An interesting observation here is that after applying the reduction only thrice, the block length parameter becomes $b' \le \log\log\log b$, which is so small that we can actually use brute force to find the "optimal" construction $\psi_{b',\ell}$ in $b^{o(1)}$ time instead of recursing deeper. Hence, to find a construction better than Theorem 4.4.1, we only need to prove the existence of such a construction. See Section 4.7 for details.

4.4.2 Improved Hardness for Hopcroft's Problem

In this subsection we are going to prove Theorem 4.1.7 using our new dimensionality reduction Lemma 4.1.6, we recap its statement here for completeness.

Reminder of Theorem 4.1.7 *[Hardness of Hopcroft's Problem in $c^{\log^* n}$ Dimension] Assuming SETH (or OVC), there is a constant c such that $\mathbb{Z}$-$\mathsf{OV}_{n,c^{\log^* n}}$ with vectors of $O(\log n)$-bit entries requires $n^{2-o(1)}$ time.*

Proof. The proof here follows roughly the same as the proof for Theorem 1.1 in [172].

Let c be an arbitrary constant and $d := c \cdot \log n$. We show that an algorithm $\mathbb{A}$ solving $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$ where $\ell = 7^{\log^* n}$ in $O(n^{2-\delta})$ time for some $\delta > 0$ can be used to construct an $O(n^{2-\delta+o(1)})$ time algorithm for $\mathsf{OV}_{n,d}$, and therefore contradicts the OVC.

We simply invoke Lemma 4.1.6, note that we have

$$\begin{aligned}\log\left\{\ell^{O\left(6^{\log^* d}\cdot(d/\ell)\right)}\right\} &= \log\ell\cdot O\left(6^{\log^* d}\cdot(d/\ell)\right)\\ &= O\left(\log^* n\cdot 6^{\log^* n}\cdot c\cdot\log n/7^{\log^* n}\right)\\ &= O\left(\log^* n\cdot(6/7)^{\log^* n}\cdot c\cdot\log n\right)\\ &= o(\log n).\end{aligned}$$

Therefore, the reduction takes $O(n\cdot\ell^{O\left(6^{\log^* d}\cdot(d/\ell)\right)}\cdot\mathrm{poly}(d)) = n^{1+o(1)}$ time, and an $\mathsf{OV}_{n,d}$ instance is reduced to $n^{o(1)}$ instances of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$, and the reduced vectors have bit length $o(\log n)$ as calculated above. We simply solve all these $n^{o(1)}$ instances using $\mathbb{A}$, and this gives us an $O(n^{2-\delta+o(1)})$ time algorithm for $\mathsf{OV}_{n,d}$, which completes the proof. □

4.4.3 Hardness for $\mathbb{Z}$-Max-IP

Now we move to hardness of exact $\mathbb{Z}$-Max-IP.

Theorem 4.4.3 (Implicit in Theorem 1.2 [172]). *There is an $O(\mathrm{poly}(d)\cdot n)$-time algorithm which reduces a $\mathbb{Z}$-$\mathsf{OV}_{n,d}$ instance into a $\mathbb{Z}$-Max-IP_{n,d^2} instance.*

Proof. We remark here that this reduction is implicitly used in the proof of Theorem 1.2 in [172], we abstract it here only for our exposition.

Given a $\mathbb{Z}$-$\mathsf{OV}_{n,d}$ instance with sets A, B. Consider the following polynomial $P(x,y)$, where $x, y \in \mathbb{Z}^d$.

$$P(x,y) = -(x\cdot y)^2 = \sum_{i,j\in[d]} -(x_i\cdot y_i)\cdot(x_j\cdot y_j) = \sum_{i,j\in[d]} -(x_i\cdot x_j)\cdot(y_i\cdot y_j).$$

It is easy to see that whether there is a $(x,y)\in A\times B$ such that $x\cdot y = 0$ is equivalent to whether the maximum value of $P(x,y)$ is 0.

Now, for each $x\in A$ and $y\in B$, we identify $[d^2]$ with $[d]\times[d]$ and construct

$\widetilde{x}, \widetilde{y} \in \mathbb{Z}^{d^2}$ such that

$$\widetilde{x}_{(i,j)} = x_i \cdot x_j \quad \text{and} \quad \widetilde{y}_{(i,j)} = -y_i \cdot y_j.$$

Then we have $\widetilde{x} \cdot \widetilde{y} = P(x, y)$. Hence, let $\widetilde{A}$ be the set of all these $\widetilde{x}$'s, and $\widetilde{B}$ be the set of all these $\widetilde{y}$'s, whether there is a $(x, y) \in A \times B$ such that $x \cdot y = 0$ is equivalent to whether $\mathsf{OPT}(\widetilde{A}, \widetilde{B}) = 0$, and our reduction is completed.

□

Now, Theorem 4.1.3 (restated below) is just a simple corollary of Theorem 4.4.3 and Theorem 4.1.7.

Reminder of Theorem 4.1.3 *Assuming SETH (or OVC), there is a constant c such that every exact algorithm for $\mathbb{Z}$-Max-IP$_{n,d}$ for $d = c^{\log^* n}$ dimensions requires $n^{2-o(1)}$ time, with vectors of $O(\log n)$-bit entries.*

A Dimensionality Reduction for Max-IP

The reduction $\psi_{b,\ell}$ from Theorem 4.4.1 actually does more: for $x, y \in \{0,1\}^{b \cdot \ell}$, from $\psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y)$ we can in fact determine the inner product $x \cdot y$ itself, not only whether $x \cdot y = 0$.

Starting from this observation, together with Theorem 4.4.3, we can in fact derive a similar dimensionality self reduction from Max-IP to $\mathbb{Z}$-Max-IP, we deter its proof to Section 4.6.

Corollary 4.4.4. *Let $1 \le \ell \le d$. There is an*

$$O\left(n \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))} \cdot \mathrm{poly}(d)\right) \textit{-time}$$

reduction from Max-IP$_{n,d}$ to $d \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))}$ instances of $\mathbb{Z}$-Max-IP$_{n,(\ell+1)^2}$, with vectors of entries with bit-length $O\left(d/\ell \cdot \log \ell \cdot 6^{\log^* d}\right)$.

4.4.4 Hardness for ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair

We finish the whole section with the proof of hardness of ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair. The two reductions below are slight adaptations of the ones in the proofs of Theorem 1.2 and Corollary 2.1 in [172].

Lemma 4.4.5. *Assuming $d = n^{o(1)}$, there is an $O(\mathrm{poly}(d) \cdot n)$-time algorithm which reduces a $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ instance into an instance of ℓ_2-Furthest Pair on $2n$ points in $\mathbb{R}^{d+2}$. Moreover, if the $\mathbb{Z}$-Max-IP instance consists of vectors of $O(\log n)$-bit entries, so does the ℓ_2-Furthest Pair instance.*

Proof. Let A, B be the sets in the $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ instance, and k be the smallest integer such that all vectors from A and B consist of $(k \cdot \log n)$-bit entries.

Let W be $n^{C \cdot k}$ where C is a large enough constant. Given $x \in A$ and $y \in B$, we construct point

$$\widetilde{x} = \left(x, \sqrt{W - \|x\|^2}, 0\right) \quad \text{and} \quad \widetilde{y} = \left(-y, 0, \sqrt{W - \|y\|^2}\right),$$

that is, appending two corresponding values into the end of vectors x and $-y$.

Now, we can see that for $x_1, x_2 \in A$, the squared distance between their reduced points is

$$\|\widetilde{x_1} - \widetilde{x_2}\|^2 \le \|x_1 - x_2\|^2 + W \le 4 \cdot d \cdot n^{2k} + W.$$

Similarly we have

$$\|\widetilde{y_1} - \widetilde{y_2}\|^2 \le 4 \cdot d \cdot n^{2k} + W$$

for $y_1, y_2 \in B$.

Next, for $x \in A$ and $y \in B$, we have

$$\|\widetilde{x} - \widetilde{y}\|^2 = \|\widetilde{x}\|^2 + \|\widetilde{y}\|^2 - 2 \cdot \widetilde{x} \cdot \widetilde{y} = 2 \cdot W + 2 \cdot (x \cdot y) \ge 2 \cdot W - d \cdot n^{2k} \gg 4 \cdot d \cdot n^{2k} + W,$$

the last inequality holds when we set C to be 5.

Putting everything together, we can see the ℓ_2-furthest pair among all points $\widetilde{x}$'s and $\widetilde{y}$'s must be a pair of $\widetilde{x}$ and $\widetilde{y}$ with $x \in A$ and $y \in B$. And maximizing $\|\widetilde{x} - \widetilde{y}\|$ is equivalent to maximize $x \cdot y$, which proves the correctness of our reduction. Furthermore, when k is a constant, the reduced instance clearly only needs vectors with $O(k) \cdot \log n = O(\log n)$-bit entries. □

Lemma 4.4.6. *Assuming $d = n^{o(1)}$, there is an $O(\text{poly}(d) \cdot n)$-time algorithm which reduces a $\mathbb{Z}$-Max-IP$_{n,d}$ instance into an instance of Bichromatic ℓ_2-Closest Pair on $2n$ points in $\mathbb{R}^{d+2}$. Moreover, if the $\mathbb{Z}$-Max-IP instance consists of vectors of $O(\log n)$-bit entries, so does the Bichromatic ℓ_2-Closest Pair instance.*

Proof. Let A, B be the sets in the $\mathbb{Z}$-Max-IP$_{n,d}$ instance, and k be the smallest integer such that all vectors from A and B consist of $(k \cdot \log n)$-bit entries.

Let W be $n^{C \cdot k}$ where C is a large enough constant. Given $x \in A$ and $y \in B$, we construct point

$$\widetilde{x} = \left(x, \sqrt{W - \|x\|^2}, 0\right) \quad \text{and} \quad \widetilde{y} = \left(y, 0, \sqrt{W - \|y\|^2}\right),$$

that is, appending two corresponding values into the end of vectors x and y. And our reduced instance is to find the closest point between the set $\widetilde{A}$ (consisting of all these $\widetilde{x}$ where $x \in A$) and the set $\widetilde{B}$ (consisting of all these $\widetilde{y}$ where $y \in B$).

Next, for $x \in A$ and $y \in B$, we have

$$\|\widetilde{x} - \widetilde{y}\|^2 = \|\widetilde{x}\|^2 + \|\widetilde{y}\|^2 - 2 \cdot \widetilde{x} \cdot \widetilde{y} = 2 \cdot W - 2 \cdot (x \cdot y) \geq 2 \cdot W - d \cdot n^{2k} \gg 4 \cdot d \cdot n^{2k},$$

the last inequality holds when we set C to be 5.

Hence minimizing $\|\widetilde{x} - \widetilde{y}\|$ where $x \in A$ and $y \in B$ is equivalent to maximize $x \cdot y$, which proves the correctness of our reduction. Furthermore, when k is a constant, the reduced instance clearly only needs vectors with $O(k) \cdot \log n = O(\log n)$-bit entries. □

Now Theorem 4.1.4 and Theorem 4.1.5 are simple corollaries of Lemma 4.4.5, Lemma 4.4.6 and Theorem 4.1.3.

4.5 NP · UPP Communication protocols and Exact Hardness for $\mathbb{Z}$-Max-IP

We note that the inapproximability results for (Boolean) Max-IP is established via a connection to the MA communication complexity protocol of Set-Disjointness [12]. In the light of this, in this section we view our reduction from OV to $\mathbb{Z}$-Max-IP (Lemma 4.1.6 and Theorem 4.4.3) in the perspective of communication complexity.

We observe that in fact, our reduction can be understood as an NP · UPP communication protocol for Set Disjointness. Moreover, we show that if we can get a slightly better NP · UPP communication protocol for Set-Disjointness, then we would be able to prove $\mathbb{Z}$-Max-IP is hard even for $\omega(1)$ dimensions (and also ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair).

4.5.1 An NP·UPP Communication Protocol for Set-Disjointness

First, we rephrase the results of Lemma 4.1.6 and Theorem 4.4.3 in a more convenience way for our use here.

Lemma 4.5.1 (Rephrasing of Lemma 4.1.6 and Theorem 4.4.3). *Let $1 \le \ell \le d$, and $m = \ell^{O(6^{\log^* d} \cdot (d/\ell))}$. There exists a family of functions*

$$\psi^i_{\mathsf{Alice}}, \psi^i_{\mathsf{Bob}} : \{0,1\}^d \to \mathbb{R}^{(\ell+1)^2}$$

for $i \in [m]$ such that:

- *when $x \cdot y = 0$, there is an i such that $\psi^i_{\mathsf{Alice}}(x) \cdot \psi^i_{\mathsf{Bob}}(y) \ge 0$;*
- *when $x \cdot y > 0$, for all i $\psi^i_{\mathsf{Alice}}(x) \cdot \psi^i_{\mathsf{Bob}}(y) < 0$;*
- *all $\psi^i_{\mathsf{Alice}}(x)$ and $\psi^i_{\mathsf{Bob}}(y)$ can be computed in* poly(d) *time.*

We also need the standard connection between UPP communication protocols and sign-rank [143] (see also Chapter 4.11 of [105]).

Recall that for a function $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, a UPP protocol for F is a private-coin randomized communication protocol such that: Alice and Bob hold $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively; $F(x, y) = 1$ if and only if Alice and Bob accepts with probability $> 1/2$. The cost of the protocol is the maximum bits communicated over all pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and Alice and Bob's corresponding private random coins.

Lemma 4.5.2 (Equivalence of sign-rank and UPP communication protocol (Theorem 3 of [143])). *The following holds*

- *There is a d-cost UPP protocol for F implies that for $\ell = d + 1$, there are mappings $\psi^{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{2^\ell}$ and $\psi^{\mathcal{Y}} : \mathcal{Y} \to \mathbb{R}^{2^\ell}$ such that for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$:*
  - *if $F(x, y) = 1$, $\psi^{\mathcal{X}}(x) \cdot \psi^{\mathcal{Y}}(y) \geq 0$;*
  - *otherwise, $\psi^{\mathcal{X}}(x) \cdot \psi^{\mathcal{Y}}(y) < 0$.*
- *There are mappings $\psi^{\mathcal{X}} : \mathcal{X} \to \mathbb{R}^{2^\ell}$ and $\psi^{\mathcal{Y}} : \mathcal{Y} \to \mathbb{R}^{2^\ell}$ satisfying the above conditions implies that for $d = \ell + 1$, there is a d-cost UPP protocol for F.*

From the above lemmas, we immediately get the needed communication protocol and prove Theorem 4.1.10 (restated below for convenience).

Reminder of Theorem 4.1.10 *For all $1 \leq \alpha \leq n$, there is an*

$$\left(\alpha \cdot 6^{\log^* n} \cdot (n/2^\alpha), O(\alpha)\right)\text{-computational-efficient}$$

NP $\cdot$ UPP communication protocol for DISJ_n.

Proof Sketch. We set $\alpha = \log \ell$ here. Given the function families $\{\psi^i_{\mathsf{Alice}}\}, \{\psi^i_{\mathsf{Bob}}\}$ from Lemma 4.5.1, Merlin just sends the index $i \in [m]$ and the rest follows from Lemma 4.5.2. □

4.5.2 Slightly Better Protocols Imply Hardness in $\omega(1)$ Dimensions

Finally, we show that if we have a slightly better $\mathsf{NP}\cdot\mathsf{UPP}$ protocol for Set-Disjointness, then we can show $\mathbb{Z}$-Max-IP requires $n^{2-o(1)}$ time even for $\omega(1)$ dimensions (and so do ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair). We restate Theorem 4.1.11 here for convenience.

Reminder of Theorem 4.1.11 *Assuming SETH (or OVC), if there is an increasing and unbounded function f such that for all $1 \le \alpha \le n$, there is a*

$$(n/f(\alpha), \alpha)\text{-}\textit{computational-efficient}$$

$\mathsf{NP}\cdot\mathsf{UPP}$ communication protocol for DISJ_n, then $\mathbb{Z}$-Max-$\mathsf{IP}_{n,\omega(1)}$ requires $n^{2-o(1)}$ time with vectors of polylog(n)*-bit entries. The same holds for ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair.*

Proof. Suppose otherwise, there is an algorithm $\mathbb{A}$ for $\mathbb{Z}$-Max-$\mathsf{IP}_{n,d}$ running in $n^{2-\varepsilon_1}$ time for all constant d and for a constant $\varepsilon_1 > 0$ (note from Lemma 4.4.5 and Lemma 4.4.6, we only need to consider $\mathbb{Z}$-Max-IP here).

Now, let c be an arbitrary constant, we are going to construct an algorithm for $\mathsf{OV}_{n,c\log n}$ in $n^{2-\Omega(1)}$ time, which contradicts OVC.

Let $\varepsilon = \varepsilon_1/2$, and α be the first number such that $c/f(\alpha) < \varepsilon$, note that α is also a constant. Consider the $(c\log n/f(\alpha), \alpha)$-computational-efficient $\mathsf{NP}\cdot\mathsf{UPP}$ protocol Π for $\mathsf{DISJ}_{c\log n}$, and let A, B be the two sets in the $\mathsf{OV}_{n,c\log n}$ instance. Our algorithm via reduction works as follows:

- There are 2^α possible messages in $\{0,1\}^\alpha$, let $m_1, m_2, \dots, m_{2^\alpha}$ be an enumeration of them.

- We first enumerate all possible advice strings from Merlin in Π, there are $2^{c\log n/f(\alpha)} \le 2^{\varepsilon\cdot\log n} = n^\varepsilon$ such strings, let $\phi \in \{0,1\}^{\varepsilon\cdot\log n}$ be such an advice string.

  - For each $x \in A$, let $\psi_{\mathsf{Alice}}(x) \in \mathbb{R}^{2^\alpha}$ be the probabilities that Alice accepts each message from Bob. That is, $\psi_{\mathsf{Alice}}(x)_i$ is the probability that Alice accepts the message m_i, given its input x and the advice ϕ.
  - Similarly, for each $y \in B$, let $\psi_{\mathsf{Bob}}(y) \in \mathbb{R}^{2^\alpha}$ be the probabilities that Bob sends each message. That is, $\psi_{\mathsf{Bob}}(y)_i$ is the probability that Bob sends the message m_i, give its input y and the advice ϕ.
  - Then, for each $x \in A$ and $y \in B$, $\psi_{\mathsf{Alice}}(x) \cdot \psi_{\mathsf{Bob}}(y)$ is precisely the probability that Alice accepts at the end when Alice and Bob holds x and y correspondingly and the advice is ϕ. Now we let A_ϕ be the set of all the $\psi_{\mathsf{Alice}}(x)$'s, and B_ϕ be the set of all the $\psi_{\mathsf{Bob}}(y)$'s.

- If there is a ϕ such that $\mathsf{OPT}(A_\phi, B_\phi) \ge 1/2$, then we output yes, and otherwise output no.

From the definition of Π, it is straightforward to see that the above algorithm solves $\mathsf{OV}_{n,c\cdot\log n}$. Moreover, notice that from the computational-efficient property of Π, the reduction itself works in $n^{1+\varepsilon} \cdot \mathrm{polylog}(n)$ time, and all the vectors in A_ϕ's and B_ϕ's have at most $\mathrm{polylog}(n)$ bit precision, which means $\mathsf{OPT}(A_\phi, B_\phi)$ can be solved by a call to $\mathbb{Z}\text{-}\mathsf{Max}\text{-}\mathsf{IP}_{n,2^\alpha}$ with vectors of $\mathrm{polylog}(n)$-bit entries.

Hence, the final running time for the above algorithm is bounded by $n^\varepsilon \cdot n^{2-\varepsilon_1} = n^{2-\varepsilon}$ (2^α is still a constant), which contradicts the OVC. □

4.6 A Dimensionality Reduction for Max-IP

Tracing the proof of Theorem 4.4.1, we observe that it is possible to compute the inner product $x \cdot y$ itself from $\psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y)$, that is:

Corollary 4.6.1. *Let b, ℓ be two sufficiently large integers. There is a reduction $\psi_{b,\ell} : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^\ell$ and $b \cdot \ell + 1$ sets $V_{b,\ell}^0, V_{b,\ell}^1, \dots, V_{b,\ell}^{b\cdot\ell} \subseteq \mathbb{Z}$, such that for every $x, y \in \{0,1\}^{b\cdot\ell}$,*

$$x \cdot y = k \Leftrightarrow \psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y) \in V_{b,\ell}^k \quad \text{for all } 0 \le k \le b \cdot \ell,$$

and

$$0 \leq \psi_{b,\ell}(x)_i < \ell^{6^{\log^*(b)} \cdot b}$$

for all possible x and $i \in [\ell]$. Moreover, the computation of $\psi_{b,\ell}(x)$ takes $\operatorname{poly}(b \cdot \ell)$ *time, and the sets $V_{b,\ell}^k$'s can be constructed in $O\left(\ell^{O(6^{\log^*(b)} \cdot b)} \cdot \operatorname{poly}(b \cdot \ell)\right)$ time.*

Together with Theorem 4.4.3, it proves Corollary 4.4.4 (restated below).

Reminder of Corollary 4.4.4 *Let $1 \leq \ell \leq d$. There is an*

$$O\left(n \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))} \cdot \operatorname{poly}(d)\right) \text{-time}$$

reduction from $\mathsf{Max\text{-}IP}_{n,d}$ to $d \cdot \ell^{O(6^{\log^* d} \cdot (d/\ell))}$ instances of $\mathbb{Z}$-$\mathsf{Max\text{-}IP}_{n,(\ell+1)^2}$, with vectors of entries with bit-length $O\left(d/\ell \cdot \log \ell \cdot 6^{\log^* d}\right)$.

Proof Sketch. Let $b = d/\ell$ (assume ℓ divides d here for simplicity), A and B be the sets in the given $\mathsf{Max\text{-}IP}_{n,d}$ instance, we proceed similarly as the case for OV.

We first enumerate a number k from 0 to d, for each k we construct the set $V_{b,\ell}^k$ as specified in Corollary 4.6.1. Then there is $(x, y) \in A \times B$ such that $x \cdot y = k$ if and only if there is $(x, y) \in A \times B$ such that $\psi_{b,\ell}(x) \cdot \psi_{b,\ell}(y) \in V_{b,\ell}^k$. Using exactly the same reduction as in Lemma 4.1.6, we can in turn reduce this into $\ell^{O(6^{\log^*(b)} \cdot b)}$ instances of $\mathbb{Z}$-$\mathsf{OV}_{n,\ell+1}$.

Applying Theorem 4.4.3, with evaluation of $(d+1) \cdot \ell^{O(6^{\log^*(b)} \cdot b)}$ $\mathbb{Z}$-$\mathsf{Max\text{-}IP}_{n,(\ell+1)^2}$ instances, we can determine whether there is $(x, y) \in A \times B$ such that $x \cdot y = k$ for every k, from which we can compute the answer to the $\mathsf{Max\text{-}IP}_{n,d}$ instance. □

4.7 Nonuniform to Uniform Transformation for Dimensionality Reduction for OV

In this section we discuss the transformation from nonuniform construction to uniform one for dimensionality reduction for OV. In order to state our result formally, we need to introduce some definitions.

Definition 4.7.1 (Nonuniform Reduction). Let $b, \ell, \kappa \in \mathbb{N}$. We say a function $\varphi : \{0,1\}^{b\cdot\ell} \to \mathbb{Z}^{\ell}$ together with a set $V \subseteq \mathbb{Z}$ is a (b, ℓ, κ)-reduction, if the following holds:

- For every $x, y \in \{0,1\}^{b\cdot\ell}$,

$$x \cdot y = 0 \Leftrightarrow \varphi(x) \cdot \varphi(y) \in V.$$

- For every x and $i \in [\ell]$,

$$0 \leq \varphi(x)_i < \ell^{\kappa \cdot b}.$$

Similarly, let τ be an increasing function, we say a function family $\{\varphi_{b,\ell}\}_{b,\ell}$ together with a set family $\{V_{b,\ell}\}_{b,\ell}$ is a τ-reduction family, if for every b and ℓ, $(\varphi_{b,\ell}, V_{b,\ell})$ is a $(b, \ell, \tau(b))$-reduction.

Moreover, if for all b and all $\ell \leq \log\log\log b$, there is an algorithm $\mathbb{A}$ which computes $\varphi_{b,\ell}(x)$ in $\mathrm{poly}(b)$ time given b, ℓ and $x \in \{0,1\}^{b\cdot\ell}$, and constructs the set $V_{b,\ell}$ in $O\left(\ell^{O(\tau(b)\cdot b)} \cdot \mathrm{poly}(b)\right)$ time given b and ℓ, then we call $(\varphi_{b,\ell}, V_{b,\ell})$ a uniform-τ-reduction family.

Remark 4.7.2. *The reason we assume ℓ to be small is that in our applications we only care about very small ℓ, and that greatly simplifies the notation. From Theorem 4.4.1, there is a uniform-$(6^{\log^* b})$-reduction family, and a better uniform-reduction family implies better hardness for $\mathbb{Z}$-OV and other related problems as well (Lemma 4.1.6, Theorem 4.4.3, Lemma 4.4.6 and Lemma 4.4.5).*

Now we are ready to state our nonuniform to uniform transformation result formally.

Theorem 4.7.3. *Letting τ be an increasing function such that $\tau(n) = O(\log\log\log n)$ and supposing there is a τ-reduction family, then there is a uniform-$O(\tau)$-reduction family.*

Proof Sketch. The construction in Theorem 4.4.1 is recursive, it constructs the reduction $\psi_{b,\ell}$ from a much smaller reduction $\psi_{b_{\mathsf{m}},\ell}$, where $b_{\mathsf{m}} \leq \log b$. In the original construction, it takes $\log^* b$ recursions to make the problem sufficiently small so that a direct construction can be used. Here we only apply the reduction thrice. First let us abstract the following lemma from the proof of Theorem 4.4.1.

Lemma 4.7.4 (Implicit in Theorem 4.4.1). *Letting $b, \ell, b_{\mathsf{m}}, \kappa \in \mathbb{N}$ and supposing $\ell^{\kappa \cdot b_{\mathsf{m}}} = b$ and there is a $(b_{\mathsf{m}}, \ell, \kappa)$-reduction (φ, V'), the following holds:*

- *There is a $(b, \ell, 6 \cdot \kappa)$-reduction (ψ, V).*
- *Given (φ, V'), for all $x \in \{0,1\}^{b \cdot \ell}$, $\psi(x)$ can be computed in* $\operatorname{poly}(b \cdot \ell)$*, and V can be constructed in $O\left(\ell^{O(\kappa \cdot b)} \cdot \operatorname{poly}(b \cdot \ell)\right)$ time.*

Now, let $b, \ell \in \mathbb{N}$, we are going to construct our reduction as follows.

Let b_1 be the number such that

$$\ell^{\tau(b) \cdot 6^2 \cdot b_1} = b,$$

and similarly we set b_2 and b_3 so that

$$\ell^{\tau(b) \cdot 6 \cdot b_2} = b_1 \quad \text{and} \quad \ell^{\tau(b) \cdot b_3} = b_2.$$

We can calculate from above that $b_3 \leq \log \log \log b$.

From the assumption that there is a τ-reduction, there is a $(b_3, \ell, \tau(b_3))$-reduction $(\varphi_{b_3,\ell}, V_{b_3,\ell})$, which is also a $(b_3, \ell, \tau(b))$-reduction, as τ is increasing. Note that we can assume $\ell \leq \log \log \log b$ and $\tau(b) \leq \log \log \log b$ from assumption. Now we simply use a brute force algorithm to find $(\varphi_{b_3,\ell}, V_{b_3,\ell})$. There are

$$\ell^{\tau(b) \cdot b_3 \cdot \ell \cdot 2^{b_3 \cdot \ell}} = b^{o(1)}$$

possible functions from $\{0,1\}^{b_3 \cdot \ell} \to \{0, \ldots \ell^{\tau(b_3) \cdot b_3} - 1\}^{\ell}$. Given such a function φ, one can check in $\operatorname{poly}(2^{b_3 \cdot \ell}) = b^{o(1)}$ time that whether one can construct a corresponding set V to obtain our $(b_3, \ell, \tau(b))$-reduction.

Applying Lemma 4.7.4 thrice, one obtain a $(b, \ell, O(\tau(b)))$-reduction (ψ, V). And since $\varphi_{b_3,\ell}$ can be found in $b^{o(1)}$ time, together with Lemma 4.7.4, we obtain a uniform-τ-reduction family.

□

Finally, we give a direct corollary of Theorem 4.7.3 that the existence of an $O(1)$-reduction family implies hardness of $\mathbb{Z}$-OV, $\mathbb{Z}$-Max-IP, ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair in $\omega(1)$ dimensions.

Corollary 4.7.5. *If there is an $O(1)$-reduction family, then for every $\varepsilon > 0$, there exists a $c \geq 1$ such that $\mathbb{Z}$-OV, $\mathbb{Z}$-Max-IP, ℓ_2-Furthest Pair and Bichromatic ℓ_2-Closest Pair in c dimensions with $O(\log n)$-bit entries require $n^{2-\varepsilon}$ time.*

Proof Sketch. Note that since its hardness implies the harnesses of other three, we only need to consider $\mathbb{Z}$-OV here.

From Theorem 4.7.3 and the assumption, there exists a uniform-$O(1)$-reduction. Proceeding similar as in Lemma 4.1.6 with the uniform-$O(1)$-reduction, we obtain a better dimensionality self reduction from OV to $\mathbb{Z}$-OV. Then exactly the same argument as in Theorem 4.1.7 with different parameters gives us the lower bound required. □

Chapter 5

IP Protocols and Hardness for Approximation Problems Under Stronger Conjectures

5.1 Introduction

In this chapter we apply the classical $\mathsf{IP} = \mathsf{PSPACE}$ theorem and the connection between communication complexity and fine-grained complexity to obtain various reductions between exact and approximate problems in fine-grained complexity.

5.1.1 Exact to Approximate Reduction for Nearest Neighbor Search for LCS

We begin with one of our most interesting results: an equivalence between ***exact*** and ***approximate*** Nearest Neighbor Search for LCS.

- Nearest Neighbor Search for LCS ($\mathsf{NNS}_{\mathsf{LCS}}$): Preprocess a database $\mathcal{D}$ of N strings of length $D \ll N$, and then for each query string x, find $y \in \mathcal{D}$ maximizing $\mathsf{LCS}(x, y)$.

  The approximate version only requires to find $y \in \mathcal{D}$ such that $\mathsf{LCS}(x, y)$ is an

$f(D)$-approximation of the maximum value.

Approximate data structures for the above problem that support fast preprocessing and queries would be highly relevant for bioinformatics. For the similar $\mathsf{NNS}_{\mathsf{Edit\text{-}Distance}}$ problem, a breakthrough work of [138] used a metric embedding technique to obtain an $2^{O(\sqrt{\log D \log\log D})}$-approximate data structure with polynomial preprocessing time, $\mathrm{poly}(D, \log n)$ query time.

In contrast, our first result shows that exact $\mathsf{NNS}_{\mathsf{LCS}}$ and approximate $\mathsf{NNS}_{\mathsf{LCS}}$ are essentially equivalent. That is, a similar data structure for *approximate* $\mathsf{NNS}_{\mathsf{LCS}}$ would directly imply a data structure for *exact* $\mathsf{NNS}_{\mathsf{LCS}}$ with essentially the same complexity!

Theorem 5.1.1 (Informal). *For $D = 2^{(\log N)^{o(1)}}$, suppose there is a data structure for $\mathsf{NNS}_{\mathsf{LCS}}$ with approximation ratio $2^{(\log D)^{1-\Omega(1)}}$, then there is another data structure for exact $\mathsf{NNS}_{\mathsf{LCS}}$ with essentially the same preprocessing time/space and query time.*

In the following, we first discuss Closest-LCS-Pair (a natural *offline* version of $\mathsf{NNS}_{\mathsf{LCS}}$) to illustrate our techniques, and then discuss our other results in details.

5.1.2 Techniques: Hardness of Approximation in P via Communication Complexity and the Theory of Interactive Proofs

Closest-LCS-Pair is the problem that given two sets of strings A and B, compute the maximum $\mathsf{LCS}(a, b)$ with $(a, b) \in A \times B$. We show how to reduce exact Closest-LCS-Pair to approximate Closest-LCS-Pair as an illustration of our proof techniques.

Theorem 5.1.2 (Informal). *There is a near-linear time[1] reduction from Closest-LCS-Pair to $2^{(\log N)^{1-\Omega(1)}}$ factor approximate Closest-LCS-Pair, when A, B are two sets of N strings of length $D = 2^{(\log N)^{o(1)}}$.*

[1] Throughout this chapter, we use near-linear time to denote the running time of $N^{1+o(1)}$.

We first introduce the concept of $\mathcal{A}$-*Satisfying-Pair* problem. This problem asks whether there is a pair of (a, b) from two given sets A and B such that (a, b) is a yes-instance of $\mathcal{A}$. By binary search, Closest-LCS-Pair can be easily formulated as an $\mathcal{A}$-*Satisfying-Pair* problem: is there a pair $(a, b) \in A \times B$ such that $\mathsf{LCS}(a, b) \geq k$? The key property we are going to use is that the function $A_{\mathsf{LCS}}^{\geq k}(a, b) := [\mathsf{LCS}(a, b) \geq k]$ can be computed in very small space, i.e., it is in NL (see Lemma 5.6.4). Indeed, we will show in this chapter that for all $\mathcal{A}$-*Satisfying-Pair* such that $\mathcal{A}$ can be computed in small space, $\mathcal{A}$-*Satisfying-Pair* can be reduced to approximate Closest-LCS-Pair.

The Reduction in a Nutshell. First, we consider an IP communication protocol for LCS. In this setting, Alice and Bob hold strings a and b, and they want to figure out whether $\mathsf{LCS}(a, b) \geq k$. To do so, they seek help from an untrusted prover Merlin by engaging in a conversation with him. The protocol should satisfy that when $\mathsf{LCS}(a, b) \geq k$, Merlin has a strategy to convince Alice and Bob w.h.p., and when $\mathsf{LCS}(a, b) < k$, no matter what Merlin does, Alice and Bob will reject w.h.p. The goal is to minimize the total communication bits (between Alice and Bob, or Alice/Bob and Merlin).

Next, by a result of Aaronson and Wigderson [2], it is shown that any function $f(a, b)$ which can be computed in NL admits an IP communication protocol with $\mathrm{polylog}(N)$ total communication bits. Finally, using an observation from [11], an efficient IP communication protocol can be embedded into approximate LCS, which completes the reduction. In the following we explain each step in details.

IP Communication Protocols for Low Space Computation. The key technical ingredient of our results is the application of IP communication protocols for low space computation by Aaronson and Wigderson [2]. It would be instructive to explain how it works.

Let us use the sum-check IP protocol for the Inner Product problem as an example. Arthur gets access to a function $f : \{0, 1\}^n \to \{0, 1\}$ and its multilinear extension $\widetilde{f} : \mathbb{F}_q^n \to \mathbb{F}_q$ over a finite field $\mathbb{F}_q$. Let $f_i(x) = f(i \circ x)$ for $i \in \{0, 1\}$ be the restrictions of f after setting the first bit of the input, Arthur wants to compute the inner product $\sum_{x \in \{0,1\}^{n-1}} f_0(x) \cdot f_1(x)$. To do so, he engages in a conservation with an untrusted Merlin who tries to convince him. During the conversation, Merlin is doing all the "real work", while Arthur only has to query $\widetilde{f}$ once at the last step, which is the crucial observation in [2].

Now, imagine a slightly different setting where f_0 and f_1 are held by Alice and Bob respectively, which means each of them holds a string of length $N = 2^{n-1}$. They still want to compute the inner product with Merlin, while using minimum communication between each other.

In this setting, Alice (pretending she is Arthur) can still run the previous IP protocol with Merlin. When she has to query $\widetilde{f}(z)$ for a point z at the last step, she only needs Bob to send her the contribution of his part to $\widetilde{f}(z)$, which only requires $O(\log q)$ bits. In terms of the input size of Alice and Bob, this IP communication protocol runs in $\text{poly}(n) = \text{polylog}(N)$ time. The same also extends to any $\text{poly}(n) = \text{polylog}(N)$ space computation on f, if we use the IP protocol for PSPACE [123, 155].

In Section 5.5, we provide a parameterized IP communication protocol for Branching Program[2] (Theorem 5.5.5). Informally, we have:

Theorem 5.1.3 (IP Communication Protocol for BP (Informal)). *Let P be a branching program of length T and width W with n input bits, equally distributed among Alice and Bob. For every soundness parameter $\varepsilon > 0$, there is an IP-protocol for P, such that:*

- *Merlin and Alice exchange $\widetilde{O}\left(\log^2 W \log^2 T \log \varepsilon^{-1}\right)$ bits, and toss the same amount of public coins;*
- *Bob sends $O(\log\log(WT) \cdot \log \varepsilon^{-1})$ bits to Alice;*
- *Alice always accepts if P accepts the input, and otherwise rejects with probability at least $1 - \varepsilon$.*

[2] Informally speaking, a branching program with length T and width W formulates a *non-uniform* low-space computation with running time T and space $\log W$, see Definition 5.2.1 for a formal definition.

Since $A_{\mathsf{LCS}}^{\geq k}(a,b)$ is in NL, the above in particular implies that there is an IP communication protocol for $A_{\mathsf{LCS}}^{\geq k}$ with polylog(D) total communication bits (D is length of strings).

IP Communication Protocols and Tropical Tensors. The next technical ingredient is the reduction from an IP communication protocol to a certain Tropical Tensors problem [11]. We use a 3-round IP communication protocol as an example to illustrate the reduction. Consider the following 3-round IP communication protocol Π:

- Alice and Bob hold strings x and y, Merlin knows both x and y.
- Merlin sends Alice and Bob a string $z_1 \in \mathcal{Z}_1$.
- Alice sends Merlin a uniform random string $z_2 \in \mathcal{Z}_2$.
- Merlin sends Alice and Bob another string $z_3 \in \mathcal{Z}_3$.
- Bob sends Alice a string $z_4 \in \mathcal{Z}_4$, and Alice decides whether to accept or reject.

The main idea of [11] is that the above IP protocol can be reduced into a certain Tropical Similarity function. That is, for x and y, we build two tensors $u = u(x)$ and $v = v(y)$ of size $|\mathcal{Z}_1| \times |\mathcal{Z}_2| \times |\mathcal{Z}_3| \times |\mathcal{Z}_4|$ as follows: we set u_{z_1,z_2,z_3,z_4} to indicate whether Alice accepts, given the transcript (z_1, z_2, z_3, z_4) and input x; we also set v_{z_1,z_2,z_3,z_4} to indicate whether Bob sends the string z_4, given the previous transcript (z_1, z_2, z_3) and input y. Then, by the definition of IP protocols, it is not hard to see the acceptance probability when Merlin uses optimal strategy is:

$$\mathsf{acc}(u,v) := \max_{z_1 \in \mathcal{Z}_1} \mathsf{E}_{z_2 \in \mathcal{Z}_2} \max_{(z_3,z_4) \in \mathcal{Z}_3 \times \mathcal{Z}_4} u_{z_1,z_2,z_3,z_4} \cdot v_{z_1,z_2,z_3,z_4}.$$

In the above equality, the max operator corresponds to the actions of Merlin, who wishes to maximize the acceptance probability, while the E operator corresponds to actions of Alice, who sends a uniform random string. It can be easily generalized to IP protocols of any rounds, by replacing acc with a series of max and E operators, which is called Tropical Similarity (denoted by $s(u, v)$) in [11] (see also Definition 5.2.8). When the number of total communication bits is d, both u and v are of size 2^d.

Applying the above to the polylog(D) bits IP protocol for $A_{\mathsf{LCS}}^{\geq k}$, it means for two strings a, b, we can compute two tensors u, v of length $2^{\mathrm{polylog}(D)} = 2^{(\log N)^{o(1)}}$, such that when $\mathsf{LCS}(a, b) \geq k$, $s(u, v)$ is large, and otherwise $s(u, v)$ is small.

Simulating Tropical Tensors by Composing max **and** Σ **Gadgets.** While the above reduction is interesting in its own right, the Tropical Similarity function seems quite artificial. Another key idea from [11] is that $s(u, v)$ can be simulated by LCS. The reduction works by noting that with LCS, one can implement the max and Σ (which is equivalent to E) gadgets straightforwardly, and composing them recursively leads to gadgets for Tropical Similarity. That is, for tensors u and v, one can construct strings $S(u)$ and $T(v)$ of similar sizes, such that $\mathsf{LCS}(S(u), T(v))$ is proportional to $s(u, v)$.

Putting everything together, for two strings a, b, we can compute two other strings $S(a)$ and $T(b)$ of length $2^{\mathrm{polylog}(D)} = 2^{(\log N)^{o(1)}}$, such that $\mathsf{LCS}(a, b) \geq k$, $\mathsf{LCS}(S(a), T(b))$ is large, and otherwise $\mathsf{LCS}(S(a), T(b))$ is small. This completes our reduction.

Our Results In Detail

5.1.3 From Exact to Approximate in the Fine-Grained World

More generally, we consider the following four (general flavor) problems.

- The Closest-LCS-Pair problem.

- The Closest-RegExp-String-Pair problem: Given a set A of N regular expressions of length $2^{(\log N)^{o(1)}}$ and a set B of N strings of length $2^{(\log N)^{o(1)}}$, find $(a, b) \in A \times B$ with maximum Hamming Similarity[3].

[3] Hamming Similarity between two strings are defined as the fraction of positions that they are equal, while the hamming similarity between a regular expression a and a string b is the maximum of the hamming similarity between z and b where z is in the language of a.

- The Max-LCST-Pair problem: Given two sets A, B of N bounded-degree trees with size $2^{(\log N)^{o(1)}}$, find a pair $(a, b) \in A \times B$ such that a and b's have the largest common subtree.

- The Max-TropSim problem: Given two sets A, B of N binary tensors with size $2^{(\log N)^{o(1)}}$, find the pair with maximum Tropical Similarity[4].

Our main theorem shows equivalence between exact and approximation versions of the above problems. In fact, we show that all these problems, together with the following two closely related decision problems and a generic satisfying pair problem, are *equivalent* under *near-linear* time reductions. See Theorem 5.3.1 for a formal statement of the equivalence class.

- The RegExp-String-Pair problem: Given a set A of N regular expressions of length $2^{(\log N)^{o(1)}}$ and a set B of N strings of length $2^{(\log N)^{o(1)}}$, is there a pair $(a, b) \in A \times B$ such that b matches a?

- The Subtree-Isomorphism-Pair problem: Given two sets A, B of N bounded-degree trees with size $2^{(\log N)^{o(1)}}$, is there a pair $(a, b) \in A \times B$ such that a is isomorphic to a subtree of b?

- The BP-Satisfying-Pair problem: Given a branching program[5] P of size $2^{(\log N)^{o(1)}}$ and two sets A, B of N strings, is there a pair $(a, b) \in A \times B$ making P accepts the input (a, b)?

We will refer to this set of problems as BP-Pair-Class.

Remark 5.1.4. *Subtree-Isomorphism-Pair and Max-LCST-Pair may seem artificial, but they are nice intermediate problems for showing hardness of the closely related problems Subtree Isomorphism and Longest Common Subtree, which are extensively studied natural problems (see Section 5.1.5).*

[4] see Definition 5.2.8 for a formal definition.

[5] see Definition 5.2.1 for a formal definition

Equivalence in the Data Structure Setting

These pair problems are interesting as they are natural *off-line* versions of closely related data structure problem, which are highly relevant in the practice [162, 135, 51, 134, 175, 111, 133, 136, 47]. Therefore, a lower bound on the time complexity of these pair problems directly implies a lower bound for corresponding data structure problems (see Theorem 5.9.4). In the next section, we will show under some conjecture which is much more plausible than SETH, these problems requires essentially quadratic time.

Our equivalence continues to hold in the data structure version, in particular, for the following data structure problems, any algorithm for one of them implies an algorithm for all of them with essentially the same preprocessing time/space and query time (up to a factor of $N^{o(1)}$). See Section 5.9 for the details.

- $\mathsf{NNS}_{\mathsf{LCS}}$[6]: Preprocess a database $\mathcal{D}$ of N strings of length $D = 2^{(\log N)^{o(1)}}$, and then for each query string x, find $y \in \mathcal{D}$ maximizing $\mathsf{LCS}(x, y)$.
- Approx. $\mathsf{NNS}_{\mathsf{LCS}}$: Find $y \in \mathcal{D}$ s.t. $\mathsf{LCS}(x, y)$ is a $2^{(\log D)^{1-\Omega(1)}}$ approximation to the maximum value.
- Regular Expression Query: Preprocess a database $\mathcal{D}$ of N strings of length $D = 2^{(\log N)^{o(1)}}$, and then for each query regular expression y, find an $x \in \mathcal{D}$ matching y.
- Approximate Regular Expression Query: For a query expression y, distinguish between[7]: (1) there is an $x \in \mathcal{D}$ matching y; and (2) for all $x \in \mathcal{D}$, the Hamming distance between x and all $z \in L(y)$ is at least $(1 - o(1)) \cdot D$, where $L(y)$ is the set of all strings matched by y.

That is, a non-trivial data structure for finding *approximate* nearest point with LCS metric would imply a non-trivial data structure for answering regular expression query! The latter one is supported by most modern database systems such as MySQL,

[6] already discussed in Section 5.1.1

[7] behavior can be arbitrary when neither of the two cases hold

Oracle Database, Microsoft SQL etc., but all of them implement it by simply using full table scan for the most general case.

LCS is the Hardest Distance Function for Approximate NNS. In fact, our results also suggest in a formal sense that LCS is the *hardest* distance function for approximate Nearest Neighbor Search. We show that for all distance function dist which is computable in poly-logarithmic *space*[8], exact NNS for dist can be reduced to approximate $\mathsf{NNS}_{\mathsf{LCS}}$.

Theorem 5.1.5 (Informal). *For a distance function dist that is computable in poly-logarithmic space, exact NNS for dist can be reduced to $2^{(\log D)^{1-\Omega(1)}}$-approximate $\mathsf{NNS}_{\mathsf{LCS}}$ in near-linear time.*

5.1.4 Weaker Complexity Assumptions for Approximation Hardness

An important goal in the study of fine-grained complexity is to find more plausible conjectures, under which to base hardness. For example, although SETH is based on the historically unsuccessful attempts on finding better algorithms for k-SAT, there is no consensus on its validity (see, e.g. [167, 173]).

This concern has been addressed in various ways. For example, in [16], the authors prove hardness for several problems, basing on *at least one of* the SETH, the APSP conjecture, or the 3-SUM Conjecture being true. In [10], Abboud et al. introduce a hierarchy of $\mathcal{C}$-SETH assumptions: the $\mathcal{C}$-SETH asserts that there is no $2^{(1-\varepsilon)n}$ time satisfiability algorithm for circuits from $\mathcal{C}$.[9] They show that the quadratic time hardness of Edit-Distance, LCS and other related sequence alignment problems can be based on the much weaker and much more plausible assumption NC-SETH. However, this has not been shown for approximation version of fine-grained problems.

[8] Which is true for almost all nature distance functions. For example, edit distance and LCS can be computed in NL, thus in $(\log^2 N)$ space by Savitch's Theorem [153].

[9] In this way, the original SETH assert that there is no $2^{(1-\varepsilon)n}$ time algorithm for satisfiability of CNF with arbitrary constant bottom fan-in.

In this work, we show that all problems in BP-Pair-Class require essentially quadratic time under NC-SETH. Indeed, our hardness results are based on a weaker assumption which we call $2^{n^{o(1)}}$*-size BP-SETH*:[10]

Hypothesis 5.1.6 ($2^{n^{o(1)}}$-size BP-SETH). *The satisfiability of a given $2^{n^{o(1)}}$-size nondeterministic branching program cannot be solved in $O(2^{(1-\delta)n})$ time for any $\delta > 0$.*

Theorem 5.1.7. *All problems in BP-Pair-Class require $N^{2-o(1)}$ time if we assume $2^{n^{o(1)}}$-size BP-SETH.*

Note that $2^{n^{o(1)}}$-size BP-SETH is even weaker than $n^{o(1)}$-depth circuit SETH: satisfiability for $n^{o(1)}$-depth bounded fan-in circuits cannot be solved in $O(2^{(1-\delta)n})$ time for any $\delta > 0$. This is because by Barrington's Theorem [46], $n^{o(1)}$-depth bounded fan-in circuits can be simulated by branching programs of size $2^{n^{o(1)}}$.

It is worthwhile to compare with [12]. It is shown in [12] that assuming SETH, a $2^{(\log N)^{1-o(1)}}$-approximation to Closest-LCS-Pair (Closest-RegExp-String-Pair) requires $N^{2-o(1)}$ time for $D = N^{o(1)}$. (The $2^{(\log N)^{1-o(1)}}$ factor is later improved to $N^{o(1)}$ in [151, 70].)

Although our results here are quantitatively worse, it is "qualitatively" better in many ways: (1) the results in [12] is based on SETH, while our hardness results are based on the assumptions in Theorem 5.1.7, which are much more plausible than SETH; (2) we in fact have established an equivalence between Closest-LCS-Pair and its approximation version, which seems not possible with the techniques in [12]; (3) our framework allows us to show that even a tiny improvement on the running time would have important algorithmic and circuit lower bound consequences (see Theorem 5.1.10), which again seems not possible with the techniques in [12].

5.1.5 BP-Pair-Class Hard Problems

We also identify a set of other problems which are at least as hard as any problem in BP-Pair-Class, but not necessarily in it. We say these problems are BP-Pair-Hard.

[10] Indeed, results in [10] are also based on a conjecture about branching programs, which can be seen as $O(1)$-width and $2^{o(n)}$-length BP-SETH or $2^{o(\sqrt{n})}$-size BP-SETH using the terminology in Hypothesis 5.1.6.

Theorem 5.1.8 (BP-Pair-Hard Problems). *There are near-linear time reductions from all the problems in BP-Pair-Class to any of the following problems:*

1. *(Subtree Isomorphism) Given two trees a, b of size at most N, determine whether a is isomorphic to a subtree of b (even if restricted to the case of binary rooted trees);*

2. *(Largest Common Subtree) Given two trees a, b of size at most N, compute the exact value or a $2^{(\log N)^{o(1)}}$-approximation of the size of the largest common subtree of a and b (even if restricted to the case of binary rooted trees);*

3. *(Regular Expression Membership Testing) Given a regular expression a of length M and a string b of length N, determine whether b is in the language of a;*

Corollary 5.1.9. *All the above BP-Pair-Hard problems require $N^{2-o(1)}$ time (or $(NM)^{1-o(1)}$ time for Regular Expression Membership Testing) under the same assumption as in Theorem 5.1.7.*

We remark that both Subtree Isomorphism and Largest Common Subtree are studied in [4]. In particular, they showed that Subtree Isomorphism and Largest Common Subtree require quadratic-time under SETH, even for binary rooted trees.

Our results improve theirs in many ways: (1) for Subtree Isomorphism, we establish the same quadratic time hardness, with a much safer conjecture; (2) for Largest Common Subtree, we not only put its hardness under a better conjecture, but also show that even a $2^{(\log N)^{o(1)}}$-approximation would be hard; (3) for both of these problems, we demonstrate that even a tiny improvement on the running time would have interesting algorithmic and circuit lower bound consequences (see Theorem 5.1.11).

[55] (which builds on [40]) classified the running time of constant-depth regular expression membership testing. In particular, they showed a large class of regular expression testing requires quadratic-time, under SETH. Our results are incomparable with theirs, as our hard instances may have unbounded depth regular expressions. On the bright side, our hardness results rely on a much safer conjecture, and we show interesting consequences even for a tiny improvement of the running time.

5.1.6 The Consequence of "shaving-logs" for Approximation Problems

There has been a large number of works focusing on "shaving logs" of the running time of fundamental problems [34, 44, 66, 178] (see also a talk by Chan [65], named "The Art of Shaving Logs"). In a recent exciting algorithmic work by Williams [170], the author shaves "all the logs" on the running time of APSP, by getting an $n^3/2^{\Theta(\sqrt{\log n})}$ time algorithm.

However, the best exact algorithms for LCS and Edit distance [124, 96] remain $O(n^2/\log^2 n)$, which calls for an explanation. An interesting feature of [10] is that their results show that even shaving logs on LCS or Edit Distance would be very hard. In particular, they prove that an $n^2/\log^{\omega(1)} n$ time algorithm for either of them would imply a $2^n/n^{\omega(1)}$ time algorithm for polynomial-size formula satisfiability, which is much better than the current state of arts [152, 161]. Such an algorithm would also imply that NEXP is not contained in non-uniform NC^1, thereby solving a notorious longstanding open question in complexity theory.

The "shaving logs barrier" only has been studied for a few problems. It was not clear whether we can get the same barriers for some *approximation problems.*

In this work we show that slightly improved algorithms (such as shaving all the logs) for any BP-Pair-Class or BP-Pair-Hard problems, would imply *circuit lower bounds* which are notoriously hard to prove. This extends all the results of [10] to approximation problems.

Theorem 5.1.10. *If there is an* $O\left(N^2\,\mathrm{poly}(D)/2^{(\log\log N)^3}\right)$ *or* $O\left(N^2/(\log N)^{\omega(1)}\right)$ *time deterministic algorithm for any decision, exact value or* $\mathrm{polylog}(D)$*-approximation problems in BP-Pair-Class, where* D *is the maximum length (or size) of elements in sets, then the following holds:*

- *NTIME*$[2^{O(n)}]$ *is not contained in non-uniform* NC^1 *and*
- *Formula-SAT with* $n^{\omega(1)}$ *size can be solved in* $2^n/n^{\omega(1)}$ *time.*

Theorem 5.1.11. *If there is a deterministic algorithm for any decision, exact value or* $\operatorname{polylog}(N)$*-approximation problems among BP-Pair-Hard problems listed in Theorem 5.1.8 running in* $O\left(N^2/2^{\omega(\log\log N)^3}\right)$ *time (or* $O\left(NM/2^{\omega(\log\log(NM))^3}\right)$ *time for Regular Expression Membership Testing), then the same consequences in Theorem 5.1.10 follows.*

5.1.7 Circuit Lower Bound Consequence of Improving Approximation Algorithms for P Time Problems

Finally, we significantly improve the results from [11], by showing much stronger circuit lower bound consequences for deterministic approximation algorithms to LCS.

Theorem 5.1.12. *The following holds for deterministic approximation to LCS:*

1. *A* $2^{(\log N)^{1-\Omega(1)}}$*-approximation algorithm in* $N^{2-\delta}$ *time for some constant* $\delta > 0$ *implies that* E^{NP} *has no* $n^{o(1)}$*-depth bounded fan-in circuits;*

2. *A* $2^{o(\log N/(\log\log N)^2)}$*-approximation algorithm in* $N^{2-\delta}$ *time for some constant* $\delta > 0$ *implies that* $\mathsf{NTIME}[2^{O(n)}]$ *is not contained in non-uniform* NC^1*;*

3. *An* $O(\operatorname{polylog}(N))$*-approximation algorithm in* $N^2/2^{\omega(\log\log N)^3}$ *time implies that* $\mathsf{NTIME}[2^{O(n)}]$ *is not contained in non-uniform* NC^1*.*

In comparison with [11], they show that an $O(N^{2-\varepsilon})$ time algorithm for constant factor deterministic approximation algorithm to LCS would imply that E^{NP} does not have non-uniform linear-size NC^1 circuits or VSP circuits. Our results here generalize theirs in all aspects: (1) we show that a much stronger lower bound consequence would follow from even a sub-quadratic time $2^{(\log N)^{1-\Omega(1)}}$-approximation algorithm; (2) we also show that a modestly stronger lower bound would follow even from a quasi-polylogarithmic improvement over the quadratic time, for approximate LCS.

More generally, following a similar argument to [10], we can show that truly-subquadratic time algorithms for these BP-Pair-Class or BP-Pair-Hard problems would imply strong circuit lower bounds against E^{NP}.

Corollary 5.1.13. *If any of the problems listed in Theorem 5.3.1 and Theorem 5.1.8 admits an $N^{2-\varepsilon}$ time algorithm (or $(NM)^{1-\varepsilon}$ time algorithm for Regular Expression Membership Testing) for some $\varepsilon > 0$, then E^{NP} does not have:*

1. *non-uniform $2^{n^{o(1)}}$-size Boolean formulas,*

2. *non-uniform $n^{o(1)}$-depth circuits of bounded fan-in, and*

3. *non-uniform $2^{n^{o(1)}}$-size nondeterministic branching programs.*

5.1.8 Discussions and Open Problems

Here we discuss some open problems arising from our work.

Find More Members for BP-Pair-Class

One immediate question is to find more natural quadratic-time problems belonging to BP-Pair-Class:

Open Question 1. *Find more natural problems which belong to BP-Pair-Class.*

It could be helpful to revisit all SETH-hard problems to see whether they can simulate BP-Satisfying-Pair. In particular, one may ask whether the Orthogonal Vectors problem (OV), the most studied problem in fine-grained complexity, belongs to this equivalence class:

Open Question 2. *Does OV belong to BP-Pair-Class?*

If it does, then it would open up the possibility that perhaps all SETH-hard quadratic-time problems are equivalent. However, some evidence suggests that the answer may be negative, as OV seems to be much easier than problems in BP-Pair-Class:

- **The Inner Function in OV is Much Weaker.** When viewing as a Satisfying-Pair problem, the inner function in OV is just a simple *Set-Disjointness*, which seems incapable of simulating generic low-space computation.

- **There are Non-trivial Algorithms for OV.** We know that for OV with N vectors of length $D = c \log n$, there are algorithms with running time $N^{2-1/O(\log c)}$ [13, 68]. This type of non-trivial speed up seems quite unlikely (or at least much harder to obtain) for problems in BP-Pair-Class (see Theorem 5.1.10).

It would be interesting to show that OV and BP-Satisfying-Pair are not equivalent under certain plausible conjectures, perhaps ideas from [63] could help.

Quasi-Polynomial Blow Up of the Dimension

In the reductions between our BP-Pair-Class problems, we get a quasi-polynomial blowup on the dimensions: that is, a problem with element size (vector dimension, string length or tree size) D is transformed into another problem with element size $2^{\text{polylog}(D)}$. This is the main reason that we have to restrict the element size to be small, i.e., $D = 2^{(\log N)^{o(1)}}$.

This technical subtlety arises from the polynomial blow-up in the $\mathsf{IP} = \mathsf{PSPACE}$ proof: given a language in $\mathsf{NSPACE}[S]$, it is first transformed into a TQBF instance of size $O(S^2)$, which is proved by an $\widetilde{O}(S^4)$ time IP protocols, using arithmetization.

Applying that into our setting, an NL computation on two strings of length D (like LCS), is transformed into an $\widetilde{O}(\log^4 D)$ time IP communication protocol, which is then embedded into an approximate problem with at least $2^{\widetilde{O}(\log^4 D)}$ dimensions (say approximate LCS).

However, if we have an $O(\log D)$ time IP communication protocol for NL. The new dimension would be $2^{O(\log D)} = D^{O(1)}$, only a polynomial blow up. Which motivates the following interesting question:

Open Question 3. *Is there an $O(\log D)$ time IP communication protocol for every problems in NL?*

A positive resolution of the above question would also tighten several parameters in many of our results. For example, in Theorem 5.1.7, $n^{o(1)}$-depth circuit SETH could be replaced by $o(n)$-depth circuit SETH, and Theorem 5.1.10, Theorem 5.1.11, Theorem 5.1.8 and Corollary 5.1.13 would also have improved parameters.

It is worth noting that IP communication lower bounds are extremely hard to prove—proving a non-trivial lower bound for AM communication protocols is already a long-standing open question [121, 93, 94]. Hence, resolving Open Question 3 negatively could be hard.

5.1.9 Related Works

Hardness for Shaving Logs

In [10], it is shown that an $n^2/\log^{\omega(1)}$ time algorithm for LCS would imply that the Formula-SAT have a $2^n/n^{\omega(1)}$ time algorithm. The construction is later tightened in [6], which shows that an $n^2/\log^{7+\varepsilon} n$ time algorithm for any of LCS, regular expression pattern matching or Fréchet distance is already enough to imply new algorithm for Formula-SAT.

Related Works for Specific Problems

Longest Common Subsequence. LCS is a very basic problem in computer science and has been studied for decades [75, 50, 84, 79]. In recent years, a series of works [41, 5, 56, 10, 11] have shown different evidences that LCS may not have truly sub-quadratic time algorithm, even for approximation.

Subtree Isomophism and Largest Common Subtree. Subtree Isomorphism has been studied in [127, 128, 119, 148, 74, 120, 90, 156]. Largest Common Subtree is an NP-hard problem when the number of trees is not fixed, and has been studied in [108, 19, 20]. For (rooted or unrooted) bounded-degree trees, both the two problems can be solved in $O(N^2)$ time, and the fastest algorithm for Subtree Isomorphism runs in $O(N^2/\log N)$ time [119, 156]. In [4], these two problems are shown to be SETH-hard.

Regular Expression. The found of $O(NM)$ algorithm for regular expression matching and membership testing in [162] is a big success in 70s, but after that no algorithm has been found to improve it to truly sub-quadratic time [135, 51]. Related works about the hardness of exact regular expression matching or membership testing include [40, 55]. There are many works on approximate regular expression matching in different formulations [134, 175, 111, 133, 136, 47], and its hardness has been analyzed in [12].

5.2 Preliminaries

In this section, we define the SAT problem for branching program (BP-SAT), and then we introduce the formal definitions for each problem in BP-Pair-Class and BP-Pair-Hard.

Definition 5.2.1 (Branching Program). A (nondeterministic) *Branching Program* (BP) on n Boolean inputs $x_1, \dots, x_n$ is defined as a layered directed graph with T layers $L_1, \dots, L_T$. Each layer L_i contains W nodes. Except the last layer, every node in L_i $(1 \leq i < T)$ is asscociated with the same variable $x_{f(i)}$ for some $f(i) \in [n]$. T and W are called the *length* and *width* of the BP.

For every two adjacent layers L_i and L_{i+1}, there are edges between nodes in L_i to nodes in L_{i+1}, and each edge is marked with either 0 or 1. The *size* of a BP is defined as the total number of edges $O(W^2T)$.

For $1 \leq i \leq T, 1 \leq j \leq W$, the j-th node in the i-th layer L_i is labeled as (i, j). $u_{\text{start}} = (1, 1)$ is the starting node, and $u_{\text{acc}} = (T, 1)$ is the accepting node. A BP accepts an input x iff there is a path from the starting node to the accepting node consisting of only the edges marked with the value of the variable associated with its starting endpoint.

Definition 5.2.2 (BP-SAT). Given a branching program P on n Boolean inputs, the BP-SAT problem asks whether there is an input making P accept.

Like the relationship between k-SAT and Orthogonal Vectors (OV), we can define BP-Satisfying-Pair problem as the counterpart of BP-SAT in the P world. BP-Satisfying-Pair can be trivially solved in $O(N^2 \cdot \text{poly}(W, T))$ time, and a faster algorithm for BP-Satisfying-Pair running in $O(N^{2-\varepsilon})$ time implies a faster algorithm for BP-SAT running in $O(2^{(1-\varepsilon/2)n})$ time.

Definition 5.2.3 (BP-Satisfying-Pair)**.** Given a branching program P on n Boolean inputs (assume n is even) and two sets of N strings $A, B \subseteq \{0,1\}^{n/2}$, the BP-Satisfying-Pair problem asks whether there is a pair $a \in A, b \in B$ such that P accepts the concatenation of a and b.

Throughout this chapter, unless otherwise stated, we use BP-Satisfying-Pair to denote the BP-Satisfying-Pair problem on branching program of size $2^{(\log N)^{o(1)}}$ for convenience.

5.2.1 Satisfying Pair and Best Pair Problems

Satisfying Pair Problems. Note that problems like *Orthogonal Vectors* are in the form of deciding whether there is a "satisfying pair". In general, we can define the $\mathcal{A}$*-Satisfying-Pair* problem, where $\mathcal{A}$ is an arbitrary decision problem on two input strings x, y:

Definition 5.2.4 ($\mathcal{A}$-Satisfying-Pair)**.** Given two sets A, B of N strings, the $\mathcal{A}$-*Satisfying-Pair* problem asks whether there is a pair of $a \in A, b \in B$ such that (a, b) is an Yes-instance of $\mathcal{A}$.

In this work, we study a series of $\mathcal{A}$-Satisfying-Pair problems, including OAPT, RegExp-String-Pair and Subtree-Isomorphism-Pair, which will be formally defined in later subsections.

Best Pair Problems. For an optimization problem $\mathcal{A}$ on two input strings x, y, we can define the Max-$\mathcal{A}$-Pair and the Min-$\mathcal{A}$-Pair problems:

Definition 5.2.5 (Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair)**.** Given two sets A, B of N strings, the Max-$\mathcal{A}$-Pair (or Min-$\mathcal{A}$-Pair) problem asks to compute the maximum value (or minimum value) of the result of problem $\mathcal{A}$ on input (a, b).

In this work, we study a series of Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair problems, including Max-TropSim, Min-TropSim, Closest-LCS-Pair, Furthest-LCS-Pair, Closest-RegExp-String-Pair, Max-LCST-Pair and Min-LCST-Pair, which will be formally defined in later subsections.

Note that both satisfying pair problems and best pair problems contain two sets A, B in the input. Without additional explanation, we use N to denote the set size, and D to denote the maximum element size in sets.

5.2.2 Two Tensor Problems

We introduce two kinds of tensor problems: the *Orthogonal Alternating Product Tensors* problem (OAPT) and the *Max/Min Tropical Similarity* problem (Max-TropSim/Min-TropSim). The former one is an $\mathcal{A}$-Satisfying-Pair problem, which helps us to prove hardness for decision problems; the latter one is a Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair problem, which helps us proving hardness of approximation for optimization problems.

First we define OAPT. OAPT implicitly appears in the reduction from BP-SAT to LCS in [10] as an intermediate problem. In our reductions, OAPT appears naturally, and we show that BP-Satisfying-Pair and OAPT of certain size are equivalent under near-linear time reduction in Section 5.5.

Definition 5.2.6 (OAPT). Let t be an even number and $d_1 = d_2 = \cdots = d_t = 2$. The *Alternating Product* $p_{\text{alt}}(u, v)$ of two tensors $u, v \in \{0,1\}^{d_1 \times \cdots \times d_t}$ is defined as an alternating sequence of logical operators $\wedge$ and $\vee$ applied to the coordinatewise product of u and v:

$$p_{\text{alt}}(u, v) = \bigwedge_{i_1 \in [d_1]} \left[\bigvee_{i_2 \in [d_2]} \left(\bigwedge_{i_3 \in [d_3]} \left[\cdots \bigvee_{i_t \in [d_t]} (u_i \wedge v_i) \cdots \right] \right) \right]. \tag{5.1}$$

Given two sets of N tensors $A, B \subseteq \{0,1\}^{d_1 \times \cdots \times d_t}$, the Orthogonal Alternating Product Tensors (OAPT) problem asks whether there is a pair $a \in A, b \in B$ such that the Alternating Product $p_{\text{alt}}(a, b) = 0$.

Restricted OAPT is a restricted version of OAPT. We mainly use this restricted version in our analysis.

Definition 5.2.7 (Restricted OAPT). We say that a tensor $x \in \{0,1\}^{d_1 \times \cdots \times d_t}$ is $\wedge$-invariant if the value of $x_{i_1 \cdots i_t}$ does not depend on $i_1, i_3, i_5, \ldots, i_{t-1}$. The Restricted OAPT problem is a restricted version of OAPT, where one set of A, B contains only $\wedge$-invariant tensors.

For proving our hardness of approximation results, we further define the Max-TropSim and Min-TropSim problems. The Max-TropSim problem was firstly proposed by Abboud and Rubinstein in [11] under the name *Tropical Tensors* in their study of LCS.

Definition 5.2.8 (Max-TropSim/Min-TropSim). 1 Let t be an even number and $d_1 = d_2 = \cdots = d_t = 2$. The *Tropical Similarity* score $s(u,v)$ of two tensors $u, v \in \{0,1\}^{d_1 \times \cdots \times d_t}$ is defined as an alternating sequence of operators E and max applied to the coordinatewise product of u and v:

$$s(u,v) = \mathsf{E}_{i_1 \in [d_1]} \left[\max_{i_2 \in [d_2]} \left\{ \mathsf{E}_{i_3 \in [d_3]} \left[\cdots \max_{i_t \in [d_t]} \{ u_i \cdot v_i \} \cdots \right] \right\} \right]. \tag{5.2}$$

Given two sets of N tensors $A, B \subseteq \{0,1\}^{d_1 \times \cdots \times d_t}$, the Max-TropSim problem asks to compute the maximum Tropical Similarity $s(a,b)$ among all pairs of $(a,b) \in A \times B$, while the Min-TropSim problem asks to compute the minimum Tropical Similarity $s(a,b)$ among all pairs of $(a,b) \in A \times B$.

Restricted OAPT is a restricted version of OAPT. We mainly use this restricted version in our analysis.

Definition 5.2.9 (Restricted Max-TropSim/Restricted Min-TropSim). We say that a tensor $x \in \{0,1\}^{d_1 \times \cdots \times d_t}$ is max-invariant if the value of $x_{i_1 \cdots i_t}$ does not depend on $i_2, i_4, i_6, \ldots, i_t$. The Restricted Max-TropSim/Restricted Min-TropSim problem is a restricted version of Max-TropSim/Min-TropSim, where one set of A, B contains only $\wedge$-invariant tensors.

Like in [11], we also define the following approximation variants of the Max-TropSim and Min-TropSim.

Definition 5.2.10 (ε-Gap-Max-TropSim). Let t be an even number and $d_1 = d_2 = \cdots = d_t = 2$. Given two sets of N tensors $A, B \in \{0,1\}^{d_1 \times \cdots \times d_t}$ of size $D = 2^t$, distinguish between the following:

- **Completeness:** There is a pair of $(a, b) \in A \times B$ with a perfect Tropical Similarity $s(a, b) = 1$;
- **Soundness:** Every pair has low Tropical Similarity score, $s(a, b) < \varepsilon$.

Here ε is a threshold value that may depend on N and D. Restricted ε-Gap-Max-TropSim is defined similarly.

Definition 5.2.11 (ε-Gap-Min-TropSim). Let t be an even number and $d_1 = d_2 = \cdots = d_t = 2$. Given two sets of N tensors $A, B \in \{0,1\}^{d_1 \times \cdots \times d_t}$ of size $D = 2^t$, distinguish between the following:

- **Completeness:** There is a pair of $(a, b) \in A \times B$ with a low Tropical Similarity $s(a, b) < \varepsilon$;
- **Soundness:** Every pair has perfect Tropical Similarity score, $s(a, b) = 1$.

Here ε is a threshold value that may depend on N and D. Restricted ε-Gap-Min-TropSim is defined similarly.

In this chapter we use the proof idea for IP = PSPACE to show that BP-Satisfying-Pair can be reduced to ε-Gap-Max-TropSim/ε-Gap-Min-TropSim of certain size in near-linear time. The proof is in Section 5.5.

5.2.3 Longest Common Subsequence

We study the hardness of *Longest Common Subsequence* (LCS) and its pair version in this chapter.

Definition 5.2.12 (LCS). Given two strings a, b of length N over alphabet Σ, the LCS problem asks to compute the length of the longest sequence that appears in both a and b as a subsequence.

Definition 5.2.13 (Closest-LCS-Pair/Furthest-LCS-Pair). Given two sets of N strings A, B, the Closest-LCS-Pair (or Furthest-LCS-Pair) problem asks to compute the maximum (or minimum) length of the longest common subsequence among all pairs of $(a, b) \in A \times B$.

5.2.4 Subtree Isomorphism and Largest Common Subtrees

We study the hardness for the following two problems on trees:

Definition 5.2.14. (Subtree Isomorphism) Given two trees G and H, the *Subtree Isomorphism* problem asks whether G is isomorphic to a subtree of H, i.e., can G and H be isomorphic after removing some nodes and edges from H.

Definition 5.2.15. (Largest Common Subtree) Given two trees G and H, the *Largest Common Subtree* problem asks to compute the size of the largest tree that is isomorphic to both a subtree of G and a subtree of H.

In this chapter, we focus on the case of unordered trees with bounded degrees. We are interested in both rooted and unrooted trees. Here "rooted" means that the root of G must be mapped to the root of H in the isomorphism.

The pair versions of these two problems are defined as follows:

Definition 5.2.16 (Subtree-Isomorphism-Pair). Given two sets of N trees A, B, the Subtree-Isomorphism-Pair problem asks whether there is a pair of trees $(a, b) \in A \times B$ such that the tree a is isomorphic to a subtree of the tree b.

Definition 5.2.17 (Max-LCST-Pair/Min-LCST-Pair). Given two sets of N trees A, B, the Max-LCST-Pair (or Min-LCST-Pair) problem asks to compute the maximum (or minimum) size of the largest common subtrees among all pairs of $(a, b) \in A \times B$.

5.2.5 Regular Expression Membership Testing

We study the hardness of testing membership for regular expression. A regular expression over an alphabet set Σ and an operator set $O = \{\circ, |, {}^{+}, {}^{*}\}$ is defined in a inductive way: (1) Every $a \in \Sigma$ is a regular expression; (2) All of $[R \mid S]$, $R \circ S$, R, $[R]^+$ are regular expressions if R and S are regular expressions. A regular expression p determines a language $L(p)$ over alphabet Σ. Specifically, for any regular expressions R, S and any $a \in \Sigma$, we have: $L(a) = \{a\}$; $L([R \mid S]) = L(R) \cup L(S)$; $L(R \circ S) = \{uv \mid u \in L(R), v \in L(S)\}$; $L([R]^+) = \bigcup_{k \geq 1}\{u_1 u_2 \cdots u_k \mid u_1, \ldots, u_k \in L(R)\}$; and $L([R]^*) = L(R^+) \cup \{\varepsilon\}$, where ε denotes the empty string. The concatenation operator $\circ$ and unnecessary parenthesis is often omitted if the meaning is clear from the context.

In this chapter, we study the *Regular Expression Membership Testing* problem, which is defined as follows:

Definition 5.2.18 (Exact Regular Expression Membership Testing)**.** Given a regular expression p of length M and a string t of length N over alphabet Σ, the *Exact Regular Expression Membership Testing* problem asks whether t is in the language $L(p)$ of p.

And its pair version is defined as follows:

Definition 5.2.19 (RegExp-String-Pair)**.** Given a set A of regular expressions of length $O(\text{poly}(D))$and a set B of N strings of length D, the RegExp-String-Pair problem asks to determine whether there is a pair (a, b) such that b is in the language $L(a)$ of a.

In [12], Abboud, Rubinstein and Williams studied a problem called RegExp Closest Pair and showed that it is SETH-hard using their distributed PCP framework. In this work, we study a slightly different problem.

Definition 5.2.20 (Closest-RegExp-String-Pair)**.** For two strings x, y of the same length n, the Hamming Similarity $\text{HamSim}(x, y)$ between x and y is defined as the fraction of positions for which the corresponding symbols are equal, i.e.,

$$\text{HamSim}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[x_i = y_i]}.$$

Given a set A of N regular expressions of length $O(\text{poly}(D))$ and a set of N strings of length D, the Closest-RegExp-String-Pair problem asks to compute the maximum

Hamming Similarity among all pairs of (x, b) satisfying $x \in L(a)$ is a string of length D for some $a \in A$ and $b \in B$.

5.3 BP-Pair-Class and an Outline of all Reductions

In this section, we first state the equivalence class formally, and outline how it is proved in this chapter.

Theorem 5.3.1 (BP-Pair-Class). *There are near linear-time*[11] *reductions between all pairs of following problems:*

1. *(Exact or Approximate Closest-LCS-Pair / Furthest-LCS-Pair) Given two sets A, B of N strings of length $D = 2^{(\log N)^{o(1)}}$, compute the exact value or a $2^{(\log D)^{1-\Omega(1)}}$-approximation of the maximum (minimum) LCS among $(a, b) \in A \times B$;*

2. *(Approximate Closest-RegExp-String-Pair) Given a set A of N regular expressions of length $2^{(\log N)^{o(1)}}$ and a set of N strings of length $D = 2^{(\log N)^{o(1)}}$, distinguish between the case that there is a pair $(a, b) \in A \times B$ such that $b \in L(a)$ (the language of a), and the case that every string in B has Hamming Similarity $< 2^{-(\log D)^{1-\Omega(1)}}$ from every string of length D in $\bigcup_{a \in A} L(a)$.*

3. *(Exact or Approximate Max-LCST-Pair / Min-LCST-Pair) Given two sets A, B of N bounded-degree trees with size at most $D = 2^{(\log N)^{o(1)}}$, compute the exact value or a $2^{(\log D)^{1-\Omega(1)}}$-approximation of the maximum (minimum) size of largest common subtree among $(a, b) \in A \times B$;*

4. *(Exact or Approximate Max-TropSim / Min-TropSim) Given two sets A, B of N binary tensors with size $D = 2^{(\log N)^{o(1)}}$, compute the exact value or a $2^{(\log D)^{1-\Omega(1)}}$-approximation of the maximum (minimum) Tropical Similarity among $(a, b) \in A \times B$;*

[11] Throughout this chapter, we use near-linear time to denote the running time of $N^{1+o(1)}$.

5. *Orthogonal-Alternating-Product-Tensors (OAPT): Given two sets A, B of N binary tensors with size $2^{(\log N)^{o(1)}}$, is there a pair $(a, b) \in A \times B$ with Alternating Product*[12] *0?*

6. *BP-Satisfying-Pair;*

7. *RegExp-String-Pair;*

8. *Subtree-Isomorphism-Pair;*

Remark 5.3.2. *A technical remark is that our reductions actually have a quasi-polynomial blow-up on the string length (tensor/tree size) D. That is, the new string length after the transformation would be at most $D' = 2^{\mathrm{polylog}(D)}$, which is still $2^{(\log N)^{o(1)}}$ assuming $D = 2^{(\log N)^{o(1)}}$. That is the reason we set the size parameter to be $2^{(\log N)^{o(1)}}$ in the equivalence class.*

For the ease of exposition, we break the proofs for Theorem 5.3.1 and Theorem 5.1.8 into many sections, each one dealing with one kind of problems. Here we give an outline of the reductions for proving Theorem 5.3.1 and Theorem 5.1.8 (Figure 5-1).

Section 5.4 **BP-Satisfying-Pair.** We present a generic reduction from $\mathcal{A}$-Satisfying-Pair and Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair problems to BP-Satisfying-Pair (Theorem 5.4.1 and Theorem 5.4.2). This implies all problems in BP-Pair-Class can be reduced to BP-Satisfying-Pair.

Section 5.5 **Tensors Problems.** We show reductions from BP-Satisfying-Pair to tensors problems OAPT, approximate Max-TropSim and Min-TropSim (Theorem 5.5.1, Theorem 5.5.6 and Theorem 5.5.7), putting these tensor problems into our BP-Pair-Class (Theorem 5.5.4 and Theorem 5.5.10).

Section 5.6 **LCS.** We show reductions from approximate Max-TropSim (Min-TropSim) to approximate Closest-LCS-Pair (Furthest-LCS-Pair) (Theorem 5.6.2 and Theorem 5.6.3), putting these LCS-Pair problems into our BP-Pair-Class (Theorem 5.6.5).

[12] see Definition 5.2.6 for a formal definition

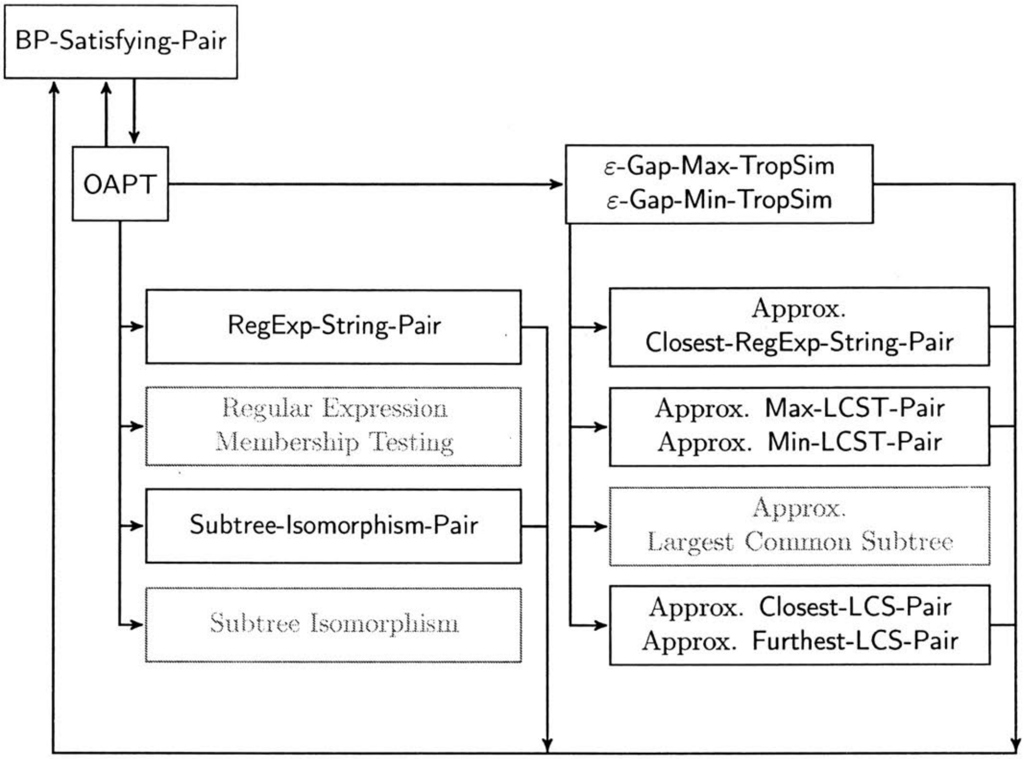

Figure 5-1: A diagram for all our reductions (red means it is BP-Pair-Hard).

Section 5.7 **Regular Expression.** We show reductions from OAPT to RegExp-String-Pair and from approximate Max-TropSim to approximate Closest-RegExp-String-Pair (Theorem 5.7.3 and Corollary 5.7.2), putting these regular expression pair problems into our BP-Pair-Class (Theorem 5.7.5).

We also show a reduction from OAPT to Regular Expression Membership Testing, showing the latter problem is BP-Pair-Hard (Theorem 5.7.6).

Section 5.8 **Subtree isomorphism.** We show reductions from OAPT to Subtree-Isomorphism-Pair and from approximate Max-TropSim (Min-TropSim) to approximate Max-LCST-Pair (Min-LCST-Pair) (Theorem 5.8.1 and Theorem 5.8.5), putting these problems related to subtree isomorphism into our BP-Pair-Class (Theorem 5.8.3 and Theorem 5.8.10).

We also show reductions from OAPT to Subtree Isomorphism and from approximate Max-TropSim to approximate Largest Common Subtree, showing that the latter problems are BP-Pair-Hard (Theorem 5.8.4 and Theorem 5.8.12).

5.4 Low-Space Algorithms Imply Reductions to BP-Satisfying-Pair

In this section, we present two important theorems for showing reductions from $\mathcal{A}$-Satisfying-Pair, Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair problems to BP-Satisfying-Pair.

The key observation is a classic result in space complexity: for $S(n) \geq \log n$, if a decision problem $\mathcal{A}$ is in $\mathsf{NSPACE}[S(n)]$, then there is a BP of length $T = 2^{O(S(n))}$ and width $W = 2^{O(S(n))}$ that decides $\mathcal{A}$ (See, e.g., [35] for the proof). This means that if $\mathcal{A}$ can be solved in small space, then we can construct a BP of not too large size to represent this algorithm.

Now we introduce our first theorem, which shows that a low-space algorithm for a decision problem $\mathcal{A}$ implies a reduction from $\mathcal{A}$-Satisfying-Pair to BP-Satisfying-Pair:

Theorem 5.4.1. *If the decision problem $\mathcal{A}$ on inputs a, b of length n can be decided in NSPACE$[\mathrm{polylog}(n)]$, then $\mathcal{A}$-Satisfying-Pair with two sets of N strings of length $2^{(\log N)^{o(1)}}$ can be reduced to BP-Satisfying-Pair on branching program of size $2^{(\log N)^{o(1)}}$ in near-linear time.*

Proof. Let $n = 2^{(\log N)^{o(1)}}$. Since $\mathcal{A}$ is in $\mathsf{NSPACE}\left[\mathrm{polylog}(n)\right]$, we can construct a BP P of size $2^{\mathrm{polylog}(n)} \leq 2^{(\log N)^{o(1)}}$ that decides $\mathcal{A}$ on inputs a, b of length n. Then to check if there is a pair of $(a, b) \in A \times B$ such that (a, b) is an Yes-instance of $\mathcal{A}$, it is sufficient to check if there is a pair of a, b making P accept. □

Our second theorem is similar. It shows that a low-space algorithm for the decision problem of an optimization problem $\mathcal{A}$ implies a reduction from Max-$\mathcal{A}$-Pair/Min-$\mathcal{A}$-Pair to BP-Satisfying-Pair:

Theorem 5.4.2. *Let $\mathcal{A}$ be an optimization problem. If the answer to $\mathcal{A}$ on input a, b of length n is bounded in $[-O(\mathrm{poly}(n)), O(\mathrm{poly}(n))]$, and the decision version of $\mathcal{A}$ (deciding whether the answer is greater than a given number k) can be decided in*

NSPACE[polylog(n)], *then Max-$\mathcal{A}$-Pair (or Min-$\mathcal{A}$-Pair) with two sets of N strings of length $2^{(\log N)^{o(1)}}$ can reduced to $2^{(\log N)^{o(1)}}$ instances of BP-Satisfying-Pair with branching programs of size $2^{(\log N)^{o(1)}}$ in near-linear time.*

Proof. For Max-$\mathcal{A}$-Pair, we enumerate each possible answer k, and then we check if there is a pair of $(a, b) \in A \times B$ with answer $> k$ by reducing to BP-Satisfying-Pair via Theorem 5.4.1. This reduction results in $O(\text{poly}(n)) \leq 2^{(\log N)^{o(1)}}$ instances of BP-Satisfying-Pair, and each branching program is of size $2^{(\log N)^{o(1)}}$. Note that NSPACE [polylog(n)] = coNSPACE [polylog(n)], thus deciding whether the answer of $\mathcal{A}$ is $\leq k$ is also in NSPACE [polylog(n)]. Using a similar argument we can reduce Min-$\mathcal{A}$-Pair to BP-Satisfying-Pair. □

5.5 Tensor Problems

In this section, we show that BP-Satisfying-Pair on branching program of size $2^{(\log N)^{o(1)}}$ is equivalent to OAPT and (exact or approximate) Max-TropSim/Min-TropSim problems on tensors of size $2^{(\log N)^{o(1)}}$ under near-linear time reductions.

5.5.1 Orthogonal Alternating Product Tensors

First we show the equivalence between BP-Satisfying-Pair and OAPT. To start with, we present the reduction from BP-Satisfying-Pair to OAPT:

Theorem 5.5.1. *There exists an $O(N \cdot 2^{O(\log W \log T)})$-time reduction from a BP-Satisfying-Pair instance with a branching program of length T and width W and two sets of N strings to an OAPT problem with two sets of N tensors of size $2^{O(\log W \log T)}$.*

Proof. Let P be a branching program of length T and width W on n Boolean inputs $x = (x_1, \dots, x_n)$. First, we follow the proof for the PSPACE-completeness of TQBF [160] to construct a quantified Boolean formula $\phi(x)$, which holds true iff the branching program P accepts x. Then, we construct two sets A', B' of N tensors such that there is a pair $(a, b) \in A \times B$ satisfying $\phi(a, b)$ is true iff there is a pair $(a', b') \in A' \times B'$ with $p_{\text{alt}}(a', b') = 0$.

Construction of Quantified Boolean Formula. We assume that $n, T-1, W$ are powers of two without loss of generality. First we construct formulas $\psi_k(x,u,v,i)$ for all $0 \le k \le \log(T-1), u, v \in [W], i \in [T]$ such that $\psi_k(x,u,v,i)$ holds true iff the node $(i+2^k, v)$ is reachable from the node (i,u) on the input $x=(x_1,\dots,x_n)$.

The construction is by induction on k. For $k=0$, we split the n input variables $x=(x_1,\dots,x_n)$ into two halves: $x^{\mathrm{a}}=(x_1,\dots,x_{n/2})$ and $x^{\mathrm{b}}=(x_{n/2+1},\dots,x_n)$. We construct two formulas $\alpha(x^{\mathrm{a}},u,v,i)$ and $\beta(x^{\mathrm{b}},u,v,i)$. We construct the formula α to be true iff the variable $x_{f(i)}$ associated with the layer L_i is in x^{a}, and there is an edge that goes from the node (i,u) to the node $(i+1,v)$ and is marked with the value of $x_{f(i)}$. We define the formula β similarly for x^{b}. Then we construct $\psi_0(x,u,v,i)=\alpha(x^{\mathrm{a}},u,v,i) \vee \beta(x^{\mathrm{b}},u,v,i)$. It is easy to see that $\psi_0(x,u,v,i)$ holds true iff the node $(i+1,v)$ is reachable from the node (i,u) on the input $x=(x_1,\dots,x_n)$. For $k \ge 1$, we construct $\psi_k(x,u,v,i)$ as:

$$(\exists m)(\forall u')(\forall v')(\forall j)[((u',v',j)=(u,m,0) \vee (u',v',j)=(m,v,1)) \Rightarrow \psi_{k-1}(x,u',v',i+j\cdot 2^{k-1})]$$

where $m, u', v' \in [W]$ and $j \in \{0,1\}$. It is easy to see that the above formula is equivalent to

$$(\exists m)[\psi_{k-1}(x,u,m,i) \wedge \psi_{k-1}(x,m,v,i+2^{k-1})],$$

thus it holds true iff the node $(i+2^k, v)$ is reachable from the node (i,u).

In the end, we construct the formula $\varphi(x)=\psi_{\log(T-1)}(x,1,1,1)$, so $\varphi(x)$ holds true iff the branching program P accepts the input $x=(x_1,\dots,x_n)$ (Recall that $u_{\mathsf{start}}=(1,1)$ and $u_{\mathrm{acc}}=(T,1)$).

We split all the variables m, u', v', j occurred in $\varphi(x)$ into t Boolean variables $z_1,\dots,z_t \in \{0,1\}$ for some $t=O(\log W \log T)$. Without loss of generality we assume t is even. Then we transform $\varphi(x)$ into the following equivalent formula ϕ:

$$\phi(x)=(\exists z_1)(\forall z_2)(\exists z_3)\cdots(\forall z_t)(f(z) \Rightarrow (g_1(x^{\mathrm{a}},z) \vee g_2(x^{\mathrm{b}},z)))$$

where $f(z)$ is the logical conjunction of all the predicates $((a,b,j)=(u,m,0) \vee$

$(a,b,j) = (m,v,1))$ in $\psi_1, \ldots, \psi_{\log(T-1)}$, and $g_1(x^{\mathrm{a}}, z), g_2(x^{\mathrm{b}}, z)$ are the innermost $\alpha(x^{\mathrm{a}}, u, v, i), \beta(x^{\mathrm{b}}, u, v, i)$. The quantifiers $\forall$ and $\exists$ appear alternatively.

Converting Quantified Boolean Formula into Tensors. Let $d_1 = \cdots = d_t = 2$. Now we construct two sets of N tensors $A', B' \subseteq \{0,1\}^{d_1 \times \cdots \times d_t}$ to be our OAPT instance. For $1 \leq k \leq t$, we associate the k-th dimension of a tensor with the variable z_k and associate each index $p \in [d_1] \times \cdots \times [d_t]$ with an assignment to $z_1, \ldots, z_t$. Note that strings in the set A correspond to assigments to x^{a}, and strings in the set B correspond to assigments to x^{b}. Thus every two strings $(a,b) \in A \times B$ along with an index p specify an assignment to x and z.

For each string $a \in A$, we construct a tensor $a' \in \{0,1\}^{d_1 \times \cdots \times d_t}$ where for every index p, a'_p is 0 iff the formula $\neg f(z) \vee g_1(x^{\mathrm{a}}, z)$ is true with corresponding assignments to x^{a} and z; for each string $b \in B$, we construct a tensor $b' \in \{0,1\}^{d_1 \times \cdots \times d_t}$ where for every index p, b'_p is 0 iff the formula $g_2(x^{\mathrm{a}}, z)$ is true with corresponding assignments to x^{b} and z.

Note that $f(z) \Rightarrow (g_1(x^{\mathrm{a}}, z) \vee g_2(x^{\mathrm{b}}, z))$ is equivalent to $\neg f(z) \vee g_1(x^{\mathrm{a}}, z) \vee g_2(x^{\mathrm{b}}, z)$, so we have

$$a'_p \wedge b'_p = 0 \iff [f(z) \Rightarrow (g_1(x^{\mathrm{a}}, z) \vee g_2(x^{\mathrm{b}}, z))].$$

Then it is easy to see that $p_{\mathrm{alt}}(a', b') = 0$ iff $\phi(a,b)$ is true, and thus P accepts a pair of $(a,b) \in A \times B$ iff $p_{\mathrm{alt}}(a', b') = 0$ for their corresponding a', b'. Note that $d_1 d_2 \cdots d_t = 2^t = 2^{O(\log W \log T)}$, so each tensor has size $2^{O(\log W \log T)}$. □

Note that in the above construction, tensors in B are all $\wedge$-invariant, so we have the following corollary for Restricted OAPT:

Corollary 5.5.2. *There exists an $O(N \cdot 2^{O(\log W \log T)})$-time reduction from a BP-Satisfying-Pair instance with a branching program of length T and width W and two sets of N strings to an Restricted OAPT problem with two sets of N tensors of size $2^{O(\log W \log T)}$.*

For the other direction, by Theorem 5.4.1, it is sufficient to show that computing Alternating Product can be done in $\mathsf{SPACE}[O(\log n)] \subseteq \mathsf{NSPACE}[\mathrm{polylog}(n)]$.

Lemma 5.5.3. *Given two tensors a, b of size $n = 2^t$, their Alternating Product $p_{\mathrm{alt}}(a, b)$ can be computed in* SPACE$[O(\log n)]$.

Proof. We compute the Alternating Product recursively according to the definition. There are t levels of recursion in total. Since $t = \log n$, space $O(\log n)$ is enough for our algorithm. □

Combining Theorem 5.5.1 and Lemma 5.5.3, we can prove the equivalence between BP-Satisfying-Pair and OAPT.

Theorem 5.5.4. *The OAPT problem on tensors of size $2^{(\log N)^{o(1)}}$ is equivalent to BP-Satisfying-Pair on branching program of size $2^{(\log N)^{o(1)}}$ under near-linear time reductions.*

5.5.2 A Communication Protocol for Branching Program

Before we turn to show the equivalence between BP-Satisfying-Pair and Max-TropSim/Min-TropSim, we introduce the following IP-protocol for branching program. Our reduction from BP-Satisfying-Pair to Max-TropSim (or Min-TropSim) directly follows by simulating the communication protocol using tropical algebra.

Theorem 5.5.5. *Let P be a branching program of length T and width W on n Boolean inputs $x_1, \ldots, x_n$. Suppose Alice holds the input $x_1, \ldots, x_{n/2}$ and Bob holds the input $x_{n/2+1}, \ldots, x_n$. For every $\varepsilon > 0$, there exists a computationally efficient IP-protocol for checking whether P accepts on $x_1, \ldots, x_n$, in which:*

1. *Merlin and Alice exchange $O(\log^2 W \log^2 T \cdot (\log\log W + \log\log T + \log \varepsilon^{-1}))$ bits;*
2. *Alice tosses $O(\log^2 W \log^2 T \cdot (\log\log W + \log\log T + \log \varepsilon^{-1}))$ public coins;*
3. *Bob sends $O(\log\log W + \log\log T + \log \varepsilon^{-1})$ bits to Alice;*
4. *Alice accepts or rejects in the end.*

If P accepts on the input $x_1, \ldots, x_n$, then Alice always accepts; otherwise, Alice rejects with probability at least $1 - \varepsilon$.

Proof. Let $\bar{a}$ be the assignment to the input variables held by Alice, and $\bar{b}$ be the assignment to the input variables held by Bob. Recall the construction of the tensors in the proof for Theorem 5.5.1. First Alice constructs a tensor $a = G(\bar{a})$, and Bob constructs a tensor $b = H(\bar{b})$. Each tensor here is of shape $d_1 \times d_2 \times \cdots d_t = 2 \times 2 \times \cdots \times 2$ for $t = O(\log T \log W)$. Then the problem reduces to check whether the Alternating Product $p_{\mathrm{alt}}(a, b)$ equals 0. Now we show that there exists a communication protocol for checking $p_{\mathrm{alt}}(a, b) = 0$, using the idea for proving $\mathsf{IP} = \mathsf{PSPACE}$ [123, 155].

Arithmetization. First we arithmetize the computation of Alternating Product. Let $q \geq 1$ be a parameter to be specified. Construct a finite field $\mathbb{F}_{2^q}$. Then Alice finds a multilinear extension α over $\mathbb{F}_{2^q}$ for her tensor a, i.e., Alice finds a function $\alpha(z_1, \ldots, z_t)$ such that α is linear in each of its variables, and $\alpha(z_1, \ldots, z_t) = a_i$ for all $i \in [d_1] \times \cdots \times [d_t]$ and $i_k = z_k + 1$ $(1 \leq k \leq t)$. Bob finds a multilinear extension β for his tensor b similarly. Recall that the definition of Alternating Product. $p_{\mathrm{a}}(a, b)$ can be rewritten as

$$p_{\mathrm{alt}}(a,b) = \bigwedge_{z_1 \in \{0,1\}} \left[\bigvee_{z_2 \in \{0,1\}} \left(\bigwedge_{z_3 \in \{0,1\}} \left[\cdots \bigvee_{z_t \in \{0,1\}} (\alpha(z_1, \ldots, z_t) \cdot \beta(z_1, \ldots, z_t)) \cdots \right] \right) \right].$$

To arithmetize $\bigwedge_{z_k \in \{0,1\}}$ and $\bigvee_{z_k \in \{0,1\}}$, we define three kinds of operators acting on polynomials:

1. Π_{z_m} operator, which arithmetizes the formula $\bigwedge_{z_m \in \{0,1\}} F(z_1, \ldots, z_{m-1}, z_m)$.

$$\Pi_{z_m} F(z_1, \ldots, z_m) = F(z_1, \ldots, z_{m-1}, 0) \cdot F(z_1, \ldots, z_{m-1}, 1)$$

2. Σ_{z_m} operator, which arithmetizes the formula $\bigvee_{z_m \in \{0,1\}} F(z_1, \ldots, z_{m-1}, z_m)$.

$$\Sigma_{z_m} F(z_1, \ldots, z_m) = 1 - (1 - F(z_1, \ldots, z_{m-1}, 0)) \cdot (1 - F(z_1, \ldots, z_{m-1}, 1)).$$

3. $\mathcal{R}_{z_i}$ operator, which is used for the degree reduction. When acting on a polynomial $F(z_1, \ldots, z_m)$, it replaces z_i^k for $k \geq 1$ by z_i in all terms. In this way, any polynomial $F(z_1, \ldots, z_m)$ can be converted into a multilinear one preserving the values at every $(z_1, \ldots, z_m) \in \{0,1\}^m$. $\mathcal{R}_{z_i}$ operator can be written as

$$\mathcal{R}_{z_i} F(z_1, \ldots, z_m) = F(z_1, \ldots, z_{i-1}, 0, z_{i+1}, \ldots z_m) + z_i \cdot F(z_1, \ldots, z_{i-1}, 1, z_{i+1}, \ldots z_m).$$

Then it is easy to see that

$$p_{\text{alt}}(a, b) = \Pi_{z_1} \Sigma_{z_2} \Pi_{z_3} \cdots \Sigma_{z_t} (\alpha \cdot \beta).$$

Note that in the computation of Alternating Product, we only use the function value at Boolean inputs, thus we can insert i operators $R_{z_1} R_{z_2} \cdots R_{z_i}$ right after each π_{z_i} or Σ_{z_i} without changing the final result:

$$p_{\text{alt}}(a, b) = \Pi_{z_1} R_{z_1} \Sigma_{z_2} R_{z_1} R_{z_2} \Pi_{z_3} R_{z_1} R_{z_2} R_{z_3} \cdots \Sigma_{z_t} R_{z_1} \cdots R_{z_t} (\alpha \cdot \beta).$$

In total we use only $M = O(t^2) \leq O(\log^2 T \log^2 W)$ operators.

The Protocol. We introduce our IP-protocol in an inductive way. Suppose that we have an IP-protocol for some polynomial $F(z_1, \ldots, z_m)$, in which for any given $(v_1, \ldots, v_m) \in \mathbb{F}_{2^q}^m$ and $u = F(v_1, \ldots, v_m)$, Merlin can convince Alice and Bob that $F(v_1, \ldots, v_m) = u$ with perfect completeness and soundness error ε_0. We show that for $G(z_1, \ldots, z_{m'}) = \mathcal{O}_{z_i} F(z_1, \ldots, z_m)$ and given $v_1, \ldots, v_{m'}$ and u' ($\mathcal{O}_{z_i} \in \{\Sigma_{z_i}, \Pi_{z_i}, \mathcal{R}_{z_i}\}$, $m' = m$ when $\mathcal{O}_{z_i} = \mathcal{R}_{z_i}$ and $m' = m - 1$ otherwise), Merlin can convince Alice and Bob that $G(v_1, \ldots, v_{m'}) = u'$ with perfect completeness and soundness error $\varepsilon_0 + O(2^{-q})$:

- First Merlin sends the coefficients of the polynomial $F(v_1, \ldots, v_{i-1}, z_i, v_{i+1}, \ldots, v_m)$ to Alice (note that it is a univariate polynomial of z_i);

- Alice calculates the value of $G(v_1, \ldots, v_{m'})$ using the information sent by Merlin (assuming Merlin is honest), and reject if $G(v_1, \ldots, v_{m'}) \neq u'$;

- Alice randomly draws an element $r \in \mathbb{F}_{2^q}$. Let $v_i = r$ (reset $v_i = r$ if v_i already exists);
- Alice checks if $F(v_1, \ldots, v_{i-1}, r, v_{i+1}, \ldots, v_m) = u$ via the IP-protocol for F.

It is easy to see that the above protocol has perfect completeness. For soundness, notice that F and G are always of $O(1)$ degree because our use of degree reduction operators, thus Alice can find $G(v_1, \ldots, v_{m'}) \neq u'$ with probability $O(2^{-q})$ if Merlin lies. By the union bound, the soundness error of the IP-protocol for G is $\varepsilon_0 + O(2^{-q})$.

Our IP-protocol starts by checking $p_{\mathrm{alt}}(a, b) = 0$. Following the inductive process above, there are M rounds of communication between Merlin and Alice. And after the last round, $p_{\mathrm{alt}}(a, b) = 0$ reduces to check if $\alpha(v_1, \ldots, v_t) \cdot \beta(v_1, \ldots, v_t) = u$ for given $(v_1, \ldots, v_t) \in \mathbb{F}_{2^q}^t, u \in \mathbb{F}_{2^q}$. Note that all the values of $v_1, \ldots, v_t$ can be inferred by the results of public coins Alice tossed. Thus the IP-protocol for $\alpha \cdot \beta$ is as follows: Bob learns the results of public coins and obtains $v_1, \ldots, v_t$. Then Bob sends the value of $\beta(v_1, \ldots, v_t)$ to Alice. Finally, Alice accepts iff $\alpha(v_1, \ldots, v_t) \cdot \beta(v_1, \ldots, v_t) = u$.

By induction, we can show that the whole IP-protocol has perfect completeness and soundness error $O(M \cdot 2^{-q})$. Setting $2^q = c \cdot M \cdot \varepsilon^{-1}$ for large enough constant c, we can achieve the soundness error ε. And in this case we have

$$q = \log M + \log \varepsilon^{-1} + \log c = O(\log \log W + \log \log T + \log \varepsilon^{-1}).$$

It can be easily seen that Alice tosses

$$O(Mq) = O(\log^2 T \log^2 W(\log \log T + \log \log W + \log \varepsilon^{-1}))$$

public coins and Bob sends

$$O(q) = O(\log \log T + \log \log W + \log \varepsilon^{-1})$$

bits to Alice in our communication protocol.

In each of the M rounds, Merlin sends $O(1)$ elements in $\mathbb{F}_{2^q}$ since F is of at most constant degree. Thus Merlin sends at most

$$O(Mq) = O(\log^2 T \log^2 W(\log\log T + \log\log W + \log \varepsilon^{-1}))$$

bits to Alice. □

5.5.3 Tropical Tensors

Following from [11], we can show a reduction from BP-Satisfying-Pair to ε-Gap-Max-TropSim based on our IP-protocol for branching program.

Theorem 5.5.6. *There is a reduction from BP-Satisfying-Pair on branching program of length T and width W and two sets of N strings to ε-Gap-Max-TropSim on two sets of N tensors of size*

$$D = 2^{O(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))},$$

and the reduction runs in $O(N \operatorname{poly}(D))$. Here ε is a threshold value that can depend on N.

Proof. For convenience, let

$$K = \log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}).$$

By Theorem 5.5.5, there is an IP-protocol using $O(K)$ bits for determining whether a branching program accepts when Alice knows the first half and Bob knows the second half, with soundness error ε. We can easily modify the communication protocol such that

- Alice and Merlin interact for $m = O(K)$ rounds, in each round Merlin sends one bit to Alice and Alice tosses one public coin;
- After the interaction between Alice and Merlin, Bob sends $\ell = O(K)$ bits to Alice, and after Bob sending each bit Merlin sends a dummy bit to Alice;

- Alice accepts or rejects in the end.

Let $t = 2\ell + 2m$ and $d_1 = \cdots = d_t = 2$. Now we construct two sets of $N = 2^{n/2}$ tensors $A, B \in \{0,1\}^{d_1 \times \cdots \times d_t}$ as our ε-Gap-Max-TropSim instance. For every $0 \le k < m$, we associate the $(t-2k)$-th dimension with the result of the public coin Alice tosses at the $(k+1)$-th round and associate the $(t-2k-1)$-th dimension with the bit Merlin sends to Alice at the $(k+1)$-th round. For every $0 \le k < \ell$, we associate the $(2\ell - 2k)$-th dimension of a tensor with the $(k+1)$-th bit sent by Bob and associate the $(2\ell - 2k - 1)$-th dimension with the bit Merlin sent to Alice right after Bob sending the $(k+1)$-th bit to Alice. In this way, every index $p \in [d_1] \times \cdots \times [d_t]$ of a tensor can be seen as a communication transcript.

Let $A, B \subseteq \{0,1\}^{n/2}$ be the two sets in the BP-Satisfying-Pair instance. For each assignment $a \in A$ to the first half variables $x_1, \ldots, x_{n/2}$, we construct a tensor $G(a) \in \{0,1\}^{d_1 \times \cdots \times d_t}$ where for every index p, $G(a)_p$ is 1 iff Alice accepts after seeing the communication transcript p when she holds the assignment a for $x_1, \ldots, x_{n/2}$. For each assignment $b \in B$ to the second half variables $x_{n/2+1}, \ldots, x_n$, we construct a tensor $H(b) \in \{0,1\}^{d_1 \times \cdots \times d_t}$ where for every index p, $H(b)_p$ is 1 iff the bits sent by Bob in the communication transcript p matches what Bob sends when he holds the assignment b for $x_{n/2+1}, \ldots, x_n$ and learning the results of Alice's public coins in p.

Let $D = 2^t = 2^{O(K)}$ be the size of each tensor. It is easy to see that when Alice holds a and Bob holds b, the maximum probability (over all Merlin's actions) that Alice accepts in our communication protocol equals the Tropical Similarity $s(G(a), H(b))$. □

By negating the branching program P, we can also show a similar reduction from BP-Satisfying-Pair to ε-Gap-Min-TropSim:

Theorem 5.5.7. *There is a reduction from BP-Satisfying-Pair on branching program of length T and width W and two sets of N strings to ε-Gap-Min-TropSim on two sets of N tensors of size*

$$D = 2^{O(\log^2 W \log^2 T (\log\log W + \log\log T + \log \varepsilon^{-1}))},$$

and the reduction runs in $O(N \operatorname{poly}(D))$. Here ε is a threshold value that can depend on N.

Proof. The IP-protocol in Theorem 5.5.5 can be easily adapted to check the branching program P does *not* accept, i.e., if P rejects on the input $x_1, \ldots, x_n$, then Alice always accepts; otherwise, Alice rejects with probability $1 - \varepsilon$. To do this, the only thing we need to change is to check whether the Alternating Product is 1 rather than 0. Then, using the same reduction as in Theorem 5.5.6, we can obtain two sets $A' = \{G(a) \mid a \in A\}, B' = \{H(b) \mid b \in B\}$ of N tensors of size

$$D = 2^{O(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))}$$

such that for every pair of strings $(a, b) \in A \times B$, the maximum probability (over all Merlin's actions) that Alice accepts in the IP-protocol equals the Tropical Similarity score of the corresponding tensor gadgets $G(a)$ and $H(b)$. Thus, to decide whether there exists a pair of $(a, b) \in A \times B$ that can make P accept, it is sufficient to distinguish from the case that there is a pair of $G(a), H(b)$ such that the Tropical Similarity score $s(G(a), H(b)) \leq \varepsilon$ and the case that every pair of $G(a), H(b)$ has perfect Tropical Similarity score $s(G(a), H(b)) = 1$. □

Note that in the above constructions in Theorem 5.5.6 and 5.5.7, tensors in B are all max-invariant, so we have the following corollary for Restricted OAPT:

Corollary 5.5.8. *There is a reduction from BP-Satisfying-Pair on branching program of length T and width W and two sets of N strings to a Restricted ε-Gap-Max-TropSim/Restricted ε-Gap-Min-TropSim on two sets of N tensors of size*

$$D = 2^{O(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))},$$

and the reduction runs in $O(N \operatorname{poly}(D))$. Here ε is a threshold value that can depend on N.

Theorem 5.5.6 and 5.5.7 also imply reductions from BP-Satisfying-Pair to exact

Max-TropSim or exact Min-TropSim. For the other direction of reduction, we have the following lemma:

Lemma 5.5.9. *Given two tensors a, b of size $n = 2^t$, their Tropical Similarity $s(a, b)$ can be computed in SPACE$[O(\log^2 n)]$.*

Proof. We compute the Tropcial Similarity recursively according to the definition. Note that there are t levels of recursion in total, and $O(t)$-bit precision is sufficient in this computation. Thus this algorithm uses only $O(t^2) \leq O(\log^2 n)$ space. □

Combining Theorem 5.5.6, Theorem 5.5.7 and Lemma 5.5.9, we can establish the equivalence between BP-Satisfying-Pair and the exact or approximate Tropical Similarity problems:

Theorem 5.5.10. *For the case that the maximum element size $D = 2^{(\log N)^{o(1)}}$, there are near-linear time reductions between all pairs of the following problems:*

- *BP-Satisfying-Pair;*
- *Exact Max-TropSim or Min-TropSim;*
- *$2^{-(\log D)^{1-\Omega(1)}}$-Gap-Max-TropSim or $2^{-(\log D)^{1-\Omega(1)}}$-Gap-Min-TropSim;*
- *$2^{(\log D)^{1-\Omega(1)}}$-approximate Max-TropSim or Min-TropSim;*

Proof. Let $c > 0$ be a constant. For any instance of BP-Satisfying-Pair on BP of size S, by Theorem 5.5.6 with parameter $\varepsilon = 2^{-(\log S)^c}$, we can reduce it to an ε-Gap-Max-TropSim instance on tensors of size $D = 2^{\Theta(\log^4 S \log \varepsilon^{-1})} = 2^{\Theta(\log^{4+c} S)}$ (adding dummy dimensions to tensors if necessary).

Thus, we have $\varepsilon = 2^{-\Theta(\log D)^{c/(4+c)}}$. For any $0 < \delta \leq 1$, by choosing an appropriate value for c, we can obtain a reduction from BP-Satisfying-Pair on BP of size $S = 2^{(\log N)^{o(1)}}$ to $2^{-(\log D)^{1-\delta}}$-Gap-Max-TropSim.

$2^{-(\log D)^{1-\delta}}$-Gap-Max-TropSim can be trivially reduced to $2^{(\log D)^{1-\delta}}$-approximate Max-TropSim, and $2^{(\log D)^{1-\delta}}$-approximate Max-TropSim can be trivially reduced to Max-TropSim. By Lemma 5.5.9 and Theorem 5.4.2, Max-TropSim can be reduced to BP-Satisfying-Pair.

Therefore under near-linear time reductions BP-Satisfying-Pair, exact Max-TropSim, $2^{-(\log D)^{1-\Omega(1)}}$-Gap-Max-TropSim and $2^{(\log D)^{1-\Omega(1)}}$-approximate Max-TropSim are all equivalent. Using a similar argument, we can also prove the same result for Min-TropSim.

□

5.6 Longest Common Subsequence

In this section, we show near-liear time reductions between BP-Satisfying-Pair and (exact or approximate) Closest-LCS-Pair/Furthest-LCS-Pair.

Our reduction from BP-Satisfying-Pair to Closest-LCS-Pair / Furthest-LCS-Pair relies on the following LCS gadgets in [11].

Theorem 5.6.1 ([11]). *Let t be an even number and $d_1 = \cdots = d_t = 2$. Given two sets of N tensors A, B in $\{0,1\}^{d_1 \times \cdots \times d_t}$, there is a deterministic algorithm running in $O(N \operatorname{poly}(2^t))$ time which outputs two sets A', B' of N strings of length 2^t over an $O(2^t)$-size alphabet Σ, such that each $a \in A$ corresponds to a string $a' \in A'$, each $b \in B$ corresponds to a string $b \in B'$, and $\mathsf{LCS}(a', b') = 2^{t/2} \cdot s(a, b)$ for every $a \in A, b \in B$, where $s(a, b)$ stands for the Tropical Similarity score of two tensors a and b.*

Theorem 5.6.1 directly leads to the following two theorems:

Theorem 5.6.2. *There exists an $O(N \operatorname{poly}(D))$-time reduction from an $\varepsilon(D)$-Gap-Max-TropSim instance with two sets of N tensors of size D to an instance of the following approximation variant of Closest-LCS-Pair: Given two sets of N strings of length D, distinguish between the following:*

- ***Completeness:** There exists a pair of a, b such that $\mathsf{LCS}(a, b) = \sqrt{D}$;*
- ***Soundness:** For every pair a, b, $\mathsf{LCS}(a, b) < \sqrt{D} \cdot \varepsilon(D)$.*

Thus $\varepsilon(D)$-Gap-Max-TropSim can be $O(N \operatorname{poly}(D))$-time reduced to $\varepsilon(D)^{-1}$-approximate Closest-LCS-Pair.

Theorem 5.6.3. *There exists an $O(N \operatorname{poly}(D))$-time reduction from an $\varepsilon(D)$-Gap-Min-TropSim instance with two sets of N tensors of size D to an instance of the following approximation variant of Furthest-LCS-Pair: Given two sets of N strings of length D, distinguish between the following:*

- ***Completeness:*** *There exists a pair of a, b such that $\mathsf{LCS}(a,b) < \sqrt{D} \cdot \varepsilon(D)$;*

- ***Soundness:*** *For every pair a, b, $\mathsf{LCS}(a,b) = \sqrt{D}$.*

Thus $\varepsilon(D)$-Gap-Min-TropSim can be $O(N \operatorname{poly}(D))$-time reduced to $\varepsilon(D)^{-1}$-approximate Furthest-LCS-Pair.

According to Theorem 5.4.2, in order to reduce Closest-LCS-Pair or Furthest-LCS-Pair problem to BP-Satisfying-Pair, we only need to show that given a number k, deciding $\mathsf{LCS}(a,b) \geq k$ for two strings a, b of length n is in $\mathsf{NSPACE}[\operatorname{polylog}(n)]$:

Lemma 5.6.4 (Folklore). *Given two strings a, b of length n and a number k, deciding whether $\mathsf{LCS}(a,b) \geq k$ is in NL.*

Proof. The algorithm consists of k stages. Let c be a longest common subsequence of a and b. In the i-th stage, we nondeterministically guess which positions of a and b are matched with the i-th character of c, and then we check if the characters on the two positions of a and b are the same. Also, in each stage we store the positions being matched in the last stage, so that we can check if the matched positions in each string are increasing. Finally, we accept if the nondeterministic guesses pass all the checks. The total space for this nondeterministic algorithm is $O(\log n)$ since we only need to maintain $O(1)$ positions in each stage. □

Combining Theorem 5.6.2, 5.6.3 and Lemma 5.6.4 together, we can prove the equivalence between BP-Satisfying-Pair and exact or approximate LCS pair problems:

Theorem 5.6.5. *For the case that the maximum element size $D = 2^{(\log N)^{o(1)}}$, there are near-linear time reductions between all pairs of the following problems:*

- *BP-Satisfying-Pair;*

- *Exact* Closest-LCS-Pair *or* Furthest-LCS-Pair;

- $2^{(\log D)^{1-\Omega(1)}}$*-approximate* Closest-LCS-Pair *or* Furthest-LCS-Pair.

Proof. By Theorem 5.5.10, the BP-Satisfying-Pair problem on branching program of size $2^{(\log N)^{o(1)}}$ is equivalent to $2^{(\log D)^{1-\delta}}$-Gap-Max-TropSim under near-linear time reductions. By Theorem 5.6.2, $2^{(\log D)^{1-\delta}}$-Gap-Max-TropSim can be reduced to $2^{(\log D)^{1-\delta}}$-approximate Closest-LCS-Pair. $2^{(\log D)^{1-\delta}}$-approximate Closest-LCS-Pair can be trivially reduced to Closest-LCS-Pair. By Lemma 5.6.4 and Theorem 5.4.2, Closest-LCS-Pair can further be reduced to BP-Satisfying-Pair. Thus BP-Satisfying-Pair, exact Closest-LCS-Pair and $2^{(\log D)^{1-\delta}}$-approximate Closest-LCS-Pair are equivalent under near-linear time reductions for all $\delta > 0$.

Using a similar argument, we can also prove the same result for exact and approximate Furthest-LCS-Pair. □

5.7 Regular Expression Membership Testing

In this section, we study the hardness of regular expression problems. First we prove that BP-Satisfying-Pair, RegExp-String-Pair and Closest-RegExp-String-Pair are equivalent under near-linear time reductions, then we show the hardness for the Regular Expression Membership Testing problem.

For simplicity, we denote $\max_{x \in L(a), |x|=|b|} \text{HamSim}(x, b)$ by $\text{MaxSim}(a, b)$ for any regular expression a and string b. The following theorem gives a construction to implement the Tropical Similarity using MaxSim.

Theorem 5.7.1. *Let t be an even number and $d_1 = \cdots = d_t = 2$. Given two sets of N tensors $A, B \subseteq \{0,1\}^{d_1 \times \cdots \times d_t}$ satisfying that all the tensors in B are* max*-invariant, there is a deterministic algorithm running in $O(N \operatorname{poly}(2^t))$ time which outputs a set A' of N regular expressions and a set B' of N strings. Here strings are of length 2^t, regular expressions are of length* $\operatorname{poly}(2^t)$*, and both of them are over alphabet $\Sigma = \{0, 1, \perp\}$. Each $a \in A$ corresponds to a regular expression $a' \in A'$, each $b \in B$*

corresponds to a string $b' \in B'$, and

$$\mathrm{MaxSim}(a', b') = s(a, b).$$

Proof. For each k and each prefix of index $i_{(k)} \in [d_1] \times \cdots \times [d_k]$, we construct corresponding gadget $a' = G_{i_{(k)}}(a)$ and $b' = H_{i_{(k)}}(b)$ for each $a \in A$ and $b \in B$ (when $k = 0$, $i_{(k)}$ can only be the empty prefix, and we simply use $G(a)$ and $H(b)$ for convenience) inductively, mimicking the evaluation of the Tropical Similarity. For this purpose, we need to construct the following three types of gadgets.

Bit Gadgets. First we need bit gadgets to simulate the innermost coordinatewise product in the evaluation of Tropical Similarity. For each coordinate $i \in [d_1]\times\cdots\times[d_t]$, for every $a \in A$ and $b \in B$, we construct

$$G_i(a) = \begin{cases} \bot & \text{if } a_i = 0, \\ 1 & \text{if } a_i = 1, \end{cases} \quad \text{and} \quad H_i(b) = \begin{cases} 0 & \text{if } b_i = 0, \\ 1 & \text{if } b_i = 1. \end{cases}$$

It is easy to see that $a_i \cdot b_i = \mathrm{MaxSim}(G_i(a), H_i(b))$.

Now we combine bit gadgets recursively according to the max and E operators in the evaluation for Tropical Similarity. Starting from $k = t - 1$, there are two cases to consider.

Expectation Gadgets. The first case is when E operator is applied to $(k + 1)$-th dimension. We construct the corresponding gadgets $G_{i_{(k)}}(a)$ and $G_{i_{(k)}}(b)$ for any $i_{(k)} \in [d_1] \times \cdots \times [d_k]$, $a \in A$ and $b \in B$ as follows:

$$G_{i_{(k)}}(a) = G_{i_{(k)},0}(a) \circ G_{i_{(k)},1}(a) \quad \text{and} \quad H_{i_{(k)}}(b) = H_{i_{(k)},0}(b) \circ H_{i_{(k)},1}(b).$$

where $\circ$ stands for concatenation as usual. It is easy to see that

$$\mathrm{MaxSim}(G_{i_{(k)}}(a), H_{i_{(k)}}(b)) = \mathsf{E}_{j\in\{0,1\}}\left[\mathrm{MaxSim}(G_{i_{(k)},j}(a), H_{i_{(k)},j}(b))\right].$$

Max Gadgets. The second case is when max operator is applied to $(k+1)$-th dimension. For $i_{(k)} \in [d_1] \times \cdots \times [d_k]$ and $a \in A$, $G_{i_{(k)}}(a)$ is constructed as follows:

$$G_{i_{(k)}}(a) = \left[\; G_{i_{(k)},0}(a) \;\middle|\; G_{i_{(k)},1}(a) \;\right],$$

and we construct $H_{i_{(k)}}(b)$ for $b \in B$ to be

$$H_{i_{(k)}}(b) = H_{i_{(k)},j}(b)$$

for all $j \in \{0,1\}$, which is well-defined since b is max-invariant. It is easy to see that

$$\mathrm{MaxSim}(G_{i_{(k)}}(a), H_{i_{(k)}}(b)) = \max_{j \in \{0,1\}} \left[\mathrm{MaxSim}(G_{i_{(k)},j}(a), H_{i_{(k)},j}(b))\right].$$

Finally, we can obtain tensor gadgets $G(a), H(b)$ for each $a \in A$ and $b \in B$. □

From Theorem 5.7.1, we have the following reduction:

Corollary 5.7.2. *Let ε-Gap-Closest-RegExp-String-Pair be the approximation variant of Closest-RegExp-String-Pair: Given a set of N regular expressions of length $O(\mathrm{poly}(D))$ and a set of N strings of length D, distinguish between the following:*

- ***Completeness:*** *There exists a pair of a, b such that $\mathrm{MaxSim}(a,b) = 1$ (i.e., $b \in L(a)$);*
- ***Soundness:*** *For every pair a, b, $\mathrm{MaxSim}(a,b) < \varepsilon$.*

There exists an $O(N\,\mathrm{poly}(D))$-time reduction from a Restricted $\varepsilon(D)$-Gap-Max-TropSim instance on two sets A, B of N tensors of size D to an instance of $\varepsilon(D)$-Gap-Closest-RegExp-String-Pair on a set of N regular expressions of length $O(\mathrm{poly}(D))$ and a set of N strings of length D.

This corollary also follows that there is a reduction from ε-Gap-Max-TropSim to RegExp-String-Pair, which is enough to show that RegExp-String-Pair is no easier than BP-Satisfying-Pair. But actually it is possible to show a direct reduction from Restricted OAPT to RegExp-String-Pair, without using the reduction from Restricted OAPT to Restricted ε-Gap-Max-TropSim:

Theorem 5.7.3. *There exists an $O(N \operatorname{poly}(D))$-time reduction from a Restricted OAPT instance with two sets A, B of N tensors of size D to an RegExp-String-Pair instance with a set of N regular expressions of length $O(\operatorname{poly}(D))$ and a set of N strings of length D.*

Proof. We use nearly the same reduction as in Theorem 5.7.1. The bit gadgets are constructed as follows:

$$G_i(a) = \begin{cases} [0 \mid 1] & \text{if } a_i = 0, \\ 0 & \text{if } a_i = 1, \end{cases} \quad \text{and} \quad H_i(b) = \begin{cases} 0 & \text{if } b_i = 0, \\ 1 & \text{if } b_i = 1. \end{cases}$$

for any $a \in A$, $b \in B$. And we construct $\wedge$ gadgets in the same way as the max gadgets, $\vee$ gadgets in the same way as the E gadgets. By De Morgan's laws, we can show that $H(b) \in L(G(a))$ iff $p_{\text{alt}}(a, b) = 0$. □

For the other direction, we note that the following theorem gives a low-space algorithm for exact and approximate regular expression membership testing, then we can obtain a reduction by Theorem 5.4.1. The following theorem is noted in [104]:

Theorem 5.7.4 ([104]). *Given a regular expression a and a string b, deciding whether $b \in L(a)$ is in NL.*

Combining all the above reductions together, we can show the equivalence between all pairs of BP-Satisfying-Pair, RegExp-String-Pair and ε-Gap-Closest-RegExp-String-Pair.

Theorem 5.7.5. *For the case that the maximum element size $D = 2^{(\log N)^{o(1)}}$, there are near-linear time reductions between all pairs of the following problems:*

- *BP-Satisfying-Pair;*
- *RegExp-String-Pair;*

- $2^{(\log D)^{1-\Omega(1)}}$-*Gap-Closest-RegExp-String-Pair*.

Proof. By Theorem 5.7.1 and 5.7.4, we can show cyclic reductions between these three problems in a similar way as in the proof of Theorem 5.6.5. □

We can also show a reduction from Restricted OAPT to Regular Expression Membership Testing on two strings using the same gadgets in Theorem 5.7.3.

Theorem 5.7.6. *There exists an $O(N \operatorname{poly}(D))$-time reduction from a Restricted OAPT instance with two sets A, B of N tensors of size D to an instance of Regular Expression Membership Testing on a regular expression R of length $O(N \operatorname{poly}(D))$ and a string S of length $O(N \operatorname{poly}(D))$.*

Proof. We construct the two sets A', B' as in Theorem 5.7.1. For the construction for the regular expression, let w be the concatenation of all $a' \in A$ separated by " | ". Then we construct the regular expression R to be

$$R = \left[\bigcirc_{i=1}^{D}[0 \mid 1]\right]^* \circ [w] \circ \left[\bigcirc_{i=1}^{D}[0 \mid 1]\right]^*$$

and we construct the string S by concatenating all $b' \in B$ directly. It is easy to see that there exists a pair $(a, b) \in A \times B$ with $p_{\text{alt}}(a, b) = 0$ iff $S \in L(R)$, by noticing that all the strings $x \in L(w), b' \in B$ are of the same length D. □

5.8 Subtree Isomorphism and Largest Common Subtree

In this section, we study the hardness of Subtree Isomorphism and Largest Common Subtree. Our reductions here are inspired by [4, 11]. We begin with some notations to ease our construction of trees. Recall that all trees considered in this chapter are bounded-degree and unordered. We are interested in both rooted and unrooted trees. Here "rooted" means that the root of G must be mapped to the root of H in the isomorphism.

We use $\mathcal{T}_2$ to denote the tree with exactly two nodes. Let $\mathcal{T}_3^0$ be the 3-node tree with root degree 1, and let $\mathcal{T}_3^1$ be the 3-node tree with root degree 2. For a tree T, let $\mathcal{P}^k(T)$ be the tree constructed by joining a path of k nodes and the tree T: one end of the path is regarded as the root, the other end of the path is linked to the root of T by an edge. For two trees $\mathcal{T}_a$ and $\mathcal{T}_b$, we use $(\mathcal{T}_a \circ \mathcal{T}_b)$ to denote the tree whose root has two children $\mathcal{T}_a$ and $\mathcal{T}_b$.

5.8.1 Subtree Isomorphism

In this subsection, first we prove that $\mathsf{BP\text{-}Satisfying\text{-}Pair}$ and $\mathsf{Subtree\text{-}Isomorphism\text{-}Pair}$ are equivalent under near-linear time reductions, then we show the hardness for Subtree Isomorphism on two trees.

For two trees T_a, T_b, we use $\mathsf{STI}(T_a, T_b)$ to indicate whether T_a is isomorphic to a subtree of T_b when T_a, T_b are seen as unrooted trees. Also, we use $\mathsf{RSTI}(a, b)$ to indicate whether T_a is isomorphic to a subtree of T_b when T_a, T_b are seen as rooted trees.

Theorem 5.8.1. *Let t be an even number and $d_1 = \cdots = d_t = 2$. Given two sets of N tensors A, B in $\{0,1\}^{d_1 \times \cdots \times d_t}$ satisfying that all the tensors in A are $\wedge$-invariant, there is a deterministic algorithm running in $O(N \operatorname{poly}(2^t))$ time which outputs two sets A', B' of N binary trees of size $O(2^t)$ and depth $O(t)$, such that each $a \in A$ corresponds to a tree $a' \in A'$, each $b \in B$ corresponds to a tree $b \in B'$, and*

$$\overline{p_{\mathsf{alt}}(a, b)} = \mathsf{RSTI}(a', b') = \mathsf{STI}(a', b'),$$

where $\overline{p_{\mathsf{alt}}(a, b)}$ is the negation of the Alternating Product of a and b.

Proof. For each k and each prefix of index $i_{(k)} \in [d_1] \times \cdots \times [d_k]$, we construct corresponding tree gadgets $G_{i_{(k)}}(a)$ and $H_{i_{(k)}}(b)$ for each $a \in A$ and $b \in B$ (when $k = 0$, $i_{(k)}$ can only be the empty prefix, and we simply use $G(a)$ and $H(b)$ for convenience) inductively, mimicking the evaluation of the alternating product. Our gadgets satisfy that

$$\mathsf{RSTI}(G_{i_{(k)}}(a), H_{i_{(k)}}(b)) = \overline{p_{\text{alt}}(a_{i_{(k)}}, b_{i_{(k)}})}$$

for any subtensors $a_{i_{(k)}}, b_{i_{(k)}}$. For this purpose, we need to construct the following three types of tree gadgets.

Bit Gadgets. First we need bit gadgets to simulate the innermost coordinatewise product in the alternating product. For each coordinate $i \in [d_1] \times \cdots \times [d_t]$, for every $a \in A$ and $b \in B$, we construct

$$G_i(a) = \begin{cases} \mathcal{T}_2 & \text{if } a_i = 0, \\ \mathcal{T}_3^1 & \text{if } a_i = 1, \end{cases} \quad \text{and} \quad H_i(b) = \begin{cases} \mathcal{T}_3^1 & \text{if } b_i = 0, \\ \mathcal{T}_2 & \text{if } b_i = 1. \end{cases}$$

It is easy to see that $\mathsf{RSTI}(G_i(a), H_i(b))$ iff $a_i \wedge b_i = 0$.

Now we combine bit gadgets recursively according to the $\wedge$ and $\vee$ operators in the Alternating Product. Starting from $k = t - 1$, there are two cases to consider.

AND Gadgets. The first case is when $\wedge$ operator is applied to $(k+1)$-th dimension, then by De Morgan's laws, we need to construct our gadgets such that for all $i_{(k)} \in [d_1] \times \cdots \times [d_k]$

$$\mathsf{RSTI}(G_{i_{(k)}}(a), H_{i_{(k)}}(b)) = \mathsf{RSTI}(G_{i_{(k)},0}(a), H_{i_{(k)},0}(b)) \vee \mathsf{RSTI}(G_{i_{(k)},1}(a), H_{i_{(k)},1}(b)).$$

To do so, for $a \in A$, we construct $G_{i_{(k)}}(a)$ to be

$$G_{i_{(k)}}(a) = \mathcal{P}^1(G_{i_{(k)},0}(a)) = \mathcal{P}^1(G_{i_{(k)},1}(a)),$$

which is well-defined since a is $\wedge$-invariant. And we construct $H_{i_{(k)}}(b)$ to be

$$H_{i_{(k)}}(b) = \left(H_{i_{(k)},0}(b) \circ H_{i_{(k)},1}(b)\right).$$

In any subtree isomorphism, it is easy to see that $G_{i_{(k)},0}(a)$ (or $G_{i_{(k)},1}(a)$) can only be mapped to either $H_{i_{(k)},0}(b)$ or $H_{i_{(k)},1}(b)$, so $G_{i_{(k)}}(a), H_{i_{(k)}}(b)$ implement an $\wedge$ operator.

OR Gadgets. The second case is when $\vee$ operator is applied to $(k+1)$-th dimension, then by De Morgan's laws, we need to construct our gadgets such that for all $i_{(k)} \in [d_1] \times \cdots \times [d_k]$,

$$\mathsf{RSTI}(G_{i_{(k)}}(a), H_{i_{(k)}}(b)) = \mathsf{RSTI}(G_{i_{(k)},0}(a), H_{i_{(k)},0}(b)) \wedge \mathsf{RSTI}(G_{i_{(k)},1}(a), H_{i_{(k)},1}(b)).$$

First for any tree T, we define two auxiliary trees $\mathcal{U}_0(T), \mathcal{U}_1(T)$ to ease our construction:

$$\mathcal{U}_0(T) = \left(\mathcal{P}^3(T) \circ \mathcal{T}_3^0\right) \quad \text{and} \quad \mathcal{U}_1(T) = \left(\mathcal{P}^3(T) \circ \mathcal{T}_3^1\right).$$

It is easy to verify that for any two trees T_1, T_2,

$$\mathsf{RSTI}(\mathcal{U}_0(T_1), \mathcal{U}_1(T_2)) = \mathsf{RSTI}(\mathcal{U}_1(T_1), \mathcal{U}_0(T_2)) = 0,$$

and

$$\mathsf{RSTI}(\mathcal{U}_0(T_1), \mathcal{U}_0(T_2)) = \mathsf{RSTI}(\mathcal{U}_1(T_1), \mathcal{U}_1(T_2)) = \mathsf{RSTI}(T_1, T_2).$$

We construct the corresponding tensor gadgets $G_{i_{(k)}}(a)$ and $H_{i_{(k)}}(b)$ for $a \in A$ and $b \in B$ as follows:

$$G_{i_{(k)}}(a) = \Big(\mathcal{U}_0(G_{i_{(k)},0}(a)) \circ \mathcal{U}_1(G_{i_{(k)},1}(a))\Big),$$

and

$$H_{i_{(k)}}(b) = \Big(\mathcal{U}_0(H_{i_{(k)},0}(b)) \circ \mathcal{U}_1(H_{i_{(k)},1}(b))\Big).$$

In any subtree isomorphism, it is easy to see that $\mathcal{U}_0(G_{i_{(k)},0}(a))$ can only be mapped to $\mathcal{U}_0(H_{i_{(k)},0}(b))$, and $\mathcal{U}_1(G_{i_{(k)},1}(a))$ can only be mapped to $\mathcal{U}_1(H_{i_{(k)},1}(b))$, so $G_{i_{(k)}}(a), H_{i_{(k)}}(b)$ implement an $\vee$ operator.

Correctness. It is not hard to verify that $\mathsf{RSTI}(G(a), H(b)) = \overline{p_{\text{alt}}(a, b)}$ by De Morgan's laws. To show $\mathsf{RSTI}(G(a), H(b)) = \mathsf{STI}(G(a), H(b))$, we focus on the case that $t > 0$ since the case that $t = 0$ is obvious. Let the root of G_A be r and the height of G_A be h. The outermost operator in an Alternating Product is $\wedge$, so r has only one child which has two subtrees of equal height $h - 2$. It is easy to see that the height of H_B is also h. Suppose that r is mapped to a node r' in H_B and c is mapped to c'. If we regard c' as the root of H_B, then after deleting c', H_B should be split into two subtrees of height $\geq h - 2$ and a single node r'. The only possible case is that c' is of depth 1 w.r.t. the original root of H_B (the depth of a root is 0) and r' is the original root of H_B. □

Theorem 5.8.2. *Given two bounded-degree unrooted trees T_A and T_B, it can be decided in $\mathsf{NSPACE}[(\log n)^2]$ that whether T_A is isomorphic to a subtree of T_B.*

Proof. This algorithm works by divide and conquer on trees. At each recursion, we have two trees S_A and S_B (implicit representation) as well as a set of node pairs $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$ (initially, $S_A = T_A, S_B = T_B$ and $M = \varnothing$). We need to decide whether there is an isomorphism from S_A to some subtree of S_B satisfying a_i in S_A is mapped to b_i in S_B for all $1 \leq i \leq k$.

First we find a centroid c of S_A, i.e., a node of S_A that decomposes S_A into subtrees of size at most $\lceil |S_A|/2 \rceil$ when the node is deleted. Then we nondeterministically guess a node c' in S_B to be the node that mapped by c in the isomorphism. If $c = a_i$ for some i but $c' \neq b_i$, then we reject; otherwise, we guess an injective mapping from the neighbors of c in S_A to the neighbors of c' in S_B.

For each neighbor v of c, let v' be the neighbor of c' mapped by v, S_A^v be the subtree of S_A containing v when the edge between v and c is deleted, $S_B^{v'}$ be the subtree of S_B containing v' when the edge between v' and c' is deleted. We create a new set of node pairs $M' = \{(a_i, b_i) \in M \mid a_i \in S_A^v\}$. If $b_i \notin S_B^{v'}$ for some pair $(a_i, b_i) \in M'$, then we reject; otherwise, we recursively checking if there is an isomorphism from S_A^v to some subtree of $S_B^{v'}$ satisfying a_i is mapped to b_i for all $(a_i, b_i) \in M'$ and v is mapped to v'.

This algorithm terminates when S_A is a single node. There are at most $O(\log n)$ levels of recursion by the property of centroid. At each level, we use only $O(\log n)$ space for c, c' and their neighbors (note that S_A, S_B are bounded-degree trees), and S_A, S_B can always be accessed according to the information stored at the upper levels of recursion. Thus this algorithm runs in $\mathsf{NSPACE}[(\log n)^2]$. □

Combining the above reductions together, we can show the equivalence between BP-Satisfying-Pair and Subtree-Isomorphism-Pair.

Theorem 5.8.3. *BP-Satisfying-Pair on branching program of size $2^{(\log N)^{o(1)}}$ and Subtree-Isomorphism-Pair on (rooted or unrooted) trees of size $2^{(\log N)^{o(1)}}$ are equivalent under near-linear time reductions.*

We can also show a reduction from OAPT to Subtree Isomorphism on two trees using the same gadgets in Theorem 5.8.1.

Theorem 5.8.4. *There exists an $O(N \operatorname{poly}(D))$-time reduction from a Restricted OAPT instance with two sets A, B of N tensors of size D to an instance of Subtree Isomorphism on two (rooted or unrooted) binary trees of size $O(N \operatorname{poly}(D))$ and depth $2 \log N + O(\log D)$.*

Proof. Using the recursive construction in Theorem 5.8.1 we can obtain tensor gadgets $G(a), H(b)$ for each $a \in A$ and $b \in B$, such that $\overline{p_{\mathrm{alt}}(a, b)} = \mathsf{RSTI}(G(a), H(b)) = \mathsf{STI}(G(a), H(b))$.

We can assume the set size N is a power of 2 by adding dummy vectors into each set. Now we combine the tensor gadgets in each set respectively to construct two trees G_A, H_B as our instance for Subtree Isomorphism:

a) To construct G_A for set A:

- Initialize G_A by a complete binary tree of N leaves;
- Associate each leaf with a tensor $a \in A$;
- For all $a \in A$, construct $\mathcal{P}^{\log N}(G(a))$ and link an edge from its root to the corresponding leaf of a.

b) To construct H_B for set B:

- Initialize H_B by a complete binary tree of N leaves;
- Select one leaf node v_ℓ;
- For every unselected leaf, construct $\mathcal{P}^{\log N}(H(\mathbf{0}))$ and link an edge from its root to the leaf.
- Construct a complete binary tree of N leaves rooted at v_ℓ;
- Associate each leaf of the tree rooted at v_ℓ with a tensor $b \in B$;
- For all $b \in B$, construct $\mathcal{P}^{\log N}(H(b))$ and link an edge from its root to the corresponding leaf of b.

Correctness For any subtree isomorphism, one $G(a)$ can be mapped to any $H(b)$ or $H(\mathbf{0})$. Since there are only $N-1$ gadgets of $H(\mathbf{0})$, there must be some $G(a)$ mapped to some $H(b)$. Thus $\mathsf{RSTI}(G_A, H_B)$ iff there exists a pair of $(a,b) \in A \times B$ with $p_{\text{alt}}(a,b) = 0$. It is not hard to see that the root of G_A can only be mapped to the root of H_B by arguing about the tree height (similar as Theorem 5.8.1), so $\mathsf{RSTI}(G_A, H_B) = \mathsf{STI}(G_A, H_B)$. □

5.8.2 Largest Common Subtree

In this subsection, first we prove that under near-linear time reductions between BP-Satisfying-Pair and (exact or approximate) Max-LCST-Pair/Min-LCST-Pair are equivalent, then we show the hardness for Largest Common Subtree on two trees.

For two trees a, b, define $\mathsf{LCST}(a,b)$ to be the size of the largest common subtree of a and b when a, b are seen as unrooted trees. Also, we define $\mathsf{RLCST}(a,b)$ to be the size of the largest common subtree of a and b when a, b are seen as rooted trees.

Now we establish a connection between Restricted Max-TropSim and Max-LCST-Pair:

Theorem 5.8.5. *Let t be an even number and $d_1 = \cdots = d_t = 2$. Given two sets of N tensors A, B in $\{0,1\}^{d_1 \times \cdots \times d_t}$ satisfying that all the tensors in A are* max-*invariant,* *for any $L \geq 2^t$, there is a deterministic algorithm running in $O(N \cdot \text{poly}(2^t) \cdot L)$ time which outputs two sets A', B' of N binary trees of size $O(\text{poly}(2^t) \cdot L)$ and depth $O((2^{t/2} + \log L) \cdot t)$, such that each $a \in A$ corresponds to a tree $a' \in A'$, each $b \in B$ corresponds to a tree $b \in B'$, and*

$$\mathit{RLCST}(a', b') = (2^{t/2} s(a, b) + O(1))L$$

$$\mathit{LCST}(a', b') = (2^{t/2} s(a, b) + O(1))L$$

where $s(a, b)$ is the Tropical Similarity score of a and b. In particular, if $s(a, b) = 1$, then a', b' satisfy $\mathit{RLCST}(a', b') = \mathit{LCST}(a', b') = |a'|$.

Proof. For each k and each prefix of index $i_{(k)} \in [d_1] \times \cdots \times [d_k]$, we construct corresponding gadget $a' = G_{i_{(k)}}(a)$ and $b' = H_{i_{(k)}}(b)$ for each $a \in A$ and $b \in B$ (when $k = 0$, $i_{(k)}$ can only be the empty prefix, and we simply use $G(a)$ and $H(b)$ for convenience) inductively, mimicking the evaluation of the Tropical Similarity. For this purpose, we need to construct the following three types of gadgets.

Bit Gadgets. For each coordinate $i \in [d_1] \times \cdots \times [d_t]$, let $\mathcal{C}_i$ be the tree constructed by join a path of length $2^{t/2}$ and a complete binary tree of L nodes: one end of the path is regarded as the root, and we link an edge between the node of depth $\text{bin}_{\text{odd}}(i)$ and the root of the complete binary tree, where $\text{bin}_{\text{odd}}(i) \in [0, 2^{t/2})$ is the number whose binary representation is $i_1 i_3 \cdots i_{t-1}$.

For every $a \in A$ and for each coordinate $i \in [d_1] \times \cdots \times [d_t]$, we construct $G_i(a) = \mathcal{C}_i$ if $a_i = 1$, or simply a path of length $2^{t/2}$ if $a_i = 0$. Similarly, for every $b \in B$ and for each coordinate $i \in [d_1] \times \cdots \times [d_t]$, we construct $H_i(b) = \mathcal{C}_i$ if $b_i = 1$, or simply a path of length $2^{t/2}$ if $a_i = 0$.

If $a_i \cdot b_i = 0$, then $\mathsf{RLCST}(G_i(a), H_i(b)) = 2^{t/2}$; otherwise $\mathsf{RLCST}(G_i(a), H_i(b)) = 2^{t/2} + L$. Furthermore, for any two coordinates i, j with $\text{bin}_{\text{odd}}(i) \neq \text{bin}_{\text{odd}}(j)$, $\mathsf{RLCST}(G_i(a), H_j(b)) = 2^{t/2}$.

Let $K = 2^{t/2} + \lceil \log L \rceil + 1$ be the maximum possible height of a bit gadget. Now we combine bit gadgets recursively according to the E and max operators in the evaluation of Tropical Similarity score. Starting from $k = t - 1$, there are two cases to consider.

Expectation Gadgets. The first case is when E operator is applied to $(k+1)$-th dimension. For $i_{(k)} \in [d_1] \times \cdots \times [d_k]$ and $a \in A$, we construct $G_{i_{(k)}}(a), H_{i_{(k)}}(b)$ as follows:

$$G_{i_{(k)}}(a) = \left(\mathcal{P}^{K-1}(G_{i_{(k)},0}(a)) \circ \mathcal{P}^{K-1}(G_{i_{(k)},1}(a))\right)$$

and

$$H_{i_{(k)}}(a) = \left(\mathcal{P}^{K-1}(H_{i_{(k)},0}(b)) \circ \mathcal{P}^{K-1}(H_{i_{(k)},1}(b))\right)$$

Max Gadgets. The second case is when max operator is applied to $(k+1)$-th dimension. For $i_{(k)} \in [d_1] \times \cdots \times [d_k]$ and $a \in A$, we construct $G_{i_{(k)}}(a), H_{i_{(k)}}(b)$ as follows:

$$G_{i_{(k)}}(a) = \mathcal{P}^{K}(G_{i_{(k)},0}(a)) = \mathcal{P}^{K}(G_{i_{(k)},1}(a))$$

and

$$H_{i_{(k)}}(a) = \left(\mathcal{P}^{K-1}(H_{i_{(k)},0}(b)) \circ \mathcal{P}^{K-1}(H_{i_{(k)},1}(b))\right).$$

Note that $G_{i_{(k)}}(a)$ is well-defined since a is max-invariant.

Finally we obtain tensor gadgets $G(a), H(b)$ for every $a \in A, b \in B$. It is easy to see that the depth of trees is $O(tK) \leq O((2^{t/2} + \log L) \cdot t)$, and the size of trees is $O(L \cdot 2^t + K \cdot 2^t) = O(\mathrm{poly}(2^t) \cdot L)$.

Correctness for RLCST. First we show that $\mathsf{RLCST}(G(a), H(b)) = (2^{t/2}s(a,b) + O(1))L$. We fix two tensors a, b. For every dimension k, let U_k be a set of gadget pairs:

$$U_k = \{(G_{i_{(k)}}(a), H_{j_{(k)}}(b)) \mid i_{(k)}, j_{(k)} \in [d_1] \times \cdots \times [d_k], i_p \neq j_p \text{ for some odd } p\}.$$

Let ε_k be the maximum RLCST among the pairs in U_k.

For every $i_{(k)} \in [d_1] \times \cdots \times [d_t]$, let $f(i_{(k)}) = \mathsf{RLCST}(G_{i_{(k)}}(a), H_{i_{(k)}}(b))$. For the last dimension t, it is easy to see that $f(i) = 2^{t/2} + (a_i \cdot b_i) \cdot L$ and $\varepsilon_t = 2^{t/2}$. Now we prove by induction that $f(i_{(k)}) \geq \varepsilon_k$ holds for every $0 \leq k \leq t$ and $i_{(k)} \in [d_1] \times \cdots \times [d_k]$.

On the one hand, if an E operator is applied to the $(k+1)$-th dimension, by induction hypothesis we have $f(i_{(k)}, 0) + f(i_{(k)}, 1) \geq 2\varepsilon_{k+1}$, so $f(i_{(k)}) = f(i_{(k)}, 0) + f(i_{(k)}, 1) + 2(K-1) + 1$. Note that $\varepsilon_k \leq 2\varepsilon_{k+1} + 2(K-1) + 1$, so $f(i_{(k)}) \geq \varepsilon_k$ holds. On the other hand, If the max operator is applied to the $(k+1)$-th dimension, then we have $f(i_{(k)}) = \max\{f(i_{(k)}, 0), f(i_{(k)}, 1)\} + K$ and $\varepsilon_k \leq \varepsilon_{k-1} + K$, so $f(i_{(k)}) \geq \varepsilon_k$ holds as well.

Expanding the above recurrence relation of $f(i_{(k)})$, we have

$$\begin{aligned}\mathsf{RLCST}(G(a), H(b)) &= 2^{t/2}s(a,b)L + O(K) \cdot 2^{t/2} \\ &= 2^{t/2}s(a,b)L + O(2^{t/2} + \log L) \cdot 2^{t/2} \\ &= (2^{t/2}s(a,b) + O(1))L.\end{aligned}$$

Correctness for LCST. Now we show that $\mathsf{LCST}(G(a), H(b)) = \mathsf{RLCST}(G(a), H(b)) + O(L)$. Fix a pair of $(a,b) \in A \times B$. If a node of $G(a)$ or $H(b)$ is in a bit gadget, then we call it *bit node*. If a node of $G(a)$ or $H(b)$ is not in any bit gadget, then we call it *operator node.*

Let $I_G(a)$ and $I_H(b)$ be the largest isomorphic subtrees in $G(a)$ and $H(b)$. Let r_a be the root of $I_G(a)$, i.e., the lowest node when the tree is directed with respect to the root of $G(a)$, and let r_b be the root of $I_H(b)$. Let r'_b be the node in $G(a)$ that is mapped to r_b, and r'_a be the node in $H(b)$ that is mapped from r_a.

If $r_a = r'_b$, then let $q = 1$ and $u_1 = r_a, u'_1 = r_b$. Otherwise, let $u_1, \ldots, u_q$ be the list of nodes that are in $I_G(a)$ and are adjacent to some node on the path from r_a to r'_b. Assume $u_1, \ldots, u_q$ is in depth-increasing order (it is easy to see that no two such nodes are of same depth). Let $u'_1, \ldots, u'_q$ be the nodes in $I_H(b)$ that are mapped by $u_1, \ldots, u_q$, respectively. Each node in $u'_1, \ldots, u'_q$ should be adjacent to some node on the path from r'_a to r_b in $I_H(b)$.

For a node u_i, we denote the whole subtree of u_i in $G(a)$ as T_{u_i}, and we define $T'_{u'_i}$ similarly. We can decompose the subtree $I_G(a)$ into two parts: the first part is the path from r_a to r'_b, and the second part is the q rooted subtrees $\widetilde{T}_{u_1} = T_{u_1} \cap I_G(a), \ldots, \widetilde{T}_{u_q} =$

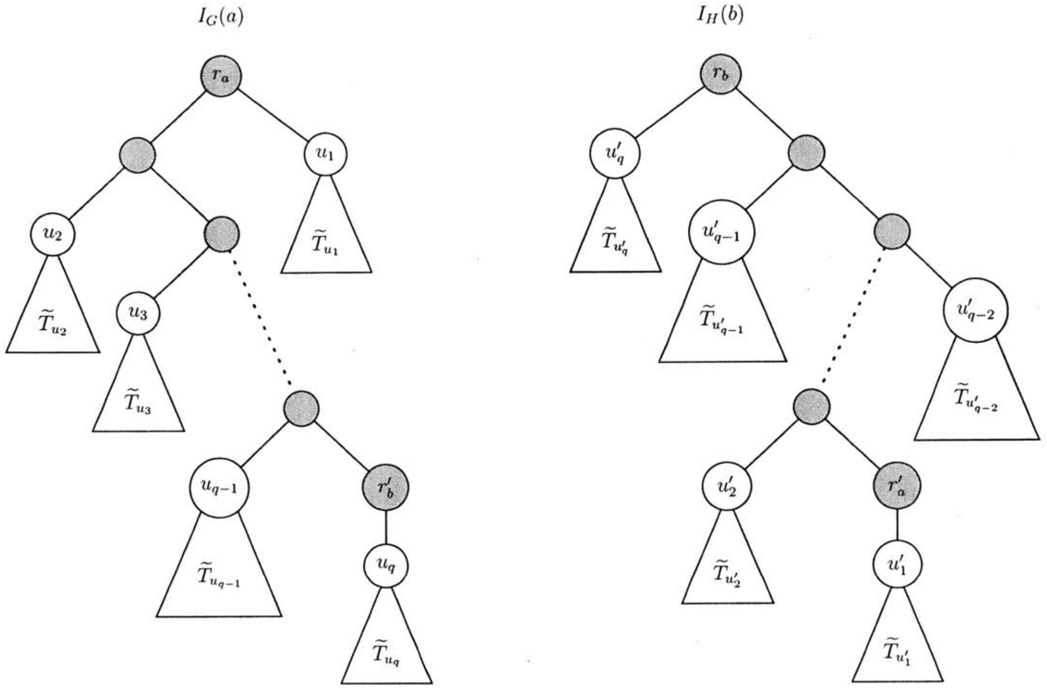

Figure 5-2: An illustration of $I_G(a)$ and $I_H(b)$.

$T_{u_q} \cap I_G(a)$. Similarly, we can decompose $I_H(b)$ into the path from r_b to r'_a and q rooted subtrees $\widetilde{T}_{u'_1} = T_{u'_1} \cap I_H(b), \ldots, \widetilde{T}_{u'_q} = T_{u'_q} \cap I_H(b)$. Thus we have

$$\mathsf{LCST}(G(a), H(b)) = q + \sum_{i=1}^{q} \left|\widetilde{T}_{u_i}\right| \leq O(tK) + \sum_{i=1}^{q} \mathsf{RLCST}(T_{u_i}, T'_{u'_i})$$

It is sufficient to obtain a bound for the sum of RLCST of $T_{u_i}, T'_{u'_i}$. If some u_j is a bit node, then u_i is also a bit node for all $i > j$, and all of them are in the same bit gadget, so $\sum_{i=j}^{q} \mathsf{RLCST}(T_{u_i}, T'_{u'_i}) \leq O(2^{t/2} + L)$. Similarly, some u'_j is a bit node, then u'_i is also a bit node for all $i < j$, and all of them are in the same bit gadget, so $\sum_{i=1}^{j} \mathsf{RLCST}(T_{u_i}, T'_{u'_i}) \leq O(2^{t/2} + L)$.

Now we consider the following three cases when both u_i and u'_i are operator nodes: (depth(u_i) stands for the depth of u_i in $G(a)$, depth(u'_i) stands for the depth of u'_i in $H(b)$)

Case 1. depth$(u_i) \not\equiv$ depth(u'_i) (mod K), then it is impossible to map some operator node with two children in T_{u_i} to an operator node with two children in $T_{u'_i}$.

If $\text{depth}(u_i) > \text{depth}(u'_i)$, then at most one bit gadget in T_{u_i} can have nodes being mapped to $T_{u'_i}$, and the case that $\text{depth}(u_i) < \text{depth}(u'_i)$ is similar. Thus $\mathsf{RLCST}(T_{u_i}, T'_{u'_i}) \leq O(tK) + (2^{t/2} + L) = O(tK + L)$.

Note that the depth of parent nodes of u_i and u'_i should also be different modulo K, so either u_i has no parent in $I_G(a)$ (this is the case when $r_a = r'_b$) or the parent of u_i has only one child in $I_G(a)$, and either case implies $i = q$.

Case 2. If $\text{depth}(u_i) \equiv \text{depth}(u'_i) \pmod K$ but $\text{depth}(u_i) > \text{depth}(u'_i)$, then since K is an upper bound of the height of any bit gadget, all the nodes in T_{u_i} can only be mapped to the operator nodes in $T_{u'_i}$, which implies $\mathsf{RLCST}(T_{u_i}, T'_{u'_i})$ is no more than the number of operator nodes in $T'_{u'_i}$. The case that $\text{depth}(u_i) \equiv \text{depth}(u'_i) \pmod K$ but $\text{depth}(u_i) < \text{depth}(u'_i)$ is similar.

All the trees $T'_{u'_i}$ are disjoint. Thus the sum $\sum_i \mathsf{RLCST}(T_{u_i}, T'_{u'_i})$ over all i in this case can be upper-bounded by the total number of operator nodes in $G(a)$, which is $O(K \cdot 2^{t/2})$.

Case 3. If $\text{depth}(u_i) = \text{depth}(u'_i)$, then there exists two prefixes of index $i_{(k)}, j_{(k)}$, such that

$$\mathsf{RLCST}(T_{u_i}, T_{u'_i}) = \mathsf{RLCST}(G_{i_{(k)}}(a), H_{j_{(k)}}(b)) + O(K)$$

Note that $u_1, \ldots, u_q$ are in depth-increasing order, and $u'_q, \ldots, u'_1$ are in depth-decreasing order, so this case can only happen for at most one pair of nodes.

Summing up all the above cases, we have

$$\mathsf{LCST}(G(a), H(b)) \leq O(tK) + O(2^{t/2} + L) + O(tK + L) + O(K \cdot 2^{t/2}) + \mathsf{RLCST}(G_{i_{(k)}}(a), H_{j_{(k)}}(b))$$

for any $i_{(k)}, j_{(k)}$. Thus $\mathsf{LCST}(G(a), H(b)) \leq \mathsf{RLCST}(G(a), H(b)) + O(L)$. □

By Theorem 5.8.5 with $L = 2^t$, we can easily have the following reductions from $\varepsilon(D)$-$\mathsf{Gap\text{-}Max\text{-}TropSim}$ and $\varepsilon(D)$-$\mathsf{Gap\text{-}Min\text{-}TropSim}$ to approximation variants of Max-

LCST-Pair and Min-LCST-Pair. Here we focus on the case that $\varepsilon(D) = \Omega(D^{-1/2})$. It is reasonable since $\sqrt{D} \cdot s(a,b)$ is always an integer, and $o(D^{-1/2})$-Gap-Max-TropSim is essentially equivalent to Max-TropSim. This argument also holds for $o(D^{-1/2})$-Gap-Min-TropSim.

Theorem 5.8.6. *For any function $\varepsilon(D) = \Omega(D^{-1/2})$, there exists an $O(N \operatorname{poly}(D))$-time reduction from a Restricted $\varepsilon(D)$-Gap-Max-TropSim instance with two sets of N tensors of size D to an instance of the following approximation variant of Max-LCST-Pair: Given two sets A, B of N trees of size $\operatorname{poly}(D)$ and a set B of N strings of length D over a constant-size alphabet, distinguish between the following:*

- ***Completeness:*** *There exists a pair of a, b such that LCST$(a,b) = |a| = (1+o(1))D^{3/2}$;*
- ***Soundness:*** *For every pair a, b, LCST$(a,b) \le O(\varepsilon(D)D^{3/2})$.*

And this conclusion also holds for RLCST.

Remark 5.8.7. *Note that Theorem 5.8.6 also implies a reduction from BP-Satisfying-Pair to Subtree-Isomorphism-Pair, but the trees constructed by the reduction in Theorem 5.8.1 have a smaller size and a lower depth.*

Theorem 5.8.8. *For any function $\varepsilon(D) = \Omega(D^{-1/2})$, there exists an $O(N \operatorname{poly}(D))$-time reduction from a Restricted $\varepsilon(D)$-Gap-Min-TropSim instance with two sets of N tensors of size D to an instance of the following approximation variant of Min-LCST-Pair: Given two sets A, B of N trees of size $\operatorname{poly}(D)$ and a set B of N strings of length D over a constant-size alphabet, distinguish between the following:*

- ***Completeness:*** *There exists a pair of a, b such that LCST$(a,b) \le O(\varepsilon(D)D^{3/2})$;*
- ***Soundness:*** *For every pair a, b, LCST$(a,b) = |a| = (1+o(1))D^{3/2}$.*

And this conclusion also holds for RLCST.

By Theorem 5.4.2, we show reductions from Max-LCST-Pair (or Min-LCST-Pair) to BP-Satisfying-Pair via the following theorem:

Theorem 5.8.9. *Given two bounded-degree unrooted trees T_A and T_B and a number q, it can be decided in NSPACE$[(\log n)^2]$ that whether there is an isomorphism between a subtree of T_A and a subtree of T_B of size q.*

Proof. The algorithm in Theorem 5.8.2 suffices to fulfill the requirement if modified slightly. At each level of recursion, we have two trees S_A and S_B, a number q, and a set of node pairs $M = \{(a_1, b_1), \ldots, (a_k, b_k)\}$. We need to decide whether there is an isomorphism from a subtree of S_A to a subtree of S_B satisfying it is of size q and a_i in S_A is mapped to b_i in S_B for all $1 \leq i \leq k$.

First we find a centroid c of S_A, then we guess if there is a subtree of size q that contains c and is isomorphic to a subtree of S_B. If not, then we delete c to decompose S_A into subtrees, guess which subtree contains a subtree that is isomorphic to a subtree of S_B of size q, and runs our algorithm to check recursively; If it is, then follow the same routine as in Theorem 5.8.2: we guess a node c' in S_B to be the node that mapped by c in the isomorphism and a bijective mapping from some of the neighbors of c in S_A to some of the neighbors of c' in S_B. Additionally, we guess a number q_v for each neighbor v of c and ensure the sum of q_v over all neighbors equals to $q - 1$. We then check recursively if there is an isomorphism from a subtree of S_A^v to a subtree of $S_B^{v'}$ of size q subject to the constraint that some set of node pairs are matched. It is clear that this algorithm runs in NSPACE$[(\log n)^2]$. □

Theorem 5.8.10. *For the case that the maximum element size $D = 2^{(\log N)^{o(1)}}$, there are near-linear time reductions between all pairs of the following problems:*

- *BP-Satisfying-Pair;*
- *Max-LCST-Pair or Min-LCST-Pair on (rooted or unrooted) trees;*
- *$2^{(\log D)^{1-\Omega(1)}}$-approximate Max-LCST-Pair or Min-LCST-Pair on (rooted or unrooted) trees;*

Proof. By Theorem 5.8.6, 5.8.8 and 5.8.9, we can show cyclic reductions between these three problems in a similar way as in the proof of Theorem 5.6.5. □

We can also show a reduction from $\varepsilon(N)$-Gap-Max-TropSim to Largest Common Subtree on two large trees using the same gadgets in Theorem 5.8.5.

Theorem 5.8.11. *Let t be an even number and $d_1 = \cdots = d_t = 2$. Given two sets of N tensors A, B in $\{0,1\}^{d_1 \times \cdots \times d_t}$, for $L = \Theta(2^t \log^2 N)$, there is a deterministic algorithm running in $O(N \log^2 N 2^{O(t)})$ time which outputs two binary trees A', B' of size $O(N \log^2 N 2^{O(t)})$ and depth $O(\log^2 N \operatorname{poly}(t) 2^{t/2})$, such that*

$$\mathit{RLCST}(A', B') = (2^{t/2} s_{\max} + O(1))L$$
$$\mathit{LCST}(A', B') = (2^{t/2} s_{\max} + O(1))L$$

where $s_{\max}$ is the maximum Tropical Similarity among all pairs of $(a, b) \in A \times B$.

Proof. Using the recursive construction in Theorem 5.8.5, we can obtain tensor gadgets $G(a), H(b)$ for each $a \in A$ and $b \in B$. Let m be the smallest number such that $N \leq 2^m$. Now we combine the tensor gadgets in each set respectively to construct two trees G_A, H_B as our instance for Largest Common Subtree.

For any number K, we define K-zoomed complete binary tree $\mathcal{Z}_K$ of 2^m as follows: first we construct a complete binary tree of 2^m leaves, then we insert $K - 1$ internal nodes between every pair of adjacent nodes (so $\mathcal{Z}_K$ is of height $mK + 1$).

Let $K_D = O((2^{t/2} + \log L)t)$ be the maximum diameter of any tensor gadget ($G(a)$ or $H(b)$). Let $K_G = 2(m+1)K_D$.

We construct $G_A = \mathcal{P}^{mK_G+1}(T_A)$, where T_A is the following auxiliary tree:

- Initialize $T_A = \mathcal{Z}_{K_D}$, and arbitrarily select N leaves;

- For each selected leaf, associate it with a tensor $a \in A$;

- Construct $G(a)$ for every $a \in A$, and link an edge from its root to the corresponding leaf of a.

And the tree H_B for set B is constructed as follows:

- Initialize $H_B = \mathcal{Z}_{K_G}$ and arbitrarily select N leaves;

- For each selected leaf, associate it with a tensor $b \in B$;
- Construct $\mathcal{P}^{mK_D+1}(b)$ for every $b \in B$, and link an edge from its root to the corresponding leaf of b.

Proof for RLCST. Note that all the tensor gadgets are of same depth, and only one gadget $G(a)$ in G_A can be mapped to a gadget $H(b)$ in H_B.

For $L = \Theta(2^t \log^2 N)$, using the fact that $\mathsf{RLCST}(G(a), H(b)) = (2^{t/2}s(a,b) + O(1))L$, one can easily show that

$$\mathsf{RLCST}(G_A, H_B) = (2^{t/2}s_{\max} + O(1))L + O(mK_G + mK_D) = (2^{t/2}s_{\max} + O(1))L.$$

Proof for LCST. Now we show that $\mathsf{LCST}(G_A, H_B) \le \mathsf{LCST}(G(a), H(b)) + O(L)$ for some $(a,b) \in A \times B$.

If a node of G_A or H_B is in a tensor gadget, then we call it *tensor node*. If a node of $G(a)$ or $H(b)$ is not in any tensor gadget, then we call it *assembly node*. Let G'_A be the tree G_A with all tensor nodes removed, and we define H'_B respectively.

We consider the following three cases:

Case 1. If none of tensor node of G_A is in the LCST, then

$$\mathsf{LCST}(G_A, H_B) = \mathsf{LCST}(G'_A, H_B) \le \mathsf{LCST}(P_A, H_B) + \mathsf{LCST}(\mathcal{Z}_{K_D}, H_B),$$

where P_A is the path of length $mK_G + 1$ linked with the root of T_A. It is easy to see that $\mathsf{LCST}(P_A, H_B) \le mK_G + 1$. Note that every pair of two tensor node from different tensor gadgets has distance at least $2K_G$, which is greater than the diameter of T_A, so the isomorphic subtree of T_A in H_B cannot contain nodes from more than one tensor gadgets. By noticing that $\mathsf{LCST}(\mathcal{Z}_{K_D}, H(b)) = O(K_D)$ for all $b \in B$ and $\mathsf{LCST}(\mathcal{Z}_{K_D}, \mathcal{Z}_{K_G}) = O(K_G)$, we have $\mathsf{LCST}(\mathcal{Z}_{K_D}, H_B) = O(K_G)$. Thus $\mathsf{LCST}(G_A, H_B) = mK_G + O(K_G) = O(mK_G)$.

Case 2. If LCST contains some tensor nodes of G_A, and all such tensor nodes are mapped to assembly nodes of H_B, then $\mathsf{LCST}(G_A, H_B)$ is no more than $\mathsf{LCST}(G'_A, H_B)$ plus the number of tensor nodes in the LCST. By Case 1, $\mathsf{LCST}(G'_A, H_B) = O(mK_G)$. Note that every two tensor nodes in G_A has distance at most K_G and any set of nodes of diameter $O(K_G)$ in H_B can only have size $O(K_G)$, so the number of tensor nodes in LCST is at most $O(K_G)$. Thus $\mathsf{LCST}(G_A, H_B) = O(mK_G) + O(K_G) = O(mK_G)$.

Case 3. If some tensor node in G_A is mapped to a tensor node in H_B, then all the tensor nodes of G_A in the LCST are in the same tensor gadget, and it also holds for G_B. This can be shown as follows: Let u_1, u_2 be two tensor node in G_A that are mapped to tensor nodes u'_1, u'_2 in H_B, then

- u_1, u_2 are in the same tensor gadget. This is because that the minimum distance between two tensor nodes from different tensor gadgets in G_A is at least $2K_D$ and at most K_G, but the distance between any two tensor nodes in H_B is either $\leq K_D$ or $\geq 2K_G$.
- u'_1, u'_2 are in the same tensor gadget. This is because that the minimum distance between two tensor nodes from different tensor gadgets in H_B is at least $2K_G$, but the distance between any two tensor nodes in G_A is at most K_G.

Let $G(a)$ be the unique tensor gadget in G_A that has nodes in the LCST, and let $H(b)$ be the unique tensor gadget in H_B that has nodes in the LCST. By Case 1, $\mathsf{LCST}(G'_A, H_B) = O(mK_G)$. Thus we have

$$\begin{aligned}\mathsf{LCST}(G_A, H_B) &\leq \mathsf{LCST}(G_A \setminus G(a), H_B) + \mathsf{LCST}(G(a), H(b))\\ &= O(mK_G) + \mathsf{LCST}(G(a), H(b)).\end{aligned}$$

In any case, we can show that $\mathsf{LCST}(G_A, H_B) \leq \mathsf{LCST}(G(a), H(b)) + O(L)$ for some $(a, b) \in A \times B$, which completes the proof. □

Theorem 5.8.12. *There exists an $O(N \operatorname{poly}(D))$-time reduction from an $\varepsilon(N)$-Gap-Max-TropSim instance with two sets of N tensors of size D to an instance of $o(\varepsilon(N)^{-1})$-approximate Largest Common Subtrees on two (rooted or unrooted) binary trees of size $O(N \operatorname{poly}(D, \varepsilon(N)^{-1}))$ and depth $O(\log^2 N \operatorname{poly}(D, \varepsilon(N)^{-1}))$.*

Proof. First we add dummy dimensions to each tensor such that the new size C of every tensor is at least $\Omega(\varepsilon^{-2}(N))$. We construct the two trees A', B' as in Theorem 5.8.11. By setting $L = C \log^2 N$, we have

$$\mathsf{LCST}(A', B') = (\sqrt{C} \cdot s_{\max} + O(1)) \cdot C \log^2 N.$$

Thus it reduces to distinguish $\mathsf{LCST}(A', B')$ from being $\geq C^{3/2} \log^2 N$ and $\leq O(\varepsilon(N)$ $C^{3/2} \log^2 N)$, which can be solved by an $o(\varepsilon(N)^{-1})$-approximation algorithm for LCST. This conclusion also holds for RLCST. □

5.9 Equivalence in the Data Structure Setting

In this section, we establish the equivalence between BP-Pair-Class problems in the data structure setting.

Theorem 5.9.1. *For the following data structure problems, if any of the following problems admits an algorithm with preprocessing time $T(N)$, space $S(N)$ and query time $Q(N)$, then all other problems admits a similar algorithm with preprocessing time $T(N) \cdot N^{o(1)}$, space $S(N) \cdot N^{o(1)}$ and query time $Q(N) \cdot N^{o(1)}$.*

- *$\mathsf{NNS}_{\mathsf{LCS}}$: Preprocess a database $\mathcal{D}$ of N strings of length $D = 2^{(\log N)^{o(1)}}$, and then for each query string x, find $y \in \mathcal{D}$ maximizing $\mathsf{LCS}(x, y)$.*
- *Approx. $\mathsf{NNS}_{\mathsf{LCS}}$: Find $y \in \mathcal{D}$ s.t. $\mathsf{LCS}(x, y)$ is a $2^{(\log D)^{1-\Omega(1)}}$ approximation to the maximum value.*
- *Regular Expression Query: Preprocess a database $\mathcal{D}$ of N strings of length $D = 2^{(\log N)^{o(1)}}$, and then for each query regular expression y, find an $x \in \mathcal{D}$ matching y.*

- *Approximate Regular Expression Query: for a query expression y, distinguish between: (1) there is an $x \in \mathcal{D}$ matching y; and (2) for all $x \in \mathcal{D}$, the hamming distance between x and all $z \in L(y)$ is at least $(1 - o(1)) \cdot D$.*

Proof. We show a reduction from $\mathsf{NNS}_{\mathsf{LCS}}$ to Approx. $\mathsf{NNS}_{\mathsf{LCS}}$ for illustration, and the proofs for the other pairs of problems are essentially the same.

By Lemma 5.6.4, there is a BP of poly-logarithmic size that accepts (a, b, k) iff $\mathsf{LCS}(a, b) \geq k$. Then using a similar argument as in Theorem 5.6.5, we can show a reduction from an instance $\phi = (A, B)$ of the following problem to an Approx. Closest-LCS-Pair instance $\phi' = (A', B')$: given a set of strings A and a set of string-integer pairs, determine whether there are $a \in A$ and $(b, k) \in B$ such that $\mathsf{LCS}(a, b) \geq k$. Moreover, this reduction maps each element separetely, i.e., there exist two maps f, g such that $A' = \{f(a) \mid a \in A\}, B' = \{g(b, k) \mid (b, k) \in B\}$, and both f and g are computable in $O(2^{\text{polylog}(D)}) = 2^{(\log N)^{o(1)}}$ time and space for each string of length D.

Now suppose there is a data structure for Approx. $\mathsf{NNS}_{\mathsf{LCS}}$ with preprocessing time $T(N)$, space $S(N)$ and query time $Q(N)$. For a set of string A, we construct a data structure for $\mathsf{NNS}_{\mathsf{LCS}}$ as follows. In the preprocessing stage, we map all the strings in A via f and store them in a data structure $\mathcal{D}$ for Approx. $\mathsf{NNS}_{\mathsf{LCS}}$. For each query string, we do a binary search for the maximum LCS. For every length k encountered, we first map the query string and the length k via g and then query it in the data structure $\mathcal{D}$. The time cost and space usage of the new data structure can be easily analyzed. □

A direct generalization of the above proof is that $\mathsf{NNS}_{\mathsf{LCS}}$ is actually the hardest NNS problem among all distance that can be computed in small space.

Corollary 5.9.2. *For every distance function dist that can be computed in poly-logarithmic space, the exact NNS problem with respect to dist ($\mathsf{NNS}_{\mathsf{dist}}$) can be reduced to $2^{(\log D)^{1-\Omega(1)}}$-approximate $\mathsf{NNS}_{\mathsf{LCS}}$ in near-linear time.*

Remark 5.9.3. *Here we assume that when the size parameter for $\mathsf{NNS}_{\mathsf{dist}}$ is N, the* *inputs to* dist *takes* $2^{(\log N)^{o(1)}}$ *bits to describe, and the output of* dist *takes* polylog(N) *bits to describe.*

Furthermore, basing on the hardness of solving BP-SAT, we can show that there may not be an efficient data structure for $\mathsf{NNS}_{\mathsf{LCS}}$ in the following sense:

Theorem 5.9.4. *Assuming the satisfiability for branching programs of size* $2^{n^{o(1)}}$ *cannot be decided in* $O(2^{(1-\delta)n})$ *for some* $\delta > 0$*, there is no data structure for Approx.* $\mathsf{NNS}_{\mathsf{LCS}}$ *(or other data structure problems listed in Theorem 5.9.1) with preprocessing time* $O(N^c)$ *and query time* $N^{1-\varepsilon}$ *for some* $c, \varepsilon > 0$*.*

Proof. Our proof closely follows the proof for Corollary 1.3 in [151]. We only prove that it is true for Approx. $\mathsf{NNS}_{\mathsf{LCS}}$, the case for other data structures are similar. Assume such a data structure for Approx. $\mathsf{NNS}_{\mathsf{LCS}}$ does exist. Now we show that there is an algorithm for approximate Closest-LCS-Pair that runs in $O(N^{2-\delta'})$ time for some $\delta' > 0$.

Let (A, B) be an instance of approximate Closest-LCS-Pair. Let $\gamma = 1/(2c)$. We partition the set A into $M = O(N^{1-\gamma})$ subsets $A_1, \ldots, A_M$, each of size $O(N^\gamma)$. Now for each subset A_i, we build a data structure for Approx. $\mathsf{NNS}_{\mathsf{LCS}}$, and query each $b \in B$ in the data structure. Finally, we take the maximum among all the query results. The total time for preprocessing is $O(M \cdot (N^\gamma)^c) = O(N^{3/2})$ and that for query is $O(N \cdot M \cdot (N^\gamma)^{1-\varepsilon}) = O(N^{2-\varepsilon\gamma})$. □

5.10 Faster BP-SAT Implies Circuit Lower Bounds

In [10], Abboud et al. showed that faster exact algorithms for Edit Distance or LCS imply faster BP-SAT, and it leads to circuit lower bound consequences that are far stronger than any state of art. Using a similar argument, strong circuit lower bounds can also be shown if any of BP-Pair-Class or BP-Pair-Hard problems has faster algorithms, even for shaving a quasipolylog factor.

We apply the following results from [10] to show the circuit lower bound consequences, which are direct corollaries from [169, 171]:

Theorem 5.10.1 ([169, 171]). *Let $n \leq S(n) \leq 2^{o(n)}$ be a time constructible and monotone non-decreasing function. Let $\mathcal{C}$ be a class of circuits. If the satisfiability of a function of the form*

- *AND of fan-in in $O(S(n))$ of*
- *arbitrary functions of fan-in 3 of*
- *$O(S(n))$-size circuits from $\mathcal{C}$*

can be decided in $\mathsf{DTIME}[O(2^n/n^{10})]$ time, then E^{NP} does not have $S(n)$-size $\mathcal{C}$-circuits.

Theorem 5.10.2 ([169]). *For the complexity class $\mathsf{NTIME}[2^{O(n)}]$, we have*

1. *If for every constant $k > 0$, there is a satisfiability algorithm for bounded fan-in formulas of size n^k running in $\mathsf{DTIME}[O(2^n/n^k)]$ time, then $\mathsf{NTIME}[2^{O(n)}]$ is not contained in non-uniform NC^1;*
2. *If for every constant $k > 0$, there is a satisfiability algorithm for NC-circuits of size n^k running in $\mathsf{DTIME}[O(2^n/n^k)]$ time, then $\mathsf{NTIME}[2^{O(n)}]$ is not contained in non-uniform NC.*

First we show the circuit lower bound consequences if truly-subquadratic algorithm exists:

Reminder of of Corollary 5.1.13 *If any of the BP-Pair-Class or BP-Pair-Hard problems admits an $N^{2-\varepsilon}$ time deterministic algorithm (or $(NM)^{1-\varepsilon}$ time algorithm for regular expression membership testing) for some $\varepsilon > 0$, then E^{NP} does not have:*

1. *non-uniform $2^{n^{o(1)}}$-size Boolean formulas,*
2. *non-uniform $n^{o(1)}$-depth circuits of bounded fan-in, and*
3. *non-uniform $2^{n^{o(1)}}$-size nondeterministic branching programs.*

Furthermore, $\mathsf{NTIME}[2^{O(n)}]$ is not in non-uniform NC.

Proof. A truly-subquadratic time algorithm for BP-Pair-Class or BP-Pair-Hard problems implies a $2^{(1-\Omega(1))n}$-time algorithm for BP-SAT on branching program of size $2^{n^{o(1)}}$. Let $S(n) = 2^{n^{o(1)}}$. $O(S(n))$-size Boolean formulas, $O(\log S(n))$-depth circuits, $2^{n^{o(1)}}$-size nondeterministic branching programs are all closed under AND, OR and NOT gates proscribed in Theorem 5.10.1. Note that any formula of size $2^{n^{o(1)}}$ can be transformed into an equivalent $n^{o(1)}$-depth circuit [159], and any $n^{o(1)}$-depth circuit can be transformed into $2^{n^{o(1)}}$-size branching program by Barrington's Theorem [46]. Then all the consequences in Item 1, 2, 3 follow from Theorem 5.10.1. Combining Item 2 and Theorem 5.10.2, we can obtain the consequence that NTIME$[2^{O(n)}]$ is not in non-uniform NC. □

We can also obtain results showing that even shaving a quasipolylog factor $2^{(\log\log N)^3}$ for problems in BP-Pair-Class and BP-Pair-Hard can imply new circuit lower bound. First, it is easy to see that shaving a $(\log N)^{\omega(1)}$ factor can lead to new circuit lower bound by Theorem 5.10.2.

Theorem 5.10.3. *If there is a deterministic algorithm for BP-Satisfying-Pair on BP of size $S = 2^{(\log N)^{o(1)}}$ running in $O(N^2 \operatorname{poly}(S)/(\log N)^{\omega(1)})$ time, then the following holds:*

1. *For any constant $k > 0$, SAT on bounded fan-in formula of size n^k can be solved in $O(2^n/n^{\omega(1)})$ deterministic time;*
2. *NTIME$[2^{O(n)}]$ is not contained in non-uniform NC^1.*

Proof. By Theorem 5.10.2, Item 1 implies Item 2, so we only need to show the conclusion in Item 1.

If BP-Satisfying-Pair can be solved in $O(N^2 \operatorname{poly}(S)/(\log N)^{\omega(1)})$ time, then BP-SAT on BP of size $O(\operatorname{poly}(n))$ can be solved in

$$O(2^n \operatorname{poly}(n)/n^{\omega(1)}) = O(2^n/n^{\omega(1)}).$$

Note that any formula of size n^k can be transformed into an equivalent BP of width $W = 5$ and length $T = O(n^{8k})$ (by rebalancing into a formula of depth $4k \log n$ [159]

and using Barrington's Theorem [46]). Thus SAT on bounded fan-in formulas of size n^k can also be solved in $O(2^n/n^{\omega(1)})$. □

A part of our reductions from BP-Satisfying-Pair to problems in BP-Pair-Class can be summerized below. In the rest of this section, for each problem in BP-Pair-Class (but except BP-Satisfying-Pair), we use the variable N to denote the number of elements in each set, and D to denote the maximum length (or size) of each element. We exclude BP-Satisfying-Pair here because the size S of BP is more important than D in BP-Satisfying-Pair.

Corollary 5.10.4. *For every problem $\mathcal{P}$ in BP-Pair-Class except BP-Satisfying-Pair:*

- *If $\mathcal{P}$ is a decision problem, then there is an $O(N \operatorname{poly}(D))$-time reduction from Restricted OAPT to $\mathcal{P}$;*
- *If $\mathcal{P}$ is an approximate problem, then for every $\varepsilon(D) = \Omega(D^{-1/2})$, there is an $O(N \operatorname{poly}(D))$-time reduction from Restricted $\varepsilon(D)$-Gap-Max-TropSim or Restricted $\varepsilon(D)$-Gap-Max-TropSim to $\mathcal{P}$ with approximation ratio $o(\varepsilon(D)^{-1})$, and each element has size $O(\operatorname{poly}(D))$.*

And any reduction here preserves the value of N.

Reminder of Theorem 5.1.10 *For $D = 2^{(\log N)^{o(1)}}$, if there is an*

$$O\left(N^2 \operatorname{poly}(D)/2^{(\log\log N)^3}\right) \text{ or } O\left(N^2/(\log N)^{\omega(1)}\right)$$

time deterministic algorithm for the decision, exact value, or $O(\operatorname{polylog}(D))$-approximation problems in BP-Pair-Class, then the same consequences in Theorem 5.10.3 follows.

Proof. By Corollary 5.10.4 and the fact that exact value problem can be trivially reduced to its approximation version, we only need to show that this statement is true for OAPT and $(\log D)^c$-Gap-Max-TropSim for every $c > 0$ (the proof for $(\log D)^c$-Gap-Min-TropSim should be similar).

Note that all our reductions here preserve the value of N. If there is an $O(N^2/(\log N)^{\omega(1)})$-time algorithm, then BP-Satisfying-Pair can also be solved in $O(N^2/(\log N)^{\omega(1)})$-time and the consequences in Theorem 5.10.3 follows.

Now consider the case that a $O(N^2 \operatorname{poly}(D)/2^{(\log\log N)^3})$-time algorithm exists. Recall that the hard instances of BP-Satisfying-Pair we constructed in the proof of Theorem 5.10.3 is on BP of width $W = O(1)$ and length $T = O(\operatorname{poly}(n)) = O(\operatorname{polylog}(N))$. By Theorem 5.5.1, we know that this instance can be near-linear time reduced to an OAPT instance with

$$D = 2^{O(\log W \log T)} = 2^{O(\log\log N)} = \operatorname{polylog}(N).$$

Thus shaving an $O(2^{(\log\log N)^3})$ factor to OAPT implies an $O(N^2/(\log N)^{\omega(1)})$-time algorithm for BP-Satisfying-Pair.

By Theorem 5.5.6, for $\varepsilon = \log^{-3c}(T)$, we know that a hard instance of BP-Satisfying-Pair can also be near-linear time reduced to an ε-Gap-Max-TropSim instance with (adding dummy dimensions if necessary)

$$D = 2^{\Theta(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))} = 2^{\Theta(\log^2 T \log\log T)}.$$

Then we have $(\log D)^c = o(\varepsilon^{-1})$, and thus shaving an $O(2^{(\log\log N)^3})$ factor to $(\log D)^c$-Gap-Max-TropSim implies an algorithm for the hard instances of BP-Satisfying-Pair running in the following time:

$$\begin{aligned} O(N^2 \operatorname{poly}(D)/2^{(\log\log N)^3}) &= O(N^2 \cdot 2^{\Theta(\log^2 T \log\log T)}/2^{(\log\log N)^3}) \\ &= O(N^2/(\log N)^{\omega(1)}). \end{aligned}$$

□

For BP-Pair-Hard problems, recall that part of our reduction can be summerized below:

Corollary 5.10.5. *For every problem $\mathcal{P}$ in BP-Pair-Hard:*

- *If $\mathcal{P}$ is a decision problem, then there is an $O(N \operatorname{poly}(D))$-time reduction from Restricted* OAPT *to $\mathcal{P}$ on input of length $O(N \operatorname{poly}(D))$;*

- *If $\mathcal{P}$ is an approximate problem, then for every $\varepsilon(N)$, there is an $O(N \operatorname{poly}(D, \varepsilon(N)^{-1}))$-time reduction from Restricted $\varepsilon(N)$-Gap-Max-TropSim or Restricted $\varepsilon(N)$-Gap-Max-TropSim to $\mathcal{P}$ with approximation ratio $o(\varepsilon(N)^{-1})$ on input of length $O(N \operatorname{poly}(D, \varepsilon(N)^{-1}))$.*

Then we can obtain the following result:

Reminder of Theorem 5.1.11 *If there is an deterministic algorithm for any decision, exact value or* polylog(N)*-approximation problems among BP-Pair-Hard problems listed in Theorem 5.1.8 running in running in*

$$O\left(N^2/2^{\omega(\log\log N)^3}\right)$$

time (or $O\left(NM/2^{\omega(\log\log(NM))^3}\right)$ time for Regular Expression Membership Testing), then the same consequences in Theorem 5.1.10 follows.

Proof. By Corollary 5.10.5 and the fact that exact value problem can be trivially reduced to its approximation version, we only need to show that this statement is true for OAPT and $(\log N)^c$-Gap-Max-TropSim for every $c > 0$.

The proof for OAPT is similar as in Theorem 5.1.10. For $(\log N)^c$-Gap-Max-TropSim, we know that the hard instances of BP-Satisfying-Pair in Theorem 5.10.3 can be reduced to a ε-Gap-Max-TropSim instance with

$$D = 2^{O(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))} = 2^{O(\log\log N)^3}$$

for $\varepsilon = (\log N)^c$. Thus shaving an $O(2^{\omega(\log\log N)^3})$ factor to $(\log N)^c$-Gap-Max-TropSim implies an algorithm for the hard instances of BP-Satisfying-Pair running in $O(N^2/(\log N)^{\omega(1)})$ time. □

5.11 Derandomization Implies Circuit Lower Bounds

For the some problems $\mathcal{A}$ like *Longest Common Subsequence*, despite its approximating for the pair version of $\mathcal{A}$ (Approximate Max-$\mathcal{A}$-Pair) is subquadratically equivalent to Max-TropSim, it is still hard to find a reduction from approximating $\mathcal{A}$. The main barrier is when trying to construct gadgets to reduce Approximate Max-$\mathcal{A}$-Pair to Approximate $\mathcal{A}$, the contribution to the final result for just one pair is too small to make a large approximating gap.

To overcome this barrier, we follows from [11] to define $\varepsilon(N)$-Super-Gap-Max-TS, which is a variant of $\varepsilon(N)$-Gap-Max-TropSim with a large fraction of pairs having perfect Tropical Similarities:

Definition 5.11.1 (ε-Super-Gap-Max-TS). Let t be an even number and $d_1 = d_2 = \cdots = d_t = 2$. Given two sets of tensors $A, B \in \{0,1\}^{d_1 \times \cdots \times d_t}$ of size $D = 2^t$, distinguish between the following:

- **Completeness:** A $(1 - 1/\log^{10} N)$-fraction of the pairs of $a \in A, b \in B$ have a perfect Tropical Similarity, $s(a, b) = 1$;
- **Soundness:** Every pair has low Tropical Similarity score, $s(a, b) < \varepsilon$.

where ε is a threshold that can depend on N and D.

In [11], Abboud and Rubinstein has shown that $o(1)$-Super-Gap-Max-TS can be reduced to $O(1)$-approximate LCS. Using the same reduction, we have the following corollary for arbitrary approximation ratio:

Theorem 5.11.2 ([11]). *Given an $\varepsilon(N)$-Super-Gap-Max-TS instance on N tensors of size D, we can construct two strings x, y of length ND in $O(N \operatorname{poly}(D))$ deterministic time such that:*

- *If $(1 - 1/\log^{10} N)$-fraction of the pairs have a perfect Tropical Similarity, then $\mathsf{LCS}(x, y) > (1/3)ND$;*
- *If every pair has low Tropical Similarity score, then $\mathsf{LCS}(x, y) < 2\varepsilon(N)ND$*

Thus, if there is an $\varepsilon(N)^{-1}$-approximation algorithm for such kind of (x, y) pairs, then there is a faster algorithm for $(\varepsilon(N)/6)$-Super-Gap-Max-TS.

Proof. We construct strings $G(a), H(b)$ as tensor gadgets for each tensor $a \in A, b \in B$ as in the reduction in [11] (stated in Theorem 5.6.1). Then we construct the final strings x, y by concatenating all the tensor gadgets. Using a similar argument as in [11], we can show that if there are at least $(1 - 1/\log^{10} N)N^2$ pairs of tensors with perfect Tropcial Similarities, then $\mathsf{LCS}(x, y) > (1 - 1/\log^{10} N) \cdot ND/2$; if every pair has low Tropical Similarity score, then $\mathsf{LCS}(x, y) < \varepsilon(N) \cdot 2ND$. □

There is no obvious reduction from BP-Satisfying-Pair to $\varepsilon(N)$-Super-Gap-Max-TS, and a randomized algorithm can even solve $\varepsilon(N)$-Super-Gap-Max-TS in nearly linear time. But finding a *deterministic* algorithm for $\varepsilon(N)$-Super-Gap-Max-TS is still hard: as noted by Abboud and Rubinstein in [11], a truly-subquadratic time deterministic algorithm for $\varepsilon(N)$-Super-Gap-Max-TS can imply some circuit lower bound for E^{NP}. Combining their ideas with the connection between Tropical Tensors and BP-SAT we established, we can show even stronger circuit lower bounds if such algorithm exists.

We base our proof on the following results in the literature:

Theorem 5.11.3 ([48]). *Let F_n be a set of function from $\{0, 1\}^n$ to $\{0, 1\}$ that are efficiently closed under projections. If the acceptance probability of a function of the form*

- *AND of fan-in in $n^{O(1)}$ of*

- *OR's of fan-in 3 of*

- *functions from $F_{n+O(\log n)}$*

can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in $\mathsf{DTIME}[2^n/n^{\omega(1)}]$, then there is a function $f \in \mathsf{E}^{\mathsf{NP}}$ on n variables and $f \notin F_n$.

Theorem 5.11.4 ([169, 48]). *If the acceptance probability of a function from NC^1 can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in $\mathsf{DTIME}[2^n/n^{\omega(1)}]$, then $\mathsf{NTIME}[2^{O(n)}]$ is not contained in NC^1.*

Theorem 5.11.5. *Let AC-BP-SAT be the following problem: given a branching program P of length T and width W on n inputs, distinguish the acceptance probability of P from being $= 1$ or $\leq 1/n^{10}$.*

There is a reduction from AC-BP-SAT to ε-Super-Gap-Max-TS on two sets of $N = 2^{n/2}$ tensors of size $D = 2^{O(\log^2 W \log^2 T(\log\log W + \log\log T + \log \varepsilon^{-1}))}$, and the reduction runs in $O(N \operatorname{poly}(D))$. Here ε is a threshold value that can depend on N (but cannot depend on D).

Reminder of Theorem 5.1.12 *The following holds for deterministic approximation to LCS:*

1. *A $2^{(\log N)^{1-\Omega(1)}}$-approximation algorithm in $N^{2-\delta}$ time for some constant $\delta > 0$ implies that E^{NP} has no $n^{o(1)}$-depth bounded fan-in circuits;*

2. *A $2^{o(\log N/(\log\log N)^2)}$-approximation algorithm in $N^{2-\delta}$ time for some constant $\delta > 0$ implies that NTIME$[2^{O(n)}]$ is not contained in non-uniform NC^1;*

3. *An $O(\operatorname{polylog}(N))$-approximation algorithm in $N^2/2^{\omega(\log\log N)^3}$ time implies that NTIME$[2^{O(n)}]$ is not contained in non-uniform NC^1.*

Proof. By Theorem 5.11.3 and Theorem 5.11.4, for Item 1, it is sufficient to show AC-BP-SAT on BP of length $2^{n^{o(1)}}$ and width $O(1)$ on n inputs can be solved in $2^{(1-\Omega(1))n}$ time; for Item 2 and 3, it is sufficient to show AC-BP-SAT on BP of length $O(\operatorname{poly}(n))$ and width $O(1)$ on n inputs can be solved in $2^n/n^{\omega(1)}$ time (the former scale of BP is able to simulate $n^{o(1)}$-depth circuit, while the later one is able to simulate NC^1 by Barrington's Theorem [46]).

Item 1. Assume there exists a $2^{(\log N)^{1-c}}$-approximation algorithm for LCS in $N^{2-\delta}$ time for some $c > 0$ and $\delta > 0$. By Theorem 5.11.5, AC-BP-SAT on BP of length $T = 2^{n^{o(1)}}$ and width $W = O(1)$ on n inputs can be reduced to $(2^{-(\log K)^{1-c}}/6)$-Super-Gap-Max-TS on $K = 2^{n/2}$ tensors of size

$$D = 2^{O(n^{o(1)} \cdot (o(\log n) + (\log K)^{1-c}))} = 2^{n^{1-c+o(1)}}.$$

Then by Theorem 5.11.2, $(2^{-(\log K)^{1-c}}/6)$-Super-Gap-Max-TS can be reduced to $2^{(\log K)^{1-c}}$-approximate LCS for strings of length

$$N = KD = 2^{n/2+n^{1-c+o(1)}} = 2^{(1/2+o(1))n}.$$

By our assumption, the last problem can be solved in $N^{2-\delta}$ time, so AC-BP-SAT on branching program of length $2^{n^{o(1)}}$ and width $O(1)$ on n inputs can be solved in $2^{(1-\delta/2+o(1))n}$ time. Applying Theorem 5.11.3 completes the proof.

Item 2. Assume there exists a $2^{f(\log N)}$-approximation algorithm for LCS in $N^{2-\delta}$ time for some constant $\delta > 0$ and some function $f(k) = o(k/\log^2 k)$. Let $g(k) = 2f(k) + \log k$. Then we have $g(\log N) = o(\log N/(\log\log N)^2)$ and

$$2^{f((1+o(1))\log K)} \leq 2^{(1+o(1))f(\log K)} \leq 2^{2f(\log K)} = o(2^{g(\log K)}).$$

By Theorem 5.11.5, AC-BP-SAT on BP of length $T = O(\text{poly}(n))$ and width $W = O(1)$ on n inputs can be reduced to $2^{-g(\log K)}$-Super-Gap-Max-TS on $K = 2^{n/2}$ tensors of size

$$D = 2^{O(\log^2 n\cdot(\log\log n+g(\log K)))} = 2^{O(\log^2 n\cdot o(\log K/(\log\log K)^2))} = 2^{o(n)}.$$

Then by Theorem 5.11.2, $2^{-g(\log K)}$-Super-Gap-Max-TS can be reduced to $o(2^{g(\log K)})$-approximate LCS for strings of length $N = KD = 2^{(1/2+o(1))n}$. Note that $2^{f(\log(K^{1+o(1)}))} = o(2^{g(\log K)})$. Thus by our assumption, the last problem can be solved in $N^{2-\delta}$ time, which means AC-BP-SAT on branching program of length $2^{n^{o(1)}}$ and width $O(1)$ on n inputs can be solved in $2^{(1-\delta/2+o(1))n}$ time. Applying Theorem 5.11.4 completes the proof.

Item 3. Assume there exists a $\log^c(N)$-approximation algorithm for LCS in $N^2/2^{\omega(\log\log N)^3}$ time for some $c > 0$. Using a similar calculation as in Theorem 5.1.11, we know that AC-BP-SAT on branching program of length $O(\text{poly}(n))$ and width $O(1)$ on n inputs can be solved in

$$N^2/2^{\omega(\log\log N)^3} \leq 2^{n+O(\log^3 n)}/2^{\omega(\log^3 n)} \leq 2^n/n^{\omega(1)}$$

time. Applying Theorem 5.11.4 completes the proof. □

Chapter 6

BQP Protocols and Approximate Counting Algorithms

6.1 Introduction

6.1.1 Motivation: from the Polynomial Method in Algorithm Design to Communication Complexity

Recent works have shown that the polynomial method, a classical technique for proving circuit lower bounds [144, 157], can be useful in designing efficient algorithms [170, 166, 13, 24, 22, 122, 21].

At a very high level, these algorithms proceed as follows: (1) identify a key subroutine of the core algorithm which has a certain low-degree polynomial representation; (2) replace that subroutine by the corresponding polynomials, and reduce the whole problem to a certain *batched evaluation problem of sparse polynomials*; (3) embed that polynomial evaluation problem to *multiplication of two low-rank (rectangular) matrices*, and apply the fast rectangular matrix multiplication algorithm [77].

As [23] point out, in term of step (3), these algorithms are ultimately making use of the fact that the corresponding matrices of some circuits or subroutines have low *probabilistic rank.*[1] [23] suggest that the *probabilistic rank*, or various low-rank

[1] See Section 7.5 for an illustration with the $n^{2-1/O(\log c)}$ time algorithm for $\mathsf{OV}_{n,c\log n}$ in [13].

decompositions of matrices in general[2], could be more powerful than the polynomial method, and lead to more efficient algorithms, as the polynomial method is just one way to construct them.

It has been noted for a long time that communication protocols are closely related to various notions of rank of matrices. To list a few: deterministic communication complexity is lower bounded by the logarithm of the *rank* of the matrix [129]; quantum communication complexity is lower bounded by the logarithm of the *approximate rank* of the matrix [30, 60]; UPP communication complexity is equivalent to the logarithm of the *sign-rank* of the matrix [143].

These connections are introduced (and usually interpreted) as methods for proving communication complexity lower bounds (see, e.g. the survey by Lee and Shraibman [116]), but they can also be interpreted in the other direction, as a way to *systematically construct low-rank decompositions of matrices.*

In this chapter and the next chapter, we explore the connection between different types of communication protocols and low-rank decompositions of matrices and establish several applications in algorithm design. For all these connections, we start with an efficient communication protocol for a problem F, which implies an efficiently constructible low-rank decomposition of the corresponding communication matrix of F, from which we can obtain fast algorithms.

In fact, in our applications of quantum communication protocols, we also consider k-party protocols, and our algorithms rely on the approximate low-rank decomposition of the tensor of the corresponding communication problem. To the best of our knowledge, this is the first time that *approximate tensor rank* is used in algorithm design (approximate rank has been used before, see e.g. [25, 45, 27, 26] and the corresponding related works section).[3]

[2] A low probabilistic rank implies a probabilistic low-rank decomposition of the matrix.

[3] We remark that a concurrent work [177] makes algorithmic use of *non-negative tensor approximate rank* to construct an optimal data structure for the succinct rank problem.

6.1.2 Quantum Communication Protocols and Deterministic Approximate Counting

In this chapter we establish a generic connection between quantum communication protocols and deterministic approximate counting algorithms.

Theorem 6.1.1. *(Informal) Let $\mathcal{X}, \mathcal{Y}$ be finite sets and $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ be a Boolean function. Suppose f has a quantum communication protocol $\mathcal{P}$*[4] *with complexity $C(\mathcal{P})$ and error ε. Then there is a classical deterministic algorithm $\mathcal{C}$ that receives $A \subseteq \mathcal{X}, B \subseteq \mathcal{Y}$ as input, and outputs a number E such that*

$$\left| \sum_{(x,y)\in A\times B} f(x,y) - E \right| \leq \varepsilon \cdot |A| \cdot |B|.$$

Furthermore, $\mathcal{C}$ runs in $(|A| + |B|) \cdot 2^{O(C(\mathcal{P}))}$ time.

We remark here that there is a simple randomized algorithm running in sub-linear time via random-sampling. Thus the above algorithm is indeed a derandomization of that randomized algorithm.

The above theorem can also be easily generalized to the (number-in-hand) k-party case. See Section 6.2.3 for the definition of the multiparty quantum communication model.

Theorem 6.1.2. *(Informal) Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \to \{0,1\}$ be a Boolean function. Suppose f has a k-party quantum communication protocol $\mathcal{P}$ with complexity $C(\mathcal{P})$ and error ε. Then there is a classical deterministic algorithm $\mathcal{C}$ that receives $X_1 \subseteq \mathcal{X}_1, X_2 \subseteq \mathcal{X}_2, \ldots, X_k \subseteq \mathcal{X}_k$ as input, and outputs a number E such that*

$$\left| \sum_{x_1\in X_1, x_2\in X_2, \ldots, x_k\in X_k} f(x_1, x_2, \ldots, x_k) - E \right| \leq \varepsilon \cdot \prod_{i=1}^{k} |X_i|.$$

Furthermore, $\mathcal{C}$ runs in $(|X_1| + |X_2| + \ldots + |X_k|) \cdot 2^{O(C(\mathcal{P}))}$ time.

[4] We need some technical condition on $\mathcal{P}$, see Corollary 6.3.2 for details.

Sketching Algorithms. In fact, Theorem 6.1.2 implies a stronger *sketching algorithm.* Given subsets $X_1, X_2, \dots, X_k$, the algorithm first computes a $w = 2^{O(C(\mathcal{P}))}$ size sketch sk_i from each X_i in $O(|X_i| \cdot w)$ time deterministically, and the number E can be computed from these sk_i's in $O(k \cdot w)$ time.

The sketch computed by the algorithm is in fact a vector in $\mathbb{R}^w$, and it satisfies a nice additive property. That is, the sketch of $X_1 \sqcup X_2$ (union as a multi-set) is simply $\mathsf{sk}(X_1) + \mathsf{sk}(X_2)$.

Applying existing quantum communication protocols, we obtain several applications of Theorem 6.1.1 and Theorem 6.1.2.

Set-Disjointness and Approximate #OV and #k-OV

We first consider the famous Set-Disjointness problem (Alice and Bob get two vectors u and v in $\{0,1\}^d$ correspondingly, and want to determine whether $\langle u, v\rangle = 0$), which has an efficient quantum communication protocol [1] with communication complexity $O(\sqrt{d})$.

The corresponding count problem for Set-Disjointness is the counting version of the Orthogonal Vectors problem (OV), denoted as $\#\mathsf{OV}_{n,d}$. In this problem, we are given two sets of n vectors $S, T \subseteq \{0,1\}^d$, and the goal is to count the number of pairs $u \in S, v \in T$ such that $\langle u, v\rangle = 0$.

Applying the quantum communication protocol for Set-Disjointness and Theorem 6.1.2, we immediately get an algorithm for the approximate version of #OV.

Theorem 6.1.3. *For any d and any constant $\varepsilon > 0$, $\#\mathsf{OV}_{n,d}$ can be approximated deterministically with additive error $\varepsilon \cdot n^2$ in $n \cdot 2^{O(\sqrt{d})}$ time. In particular, it runs in $n^{1+o(1)}$ time when $d = o(\log^2 n)$.*

Comparison with [68]. [68] gives a *deterministic exact counting* algorithm for $\#\mathsf{OV}_{n,c\log n}$, which runs in $n^{2-O(1/\log c)}$ time. Note that their running time is $n^{2-o(1)}$ when $d = \omega(\log n)$, while our algorithm only achieves an additive approximation, but runs in *near-linear* time for all $d = o(\log^2 n)$.

Another closely related problem, COUNTING PARTIAL MATCH, is the problem that given n query strings from $\{0,1,\star\}^d$ ($\star$ is a "don't care") and n strings from $\{0,1\}^d$, and the goal is to count the number of matching string and query pairs.

Using known reductions between PARTIAL MATCH and OV (see, e.g., Section 2 in [13]), together with the approximate counting algorithm for #OV, we can also solve COUNTING PARTIAL MATCH approximately in the same running time.

The approximate counting algorithm for #OV can be easily generalized to solve #k-OV, which is the problem that given k sets of n vectors $X_1, X_2, \ldots, X_k \subseteq \{0,1\}^d$, and count the number of k-tuples $u_1 \in X_1, u_2 \in X_2, \ldots, u_k \in X_k$ such that $\langle u_1, u_2, \ldots, u_k\rangle = 0$.[5]

Applying Theorem 6.1.2 and observe that the 2-party SET-DISJOINTNESS protocol in [1] can be easily generalized to solve the k-party case (in k-party SET-DISJOINTNESS, there are k players getting $u_1, u_2, \ldots, u_k$ respectively, and they want to determine whether $\langle u_1, u_2, \ldots, u_k\rangle = 0$), we obtain the following approximate counting algorithm for #k-OV.

Theorem 6.1.4. *For any integers k, d and any constant $\varepsilon > 0$, #k-$\mathsf{OV}_{n,d}$ can be approximated deterministically with additive error $\varepsilon \cdot n^k$ in $n \cdot 2^{O(k\sqrt{d})}$ time. In particular, it runs in $n^{1+o(1)}$ time when k is a constant and $d = o(\log^2 n)$.*

Remark 6.1.5. *We remark that similar algorithms with slightly worse running time ($n \cdot d^{O(\sqrt{d})}$ time for additive approximation to #$\mathsf{OV}_{n,d}$) can also be derived using the polynomial method, see Section 6.4 for details. However, we think our new algorithms via quantum communication protocols have the following extra benefits: (1) our algorithm is slightly faster, with a running time of $n \cdot 2^{O(\sqrt{d})}$; (2) our algorithm is derived via a general connection. Once the connection is set up, the algorithm follows in an elegant and black-box way. We hope this general connection could stimulate more applications of quantum communication protocols.*

[5] the generalized inner product of k vectors, is defined as $\langle u_1, u_2, \ldots, u_k\rangle = \sum_{i=1}^{d} \prod_{j=1}^{k} (u_j)_i$.

Sparse Set-Disjointness and Approximate Sparse #OV

Next we consider a sparse version of Set-Disjointness, in which Alice and Bob get two sparse vectors $u, v \in \{0,1\}^m_{\le d}$[6], and want to decide whether $\langle u, v\rangle = 0$.

Using the famous quantum-walk algorithm for Element Distinctness [28], there is an $O(d^{2/3} \log m)$ communication protocol for sparse Set-Disjointness, which is much better than the $O(\sqrt{m})$ protocol for Set-Disjointness when $m \gg d$.

Applying this protocol and Theorem 6.1.1, we can give an algorithm for a sparse version of #OV, denoted as $\#\mathsf{Sparse\text{-}OV}_{n,m,d}$, in which we are given sets $A, B \subseteq \{0,1\}^m_{\le d}$ of n vectors, and the goal is to count the number of distinct $(a,b) \in A \times B$ such that $\langle a, b\rangle = 0$. Formally, we have:

Theorem 6.1.6. *For integers n, m, d and any constant $\varepsilon > 0$, $\#\mathsf{Sparse\text{-}OV}_{n,m,d}$ can be approximated deterministically with additive error $\varepsilon \cdot n^2$ in*

$$n \cdot 2^{O(d^{2/3} \log(m))}$$

time. In particular, when $m = \operatorname{poly}(d)$ and $d = o\left(\left(\frac{\log n}{\log\log n}\right)^{1.5}\right)$, it runs in $n^{1+o(1)}$ time.

We remark that it is possible to improve Theorem 6.1.6 via the polynomial method (see Section 6.4 for details). Again, we emphasize that our focus here is to provide direct applications of our general framework, with the hope that it could stimulate more applications of quantum communication protocols in the classical settings.

Approximate Counting for Formula ∘ SYM Circuits

Finally, we apply our algorithm to approximately count solutions (i.e., satisfying assignments) to a class of circuits, for which no non-trivial algorithms were previously known.

A $\mathsf{Formula} \circ \mathsf{SYM}$ circuit of size m is a formula with $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ basis on m SYM gates[7] at the bottom. Using the quantum query algorithm for Formula

[6] We use $\{0,1\}^m_{\le d}$ to denote all Boolean vectors of length m with at most d ones.

[7] A SYM gate is a gate whose output only depends on the number of ones in the input.

EVALUATION [29] and the split-and-list technique, we obtain the following deterministic approximate counting algorithm for Formula $\circ$ SYM circuits:

Theorem 6.1.7. *For any constant $\varepsilon > 0$, the number of solutions to a Formula $\circ$ SYM circuit of size m can be approximated deterministically within $\varepsilon \cdot 2^n$ additive error in*

$$2^{O(n^{1/2}m^{1/4+o(1)}\sqrt{\log n+\log m})}$$

time. In particular, when $m = n^{2-\delta}$ for some $\delta > 0$, the running time is $2^{o(n)}$.

Previously, even no non-trivial deterministic approximate counting algorithms for AND $\circ$ SYM circuits were known. A recent line of works [95, 97, 154], culminating in [137], construct a PRG for $\mathsf{AND}_m \circ \mathsf{THR}$ circuits with seed length $\mathrm{poly}(\log m, \delta^{-1}) \cdot \log n$, using which one can obtain a quasi-polynomial time deterministic approximate counting algorithm for polynomial size AND $\circ$ THR circuits. However, their PRG constructions rely on the fact that the solution set of an $\mathsf{AND}_m \circ \mathsf{THR}$ circuit is a *polytope*, while the solution set of an AND $\circ$ SYM circuit may not have such a nice geometric structure.

In fact, the only property we need for SYM gates is that they admit an efficient *classical* k-party communication protocol when the inputs are divided to k players (each player sends the contribution of her part). Our algorithm actually works for the following more general problem.

Problem 1. *Given k sets of n vectors $X_1, X_2, \ldots, X_k \subseteq \{0, \ldots, r\}^d$ and d functions $f_1, f_2, \ldots, f_d$ where each f_i is from $[r]^k$ to $\{0,1\}$, and a Boolean formula $\mathcal{F} : \{0,1\}^d \to \{0,1\}$ of $O(1)$ fan-in. Count the number of k-tuples $u_1 \in X_1, u_2 \in X_2, \ldots, u_k \in X_k$ such that*

$$\mathcal{F}(f_1(u_{1,1}, u_{2,1}, \ldots, u_{k,1}), f_2(u_{1,2}, u_{2,2}, \ldots, u_{k,2}), \ldots, f_d(u_{1,d}, u_{2,d}, \ldots, u_{k,d})) = 1.$$

Theorem 6.1.8. *For any constant $\varepsilon > 0$, the above problem can be solved deterministically in $n \cdot 2^{O(d^{1/2+o(1)} \cdot k(\log d + \log r))}$ time, within $\varepsilon \cdot n^k$ additive error.*

6.1.3 Related Works

Other Algorithmic Applications of Approximate Rank

Alon studies the approximate rank of the identity matrix I_n in [25]. It is shown that it is at least $\Omega\left(\frac{\log n}{\varepsilon^2 \log(1/\varepsilon)}\right)$ and at most $O\left(\frac{\log n}{\varepsilon^2}\right)$. Built upon this result, several applications in geometry, coding theory, extremal finite set theory and the study of sample spaces supporting nearly independent random variables are derived. The lower bound also has applications in combinatorial geometry and in the study of locally correctable codes over real and complex numbers, as shown in [45]. In [27, 26], several bounds on approximate rank are derived, together with applications of approximate rank in approximating Nash Equilibria, approximating densest bipartite subgraph and covering convex bodies.

6.2 Preliminaries

6.2.1 Tensor Ranks

In this chapter we are interested in the approximate tensor rank with respect to the ℓ_∞ norm. For more on approximate tensor rank with respect to other norms and their applications, see [158] and the references therein. Now we introduce some relevant definitions.

Definition 6.2.1. We say a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_k}$ is *simple* if $T = v_1 \otimes v_2 \otimes \ldots \otimes v_k$ where $v_i \in \mathbb{R}^{n_i}$.

Definition 6.2.2. For a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_k}$, its rank$(T)$ is defined to be the smallest integer r such that $T = \sum_{i=1}^{r} A_r$ and A_i is simple for all $i \in [r]$.

Definition 6.2.3. For a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_k}$, the approximate rank of T is defined as follows:

$$\operatorname{rank}_\varepsilon(T) = \min\{\operatorname{rank}(S) \mid \|T - S\|_\infty \leq \varepsilon\}.$$

Here $\| \cdot \|_\infty$ is the entry-wise ℓ_∞-norm of a tensor.

6.2.2 Quantum Query Complexity

In this section we recall some previous results on quantum query complexity. Here we emphasize the number of qubits used by the algorithms, which will be crucial when simulating them using classical algorithms (see Section 6.5).

Definition 6.2.4. In the FORMULA EVALUATION problem, we are given a formula $\mathcal{F}$ with $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ basis and $O(1)$ fan-in on n variables $x_1, x_2, \ldots, x_n$. In each query, the algorithm gets the value of x_i, where $i \in [n]$ is determined by the algorithm. The goal is to evaluate the formula.

Theorem 6.2.5 ([29]). *The* FORMULA EVALUATION *problem can be solved in* $O(n^{1/2+o(1)})$ *queries using* $O(\text{polylog}(n))$ *qubits, with failure probability at most* $1/3$.

Remark 6.2.6. *There is an optimal* $O(n^{1/2})$ *query algorithm for* FORMULA EVALUATION *[147]. However, that query algorithm doesn't fit in our applications here for two reasons: (1) the algorithm needs* $O(n)$ *qubits, which is too much for classical simulation; (2) the algorithm is not* computationally efficient *and it takes too much time to compute the corresponding unitary transformation.*

Definition 6.2.7. In the ELEMENT DISTINCTNESS problem, we are given n elements $X = (x_1, x_2, ..., x_n) \in [m]^n$. In each query, the algorithm gets the value of x_i, where $i \in [n]$ is determined by the algorithm. The goal is to decide whether there are two distinct indices $i \neq j$ such that $x_i = x_j$.

Theorem 6.2.8 ([28]). *The* ELEMENT DISTINCTNESS *problem can be solved in* $O(n^{2/3})$ *queries using* $O(n^{2/3} \log m)$ *qubits, with failure probability at most* $1/3$.

6.2.3 Multiparty Quantum Communication Protocols

In this section, we give our definition of multiparty quantum communication protocols.

Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \to \{0,1\}$ be a function. In a k-part quantum communication protocols, there are k players $P_1, P_2, \ldots, P_k$, together with a Hilbert space $H = H_1 \otimes H_2 \otimes \ldots \otimes H_k \otimes \overline{H}$. Here H_i serves as the inner working space for player P_i, and $\overline{H}$ is the communication channel between all the players. Each player P_i receives an input $x_i \in \mathcal{X}_i$ and the goal is to determine $f(x_1, x_2, \ldots, x_k)$.

Now we give the formal definition of a k-party quantum communication protocol.

Definition 6.2.9. A k-part quantum communication protocol $\mathcal{P} = \mathcal{P}(x_1, x_2, \ldots, x_k)$ is a sequence of r unitary transforms $\mathcal{P} = (U_1^{p1}(x_{p1}), U_2^{p2}(x_{p2}), \ldots, U_r^{p_r}(x_{p_r}))$, such that:

- $U_i^{p_i}(x_{p_i})$ is a unitary transform *acting on* $H_{p_i} \otimes \overline{H_i}$ where $\overline{H_i}$ is a subspace spanned by some qubits of $\overline{H}$[8]. That is, it is the action of p_i-th player P_{p_i}, who is in charge of the i-th turn.

- The sequence $p_1, p_2, \ldots, p_r$, and $\overline{H_1}, \overline{H_2}, \ldots, \overline{H_r}$ are fixed and do not depend on $x_1, x_2, \ldots, x_k$. In other words, $\overline{H_i}$ corresponds to the qubits in the channel $\overline{H}$ that player P_{p_i} will modify during its action in the i-th turn, and all players take actions in a fixed, predefined order.

- The communication complexity of $\mathcal{P}$ is defined to be $C(\mathcal{P}) = \sum_{i=1}^{r} \log(\dim(\overline{H_i}))$. The space complexity of P_i is defined to be $S_i(\mathcal{P}) = \log(\dim(H_i \otimes \overline{H}))$.

For a protocol $\mathcal{P} = (U_1^{p1}(x_{p1}), U_2^{p2}(x_{p2}), \ldots, U_r^{p_r}(x_{p_r}))$, we say $\mathcal{P}$ computes f with error ε if we measure the *first* qubit in $\overline{H}$ on the state $U_r^{p_r}(x_{p_r}) \cdot U_{r-1}^{p_{r-1}}(x_{p_{r-1}}) \cdot \ldots \cdot U_2^{p2}(x_{p2}) \cdot U_1^{p1}(x_{p1}) \cdot |0\rangle$, we get $f(x_1, x_2, \ldots, x_k)$ with probability at least $1 - \varepsilon$, for all $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots, x_k \in \mathcal{X}_k$.

Remark 6.2.10. *We remark that our definition here is more complicated than the usual definition of quantum communication protocols in the literature (see, e.g., [114]), but nonetheless, it is equivalent to them. We choose to formulate it in such a way*

[8] i.e., $U_i^{p_i}(x_{p_i})$ does not alter qubits other than those in $H_{p_i} \otimes \overline{H_i}$.

because it is easier to describe the classical simulation of quantum communication protocols for low approximate rank decompositions (see Section 6.5), and the simulation of quantum query algorithms (see below).

Simulating Quantum Query Algorithm in Quantum Communication Protocols

Quantum communication protocols can be built upon quantum query algorithms (see, e.g., [61]). Here we give an example to show how to simulate a quantum query algorithm for FORMULA EVALUATION to construct a quantum communication protocol for the communication problem corresponding to Problem 1, under our definition.

In the corresponding k-party communication problem, there are k players, and the i-th player P_i is given a vector $u_i \in [r]^d$. There are d functions $f_1, f_2, \ldots, f_d$ where each f_i is from $[r]^k$ to $\{0,1\}$, and a Boolean formula $\mathcal{F} : \{0,1\}^d \to \{0,1\}$ of $O(1)$ fan-in. Set $v(i) = f_i(u_{1,i}, u_{2,i}, \ldots, u_{k,i})$. Their goal is to compute $\mathcal{F}(v(1), v(2), \ldots, v(d))$.

Now we show how to construct a quantum communication protocol for the above problem.

Example 1. *Assuming the first player runs a quantum query algorithm for* FORMULA EVALUATION. *For the simulation, we only need to implement the following query gate* O_v: $|i\rangle\,|b\rangle \to |i\rangle\,|b \oplus v(i)\rangle$, *where i is the index of a variable written in binary form and $v(i)$ is the corresponding input bit to $\mathcal{F}$.*

We first specify the channel, $\overline{H}$ is defined as $\overline{H}_{\mathsf{index}} \otimes \overline{H}_{\mathsf{output}} \otimes \overline{H}_1 \otimes \cdots \otimes \overline{H}_k$. $\overline{H}_{\mathsf{index}}$ and $\overline{H}_{\mathsf{output}}$ together simulate the query gate, and $\overline{H}_i$ is the place for player P_i to write her number.

In the beginning, all qubits in $\overline{H}$ are $|0\rangle$. When the first player wants to apply O_v on some qubits in H_1, it first swaps the qubits containing i and b in H_1 with $\overline{H}_{\mathsf{index}}$ and $\overline{H}_{\mathsf{output}}$ in $\overline{H}$.

Each player P_j in turn reads i in $\overline{H}_{\mathsf{index}}$ and writes the value of $u_{j,i}$ to qubits in $\overline{H}_j$. Note that each player can write the value of $u_{j,i}$ to qubits in $\overline{H}_j$ using a unitary transformation since all qubits in $\overline{H}_j$ are $|0\rangle$ at the beginning, by assumption.

Now, given the value of i and $u_{1,i}, u_{2,i}, \dots, u_{k,i}$, the first player maps $|i\rangle\,|b\rangle$ to $|i\rangle\,|b \oplus v(i)\rangle$ via a unitary transformation. Now the gate O_v is implemented, but we still have to clean up the garbages in $\overline{H}_j$'s, and set them back to $|0\rangle$'s. This can be done by applying reverse transforms of all applied unitary transformation, in the reverse order.

The communication complexity of this protocol is $O(Q \cdot k(\log d + \log r))$, where Q is the query complexity of the quantum query algorithm. Also, using the algorithm in Theorem 6.2.5, the communication complexity of this protocol is $O(n^{1/2+o(1)} \cdot k(\log d + \log r))$.

6.3 Approximate Counting Algorithms from Quantum Communication Protocols

Let $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k \to \{0,1\}$ be a function. Let $M_f \in \{0,1\}^{|\mathcal{X}_1| \times |\mathcal{X}_2| \times \dots \times |\mathcal{X}_k|}$ denote the Boolean tensor whose $(x_1, x_2, \dots, x_k)$ entry is $f(x_1, x_2, \dots, x_k)$. The following connection between 2-party quantum communication complexity and approximate rank is first observed in [61]. This result can be generalized to the k-party case to get the following theorem. Full details can be found in Section 6.5.

Theorem 6.3.1. *Let $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k \to \{0,1\}$ be a Boolean function. Suppose there exists a k-party efficient quantum communication protocol $\mathcal{P}$, such that $\mathcal{P}$ gives the correct answer with probability at least $1 - \varepsilon$ on every input, then $\mathrm{rank}_\varepsilon(M_f) \le 2^{O(C(\mathcal{P}))}$, or equivalently, there exist simple tensors $A_1, A_2, \dots, A_{2^{O(C(\mathcal{P}))}}$ such that*

$$\left\| M_f - \sum_{i=1}^{2^{O(C(\mathcal{P}))}} A_i \right\|_\infty \le \varepsilon.$$

In Section 6.5 we further show how to use classical deterministic algorithms to simulate quantum communication protocols. Notice that here the time complexity depends on the space complexity of the quantum communication protocol to use.

Corollary 6.3.2. *Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \to \{0,1\}$ be a Boolean function. Suppose there exists a k-party efficient quantum communication protocol $\mathcal{P}$, such that $\mathcal{P}$ gives the correct answer with probability at least $1-\varepsilon$ on every input, and all the unitary transformation used in the $\mathcal{P}$ can be constructed in polynomial time (with respect to their sizes) by a deterministic classical algorithm. Then there exists k deterministic classical algorithms $\mathbb{A}_{\mathcal{X}_1}, \mathbb{A}_{\mathcal{X}_2}, \ldots, \mathbb{A}_{\mathcal{X}_k}$ such that $\mathbb{A}_{\mathcal{X}_i}$runs in $2^{O(C(\mathcal{P})+S_i(\mathcal{P}))}$ time, receives $x_i \in \mathcal{X}_i$ as input and outputs a vector $\mathbb{A}_{\mathcal{X}_i}(x_i) \in \mathbb{R}^{2^{O(C(\mathcal{P}))}}$, and for any $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots, x_k \in \mathcal{X}_k$,*

$$-\varepsilon \leq \langle \mathbb{A}_{\mathcal{X}_1}(x_1), \mathbb{A}_{\mathcal{X}_2}(x_2), \ldots, \mathbb{A}_{\mathcal{X}_k}(x_k) \rangle - f(x_1, x_2, \ldots, x_k) \leq \varepsilon.$$

Based on Corollary 6.3.2, for any Boolean function $f : \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \to \{0,1\}$ with an efficient efficient quantum communication protocol, there also exists an efficient approximate counting algorithm for f.

Theorem 6.3.3. *Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be finite sets and $f : \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \to \{0,1\}$ be a Boolean function. Suppose there exists a k-party efficient quantum communication protocol $\mathcal{P}$, such that $\mathcal{P}$ gives the correct answer with probability at least $1-\varepsilon$ on every input, and all the unitary transformation used in the $\mathcal{P}$ can be constructed in polynomial time (with respect to their sizes) by a deterministic classical algorithm. Then there exists a classical deterministic algorithm $\mathcal{C}$ that receives $X_1 \subseteq \mathcal{X}_1, X_2 \subseteq \mathcal{X}_2, \ldots, X_k \subseteq \mathcal{X}_k$ as input, and outputs a number E such that*

$$\left| \sum_{x_1 \in X_1, x_2 \in X_2, \ldots, x_k \in X_k} f(x_1, x_2, \ldots, x_k) - E \right| \leq \varepsilon \cdot \prod_{i=1}^{k} |X_i|.$$

Furthermore, $\mathcal{C}$ runs in $\sum_{i=1}^{k} |X_i| \cdot 2^{C(\mathcal{P})+S_i(\mathcal{P})}$ time.

Proof. For all $x_i \in \mathcal{X}_i$ we first use $\mathbb{A}_{\mathcal{X}_i}$ in Corollary 6.3.2 to calculate $\mathbb{A}_{\mathcal{X}_i}(x_i) \in$

$\mathbb{R}^{2^{O(C(\mathcal{P}))}}$, in $\sum_{i=1}^{k} |X_i| \cdot 2^{C(\mathcal{P})+S_i(\mathcal{P})}$ time. Then we directly output

$$\left\langle \sum_{x_1 \in X_1} \mathbb{A}_{\mathcal{X}_1}(x_1), \sum_{x_2 \in X_2} \mathbb{A}_{\mathcal{X}_2}(x_2), \dots, \sum_{x_k \in X_k} \mathbb{A}_{\mathcal{X}_k}(x_k) \right\rangle.$$

The correctness simply follows from the fact that for all $(x_1, x_2, \dots, x_k) \in \prod_i \mathcal{X}_i$,

$$-\varepsilon \le \langle \mathbb{A}_{\mathcal{X}_1}(x_1), \mathbb{A}_{\mathcal{X}_2}(x_2), \dots, \mathbb{A}_{\mathcal{X}_k}(x_k) \rangle - f(x_1, x_2, \dots, x_k) \le \varepsilon.$$

□

Remark 6.3.4. *The algorithm described above is actually a sketching algorithm. We may define the sketch for X_i as $\mathsf{sk}_i(X_i) = \sum_{x_i \in X_i} \mathbb{A}_{\mathcal{X}_i}(x_i) \in \mathbb{R}^{2^{O(C(\mathcal{P}))}}$ and the number E can be computed from these sk_i's. This sketching algorithm satisfies a nice additive property, i.e., the sketch of $A \sqcup B$ (union as a multi-set) is simply $\mathsf{sk}_i(A) + \mathsf{sk}_i(B)$.*

Now we give approximate counting algorithms for concrete problems, using Theorem 6.3.3.

6.3.1 Counting the k-Tuples of Orthogonal Vectors

The goal of this section is to prove the following theorem.

Reminder of Theorem 6.1.4 *For any integers k, d and any constant $\varepsilon > 0$, $\#k$-$\mathsf{OV}_{n,d}$ can be approximated deterministically with additive error $\varepsilon \cdot n^k$ in $n \cdot 2^{O(k\sqrt{d})}$ time. In particular, it runs in $n^{1+o(1)}$ time k is a constant and $d = o(\log^2 n)$.*

We first consider quantum communication protocols for the following function f.

Definition 6.3.5. Let $\mathcal{X}_1 = \mathcal{X}_2 = \ldots = \mathcal{X}_k = \{0,1\}^d$ and

$$f(x_1, x_2, \dots, x_k) = \begin{cases} 1 & \text{if } \langle x_1, x_2, \dots, x_k \rangle = 0 \\ 0 & \text{otherwise} \end{cases}.$$

The corresponding communication problem can be solved using the quantum communication protocol in [1] with communication complexity $O(k\sqrt{d})$ and space complexity $O(\text{polylog}(d))$, with constant failure probability. If we use the algorithm in Theorem 6.3.3, together with the efficient quantum communication protocol mentioned above, we can then deterministically count the number of k-tuples of orthogonal vectors, in time $n \cdot 2^{O(k \cdot \sqrt{d})}$ time, with an additive $\varepsilon \cdot n^k$ error.

6.3.2 Counting the Pairs of Orthogonal Sparse Vectors

The goal of this section is to prove the following theorem.

Reminder of Theorem 6.1.6 *For integers n, m, d and any constant $\varepsilon > 0$, #Sparse-OV$_{n,m,d}$ can be approximated deterministically with additive error $\varepsilon \cdot n^2$ in*

$$n \cdot 2^{O(d^{2/3} \log(m))}$$

time. In particular, when $m = \text{poly}(d)$ and $d = o\left(\left(\frac{\log n}{\log \log n}\right)^{1.5}\right)$, it runs in $n^{1+o(1)}$ time.

Again we consider quantum communication protocols for the following function f.

Definition 6.3.6. Let $\mathcal{X} = \mathcal{Y} = \{0,1\}^m_{\leq d}$ and

$$f(x,y) = \begin{cases} 1 & \text{if } \langle x, y \rangle = 0 \\ 0 & \text{otherwise} \end{cases}.$$

The corresponding communication problem can be solved with communication complexity $O(d^{2/3} \log m)$, by simulating the quantum query algorithm in Theorem 6.2.8 for ELEMENT DISTINCTNESS. Too see the connection, let $S = \{i \mid x_i = 1\}$ and $T = \{i \mid y_i = 1\}$. We will have $f(x,y) = 1$ if and only if all elements in $S \sqcup T$ (union as a multi-set) are distinct. Now, using the algorithm in Theorem 6.3.3, together with the efficient quantum communication protocol mentioned above, we can deterministically count the number of orthogonal pairs in S and T, in $n \cdot 2^{O(d^{2/3} \log(m))}$ time, with an additive $\varepsilon \cdot n^k$ error.

6.3.3 Counting Solutions to $\mathsf{Formula} \circ \mathsf{SYM}$ Circuits

The goal of this section is to solve the following problem.

Reminder of Problem 1 *Given k sets of n vectors $S_1, S_2, \ldots, S_k \subseteq \{0, \ldots, r\}^d$ and d functions $f_1, f_2, \ldots, f_d$ where each f_i is from $\{0, \ldots, r\}^k$ to $\{0, 1\}$, and a Boolean formula $\mathcal{F} : \{0,1\}^d \to \{0,1\}$ of $O(1)$ fan-in. Count the number of k-tuples $u_1 \in S_1, u_2 \in S_2, \ldots, u_k \in S_k$ such that*

$$\mathcal{F}(f_1(u_{1,1}, u_{2,1}, \ldots, u_{k,1}), f_2(u_{1,2}, u_{2,2}, \ldots, u_{k,2}), \ldots, f_d(u_{1,d}, u_{2,d}, \ldots, u_{k,d})) = 1.$$

Reminder of Theorem 6.1.8 *For any constant $\varepsilon > 0$, the above problem can be solved deterministically in $n \cdot 2^{O(d^{1/2+o(1)} \cdot k(\log d + \log r))}$ time, within $\varepsilon \cdot n^k$ additive error.*

The corresponding k-party communication problem can be solved by a quantum communication protocol with communication complexity $O(d^{1/2+o(1)} \cdot k(\log d + \log r))$, by simulating the quantum query algorithm for $\mathsf{Formula}$-Evaluation in Theorem 6.2.5. For details see Example 1. By our framework, this implies an approximate counting algorithm to the problem mentioned above in time $n \cdot 2^{O(d^{1/2+o(1)} \cdot k(\log d + \log r))}$, with an additive $\varepsilon \cdot n^k$ error.

Here we mention one application to the approximate counting algorithm above.

Reminder of Theorem 6.1.7 *For any constant $\varepsilon > 0$, the number of solutions to a $\mathsf{Formula} \circ \mathsf{SYM}$ circuit of size m can be approximated deterministically within $\varepsilon \cdot 2^n$ additive error in*

$$2^{O(n^{1/2} m^{1/4+o(1)} \sqrt{\log n + \log m})}$$

time. In particular, when $m = n^{2-\delta}$ for some $\delta > 0$, the running time is $2^{o(n)}$.

Proof of Theorem 6.1.7. Consider a $\mathsf{Formula} \circ \mathsf{SYM}$ circuit $C : \{0,1\}^n \to \{0,1\}$ with m symmetric gates $X_1, X_2, \ldots, X_m$ and a Boolean formula $\mathcal{F}$ of $O(1)$ fan-in. Here we slightly abuse of notation by regarding X_i as a function that maps the number of inputs bits with value one to an output in $\{0,1\}$. We can approximately count the number of solutions to C as follows.

We split the n inputs bits into s groups, each with n/s input bits. Then for each group, we enumerate all the $2^{n/s}$ possible assignments to the n/s input bits. We create a vector in $\{0, \dots, n/s\}^m$ for each possible assignment, where the i-th entry is simply the number of ones in the assignment which is an input bit to the i-th symmetric gate X_i. Now, the number of solutions to the circuit $\mathcal{C}$, is simply the same as Problem 1, by setting

$$f_i(u_{1,i}, u_{2,i}, \dots, u_{k,i}) = X_i(u_{1,i} + u_{2,i} + \dots + u_{k,i}).$$

The total time complexity would be $2^{n/s} \cdot 2^{O\left(m^{1/2+o(1)} \cdot s(\log m + \log(n/s))\right)}$, with an additive $\varepsilon \cdot 2^n$ error. Setting $s = \frac{n^{1/2}}{m^{1/4+o(1)}\sqrt{\log n + \log m}}$, the final time complexity would be $2^{O(n^{1/2} m^{1/4+o(1)} \sqrt{\log n + \log m})}$. □

6.4 Deterministic Approximate Counting Algorithm for #OV via Approximate Polynomial

Here we show that using the approximate polynomial for OR, one can also derive a deterministic approximate counting algorithm for #OV, but with running time worse than Theorem 6.1.3.

Theorem 6.4.1. *For any d and any $\varepsilon > 0$, $\#\mathsf{OV}_{n,d}$ can be approximated with additive error $\varepsilon \cdot n^2$ in $n \cdot \binom{d}{\le O(\sqrt{d \log 1/\varepsilon})}$ time. In particular, it runs in $n^{1+o(1)}$ time when ε is a constant and $d = o((\log n / \log\log n)^2)$.*

Proof. By [59, 85], there is a polynomial $P_\varepsilon : \{0,1\}^d \to \mathbb{R}$ such that:

- P_ε is of degree $D = O\left(\sqrt{d \log 1/\varepsilon}\right)$.
- Given $z \in \{0,1\}^d$, if $\mathsf{OR}(z) = 0$, then $P_\varepsilon(z) \in [1-\varepsilon, 1]$, otherwise $P_\varepsilon(z) \in [0, \varepsilon]$.
- P_ε can be constructed in time polynomial in its description size.

Let $z_S := \prod_{i \in S} z_i$, one can write $P_\varepsilon(z) := \sum_{|S| \le D} c_S \cdot z_S$. Let $A, B \subseteq \{0,1\}^d$ with

$|A| = |B| = n$ be the given #OV instance, we compute

$$\begin{aligned}
E &= \sum_{(x,y)\in A\times B} P_\varepsilon(x_1 \cdot y_1, x_2 \cdot y_2, \dots, x_d \cdot y_d) \\
&= \sum_{(x,y)\in A\times B} \sum_{|S|\le D} c_S \cdot x_S \cdot y_S \\
&= \sum_{|S|\le D} c_S \cdot \sum_{(x,y)\in A\times B} x_S \cdot y_S \\
&= \sum_{|S|\le D} c_S \cdot \sum_{x\in A} x_S \cdot \sum_{y\in B} y_S \qquad (6.1)
\end{aligned}$$

In above x_S and y_S denote $\sum_{i\in S} x_i$ and $\sum_{i\in S} y_i$ respectively.

By the property of P_ε, is easy to see that E approximates the number of orthogonal pairs with an additive error $O(\varepsilon \cdot n^2)$. And by (6.1), E can be computed in

$$n \cdot \binom{d}{\le D} = n \cdot \binom{d}{\le O(\sqrt{d \log 1/\varepsilon})}$$

time.

When ε is a constant, the above simplifies to $n \cdot d^{O(\sqrt{d})}$, which is $n^{1+o(1)}$ when $d = o((\log n / \log\log n)^2)$. □

Remark 6.4.2. *We remark that the above algorithm also works for #Sparse-$\mathsf{OV}_{n,m,d}$ and #k-$\mathsf{OV}_{n,d}$, with running times $n \cdot \binom{m}{\le O\left(\sqrt{d \log 1/\varepsilon}\right)}$ and $n \cdot k \cdot \binom{d}{\le O(\sqrt{d \log 1/\varepsilon})}$, respectively. In particular, it improves the running time in Theorem 6.1.6.*

6.5 Quantum Communication Protocols and Approximate Rank

In this section we explain how to simulate a quantum communication protocol by a deterministic algorithm, and thus prove Theorem 6.3.1 and Corollary 6.3.2.

We first prove the following theorem under our definition of k-party quantum communication protocol. The proof itself follows closely previous proof for 2-party quantum communication protocols in [61].

Theorem 6.5.1. *For a k-party quantum communication protocol. The final state of $\mathcal{P}$ on input $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2, \ldots, x_k \in \mathcal{X}_k$, can be written as*

$$\sum_{i \in S} a_i^1(x_1) \cdot a_i^2(x_2) \cdot \ldots a_i^k(x_k) \cdot |A_i^1(x_1)\rangle \, |A_i^2(x_2)\rangle \ldots |A_i^k(x_k)\rangle \, |f(i)\rangle \, ,$$

where $a_i^1(x_1), a_i^2(x_2), \ldots, a_i^k(x_k)$ are complex numbers and $|A_i^1(x_1)\rangle, |A_i^2(x_2)\rangle, \ldots, |A_i^k(x_k)\rangle$ are unit vectors, $S = \{0,1\}^{C(\mathcal{P})}$ is the set of all possible histories of modifications, and $f : S \to \overline{H}$ is a function which maps the history of modifications to a state in $\overline{H}$.

Proof. The proof is by induction. When $r = 0$ the theorem is obvious. Suppose after applying $U_1^{p_1}, U_2^{p_2}, \ldots, U_{r-1}^{p_{r-1}}$ the final state is

$$\sum_{i \in S'} a_i^1(x_1) \cdot a_i^2(x_2) \cdot \ldots a_i^k(x_k) \cdot |A_i^1(x_1)\rangle \, |A_i^2(x_2)\rangle \ldots |A_i^k(x_k)\rangle \, |f(i)\rangle \, ,$$

where $S' = \{0,1\}^{\sum_{i=1}^{r-1} \log(\dim(\overline{H_i}))}$. Now we apply $U_r^{p_r}$. Since $U_r^{p_r}$ acts on $H_{p_r} \otimes \overline{H_r}$, thus every element of the superposition in the previous states splits into at most $2^{\log \dim(\overline{H_r})}$ terms, depending on the state of the qubits in $\overline{H_r}$ after applying $U_r^{p_r}$. Thus, after applying $U_r^{p_r}$ there will be at most $|S'| \cdot 2^{\log \dim(\overline{H_r})} = 2^{\sum_{i=1}^{r} \log(\dim(\overline{H_i}))}$ terms in the superposition. □

Now let S_1 to be the set that contains all $i \in S$ such that the first qubit in $f(i)$ is $|1\rangle$, and let

$$\phi(x_1, x_2, \ldots, x_k) = \sum_{i \in S_1} a_i^1(x_1) \cdot a_i^2(x_2) \cdot \ldots a_i^k(x_k) \cdot |A_i^1(x_1)\rangle \, |A_i^2(x_2)\rangle \ldots |A_i^k(x_k)\rangle \, |f(i)\rangle$$

be the part of the final state that corresponds to a 1-output of the protocol.

Now for $i, j \in S_1$, we define

$$a_{i,j}^p(x_p) = \overline{a_i^p(x_p)} a_j^p(x_p) \, \langle A_i^p(x_p) | A_j^p(x_p) \rangle \, .$$

Thus, the probability of outputting 1 is

$$\langle \phi(x_1, x_2, \ldots, x_k) | \phi(x_1, x_2, \ldots, x_k) \rangle = \sum_{i,j \in S_1} \prod_{p=1}^{k} a_{i,j}^{p}(x_p).$$

Thus, we have

$$\left| \sum_{i,j \in S_1} \prod_{p=1}^{k} a_{i,j}^{p}(x_p) - f(x_1, x_2, \ldots, x_k) \right| \leq \varepsilon.$$

Now we have finished our prove for Theorem 6.3.1, since the tensor defined by

$$A(x_1, x_2, \ldots, x_k) = \prod_{p=1}^{k} a_{i,j}^{p}(x_p)$$

is a simple tensor, and thus $\text{rank}_\varepsilon(M_f) \leq |S_1|^2 = 2^{O(C(\mathcal{P}))}$.

Scrutinizing the proof of Thereom 6.5.1, for each player P_p, for any input $x_p \in \mathcal{X}_p$, we can calculate $a_{i,j}^{p}(x_p)$ in $2^{O(S_p(\mathcal{P}))} \cdot \text{poly}(|S|) = 2^{O(S_p(\mathcal{P})+C(\mathcal{P}))}$ time, by using a classical deterministic algorithm to simulate the procedure above, as long as all unitary transforms U_i^{pi} can be constructed in polynomial time (with respect to its size).

Chapter 7

$\mathsf{PH}^{\mathsf{cc}}$ Protocols and Efficient SAT Algorithms

7.1 Introduction

In this chapter we consider $\mathsf{PH}^{\mathsf{cc}}$ protocols and $\mathsf{AM}^{\mathsf{cc}}$ protocols (note that the latter protocols are special cases of the first protocols). We show that these protocols imply non-trivial f-Satisfying-Pair algorithms.

7.1.1 Arthur-Merlin Communication Protocols and a New Approximate Max-IP Algorithm

We first consider $\mathsf{AM}^{\mathsf{cc}}$ protocols, which are defined as below.

Definition 7.1.1. An Arthur-Merlin communication protocol ($\mathsf{AM}^{\mathsf{cc}}$) Π for a partial function $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1, \perp\}$[1] proceeds as follows:

- Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$.
- Alice and Bob toss some public coins jointly and send the random string $r \in \{0, 1\}^*$ to Merlin (r is called the random challenge).

[1] $F(x, y) = \perp$ means $F(x, y)$ is undefined.

- Based on x, y and the random challenge r, Merlin sends Alice and Bob a proof ψ, and Alice and Bob decide to accept or not independently and deterministically. We require the following conditions:
  - If $F(x, y) = 1$, with probability $1 - \varepsilon$ over the random challenge r, there is a proof ψ from Merlin such that Alice and Bob both accept.
  - If $F(x, y) = 0$, with probability $1 - \varepsilon$ over the random challenge r, there is no proof ψ from Merlin such that Alice and Bob both accept.

We call the parameter ε the error of the protocol Π. Moreover, we say the protocol is *computationally efficient* if Alice and Bob's behavior can be computed in polynomial-time w.r.t. their input lengths.

We show that for any function F, a low-complexity and computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocol implies a faster algorithm for the corresponding F-Satisfying-Pair problem (defined below).

For a partial function $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1, \perp\}$, where $\mathcal{X}$ and $\mathcal{Y}$ are two sets, we define F-Satisfying-Pair$_n$ as the problem that given two sets $A \subseteq \mathcal{X}$ and $B \subseteq \mathcal{Y}$ of size n, distinguish between the following two cases: (1) There is an $(x, y) \in A \times B$ such that $F(x, y) = 1$. (2) For all $(x, y) \in A \times B$, $F(x, y) = 0$.

Theorem 7.1.2 (Algorithms from $\mathsf{AM}^{\mathsf{cc}}$ protocols). *Let $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1, \perp\}$ be a partial function. Suppose there is a computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocol for F with communication complexity T and error ε. Then for n such that $2^T \le (\sqrt{\varepsilon} n)^{0.1}$, there is an $O\left(\varepsilon n^2 \cdot \operatorname{polylog}(n) + n \cdot 2^T\right)$ time randomized algorithm for F-Satisfying-Pair$_n$.*

A New Algorithm for Approximate Max-IP

The first application of Theorem 7.1.2 is a new algorithm for approximate Maximum Inner Product. We use $\mathsf{Max\text{-}IP}_{n,d}$ to denote the problem that given sets $A, B \subseteq \{0, 1\}^d$ with size n, compute $\mathsf{Max}(A, B) := \max_{(a,b) \in A \times B} \langle a \cdot b \rangle$.

To phrase this as an F-Satisfying-Pair problem, we first define the following gap inner product problem.

Definition 7.1.3 (Multiplicative-Gap Inner Product). Consider the following problem, denoted as $\textsf{Gap-Inner-Product}_d$, Alice and Bob hold strings $x, y \in \{0,1\}^d$ respectively, and they are given an integer τ. They want to distinguish between the following two cases: (Yes) $x \cdot y \geq 2\tau$; (No) $x \cdot y \leq \tau$.

Adapting the classical Goldwasser-Sisper $\textsf{AM}$ protocol for approximating set size [92], we can derive an efficient $\textsf{AM}^{\textsf{cc}}$ protocol for $\textsf{Gap-Inner-Product}_d$.

Lemma 7.1.4 ($\textsf{AM}^{\textsf{cc}}$ protocol for $\textsf{Gap-Inner-Product}_d$). *There is an $\textsf{AM}^{\textsf{cc}}$ protocol for $\textsf{Gap-Inner-Product}_d$ with error ε and communication complexity*

$$\log \binom{d}{\leq O(\log \varepsilon^{-1})}.^{2}$$

Applying Theorem 7.1.2, the following algorithm for approximating $\textsf{Max-IP}$ follows directly, matching the previous best algorithm in [70].

Corollary 7.1.5. *There is an algorithm for computing a 2-approximation to $\textsf{Max-IP}_{n,c\log n}$, which runs in $n^{2-1/O(\log c)}$ time.*

Remark 7.1.6. *The constant 2 in Corollary 7.1.5 can be replaced by any other constant $\kappa > 1$.*

We remark here that a direct application of the Goldwasser-Sisper protocol and parallel repetition leads to a communication protocol with communication complexity $O(\log d \log \varepsilon^{-1})$, which is slightly worse than Lemma 7.1.4. In particular, such a protocol only gives an algorithm with running time $n^{2-1/O(\log d)}$, which is worse than $n^{2-1/O(\log c)}$ when $c \ll d = c \log n$. In order to get the improved complexity in Lemma 7.1.4, we make use of a clever sampling scheme using Poisson distributions, see Section 7.3.1 for details.

[2] $\binom{n}{\leq m}$ denotes $\sum_{i=0}^{m} \binom{n}{i}$.

Evidences that Longest Common Subsequence and Edit Distance do not Have Fast $\mathsf{AM}^{\mathsf{cc}}$ Protocols

It has been a long-standing open problem in communication complexity to prove an $\omega(\log n)$ $\mathsf{AM}^{\mathsf{cc}}$ lower bound for any explicit function [38, 93, 94]—it is consistent with our current knowledge that all known natural communication problems have $O(\log n)$ $\mathsf{AM}^{\mathsf{cc}}$ protocols.

We consider two natural communication problems here, $\mathsf{LCS}^{\mathsf{cc}}_d$ and $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}_d$, in which Alice and Bob hold strings $x, y \in \{0,1\}^d$ respectively, and are given an integer τ. Their goal is to decide whether $\mathsf{LCS}(x,y) \ge \tau$ ($\mathsf{Edit\text{-}Distance}(x,y) \ge \tau$).

Our Theorem 7.1.2 shows that if $\mathsf{LCS}^{\mathsf{cc}}$ or $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ admit low-complexity and computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols, it would imply non-trivial algorithms for the corresponding F-$\mathsf{Satisfying\text{-}Pair}$ problem. By a known reduction in [10], that would, in turn, implies non-trivial algorithms for Formula-SAT[3]—much faster than the current state-of-the-art [161]! Therefore, at least for these two problems, constructing low-complexity $\mathsf{AM}^{\mathsf{cc}}$ protocol could be hard, which may also be viewed as an evidence that they do not have efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols.

Theorem 7.1.7. *If $\mathsf{LCS}^{\mathsf{cc}}_d$ admits computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols with complexity* $\operatorname{polylog}(d)$, *then Formula-$\mathsf{SAT}$ of polynomial-size formulas admits an $2^{n-n^{1-\delta}}$ time algorithm for any constant $\delta > 0$. The same holds for $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ in place of $\mathsf{LCS}^{\mathsf{cc}}$.*

The state-of-the-art algorithm for Formula-SAT runs in $o(2^n)$ time only when the formula size is smaller than n^3 [161]. It is even purposed as a hypothesis that no $2^n/n^{\omega(1)}$ time algorithm exists for $n^{3+\Omega(1)}$-size Formula-SAT in [6]. Therefore, our results imply that if $\mathsf{LCS}^{\mathsf{cc}}$ or $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ admits fast (computationally efficient) $\mathsf{AM}^{\mathsf{cc}}$ protocols, then that would refute the hypothesis in [6]:

Corollary 7.1.8. *Under the following hypothesis*[4], *$\mathsf{LCS}^{\mathsf{cc}}_d$ and $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}_d$ do not admit computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols with complexity* $\operatorname{polylog}(d)$:

[3] Formula-SAT is the problem that deciding whether a given formula is satisfiable.

[4] which is much weaker than the hypothesis in [6]

- *There is a constant $\delta > 0$ such that Formula-SAT of polynomial-size formulas requires $2^{n-n^{1-\delta}}$ time.*

In fact, in Section 7.6, we show that the above corollary can be generalized to hold for computationally efficient $\mathsf{PH}^{\mathsf{cc}}$ protocols (see Section 7.6 for a formal definition). Formally, we have:

Theorem 7.1.9. *Under the same hypothesis as in Corollary 7.1.8, $\mathsf{LCS}_d^{\mathsf{cc}}$ and $\mathsf{Edit\text{-}Dist}_d^{\mathsf{cc}}$ do not admit computationally efficient $\mathsf{PH}^{\mathsf{cc}}$ protocols with complexity* $\operatorname{polylog}(d)$.

7.2 Preliminaries

7.2.1 Fast Rectangular Matrix Multiplication

In this chapter we make use of the algorithms for fast rectangular matrix multiplication.

Theorem 7.2.1 ([77]). *There is an $N^2 \cdot \operatorname{polylog}(N)$ time algorithm for multiplying two matrices A and B with size $N \times N^\alpha$ and $N^\alpha \times N$, where $\alpha > 0.172$.*

7.2.2 Random Variables and Poisson Distributions

Throughout the chapter, we use $X \simeq Y$ to mean that X and Y have the same distribution. We use $X \succeq Y$ to denote stochastic dominance, i.e., $X \succeq Y$ iff for any $t \in \mathbb{R}$, $\Pr[X \geq t] \geq \Pr[Y \geq t]$.

We use $\operatorname{Pois}(\lambda)$ to denote a Poisson distribution with parameter λ. We will need the following two facts about Poisson distributions in the chapter.

Lemma 7.2.2. *Suppose $\{X_i\}_{i=1}^n$ is a set of independent random variables with $X_i \sim \operatorname{Pois}(\lambda_i)$, then*

$$\sum_{i=1}^n X_i \sim \operatorname{Pois}\left(\sum_{i=1}^n \lambda_i\right).$$

Lemma 7.2.3.

$$\Pr\left[\operatorname{Pois}(\lambda) \geq 1.2\lambda\right] \leq e^{-0.01\lambda}$$

and

$$\Pr\left[\mathrm{Pois}(\lambda) \leq 0.8\lambda\right] \leq e^{-0.01\lambda}.$$

Proof. By standard tail inequalities of Poisson distribution (see Theorem 5.4 in [131]),

$$\Pr\left[\mathrm{Pois}(\lambda) \geq x\right] \leq e^{-\lambda}(e\lambda/x)^{x}$$

and

$$\Pr\left[\mathrm{Pois}(\lambda) \leq x\right] \leq e^{-\lambda}(e\lambda/x)^{x}.$$

Thus for any $\lambda > 0$, we have

$$\Pr\left[\mathrm{Pois}(\lambda) \geq 1.2\lambda\right] \leq e^{-\lambda}(e/1.2)^{1.2\lambda} < e^{-0.01\lambda}$$

and

$$\Pr\left[\mathrm{Pois}(\lambda) \leq 0.8\lambda\right] \leq e^{-\lambda}(e/0.8)^{0.8\lambda} < e^{-0.01\lambda}. \qquad \square$$

7.3 Algorithms from Arthur-Merlin Communication Protocols

In this section, we prove our algorithmic applications of $\mathsf{AM}^{\mathsf{cc}}$ protocols. We first show faster $\mathsf{AM}^{\mathsf{cc}}$ protocols for F imply faster F-Satisfying-Pair algorithms.

Reminder of Theorem 7.1.2 *Let $F : \mathcal{X} \times \mathcal{Y} \to \{0,1,\perp\}$ be a partial function. Suppose there is a computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocol for F with communication complexity T and error ε. Then for n such that $2^T \leq (\sqrt{\varepsilon}n)^{0.1}$, there is an $O\left(\varepsilon n^2 \cdot \mathrm{polylog}(n) + n \cdot 2^T\right)$ time randomized algorithm for F-Satisfying-Pair$_n$.*

Proof. We first assume $n < \frac{1}{10\sqrt{\varepsilon}}$. After drawing a random challenge, for each element $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ we construct a Boolean vector $\mathbb{A}_{\mathcal{X}}(x)$ and $\mathbb{A}_{\mathcal{Y}}(y)$ of length 2^T, where each the i-th entry indicates whether Alice (Bob) accepts when receiving the proof i from Merlin. Here we regard i as a Boolean string of length T via a natural bijection between $[2^T]$ and $\{0,1\}^T$.

According to the guarantee of an $\mathsf{AM}^{\mathsf{cc}}$ protocol, for each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, when $F(x,y) = 1$, with probability at least $1 - \varepsilon$ over the random challenge, we have $\langle \mathbb{A}_{\mathcal{X}}(x), \mathbb{A}_{\mathcal{Y}}(y)\rangle > 0$, and when $F(a,b) = 0$ we have $\langle \mathbb{A}_{\mathcal{X}}(x), \mathbb{A}_{\mathcal{Y}}(y)\rangle > 0$ with probability at most ε over the random challenge.

By a union bound on all pairs of elements in A and B, we have with probability at least 0.99, for all $a \in A$ and $b \in B$, $\langle \mathbb{A}_A(a), \mathbb{A}_B(b)\rangle > 0$ if and only if $F(a,b) = 1$. Consequently, with probability at least 0.99,

$$\left\langle \sum_{a \in A} \mathbb{A}_A(a), \sum_{b \in B} \mathbb{A}_B(b) \right\rangle > 0$$

if and only if there exist $a \in A$ and $b \in B$ such that $F(a,b) = 1$.

For general $n = |A| = |B|$, we first split A and B into $O(\sqrt{\varepsilon} n)$ groups, each with at most $\frac{1}{10\sqrt{\varepsilon}}$ elements. I.e., we assume $A = \bigcup_{i=1}^{g} A_i$ and $B = \bigcup_{i=1}^{g} B_i$ such that $g = O(\sqrt{\varepsilon} n)$ and $|A_i|, |B_i| \le \frac{1}{10\sqrt{\varepsilon}}$. For each $i, j \in [g]$, we use the algorithm mentioned above to calculate two vectors $\sum_{a \in A_i} \mathbb{A}_A(a)$ and $\sum_{b \in B_j} \mathbb{A}_B(b)$. We write $\mathcal{M}_A \in \mathbb{R}^{2^T \times g}$ to denote the matrix

$$\left[\sum_{a \in A_1} \mathbb{A}_A(a), \sum_{a \in A_2} \mathbb{A}_A(a), \cdots, \sum_{a \in A_g} \mathbb{A}_A(a) \right]$$

and $\mathcal{M}_B \in \mathbb{R}^{2^T \times g}$ to denote the matrix

$$\left[\sum_{b \in B_1} \mathbb{A}_B(b), \sum_{b \in B_2} \mathbb{A}_B(b), \cdots, \sum_{b \in B_g} \mathbb{A}_B(b) \right].$$

Since $2^T \le (\sqrt{\varepsilon} n)^{0.1} \le O(g^{0.1})$, we can use the rectangular matrix multiplication algorithm in Theorem 7.2.1 to calculate $\mathcal{M}_A^T \mathcal{M}_B$ in $O(g^2 \cdot \mathrm{polylog}(g)) = O(\varepsilon n^2 \,\mathrm{polylog}(n))$ time. We repeat this procedure for $O(\log n)$ times. For any $i, j \in [g]$, by standard concentration bounds, with probability at least $1 - \mathrm{poly}(n)$, there exist $a \in A_i$ and $b \in B_j$ such that $F(a,b) = 1$ if and only if the majority of the $O(\log n)$ repetitions satisfies $(\mathcal{M}_A^T \mathcal{M}_B)_{i,j} > 0$. Applying union bound again over all $i, j \in [g]$, we can now solve F-Satisfying-Pair$_n$ by checking whether there exist i and j such that the majority of the $O(\log n)$ repetitions satisfies $(\mathcal{M}_A^T \mathcal{M}_B)_{i,j} > 0$. The overall algorithm runs in $O(\varepsilon n^2 \cdot \operatorname{polylog}(n))$ time and succeeds with high probability, as stated. □

7.3.1 A New Algorithm for Approximate Max-IP

The first application of Theorem 7.1.2 is to use the Goldwasser-Sisper AM protocol [92] for approximating set size to obtain a new algorithm for approximating Max-IP.

We first need the following adaption of [92], which has a better dependence on ε.

Reminder of Lemma 7.1.4 *There is an $\mathsf{AM}^{\mathsf{cc}}$ protocol for Gap-Inner-Product$_d$ with error ε and communication complexity*

$$\log \binom{d}{\leq O(\log \varepsilon^{-1})}.$$

Proof. Recall that $x, y \in \{0,1\}^d$ are the inputs hold by Alice and Bob respectively.

Let $X = \{i \mid x_i = 1\}$ and $Y = \{i \mid y_i = 1\}$. The problem is equivalent to determine whether $|X \cap Y| \geq 2\tau$ or $|X \cap Y| \leq \tau$. Here we give an $\mathsf{AM}^{\mathsf{cc}}$ communication protocol with error ε and communication complexity $\log \binom{d}{\leq O(\log \varepsilon^{-1})}$.

In the communication protocol, Alice and Bob first generate i.i.d. random variables $p_i \sim \text{Pois}(k/\tau)$ for each $i \in [d]$, for a parameter $k = \Theta(\log(1/\varepsilon))$ to be determined later. When $|X \cap Y| \geq 2\tau$, Merlin finds an arbitrary set $S \subseteq X \cap Y$ of size $O(k)$ such that $\sum_{i \in S} p_i \geq 1.6k$, and then sends it to Alice and Bob. Upon receiving S, Alice (Bob) decides to accept or reject by checking whether $S \subseteq X$ ($S \subseteq Y$) and $\sum_{i \in S} p_i \geq 1.6k$. The communication complexity of this protocol is upper bounded by $\log \binom{d}{\leq O(\log \varepsilon^{-1})}$ since $|S| \leq 1.6k = O(\log(1/\varepsilon))$.

Now we prove the correctness by considering the following two cases.

Case 1: $|X \cap Y| \geq 2\tau$. For this case, we have

$$\sum_{i \in X \cap Y} p_i \sim \text{Pois}(|X \cap Y| \cdot k/\tau) \succeq \text{Pois}(2k).$$

Thus by Lemma 7.2.3, with probability at least $1 - e^{\Omega(k)}$, $\sum_{i \in X \cap Y} p_i \geq 1.6k$. Since for each $p_i > 0$ we must have $p_i \geq 1$, with probability at least $1 - e^{\Omega(k)}$, there exists a set $S \subseteq X \cap Y$ of size $O(k)$ such that $\sum_{i \in S} p_i \geq 1.6k$.

Case 2: $|X \cap Y| \leq \tau$. For this case, we have

$$\sum_{i \in X \cap Y} p_i \sim \text{Pois}(|X \cap Y| \cdot k/\tau) \preceq \text{Pois}(k).$$

Thus by Lemma 7.2.3, with probability at least $1 - e^{\Omega(k)}$, $\sum_{i \in X \cap Y} p_i \leq 1.2k$. When both Alice and Bob accept, it must be the case that $S \subseteq X \cap Y$ and $\sum_{i \in S} p_i \geq 1.6k$. However when $|X \cap Y| \leq \tau$, with probability at least $1 - e^{\Omega(k)}$, $\sum_{i \in X \cap Y} p_i \leq 1.2k$. Thus there is no S such that both Alice and Bob accept, with probability at least $1 - e^{-\Omega(k)}$.

The lemma follows by setting k to be a large enough multiple of $\log(1/\varepsilon)$.

□

By Theorem 7.1.2 and the above lemma, Corollary 7.1.5 follows from a binary search over τ.

Reminder of Corollary 7.1.5 *There is an algorithm for computing a 2-approximation to $\mathsf{Max}\text{-}\mathsf{IP}_{n,c\log n}$, which runs in $n^{2-1/O(\log c)}$ time.*

7.4 Consequence of Fast $\mathsf{AM}^{\mathsf{cc}}$ Protocols for LCS and Edit-Distance

Next we discuss the consequences of LCS and Edit-Distance having efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols. We first introduce some classical notations about the communication complexity classes (see [38, 94]). We say a function family $F = \{F_d : \{0,1\}^d \times \{0,1\}^d \to \{0,1,\perp\}\}_{d \in \mathbb{N}}$ is in $\mathsf{AM}^{\mathsf{cc}}$ if $\mathsf{AM}^{\mathsf{cc}}(F_d) = \text{polylog}(d)$ (we use $\mathsf{AM}^{\mathsf{cc}}(F_d)$ to denote the $\mathsf{AM}^{\mathsf{cc}}$ communication complexity for F_d with error $1/3$).

We also say F is $\mathsf{AM}^{\mathsf{cc}}_{\mathsf{eff}}$ if for all $d \in \mathbb{N}$, F_d admits a computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocol with error $1/3$ and complexity $\text{polylog}(d)$.

Now we prove the consequence of a function family $F \in \mathsf{AM}^{\mathsf{cc}}_{\mathsf{eff}}$.

Corollary 7.4.1 (Consequence of $F \in \mathsf{AM}^{\mathsf{cc}}_{\mathsf{eff}}$). *Let $F = \{F_d : \{0,1\}^d \times \{0,1\}^d \to \{0,1,\perp\}\}_{d \in \mathbb{N}}$ be a partial function family. If $F \in \mathsf{AM}^{\mathsf{cc}}_{\mathsf{eff}}$, then there is an $n^2/2^{\log^{1-\delta} n}$ time algorithm for $F_{\text{polylog}(n)}$-Satisfying-Pair$_n$, for any constant $\delta > 0$.*

Proof. By standard repetition arguments, there exists an $\mathsf{AM}^{\mathsf{cc}}$ communication protocol with communication complexity $\text{polylog}(d)\log(1/\varepsilon)$ and failure probability $1-\varepsilon$. In order to invoke Theorem 7.1.2 we need to make sure

$$2^{\text{polylog}(d)\log(1/\varepsilon)} = 2^{\text{polyloglog}(n)\log(1/\varepsilon)} < n^{0.1},$$

and thus we can set $\varepsilon = 2^{-\log^{1-\delta/2} n}$. For this choice of ε we will then get an $n^2/2^{\log^{1-\delta/2} n} \cdot \text{polylog}(n) \le n^2/2^{\log^{1-\delta} n}$ time algorithm for $F_{\text{polylog}(n)}$-$\mathsf{Satisfying\text{-}Pair}_n$, which completes the proof. □

Recall that in $\mathsf{LCS}^{\mathsf{cc}}_d$ ($\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}_d$), Alice and Bob hold strings $x, y \in \{0,1\}^d$ respectively, and are given an integer τ. Their goal is to decide whether $\mathsf{LCS}(x,y) \ge \tau$ ($\mathsf{Edit\text{-}Distance}(x,y) \ge \tau$). Now we are ready to prove Theorem 7.1.7.

Reminder of Theorem 7.1.7 *If $\mathsf{LCS}^{\mathsf{cc}}_d$ admits computationally efficient $\mathsf{AM}^{\mathsf{cc}}$ protocols with complexity* $\text{polylog}(d)$, *then Formula-SAT of polynomial-size formulas admits a $2^{n-n^{1-\delta}}$ time algorithm for any constant $\delta > 0$. The same holds for Edit-Dist$^{\mathsf{cc}}$ in place of LCS$^{\mathsf{cc}}$.*

We will only discuss $\mathsf{LCS}^{\mathsf{cc}}$ here, the proof for $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ follows exactly the same. We first introduce the reduction from [10] (see also [6]).

Theorem 7.4.2 (Implicit in [10]). *For a given formula $\mathcal{F}$ with n input variables and size s, let $a \in \{0,1\}^{n/2}$ be an assignment to first $n/2$ variables in $\mathcal{F}$ and $b \in \{0,1\}^{n/2}$ be an assignment to last $n/2$ variables in $\mathcal{F}$. There exists an algorithm $\mathbb{A}$ which outputs $G(a) \in \{0,1\}^{\text{poly}(s)}$ and $\overline{G}(b) \in \{0,1\}^{\text{poly}(s)}$ such that for a fixed integer Y (Y depends on $\mathcal{F}$),*

- $\mathsf{LCS}(G(a), \overline{G}(b)) = Y$ *if* $a \odot b$ *is a satisfying assignment to* $\mathcal{F}$*;*
- $\mathsf{LCS}(G(a), \overline{G}(b)) \leq Y - 1$ *if* $a \odot b$ *is not a satisfying assignment to* $\mathcal{F}$*.*

Proof of Theorem 7.1.7. For a given formula $\mathcal{F}$ of size $s = \text{poly}(n)$, we first enumerate all $2^{n/2}$ possible assignments to first $n/2$ variables in $\mathcal{F}$ and all possible assignments to last $n/2$ variables in $\mathcal{F}$. For each $a \in \{0,1\}^{n/2}$ corresponding to an assignment to first $n/2$ variables in $\mathcal{F}$ and $b \in \{0,1\}^{n/2}$ corresponding to an assignment to last $n/2$ variables in $\mathcal{F}$, we calculate $G(a)$ and $\overline{G}(b)$ using Theorem 7.4.2. Note that all $G(a)$'s and $\overline{G}(b)$'s have length $\text{poly}(s) = \text{poly}(n)$.

Now suppose $\mathsf{LCS}^{\mathsf{cc}} \in \mathsf{AM}^{\mathsf{cc}}_{\mathsf{eff}}$ for $\tau = Y$. Applying Corollary 7.4.1 with all possible $G(a)$'s and $\overline{G}(b)$'s, we can solve Formula-SAT in $2^{n-n^{1-\delta}}$ time for any constant $\delta > 0$. □

7.5 Probabilistic Rank and OV Algorithms in [13]

In this section we explain [23]'s observation with the OV algorithm in [13] as an example. [13] derived an $n^{2-1/O(\log c)}$ time algorithm for $\mathsf{OV}_{n,c\log n}$ with the classical polynomial methods [144, 157] (i.e., the probabilistic polynomials for AND and OR). Here we demonstrate that their results ultimately rely on the fact that the SET-DISJOINTNESS matrix has a low probabilistic rank.

For the rest of the section we will always work with $\mathbb{F}_2$-matrices and vectors. A probabilistic matrix $\mathcal{M}$ is a distribution of matrices over $\mathbb{F}_2^{n\times n}$. We say a probabilistic matrix $\mathcal{M}$ computes a matrix $A \in \mathbb{F}_2^{n\times n}$ with error ε, if for every entry $(i,j) \in [n]\times[n]$,

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

We say a probabilistic matrix $\mathcal{M}$ has rank r, if the maximum rank of matrices from $\mathcal{M}$ is r. We define the ε-probabilistic rank of a matrix A to be the minimum rank of a probabilistic matrix $\mathcal{M}$ computing A with error at most ε.

Consider the following SET-DISJOINTNESS matrix $M^{\mathsf{DISJ}} \in \mathbb{F}_2^{2^d\times 2^d}$. We use subsets of $[d]$ to index rows and columns of M^{DISJ}, and for subsets $S, T \subseteq [d]$, $M^{\mathsf{DISJ}}_{S,T} =$

$\mathsf{DISJ}(S,T)$ ($\mathsf{DISJ}(S,T)$ is the indicator function that whether $S \cap T = \emptyset$).

The following fact is implicit in [13]:

Proposition 7.5.1 ([13]). *The ε-probabilistic rank of M^{DISJ} is $r \le \binom{d}{\le O(\log \varepsilon^{-1})}$. Moreover, let $\mathcal{M}$ be the corresponding probabilistic matrix and $M \sim \mathcal{M}$, there are mappings $\phi_X^M, \phi_Y^M : [d] \to \mathbb{F}_2^r$ which can be computed in* poly(r) *time, such that $\phi_X^M(S) \cdot \phi_Y^M(T) = M_{S,T}$ for all $S, T \subseteq [d]$.*

Now, we derive the OV algorithm in [13] only using the above proposition.

Theorem 7.5.2 ([13]). *There is an $n^{2-1/O(\log c)}$ time algorithm for $\mathsf{OV}_{n,c\log n}$.*

Proof. Our presentation here will be different from [13] for simplicity. Let $d = c \log n$, and $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ be the given OV instance. We say $\mathsf{OV}(A,B) = 1$ if there is an orthogonal pair in $A \times B$, and $\mathsf{OV}(A,B) = 0$ otherwise.

We first set $\log \varepsilon^{-1} = \Theta(\log n / \log c)$ so that after applying Proposition 7.5.1, $r \le n^{0.1}$.

Let $m = \sqrt{\varepsilon^{-1}}/10$, and assume m divides n for simplicity. We partition A and B into $g = n/m$ groups of vectors, each of size m. Let them be $A_1, A_2, \dots, A_g$ and $B_1, B_2, \dots, B_g$ correspondingly.

Let $\mathcal{M}$ be the ε-error probabilistic matrix for M^{DISJ}. And $M \sim \mathcal{M}$ be a sample from it. We also draw two random vectors $u, v \in \mathbb{F}_2^m$.

Fix two groups A_i and B_j, consider the following quantity

$$\sum_{k=1}^{m} \sum_{\ell=1}^{m} \mathsf{DISJ}((A_i)_k, (B_j)_\ell)) \cdot u_k \cdot v_\ell. \tag{7.1}$$

Note that this is just $u^T W v$, where $W_{k,\ell} = \mathsf{DISJ}((A_i)_k, (B_j)_\ell))$. Since u and v are two random vectors, when $\mathsf{OV}(A_i, B_j) = 0$, (7.1) is always zero as W is the all-zero matrix; and otherwise (7.1) is 1 with probability at least $1/4$.

We are going to approximate (7.1) by the following

$$
\begin{aligned}
&\sum_{k=1}^{m}\sum_{\ell=1}^{m} M_{(A_i)_k,(B_j)_\ell} \cdot u_k \cdot v_\ell && (7.2)\\
=&\sum_{k=1}^{m}\sum_{\ell=1}^{m}\{\phi_X^M((A_i)_k)\cdot\phi_Y^M((B_j)_\ell)\}\cdot u_k\cdot v_\ell\\
=&\left\{\sum_{k=1}^{m}\phi_X^M((A_i)_k)\cdot u_k\right\}\cdot\left\{\sum_{\ell=1}^{m}\phi_Y^M((B_j)_\ell)\cdot v_\ell\right\}. && (7.3)
\end{aligned}
$$

Note that there are $m^2 = \varepsilon^{-1}/100$ entries of M are considered in (7.2). So by a union bound and the fact that $\mathcal{M}$ computes M^{DISJ} with error ε. With probability 0.99, (7.2) and (7.1) are equal, over M, u and v.

Let $U_i = \sum_{k=1}^{m} \phi_X^M((A_i)_k)\cdot u_k$ and $V_j = \sum_{\ell=1}^{m}\phi_Y^M((B_j)_\ell)\cdot v_\ell$. Then by (7.3), (7.2) equals $U_i \cdot V_j$. Putting everything together, for each $(i,j) \in [g]\times[g]$, we have: (1) when $\mathsf{OV}(A_i, B_j) = 0, \Pr_{M,u,v}[U_i \cdot V_j = 1] \le 0.01$; (2) when $\mathsf{OV}(A_i, B_j) = 1, \Pr_{M,u,v}[U_i \cdot V_j = 1] \ge 0.24$.

For a tuple of M, u and v, we can compute $U_i \cdot V_j$ for all i, j via a rectangular matrix multiplication between two matrices of size $g \times r$ and $r \times g$. By Theorem 7.2.1, it can be solved in $g^2 \cdot \text{polylog}(g)$ time. Repeating this for $T = 1000 \log n$ times, for each i, j, we record how many times we get $U_i \cdot V_j = 1$ as $T_{i,j}$. The algorithm outputs yes if there are i, j such that $T_{i,j} > T \cdot 0.15$, and no otherwise.

By a simple Chernoff bound, one can show the algorithm solves the OV instance with high probability, and the running time is $(n/m)^2 \cdot \text{polylog}(n) = \varepsilon n^2 \cdot \text{polylog}(n) = n^{2-1/O(\log c)}$. □

Remark 7.5.3. *From the above proof, it is easy to see that for any function F whose corresponding communication matrix M_F admits a low probabilistic rank with an efficient decomposition as in Proposition 7.5.1. A fast F-$\mathsf{Satisfying}$-Pair algorithm can be derived similarly.*

7.6 Conditional Lower Bounds for Computational-Efficient $\mathsf{PH}^{\mathsf{cc}}$ Protocols

In this section we prove Theorem 7.1.9 (restated below).

Reminder of Theorem 7.1.9 *Under the following hypothesis, $\mathsf{LCS}_d^{\mathsf{cc}}$ and $\mathsf{Edit\text{-}Dist}_d^{\mathsf{cc}}$ do not admit computationally efficient $\mathsf{PH}^{\mathsf{cc}}$ protocols with complexity* $\text{polylog}(d)$*:*

- *There is a constant $\delta > 0$ such that Formula-SAT of polynomial-size formulas requires $2^{n-n^{1-\delta}}$ time.*

We first recall the definition of a $\mathsf{PH}^{\mathsf{cc}}$ protocol from [38].[5]

Definition 7.6.1 ([38]). A PH communication protocol ($\mathsf{PH}^{\mathsf{cc}}$) Π for a function $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1, \perp\}$ proceeds as follows:

- Alice holds input $x \in \mathcal{X}$ and Bob holds input $y \in \mathcal{Y}$.
- For a constant $k \in \mathbb{N}$, there are $2k$ provers $P_1, P_2, \dots, P_{2k}$.
- For each $i \in [2k]$, the prover P_i sends both Alice and Bob a proof $z_i \in \{0, 1\}^{m_i}$.
- We use $A(x, z_1, z_2, \dots, z_{2k})$ (resp. $B(y, z_1, z_2, \dots, z_{2k})$) to be the indicator function that whether Alice (resp. Bob) accepts the proof sequence $z_1, z_2, \dots, z_{2k}$, given the input x (resp. y).
- If $F(x, y) = 1$, then

$$\underset{z_1 \in \{0,1\}^{m_1}}{\exists} \; \underset{z_2 \in \{0,1\}^{m_2}}{\forall} \cdots \underset{z_{2k-1} \in \{0,1\}^{m_{2k-1}}}{\exists} \; \underset{z_{2k} \in \{0,1\}^{m_{2k}}}{\forall} \left[A(x, z_1, \dots, z_{2k}) \wedge B(y, z_1, \dots, z_{2k}) \right].$$

- If $F(x, y) = 0$, then

$$\underset{z_1 \in \{0,1\}^{m_1}}{\forall} \; \underset{z_2 \in \{0,1\}^{m_2}}{\exists} \cdots \underset{z_{2k-1} \in \{0,1\}^{m_{2k-1}}}{\forall} \; \underset{z_{2k} \in \{0,1\}^{m_{2k}}}{\exists} \left[\neg A(x, z_1, \dots, z_{2k}) \vee \neg B(y, z_1, \dots, z_{2k}) \right].$$

[5] See also [94] for a more recent reference.

Moreover, we say the protocol is *computationally efficient* if Alice and Bob's decision functions (the functions A and B) can be computed in polynomial-time w.r.t. their input lengths. The communication complexity of Π is simply the total number of proof bits from all provers, i.e. $\sum_{i=1}^{2k} m_i$.

Theorem 7.6.2. *Let $F : \mathcal{X} \times \mathcal{Y} \to \{0, 1, \perp\}$ be a partial function. Suppose there is a computationally efficient PH^{cc} protocol for F with communication complexity T and number of provers $2k$. If $\varepsilon > 0$ satisfies*

$$\binom{2^T}{\le 10 \cdot (2 \cdot T)^{2k} \cdot \log(1/\varepsilon)} \le n^{0.1},$$

then there is an $O(\varepsilon \cdot n^2 \cdot \mathrm{polylog}(n))$ time algorithm for F-$\mathsf{Satisfying\text{-}Pair}_n$.

Proof. By Remark 7.5.3, we only need to argue the probabilistic rank of the communication matrix M_F is small, which is already established in [145]. In the following, we follow the proof structure of Theorem 7.5.2.

Recall that $T = \sum_{i=1}^{2k} m_i$. Let $M = 2^T$ and $z \in \{0, 1\}^M$. With a natural bijection between $[M]$ and $\{0, 1\}^T$, we can use a string $x \in \{0, 1\}^T$ to index z.

We define the following AC^0 function

$$F(z) := \vee_{x_1 \in \{0,1\}^{m_1}} \wedge_{x_2 \in \{0,1\}^{m_2}} \cdots \vee_{x_{2k-1} \in \{0,1\}^{m_{2k-1}}} \wedge_{x_{2k} \in \{0,1\}^{m_{2k}}} z_{(x_1 \circ x_2 \circ \ldots x_{2k})},$$

where $x_1 \circ x_2 \circ \ldots x_{2k}$ means the concatenation of the x_i's.

By standard polynomial method [144, 157], there is a

$$D = 10 \cdot (2 \cdot T)^{2k} \cdot \log(1/\varepsilon)$$

degree, ε-error probabilistic polynomial for the function F. Formally, there is an efficiently-sampleable distribution on D-degree polynomials $\mathcal{P}_\varepsilon$, such that for all $z \in \{0, 1\}^M$,

$$\Pr_{P \sim \mathcal{P}_\varepsilon} [P(z) = F(z)] \ge 1 - \varepsilon.$$

In particular, this means the ε-probabilistic rank of M_F is smaller than

$$r \leq \binom{M}{\leq D},$$

and the corresponding distribution on low-rank matrices has efficiently computable decomposition as specified in Proposition 7.5.1. Then we can proceed exactly as in Theorem 7.5.2. □

Finally, we are ready to prove Theorem 7.1.9.

Proof of Theorem 7.1.9. We proceed similarly as in Theorem 7.1.7. Below we only discuss $\mathsf{LCS}^{\mathsf{cc}}$, the proof for $\mathsf{Edit\text{-}Dist}^{\mathsf{cc}}$ is exactly the same.

For a given formula $\mathcal{F}$ of size $s = \mathrm{poly}(n)$, we first enumerate all $2^{n/2}$ possible assignments to first $n/2$ variables in $\mathcal{F}$ and all possible assignments to last $n/2$ variables in $\mathcal{F}$. For each $a \in \{0,1\}^{n/2}$ corresponding to an assignment to first $n/2$ variables in $\mathcal{F}$ and $b \in \{0,1\}^{n/2}$ corresponding to an assignment to last $n/2$ variables in $\mathcal{F}$ we calculate $G(a)$ and $\overline{G}(b)$ using Theorem 7.4.2. We can assume that both $G(a)$ and $\overline{G}(b)$ have length $\ell = \mathrm{poly}(s) = \mathrm{poly}(n)$.

Now suppose $\mathsf{LCS}^{\mathsf{cc}}$ with $\tau = Y$ has a computationally efficient $\mathsf{PH}^{\mathsf{cc}}$ protocol with $T = \mathrm{polylog}(\ell) = \mathrm{polylog}(n)$ and the number of provers $2k$ (k is a constant). We set ε such that

$$\binom{2^T}{\leq 10 \cdot (2 \cdot T)^{2k} \cdot \log(1/\varepsilon)} \leq 2^{n/20}.$$

The above can be satisfied if

$$2^{T \cdot 10 \cdot (2 \cdot T)^{2k} \cdot \log(1/\varepsilon)} = 2^{\mathrm{polylog}(n) \cdot \log(1/\varepsilon)} \leq 2^{n/20}.$$

Finally, we set $\varepsilon = 2^{-n^{1-\delta/2}}$ for the $\delta > 0$ in the hypothesis. Now we can complete the proof by applying Theorem 7.6.2. □

Bibliography

[1] Scott Aaronson and Andris Ambainis. Quantum search of spatial regions. *Theory of Computing*, 1(1):47–79, 2005.

[2] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1):2:1–2:54, 2009.

[3] Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017*, pages 11:1–11:26, 2017.

[4] Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1256–1271, 2016.

[5] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.

[6] Amir Abboud and Karl Bringmann. Tighter connections between formula-sat and shaving logs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 8:1–8:18, 2018.

[7] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. More consequences of falsifying SETH and the orthogonal vectors conjecture. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 253–266, 2018.

[8] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 477–486, 2016.

[9] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015.

[10] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016.

[11] Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018*, pages 35:1–35:14, 2018.

[12] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 25–36, 2017.

[13] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 218–230, 2015.

[14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 434–443, 2014.

[15] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014*, pages 39–51, 2014.

[16] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 41–50, 2015.

[17] Pankaj K Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete & Computational Geometry*, 6(3):407–422, 1991.

[18] Thomas Dybdahl Ahle, Rasmus Pagh, Ilya P. Razenshteyn, and Francesco Silvestri. On the complexity of inner product similarity join. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016*, pages 151–164, 2016.

[19] Tatsuya Akutsu and Magnús M Halldórsson. On the approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233(1-2):33–50, 2000.

[20] Tatsuya Akutsu, Takeyuki Tamura, Avraham A Melkman, and Atsuhiro Takasu. On the complexity of finding a largest common subtree of bounded degree. *Theoretical Computer Science*, 590:2–16, 2015.

[21] Josh Alman. An illuminating algorithm for the light bulb problem. In *2nd Symposium on Simplicity in Algorithms, SOSA 2019*, pages 2:1–2:11, 2019.

[22] Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 467–476, 2016.

[23] Josh Alman and R. Ryan Williams. Probabilistic rank and matrix rigidity. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 641–652, 2017.

[24] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 136–150, 2015.

[25] Noga Alon. Perturbed identity matrices have high rank: Proof and applications. *Combinatorics, Probability and Computing*, 18(1-2):3–15, 2009.

[26] Noga Alon, Troy Lee, and Adi Shraibman. The cover number of a matrix and its algorithmic applications. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 28. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[27] Noga Alon, Troy Lee, Adi Shraibman, and Santosh Vempala. The approximate rank of a matrix and its algorithmic applications: approximate rank. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 675–684. ACM, 2013.

[28] Andris Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37(1):210–239, 2007.

[29] 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 Journal on Computing*, 39(6):2513–2530, 2010.

[30] Andris Ambainis, Leonard J. Schulman, Amnon Ta-Shma, Umesh V. Vazirani, and Avi Wigderson. The quantum communication complexity of sampling. *SIAM J. Comput.*, 32(6):1570–1585, 2003.

[31] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[32] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. *arXiv preprint arXiv:1806.09823*, 2018.

[33] Tom M. Apostol. *Introduction to analytic number theory.* Springer Science & Business Media, 2013.

[34] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradžev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics—Doklady*, 11(5):1209–1210, 1970.

[35] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[36] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[37] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[38] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 337–347, 1986.

[39] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 81:1–81:13, 2016.

[40] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, pages 457–466, 2016.

[41] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018.

[42] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4311–4321, 2017.

[43] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 267–280, 2018.

[44] Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory of Computing*, 8(1):69–94, 2012.

[45] Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 519–528. ACM, 2011.

[46] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[47] Djamal Belazzougui and Mathieu Raffinot. Approximate regular expression matching with multi-strings. *Journal of Discrete Algorithms*, 18:14–21, 2013.

[48] Eli Ben-Sasson and Emanuele Viola. Short pcps with projection queries. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 163–173, 2014.

[49] Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, STOC 1976*, pages 220–230, 1976.

[50] Lasse Bergroth, Harri Hakonen, and Timo Raita. New approximation algorithms for longest common subsequences. In *String Processing and Information Retrieval: A South American Symposium, SPIRE 1998, Santa Cruz de la Sierra Bolivia, September 9-11, 1998*, pages 32–40, 1998.

[51] Philip Bille and Mikkel Thorup. Faster regular expression matching. In *International Colloquium on Automata, Languages, and Programming*, pages 171–182. Springer, 2009.

[52] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electr. Notes Theor. Comput. Sci.*, 322:51–67, 2016.

[53] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 312–321, 1999.

[54] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 661–670, 2014.

[55] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 307–318, 2017.

[56] Karl Bringmann and Marvin Kunnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.

[57] Karl Bringmann and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1216–1235, 2018.

[58] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

[59] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science, FOCS 1999*, pages 358–368, 1999.

[60] Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proc. of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 63–68, 1998.

[61] Harry Buhrman and Ronald de Wolf. Communication complexity lower bounds by polynomials. In *Computational Complexity, 16th Annual IEEE Conference on, 2001.*, pages 120–130. IEEE, 2001.

[62] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009*, pages 75–85, 2009.

[63] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016.

[64] Timothy M. Chan. A (slightly) faster algorithm for klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010.

[65] Timothy M. Chan. The art of shaving logs. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, page 231, 2013.

[66] Timothy M. Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 212–217, 2015.

[67] Timothy M. Chan. Orthogonal range searching in moderate dimensions: k-d trees and range trees strike back. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 27:1–27:15, 2017.

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

[69] Moses Charikar, Piotr Indyk, and Rina Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *International Colloquium on Automata, Languages, and Programming*, pages 451–462. Springer, 2002.

[70] Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 14:1–14:45, 2018.

[71] Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP = PSPACE. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1–20, 2019.

[72] Lijie Chen and Ruosong Wang. Classical algorithms from quantum and arthur-merlin communication protocols. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 23:1–23:20, 2019.

[73] Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 21–40, 2019.

[74] Moon-Jung Chung. $O(n^{2.55})$ time algorithms for the subgraph homeomorphism problem on trees. *J. Algorithms*, 8(1):106–112, 1987.

[75] Vašek Chvátal, David A Klarner, and Donald Ervin Knuth. *Selected combinatorial research problems.* Computer Science Department, Stanford University, 1972.

[76] Richard Cole, Lee-Ad Gottlieb, and Moshe Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 91–100, 2004.

[77] Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982.

[78] Svyatoslav Covanov and Emmanuel Thomé. Fast integer multiplication using \goodbreak generalized fermat primes. *Math. Comput.*, 88(317):1449–1477, 2019.

[79] Maxime Crochemore, Costas S Iliopoulos, Yoan J Pinzon, and James F Reid. A fast and practical bit-vector algorithm for the longest common subsequence problem. *Information Processing Letters*, 80(6):279–285, 2001.

[80] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 22:1–22:15, 2017.

[81] Mina Dalirrooyfard, Thuy Duong Vuong, and Virginia Vassilevska Williams. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In *STOC 2019, to appear.*

[82] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pages 253–262, 2004.

[83] Roee David, Karthik C. S., and Bundit Laekhanukit. On the complexity of closest pair via polar-pair of point-sets. In *34th International Symposium on Computational Geometry, SoCG 2018*, pages 28:1–28:15, 2018.

[84] J Boutet de Monvel. Extensive simulations for longest common subsequences. *The European Physical Journal B-Condensed Matter and Complex Systems*, 7(2):293–308, 1999.

[85] Ronald de Wolf. A note on quantum algorithms and the minimal degree of ε-error polynomials for symmetric functions. *Quantum Information & Computation*, 8(10):943–950, 2008.

[86] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms*, 25(1):19–51, 1997.

[87] Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[88] Martin Fürer. Faster integer multiplication. *SIAM J. Comput.*, 39(3):979–1005, 2009.

[89] Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2162–2181, 2017.

[90] Phillip B Gibbons, Richard M Karp, Gary L Miller, and Danny Soroker. Subtree isomorphism is in random NC. *Discrete Applied Mathematics*, 29(1):35–62, 1990.

[91] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. Conditional lower bounds for space/time tradeoffs. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017*, pages 421–436, 2017.

[92] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research*, 5:73–90, 1989.

[93] Mika Göös, Toniann Pitassi, and Thomas Watson. Zero-information protocols and unambiguity in arthur-merlin communication. *Algorithmica*, 76(3):684–719, 2016.

[94] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018.

[95] Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 223–234, 2010.

[96] Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. *Discrete Applied Mathematics*, 212:96–103, 2016.

[97] Prahladh Harsha, Adam R. Klivans, and Raghu Meka. An invariance principle for polytopes. *J. ACM*, 59(6):29:1–29:25, 2012.

[98] David Harvey, Joris van der Hoeven, and Grégoire Lecerf. Even faster integer multiplication. *J. Complexity*, 36:1–30, 2016.

[99] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[100] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 21–30, 2015.

[101] Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional hardness for sensitivity problems. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017*, pages 26:1–26:31, 2017.

[102] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[103] T. S. Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-probe lower bounds for the partial match problem. *J. Comput. Syst. Sci.*, 69(3):435–447, 2004.

[104] Tao Jiang and Bala Ravikumar. A note on the space complexity of some decision problems for finite automata. *Information Processing Letters*, 40(1):25–31, 1991.

[105] Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[106] Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 1283–1296, 2018.

[107] Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, pages 17:1–17:16, 2019.

[108] Sanjeev Khanna, Rajeev Motwani, and F Frances Yao. *Approximation algorithms for the largest common subtree problem.* Citeseer, 1995.

[109] Samir Khuller and Yossi Matias. A simple randomized sieve algorithm for the closest-pair problem. *Inf. Comput.*, 118(1):34–37, 1995.

[110] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *18th Annual IEEE Conference on Computational Complexity, CCC 2003*, pages 118–134, 2003.

[111] James R Knight and Eugene W Myers. Approximate regular expression pattern matching with concave gap penalties. *Algorithmica*, 14(1):85–121, 1995.

[112] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 1272–1287, 2016.

[113] Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. *ACM Trans. Algorithms*, 14(4):42:1–42:15, 2018.

[114] Ilan Kremer. *Quantum communication.* Citeseer, 1995.

[115] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 21:1–21:15, 2017.

[116] Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–398, 2009.

[117] Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In *ICALP 2019, to appear.*

[118] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1236–1252, 2018.

[119] Andrzej Lingas. An application of maximum bipartite c-matching to subtree isomorphism. In *CAAP'83, Trees in Algebra and Programming, 8th Colloquium, L'Aquila, Italy, March 9-11, 1983, Proceedings*, pages 284–299, 1983.

[120] Andrzej Lingas and Marek Karpinski. Subtree isomorphism is NC reducible to bipartite perfect matching. *Information Processing Letters*, 30(1):27–32, 1989.

[121] Satyanarayana V. Lokam. Spectral methods for matrix rigidity with applications to size-depth trade-offs and communication complexity. *J. Comput. Syst. Sci.*, 63(3):449–473, 2001.

[122] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202, 2017.

[123] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[124] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980.

[125] Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[126] Jiří Matoušek. Range searching with efficient hierarchical cuttings. In *Proceedings of the Eighth Annual Symposium on Computational Geometry, SOCG 1992*, pages 276–285, 1992.

[127] David W. Matula. An algorithm for subtree identification. *SIAM Review*, 10(2):273–274, 1968.

[128] David W. Matula. Subtree isomorphism in $O(n^{5/2})$. In B. Alspach, P. Hell, and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91 – 106. Elsevier, 1978.

[129] Kurt Mehlhorn and Erik Meineche Schmidt. Las vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 330–337, 1982.

[130] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

[131] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis.* Cambridge university press, 2005.

[132] Jorge Moraleda. Gregory shakhnarovich, trevor darrell and piotr indyk: Nearest-neighbors methods in learning and vision. theory and practice. *Pattern Anal. Appl.*, 11(2):221–222, 2008.

[133] Eugene Myers, Paulo Oliva, and Katia Guimarães. Reporting exact and approximate regular expression matches. In *Combinatorial pattern matching*, pages 91–103. Springer, 1998.

[134] Eugene W Myers and Webb Miller. Approximate matching of regular expressions. *Bulletin of mathematical biology*, 51(1):5–37, 1989.

[135] Gene Myers. A four russians algorithm for regular expression pattern matching. *Journal of the ACM (JACM)*, 39(2):432–448, 1992.

[136] Gonzalo Navarro. Approximate regular expression searching with arbitrary integer weights. *Nord. J. Comput.*, 11(4):356–373, 2004.

[137] Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. Fooling polytopes. *CoRR*, abs/1808.04035, 2018.

[138] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 54(5):23, 2007.

[139] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proc. of the 49th FOCS*, pages 414–423. IEEE, 2008.

[140] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610, 2010.

[141] Mihai Patrascu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. *SIAM J. Comput.*, 39(2):730–741, 2009.

[142] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*, pages 1065–1075, 2010.

[143] Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *J. Comput. Syst. Sci.*, 33(1):106–123, 1986.

[144] Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[145] Alexander A Razborov. On rigid matrices. *preprint*, 1989.

[146] Ilya P. Razenshteyn. *High-dimensional similarity search and sketching: algorithms and hardness.* PhD thesis, Massachusetts Institute of Technology, Cambridge, USA, 2017.

[147] Ben Reichardt. Reflections for quantum query algorithms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 560–569, 2011.

[148] Steven W Reyner. An analysis of a good algorithm for the subtree problem. *SIAM Journal on Computing*, 6(4):730–732, 1977.

[149] Ronald Linn Rivest. *Analysis of Associative Retrieval Algorithms.* PhD thesis, Stanford University, Stanford, CA, USA, 1974. AAI7420230.

[150] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC 2013*, pages 515–524, 2013.

[151] Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 1260–1268, 2018.

[152] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 183–192, 2010.

[153] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[154] Rocco A. Servedio and Li-Yang Tan. Fooling intersections of low-weight halfspaces. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 824–835, 2017.

[155] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[156] Ron Shamir and Dekel Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267–280, 1999.

[157] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987.

[158] Zhao Song, David P. Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2772–2789, 2019.

[159] Philip Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971.

[160] Larry J Stockmeyer and Albert R Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1973.

[161] Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:114, 2015.

[162] Ken Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[163] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *the proceedings of the ICM*, 2018.

[164] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654, 2010.

[165] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[166] Richard Ryan Williams. The polynomial method in circuit complexity applied to algorithm design (invited talk). In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 47–60, 2014.

[167] Richard Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC*, pages 2:1–2:17, 2016.

[168] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[169] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

[170] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC 2014*, pages 664–673, 2014.

[171] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[172] Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1207–1215, 2018.

[173] Ryan Williams. Some estimated likelihoods for computational complexity. 2018. Invited paper in the Springer LNCS 10,000 volume.

[174] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 1867–1877, 2014.

[175] Sun Wu, Udi Manber, and Eugene Myers. A subquadratic algorithm for approximate regular expression matching. *Journal of algorithms*, 19(3):346–360, 1995.

[176] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.

[177] Huacheng Yu. Optimal succinct rank data structure via approximate nonnegative tensor decomposition. In *STOC 2019, to appear.*

[178] Huacheng Yu. An improved combinatorial algorithm for boolean matrix multiplication. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 1094–1105, 2015.