UNIVERSITY OF CALIFORNIA, SAN DIEGO

Fine-Grained Connections Between Exponential and Polynomial Time

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Stefan Schneider

Committee in charge:

Professor Ramamohan Paturi, Chair
Professor Sanjoy Dasgupta
Professor Massimo Franceschetti
Professor Russell Impagliazzo
Professor Shachar Lovett

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Mohan Paturi whose help was invaluable in getting me through this program.

I would also like to acknowledge my other co-authors Marco Carmosino, Jiawei Gao, Russell Impagliazzo, Marvin Künnemann, Shachar Lovett, Ivan Mihajlin, and Daniel Moeller.

A special thank also to my labmates Marco Carmosino, Jiawei Gao, Ivan Mihajlin, and Michael Walter.

Chapter 3 is based on material as it appears in the following publications: Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. “A satisfiability algorithm for sparse depth two threshold circuits.” In Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, pp. 479-488. IEEE, 2013. [84] The author of this dissertation was a principal author of this publication. Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. “0-1 integer linear programming with a linear number of constraints.” arXiv preprint arXiv:1401.5512 (2014). [80] The author of this dissertation was a principal author of this publication. Material from Chapter 3 is currently in preparation for submission for publication, by Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We thank Dominik Scheder and Ryan Williams for the fruitful discussions on the material in Chapter 3.

Chapter 4 is based on material as it appears in the following publications: Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. “Subquadratic algorithms for succinct stable matching.” In International Computer Science Symposium in Russia, pp. 294-308. Springer International Publishing, 2016. [110] The author of this dissertation was a principal author of this publication. Marvin Künnemann, Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. “Subquadratic Algorithms for Succinct Stable Matching.”

arXiv preprint arXiv:1510.06452v5 (2016). [101] The author of this dissertation was a principal author of this publication. Material from Chapter 4 is currently in submission for publication, by Marvin Künnemann, Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We would like to thank Russell Impagliazzo, Vijay Vazirani, and the anonymous reviewers for helpful discussions and comments on the material in Chapter 4.

Chapter 5 is based on material as it appears in the following publications: Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. "On the Fine-grained Complexity of One-Dimensional Dynamic Programming". To appear in the International Colloquium on Automata, Languages, and Programming, 2017. The author of this dissertation was a principal author of this publication. We would like to thank Karl Bringmann and Russell Impagliazzo for helpful discussions and comments on the material in Chapter 5.

Chapter 6 is based on material as it appears in the following publications: 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, pp. 261-270. ACM, 2016. [36] The author of this dissertation was a principal author of this publication. Material from Chapter 6 is currently in preparation for submission for publication, by Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We would like to thank Amir Abboud, Karl Bringmann, Bart Jansen, Sebastian Krinninger, Virginia Vassilevska Williams, Ryan Williams and the anonymous reviewers for many helpful comments on the material in this Chapter 6.

VITA

2009	Bachelor of Science, Eidgenössisch Technische Hochschule Zürich
2010	Master of Science, Eidgenössisch Technische Hochschule Zürich
2017	Doctor of Philosophy, University of California, San Diego

ABSTRACT OF THE DISSERTATION

Fine-Grained Connections Between Exponential and Polynomial Time

by

Stefan Schneider

Doctor of Philosophy in Computer Science

University of California, San Diego, 2017

Professor Ramamohan Paturi, Chair

This dissertation presents several results in fine-grained complexity. Fine-grained complexity aims at extending traditional complexity theory by making more precise, and fine-grained, statements on the complexity of problems in various resources, including time and space. In doing so, fine-grained complexity distinguishes between false equivalences, such as the equivalence of all NP-complete problems, while also exploring the connections between traditionally separate realms, such as polynomial and exponential time.

This dissertation takes a fine-grained view on computational problems from a range of fields, including geometry, biology, computational complexity, economics and graph theory.

We study the relationship between the satisfiability problem on threshold circuits and a geometric problem, giving the first non-trivial algorithms for both. From economics, we study the stable matching problem, giving both upper and lower bounds on the problem. We also take a fine-grained view on the power of dynamic programming, again proving both upper and lower bound, as well as relating dynamic programming to other algorithmic paradigms in a fine-grained manner. Finally, we study the relationship between several problems central to the field of fine-grained complexity, including satisfiability, 3-sum and the all-pairs shortest path problem, giving the first fine-grained non-reducibility results.

Chapter 1

Introduction

One of the main goals of computational complexity theory is to distinguish between computational problems that are feasible or efficiently solvable, and those that are not.

In the early 1960's, the seminal paper "On the Computational Complexity of Algorithms" by Hartmanis and Stearns [73] laid the foundation of modern complexity theory, including the definitions of time and space complexity. Already around the same time, both Cobham [46] and Edmonds [56] suggested that a "good" algorithm is one that has a time complexity that is polynomial in the size of the input. The idea of equating the complexity class P with the class of feasible problems has therefore been as old as modern complexity theory. Inherent in this early work is a path to showing that a problem is feasible; an algorithm that can be proven to require polynomial time on any input places the computational problem into the set of feasible problems.

On the other side of coin, we need a path to showing the infeasibility of a problem. Cook [47] and Levin [102] provided this tool with the definition of NP-hardness. If a problem can be shown to be NP-hard, then a polynomial time algorithm for that problem would have grave and extremely wide-ranging consequences. In particular, it would mean $\mathsf{P} = \mathsf{NP}$ and a very large class of problems, including almost any problem we encounter in real world applications, would have efficient algorithms. At least on an intuitive level, that seems very unlikely.

The theory of NP-hardness has been a success story of complexity theory, with a large number of problems that have been shown to be NP-hard, starting with a compilation of 21 problems by Karp [90]. The success of the theory of NP-hardness is partly due to its appeal as a separator of feasible and unfeasible problems. In turn, it reinforced the idea of equating efficiency with polynomial time.

NP-hardness is built around the idea of using reductions as a tool to argue about the complexity of computational problems. We will use this idea in a refined form as a core theme throughout this dissertation.

One of the problems with NP-hardness is that proving that a problem is NP-hard does not actually prove that the problem cannot be solved in polynomial time. The question whether $\mathsf{P} = \mathsf{NP}$ is unsolved to this day and considered one of, if not the most important open question of complexity theory and computer science as a whole. A proof that a problem is NP-hard is therefore a *conditional lower bound*; under the condition that $\mathsf{P} \neq \mathsf{NP}$ we can conclude that NP-hard problems cannot be solved in polynomial time.

The other issue is that both P and the class of NP-hard problems contain problems that are very different from each other. The complexity class P encompasses both problems with $\mathrm{O}(n)$ time algorithms and problems with $\mathrm{O}(n^{100})$ time algorithms. Apart from time complexity, problems may also differ widely in other measures, such as space and parallel complexity. The notion of feasibility may also depend on the problem domain. For example, one of the most fundamental computational tasks in bioinformatics is to compare two strings of DNA. The human genome consists of approximately three billion base pairs. A problem size in this order of magnitude does not allow for quadratic time algorithms and only linear time algorithms can truly be considered feasible in that context. On the other hand, in some domains the typical sizes of instances may be much smaller, and even larger exponents are feasible. Similarly, NP-hard problems vary widely both in time complexity and other measures such as approximability. To give just a few examples, GraphColoring can be solved in time $\tilde{O}(2^n)$ [26], 3-sat in time $\tilde{O}(1.308^n)$ [76] and PlanarSteinerTree in time $2^{\tilde{O}(n^{2/3})}$ [119]. NP-hardness is also not necessarily a sign of infeasibility in practice. The CNF-sat problem is NP-hard, yet SAT-solvers have been used in practice to solve real-world problems in fields such as artificial intelligence [91], computational biology [104], and many others. See [25] for an extensive treatment on SAT-solvers in practice.

Fine-grained complexity aims to go beyond traditional complexity theory and distinguish between problems, both inside P and the class of NP-hard problem by determining their exact complexity in different measures, including their relationship with other problems. Fine-grained complexity is built on refinements of the techniques discussed above, namely reductions and conditional lower bounds.

Polynomial-time reductions are, by definition, not sufficient to distinguish between problems in P or between NP-complete problems, as they do not preserve the exact time complexity of problems. Instead, we build fine-grained complexity around a more restrictive notion of reduction. *Fine-grained reductions* have the additional property that they preserve the time complexity with respect to specific time bounds. By establishing a web of fine-grained reductions among problems we can argue about the relative hardness of problems with respect to time complexity.

In order to translate fine-grained reductions into statments on lower bounds, we introduce conjectures that act as anchors. Similar as before, the previously used conjecture of $P \neq NP$ proves to be insufficient to reason at a fine-grained level. We consider even stronger (and therefore less believable) assumptions than $P \neq NP$. Two of the main conjectures used in fine-grained complexity, the *Exponential Time Hypothesis* (ETH) and the *Strong Exponential Time Hypothesis* (SETH) [82] are stronger versions

$\mathsf{P} \neq \mathsf{NP}$ in a direct sense. If either of the exponential time hypotheses is true, then $\mathsf{P} \neq \mathsf{NP}$. We also use other conjectures, such as the 3-sum conjecture and the APSP conjecture which are not immediately related to P versus NP, but that provide fine-grained assumptions about specific problems.

In exchange for stronger assumptions, we are able to give more fine-grained conditional lower bounds. In particular we try to find the optimal k, such that the problem can be solved in time $\tilde{O}(n^k)$, for problems in P, or $\tilde{O}(k^n)$ for NP-hard problems where n is a size parameter. We reason on a more fine-grained level than is possible using the theory of NP-hardness, but it is also coarse-grained in the sense that it we do not argue about subpolynomial factors for problems in P or subexponential factors for problems in NP, which is captured by the $\tilde{O}(\cdot)$ notation, that suppresses such factors.

The fine-grained definition of feasibility and hardness is typically bound to specific natural time complexities for different problems. For example, we ask the question of which problems are hard at time n^2 or 2^n. To do this we associate problems to a nominal or conjectured time bound that is often given by a relatively simple and generic algorithm. For example, for satisfiability problems on restricted circuit classes on n variables the nominal time bound is typically 2^n, matched by a simple brute-force algorithm. A focus of fine-grained complexity is then the question of which circuit classes allow for satisfiability algorithms faster than 2^n. Interestingly, this does not lead to independent questions of hardness at different nominal time bounds. Throughout this dissertation, we extensively use connections between problems at different nominal time bounds, connecting the hardness of problems, as well as exploiting the same connections to prove upper bounds for problems with one time bound from upper bounds for problems at other time bounds. In particular, we show that easy problems in the polynomial world have a lot in common with easy problems in the exponential world, while hard problems in the polynomial world have a lot in common with hard problems in the exponential world. With this classification, both problems in P and NP-hard problems can be either hard or easy.

Fine-grained complexity is inherently *problem-centric*. Problem-centric complexity aims to understand the complexity of specific, independently motivated, problems. This approach differs from the more traditional resource-centric view, which aims to classify problems based on the resources available (e.g. polynomial time). To some degree, the problem-centric approach is a product of the incomplete picture we currently have. Other than NP-complete problems, which all reduce to each other, we have a messy network of fine-grained reductions, with few equivalences. On the other hand, the problem-centric approach has some advantages. It keeps the area of fine-grained complexity more directly related to the long history of each of the problems we discuss, and it avoids hiding results on specific problems behind layers of abstraction.

With the idea of fine-grained complexity in place, a number of research directions immediately become apparent. The first one is exploring the type of lower bounds that we can infer from our new set of conjectures and hypotheses. In this dissertation we establish lower bounds, mainly based on SETH, for a wide range of problems such as VectorDomination, variants of the StableMatching problem, and instances of the least-weight subsequence problem (LWS). The second direction focuses on upper bounds. From an algorithmic point of view, the idea of improving algorithms in more fine-grained terms is very natural, but fine-grained complexity is a useful tool in identifying particularly interesting computational problems, although all of the problems discussed in this dissertation are already interesting independently. In particular, this dissertation discusses algorithm for the satisfiability problem on depth two threshold circuits, VectorDomination, StableMatching, and LWS. The third direction explores meta-questions that arise in the framework of fine-grained complexity. We study the limits of fine-grained complexity by discussing non-reducibility results that prove (conditionally) that certain problems cannot be shown to be hard under SETH, even though they are suspected to not allow efficient algorithms. This is particularly interesting as this shines light on the relationship of various conjectures that fine-grained complexity is based on.

Chapter 2

Fine-Grained Complexity

2.1 Notation

We use Õ to suppress subpolynomial factors if the bound is polynomial and subexponential factors if the bound is exponential. Throughout the dissertation, we use ω to denote the matrix multiplication exponent. For vectors a, we use a_i to denote the ith component. For two vectors a,b, $\langle a,b\rangle$ denotes the inner product. We use $[i]$ to denote the range of integers $\{1,\ldots,i\}$ and $[i,j]$ to denote the range $\{i,\ldots,j\}$. All logarithms are base 2 unless noted otherwise. Other notational conventions only used in certain chapters are introduced there.

2.2 Satisfiability

We begin our discussion of fine-grained complexity by looking at a class of problems that are central both to traditional and fine-grained complexity.

The satisfiability problem is an example of a *meta-problem*, a computational problem that where the input is an algorithm. Satisfiability is therefore a natural candidate to try to understand the complexity of this problem. It is also the first problem proved to be NP-complete [47, 102].

In its most general form P/Poly-sat, the input is a Boolean circuit on n variables and $\mathsf{poly}(n)$ size and the question is if there is a Boolean input such that the circuit outputs 1. The brute-force algorithm evaluates the circuit on all 2^n inputs and runs in time $\tilde{\mathsf{O}}(2^n)$. An immediate question is if we can improve on the time complexity of this problem. The theory of NP-completeness gives a conditional lower bound here: If $\mathsf{P} \neq \mathsf{NP}$, then the time complexity cannot be improved to $\mathsf{poly}(n)$. A more fine-grained question is if we can improve the time bound to $2^{(1-s)n}$ for some constant $s > 0$. We call s the savings of such an algorithm. The goal of finding algorithms with constant saving is more promising, at least when we consider the satisfiability problem on restricted circuit classes.

The $\mathscr{C}$-sat problem is defined for any circuit class $\mathscr{C} \subseteq \mathsf{P}/\mathsf{Poly}$. The input is a circuit of class $\mathscr{C}$ and the problem is to determine if there is an input such that the circuit outputs 1. The most studied instance of $\mathscr{C}$-sat is CNF-sat, where $\mathscr{C}$ is a formula in conjunctive normal form (CNF).

Given a variable set V, let the set $\overline{V} = \{\overline{v} \mid v \in V\}$ be the set of negative literals and $V \cup \overline{V}$ be the set of literals. A clause is a disjunction of literals, where we call the number of literals in a clause the clause width. Finally, a CNF, or $(\mathsf{AND} \circ \mathsf{OR})$-circuit, is a conjunction of clauses. If the clause width is at most k, then the formula is called a k-CNF. The formula

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee \overline{x_3}) \tag{2.1}$$

is an example for a 3-CNF on variable set $V = \{x_1, x_2, x_3, x_4\}$.

The k-sat problem is to decide satisfiability of k-CNF formulas. The problem is NP-complete for $k \geq 3$. The first algorithm with constant savings if k is constant is due to Monien and Speckenmeyer [111]. Later several algorithms improved on the savings, for example [118, 117, 130, 76].

If we do not restrict the the clause width, then Schuler's algorithm [131] for CNF-sat runs in time $2^{(1-\Omega(n/m))n}$, where m is the number of clauses. This gives an algorithm with constant savings if $m = O(n)$.

For more general circuit classes, not many results for constant savings are known. Santhanam [128] gives an algorithm with constant savings for DeMorgan-sat, the satisfiability problem on linear sized formulas over AND and OR with fan-in 2. There is an algorithm with constant savings for AC^0-sat [81], where the circuits are constant-depth circuits over (unbounded fan-in) AND and OR gates. For ACC^0-sat, where we also allow MOD_6 gates, Williams [147] gives an algorithm with subconstant savings.

For the max-2-sat problem, where the question is if it is possible to satisfy at least k clauses of a 2-CNF, Williams [142] gives an algorithm with constant savings, while for max-3-sat, nothing is known. For the max-sat problem, where we have no restriction on the clause width, Dantsin and Wolpert [51] give an algorithm with constant savings if the number of clauses is linear. The currently fastest algorithm for max-sat is due to Chen an Santhanam [42].

In Chapter 3 we give satisfiability algorithm with constant savings for a number of circuit classes that contain threshold gates. In particular, we give algorithms for SparseDepthTwoThr-sat and SymFormula-sat, if the number of wires is linear, and 0-1-ILP if the number of gates is linear.

2.3 Exponential Time Hypotheses

In order to prove lower bounds that are more fine-grained than the theory of NP-completeness can provide, a natural approach is to condition on stronger assumptions than $P \neq NP$. Two such statements are the *Exponential Time Hypothesis* (ETH) and the *Strong Exponential Time Hypothesis* (SETH) defined by Impagliazzo and Paturi [82].

Definition 2.3.1 (Exponential Time Hypothesis (ETH))**.** *The* 3-sat *problem on n variables requires time $\Omega(2^{\varepsilon n})$ for some $\varepsilon > 0$.*

Definition 2.3.2 (Strong Exponential Time Hypothesis (SETH))**.** *For every $\varepsilon > 0$, there is an k such that* k-sat *on n variables requires time $\Omega\left(2^{(1-\varepsilon)n}\right)$.*

SETH implies ETH, which in turn implies $\mathsf{P} \neq \mathsf{NP}$. While the second implication is a direct consequence of the NP-completeness of 3-sat, the first implication is a consequence of the *sparsification lemma* [85, 34]. We discuss the sparsification lemma in Section 2.5.

Both ETH and SETH can be defined with respect to deterministic or randomized algorithms. We will generally use the hypotheses in their randomized variants, which allows us to show conditional lower bounds on randomized algorithms. In rare cases where we consider ETH and SETH in deterministic variants we point that out explicitly (for example Section 2.9).

We do not know if either ETH or SETH are true. They therefore need to be looked at in the context of how useful they are as a tool in proving interesting results, as well as if they are likely to be true.

In terms of usefulness, both ETH and SETH have been successfully used for conditional lower bounds, both for NP-complete problems and problems in P. These lower bounds come in the form of ETH-hardness or SETH-hardness. A problem is (S)ETH-hard at some time bound T, if improving the existence of an algorithm significantly faster than T implies that ETH or SETH respectively are false. Section 2.4 formalizes this notion and contains a discussion of what constitutes a significant improvement. Sections 2.6 and 2.7 outline some of the previously known key results regarding SETH-hardness while for all parts of this dissertation the concept of SETH-hardness and conditional lower bounds in general play an important part.

In terms of likeliness to be true, the main argument is mostly a historical one. The k-sat problem has a long history of attempts at fast algorithms, none of which resulted in algorithms fast enough to contradict ETH or SETH. Likewise, the network of SETH-hard problems contains a large number of problems, each with its own long history of attempts at finding fast algorithms. The fine-grained complexity lens that connects them is a fairly new development, so these problems have been studied by many disjoint communities over a significant amount of time. Many of these developments have resulted in implicit or explicit conjectures on lower bounds, that are now unified under the umbrella of SETH.

Another dimension are consequences if ETH or SETH are false. Section 2.9 outlines some results on how fast satisfiability algorithms imply circuit lower bounds. In particular, if SETH is false, then the fast algorithm for k-sat implies a number of circuit lower bounds that have been open for a long time and are considered hard to achieve, potentially requiring completely new techniques.

For the purpose of this dissertation we mostly concentrate on SETH.

One aspect of the conditional lower bounds based on SETH and the connection to circuit lower bounds that is particularly interesting is that we can now divide up the world into two possible scenarios. If SETH is true, we get lower bounds in various domains including geometry, economics and computational biology. On the other hand, if SETH is false we get new results in circuit complexity. This property is not shared by other popular conjectures that imply conditional lower bounds, such as the 3-sum conjecture and the APSP conjecture (see Section 2.8).

2.4 Fine-Grained Reductions

The key tool for showing conditional lower bounds are reductions. The theory of NP-completeness is built around polynomial time reductions. In order to show that a problem is NP-complete, it is sufficient to give a polynomial time reduction from a known NP-complete problem.

For fine-grained complexity polynomial time reductions are not sufficient. In fact, the time bounds of the known algorithms for NP-complete problems may differ greatly. To give just a few examples, GraphColoring can be solved in time $\tilde{O}(2^n)$ [26], 3-sat in time $\tilde{O}(1.308^n)$ [76] and PlanarSteinerTree in time $2^{\tilde{O}(n^{2/3}}$ [119].

We define fine-grained reductions with the motivation to control the exact complexity. For this purpose, we consider languages together with their *natural* or *conjectured* complexities. We use the pair (L,T) to denote a language L together with its time complexity T. Intuitively, if (L_1,T_1) fine-grained reduces to (L_2,T_2), then any constant savings in the exponent of the time complexity of L_2 implies some constant savings in the exponent of the time complexity of L_1.

Definition 2.4.1 (Fine-Grained Reductions ($\leq_{FGR}$)). *Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1,T_1)* fine-grained reduces *to (L_2,T_2) (denoted $(L_1,T_1) \leq_{FGR} (L_2,T_2)$) if for all $\varepsilon > 0$, there is a $\delta > 0$ and a deterministic Turing reduction $\mathscr{M}^{L_2}$ from L_1 to L_2 satisfying the following conditions.*

(a) The time complexity of the Turing reduction without counting the oracle calls is bounded by $T_1^{1-\delta}$.

$$\mathsf{TIME}[\mathscr{M}] \leq T_1^{1-\delta} \tag{2.2}$$

(b) Let $\tilde{Q}(\mathscr{M},x)$ denote the set of queries made by $\mathscr{M}$ to the oracle on an input x of length n. The query lengths obey the following time bound.

$$\sum_{q \in \tilde{Q}(\mathscr{M},x)} (T_2(|q|))^{1-\varepsilon} \leq (T_1(n))^{1-\delta}$$

If a fine-grained reduction exists from (L_1,T_1) to (L_2,T_2), algorithmic savings for

L_2 can be transferred to L_1. The definition gives us exactly what is needed to establish savings for L_1 by simulating the machine $\mathscr{M}^{L_2}$ using the faster algorithm for L_2. The role of each parameter in the definition of fine-grained reducibility makes this clear.

T_1: The presumed time to decide L_1, usually given by a trivial algorithm.

T_2: The presumed time to decide L_2.

ε: Any savings (assumed or real) on computing L_2.

δ: The savings (as a function of ε) that can be obtained over T_1 when deciding L_1 by reducing to L_2.

It is easy to verify that fine-grained reductions, just like polynomial time reductions, can be composed.

Lemma 2.4.1 (Fine-grained reductions are closed under composition)**.** *Let $(A,T_A) \leq_{FGR} (B,T_B)$ and $(B,T_B) \leq_{FGR} (C,T_C)$. It then follows $(A,T_A) \leq_{FGR} (C,T_C)$.*

Similarly, we define randomized fine-grained reductions.

Definition 2.4.2 (Randomized Fine-Grained Reductions ($\leq^s_{rFGR}$))**.** *Exactly as in the deterministic case, except the Turing reduction from (L_1,T_1) to (L_2,T_2) is a probabilistic machine with some two-sided error bound*

$$\Pr[\mathscr{M}^{L_2}(x) = L_1(x)] \geq s \tag{2.3}$$

for some s.

We denote a randomized fine grained reduction from L_1 to L_2 with error bound s by $(L_1,T_1) \leq^s_{rFGR} (L_2,T_2)$. Generally, we will use $s = 2/3$, so we denote $\leq^{2/3}_{rFGR}$ by $\leq_{rFGR}$.

Similar to deterministic reductions, randomized reductions transfer savings in the randomized complexity for L_2 to L_1.

We will have occasion to consider FGRs between function problems. This poses the problem that, in certain situations, just writing down the solution to a problem could exceed the time bound and wipe out fine-grained savings. In the deterministic case, we cope with this by adding another restriction to the definition of a fine-grained reduction:

Definition 2.4.3 (Fine-Grained Reductions for Functions ($\leq_{fFGR}$))**.** *Exactly as in the decision deterministic case, except that the Turing reduction $\mathcal{M}^{f_2}$ is to a function problem f_2 and is expected to produce a functional output. In addition to the existing resource bounds, we bound the size of answers given by the f_2 oracle.*

$$\sum_{q \in \tilde{Q}(\mathcal{M},x)} |f_2(q)| \leq (T_1(n))^{1-\delta} \tag{2.4}$$

The bound on the query answer size ensures that fine-grained reductions between functional problems also compose.

We are now able to define SETH-hardness formally.

Definition 2.4.4 (SETH-hardness)**.** *A language L is* SETH*-hard at time T, if for some function $k(n) = \omega(1)$ we have* $(\mathsf{k(n)\text{-}sat}, 2^n) \leq_{rFGR} (L, T)$.

Note that we allow for randomized fine-grained reductions as we define SETH with respect to randomized algorithms. If a problem L is SETH-hard and the fine-grained reduction from k-sat is deterministic, we say L is SETH-hard under deterministic reductions.

To state the definition of SETH-hardness less formally, a problem L is SETH-hard at time T, if the existence of and algorithm for L with time $T^{1-\varepsilon}$ for any $\varepsilon > 0$ implies that SETH is false.

We would like to point out that the definition of fine-grained reductions above captures the reductions necessary to show SETH-hardness. For ETH-hardness, we can relax the requirements on the reductions. We will not define reductions that preserve subexponential time formally, but note that there is a concept of ETH-hardness that provides conditional lower bounds on problems if ETH holds.

2.5 Sparsification Lemma

The first fine-grained reduction that we discuss in some detail is the sparsification lemma [85, 34]. The sparsification lemma reduces the satisfiability problem of a k-CNF formula on n variables and any number of clauses to the the disjunction of a subexponential number of *sparse* k-CNF formulas, that have only $f(k)n$ clauses for some function f. The sparsification lemma is a fine-grained reduction from the k-sat problem, parametrized by k, and the CNF-sat problem, where the input is a CNF with cn clauses, and the time complexity is parametrized by c.

Formally the sparsification lemma is a follows.

Lemma 2.5.1 (Sparsification Lemma [85, 34])**.** *Given a k-CNF $\mathcal{F}$, and for every $\varepsilon > 0$ and k, there is a constant $c = k^{\mathrm{O}(k)}$ such that we can construct a set of k-CNF $\mathcal{F}_1, \ldots, \mathcal{F}_t$ with*

- *$t < 2^{\varepsilon n}$*
- *$\mathcal{F}_i$ has at most cn clauses for all i*
- *$\mathcal{F}$ is satisfiable if and only if at least one formula $\mathcal{F}_i$ is satisfiable*

Furthermore the construction takes time $\mathsf{poly}(n)2^{\varepsilon n}$.

Using the sparsification lemma we can conclude that the CNF-sat problem is SETH-hard at time 2^n. Due to Schuler's algorithm [131], which provides a fine-grained reduction from CNF-sat to k-sat, we can conclude something stronger, namely that SETH holds if and only if CNF-sat requires time 2^n.

Lemma 2.5.2. SETH *holds if and only if for any $\varepsilon > 0$, there is a c such that solving the* CNF-sat *problem on $m = cn$ clauses requires time $\Omega(n^{2-\varepsilon})$.*

2.6 Orthogonal Vectors and SETH

For the remaining sections of this chapter we will concentrate on fine-grained complexity inside P.

For a lot of problems in P that are SETH-hard, the proof is a fine-grained reduction from OrthogonalVectors. The SETH-hardness of OrthogonalVectors, first proved by Williams [142], is therefore central to the fine-grained complexity inside P.

The OrthogonalVectors problem is defined as follows:

Problem 1 (OrthogonalVectors)**.** *Given vectors $a_1, \dots, a_n, b_1, \dots, b_n \in \{0,1\}^d$, determine if there is $i, j \in [n]$ satisfying $\langle a_i, b_j \rangle = 0$.*

The OrthogonalVectors problem is a natural problem, both because of its geometric interpretation, but also if the input is interpreted as sets. In the language of set theory, the goal is to decide if there are two sets such that the intersection is empty.

Lemma 2.6.1 ([142])**.** *Assuming* SETH*, for any $\varepsilon > 0$, there is a c such that solving* OrthogonalVectors *problem on $d = c \log n$ dimensions requires time $\Omega(n^{2-\varepsilon})$.*

Proof. The reduction is from CNF-sat (see Lemma 2.5.2). Let $\mathscr{F}$ be a formula in constraint normal form with $d = cn$ clauses. We reduce the satisfiability problem for $\mathscr{F}$ to OrthogonalVectors using a technique called *split and list*. Divide the variable set into two sets S, T of size $\frac{n}{2}$ and for each set consider all $N = 2^{n/2}$ assignments to the variables in the set. For every assignment we construct a d-dimensional vector where the

ith position is 1 if and only if the assignment does not satisfy the ith clause of $\mathscr{F}$. Let U be the set of vectors corresponding to the assignments to S and let V be the set of vectors corresponding to T. A pair $u \in U$, $v \in V$ is orthogonal if and only if the corresponding assignment satisfies all clauses. An algorithm for the OrthogonalVectors problem on $d = cn = 2c \log N$ dimensions and in time $\mathrm{O}(N^{2-\varepsilon}) = \mathrm{O}(2^{(1-\varepsilon/2)n})$ would contradict SETH. Hence assuming SETH, for every $\varepsilon > 0$ there is a c such that the Boolean vector orthogonality problem with $d = c \log N$ requires time $\Omega(N^{2-\varepsilon})$. □

This reduction also trivially generalizes to k-OrthogonalVectors, simply partition the variable set into k parts instead of two.

Apart from the conditional hardness of OrthogonalVectors, this reduction also has consequences for algorithms. The fastest known algorithm for OrthogonalVectors on $d = c \log n$ dimensions has a time complexity of $n^{2-1/\mathrm{O}(\log c)}$ [7]. Applying this algorithm with the reduction we get an algorithm for CNF-sat on $d = cn$ clauses with time $n^{2-1/\mathrm{O}(\log c)}$. This is particularly interesting as this time bound matches the best known algorithm for CNF-sat (up to constants in the exponent) due to Schuler [131]. Schuler's algorithm, at its heart, is a fine-grained reduction to k-sat. This draws a very interesting picture of the relationship between k-sat, CNF-sat and OrthogonalVectors. Note that in general, our definition of fine-grained reduction does not preserve the exact savings of the algorithm, but in this case we do have such a relationship. The fastest algorithms for k-sat runs in time $2^{(1-1/\mathrm{O}(k))n}$ [118, 117, 130]. The same time bound can be achieved (up to constants in the exponent) by reducing k-sat to CNF-sat using the sparsification lemma (see Section 2.5) and then reducing to OrthogonalVectors. Similarly, CNF-sat can either be reduced to k-sat or OrthogonalVectors, achieving the same time bound.

While we know a fine-grained reduction from satisfiability to OrthogonalVectors, the other direction is not known. As far as we know, it is possible that SETH is false, but OrthogonalVectors requires quadratic time for large dimensions. A fine-grained reduction from OrthogonalVectors to k-sat would show that SETH and the OrthogonalVectors conjecture are equivalent. In fact, we do not know of any (nontrivial) fine-grained reductions from any problem with time T to any problem with time T' where $T' = \omega(T)$. In other words, all our fine-grained reductions are from hard to easy problems. Note that this phenomena is for the time complexity as a function of the total input size. Problems in a number of domains are often parametrized by other measures, such as graphs by the number of nodes (where the input size can be quadratic in the number of nodes), or circuits by the number of variables. This observation is at first glance counter-intuitive, but any fine-grained reduction from an easy problem to a hard problem would in some sense need to compress the input. It is an open question how to formalize this intuition and how to argue non-reducibility from OrthogonalVectors to CNF-sat.

The split and list technique has a number of applications, both for upper and lower bounds. On the upper bound side, a classical example is the $\tilde{O}(2^{n/2})$ time algorithm for SubsetSum, where we divide the input into two subsets of equal size and compute the sums of all $2 \cdot 2^{n/2}$ subsets of these sets. The problem is then to find two sums, one from each set of size $2^{n/2}$, such that they sum to the target value. This can be done in $\tilde{O}(2^{n/2})$, i.e. linear in the number of sums. We can think of this algorithm as a fine-grained reduction from SubsetSum to Table-2-sum, combined with the $O(n)$ time algorithm for Table-2-sum. Another example of an upper bound using split and list is in Chapter 3, where we reduce the 0-1 integer linear programming problem (0-1-ILP), a generalization of CNF-sat to the VectorDomination problem, a generalization of OrthogonalVectors. This allows us the get a nontrivial algorithm for 0-1-ILP and DepthTwoThr-sat.

For lower bounds, all known fine-grained reductions from problems in EXP to problems in P are split and list in some form. Apart from SETH-hardness proofs that use OrthogonalVectors as an intermediate steps, this includes also the fine-grained reductions from CNF-sat to k-DominatingSet [123], from MaxCut to k-Clique [142], and from BranchingProgram-sat to LongestCommonSubsequence and EditDistance [4].

2.7 SETH-hardness in P

Apart from OrthogonalVectors, there is a large and growing number of SETH-hard problems inside P. Most of these problems are SETH-hard at time n^2, with some problems at time n^k for some parameter k (e.g. k-OrthogonalVectors). This section gives a very brief overview over these results. For a more comprehensive overview, see [149].

A first cluster of results are vector problems, in particular variants and generalizations of OrthogonalVectors. An example of a SETH-hard vector problem is the SetContainment problem, where the input are two sets A, B of Boolean vectors, similar to the input for OrthogonalVectors. Other than OrthogonalVectors, we consider the vectors as sets (where a 1 at position i represents the fact that i is in the set) and the problem is to find two sets $a \in A, b \in B$ such that $a \subseteq b$. We can reduce OrthogonalVectors to SetContainment by negating the vectors in B component-wise. In fact, the OrthogonalVectors problem and the SetContainment problem are subquadratic equivalent. Another vector problem that with a trivial fine-grained reduction from OrthogonalVectors is MinInnProd, where the goal is to find the pair of Boolean vectors that minimizes the inner product.

The MaxInnProd problem, where the goal is to find the maximum inner product is also SETH-hard. We use the MaxInnProd problem as a starting point for fine-grained reductions in Chapter 4. The problem is defined as follows.

Problem 2 (MaxInnProd). *Given vectors $a_1, \dots, a_n, b_1, \dots, b_n \in \{0,1\}^d$ and $k \in nats$, determine if there is $i, j \in [n]$ satisfying $\langle a_i, b_j \rangle \geq k$.*

The fine-grained reduction is from SetContainment and is given by [15]: Partition the set of vectors A into sets A_i for $0 \leq i \leq d$ where A_i contains all vectors with Hamming weight i (i.e. sets with size i). Observe that a vector $b \in B$ represents a set that contains $a \in A_i$ if and only if $\langle a,b \rangle = i$. Thus A and B have two sets such that one contains the other if and only if there is an i such that A_i and B have a pair of vectors with inner product at least i.

Other problems that are trivially SETH-hard include integer variants of the MinInnProd problem and the MaxInnProd problem, as well as the VectorDomination problem, where the goal is to find vectors a, b such that $a_i \leq b_i$ for all $1 \leq i \leq d$. The fine-grained reduction from OrthogonalVectors to the Boolean variant of VectorDomination is similar to the reduction from OrthogonalVectors to the SetContainment problem.

Gao, Impagliazzo, Kolokolova and Williams [67] extend the results on (Boolean) vector problems to their sparse variants. Instead of Boolean vectors, the input is given by a list of indices such that the corresponding entry is 1. Note that this is a more natural input in some cases like sets. Some reductions, such as the fine-grained reductions between OrthogonalVectors and SetContainment become nontrivial in the sparse setting, as complementing a set is an expensive operation if the set is small compared to the universe. Their main result is that any property that can be expressed as a first-order formula on graphs has a fine-grained reduction to OrthogonalVectors, hence OrthogonalVectors is a complete problem for that class. We study first-order properties in the context of non-reducibility in Chapter 6.

SETH-hardness has been shown for a number of graph properties, most of which fall into the first-order property umbrella of [67]. Examples include the GraphDiameter-2 problem [29] and the k-DominatingSet problem [123]. k-DominatingSet is a particularly interesting example, as it is one of the few SETH-hard problems in P where the fine-grained reduction does not use k-OrthogonalVectors as an intermediate step. The reduction is directly from CNF-sat and also uses split and list.

Another class of SETH-hard problems are alignment problems. In an alignment problem, the input are two sequences of objects, and the goal is to compute an alignment, that is crossing-free matching of the objects that maximizes some function. Examples of SETH-hard alignment problems (at time n^2) include FréchetDistance [31], EditDistance [17] and LongestCommonSubsequence [2, 33].

The results on many alignment problems have been strengthened by a fine-grained reduction from the BranchingProgram-sat problem [4]. A strongly subquadratic algorithm for EditDistance would imply a satisfiability algorithm for BranchingProgram-sat, and not just CNF-sat. We can therefore see their result as a strengthening of the hardness by basing the result on a conjecture that is more likely than SETH. Alternatively, we can exploit the fact that we do not know any algorithm with superpolynomial improvements over $O(2^n)$ for BranchingProgram-sat. If we conjecture that no such algorithm exists, then we can conclude that EditDistance and other alignment problems do not have algorithms with arbitrary log-factor improvements, i.e. there is a k such that there is no $O(n^2/\log^k(n))$ algorithm for EditDistance. Note that this is consistent with currently known upper bounds, which save a factor of $\log^2 n$ [105]. They also show that such superpolynomial improvements imply circuit lower bounds similar to the results discussed in Section 2.9.

The dynamic programming formulation of the LongestCommonSubsequence problem and other alignment problems is perhaps the conceptually simplest example of a two-dimensional dynamic programming formulation. In the standard formulation, each entry of an $n \times n$ table is computed in constant time. In Chapter 5 we take a look at examples for one-dimensional dynamic programming.

2.8 Other Hardness Conjectures

Apart from SETH, a number of other popular conjectures are used for the purpose of fine-grained complexity. Similar to SETH, all conjectures below specify a time bound for a specific problem, and are fine-grained up to subpolynomial factors. The most popular conjectures other than SETH are the 3-sum-conjecture [61] and the APSP-conjecture [150]. For a more comprehensive overview, see [149].

By not restricting ourselves to a single conjecture we broaden the power of our fine-grained approach considerably. As a downside, it seems more likely that at least one of the conjectures turns out to be false. The conjectures we discuss in this section are typically independent from each other, according to our current understanding. The fact that we currently need multiple conjectures to explain the hardness of problems inside P can be viewed as a limitation of our current techniques. A number of results aim to address this issue. Abboud, Vassilevska Williams and Yu [6] identify problems that are hard under the meta-conjecture that at least one of SETH, the 3-sum-conjecture of the APSP-conjecture is true. In Chapter 6 we give conditional impossibility results for showing 3-sum and APSP SETH-hard, which implies that the use of multiple conjecture to explain the hardness of problems might be necessary.

2.8.1 3-Sum Conjecture

The 3-sum problem is defined as follows:

Problem 3 (3-sum)**.** *Given n integers $a_1 \dots a_n$ in the range $[-W, W]$ for some $W = \mathsf{poly}(n)$, the* 3-sum *problem is the problem of determining if there is $1 \leq i, j, k \leq n$ such that $a_i + a_j + a_k = 0$.*

The 3-sum conjecture [61] is that there is no algorithm for the 3-sum problem with time $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$. Originally, the conjecture was stated as there not being an algorithm with time $o(n^2)$. However, Baran, Demaine and Pătraşcu [19] give a $\mathrm{O}(n^2/\mathsf{polylog}(n)$ algorithm for 3-sum on integer inputs. Grønlund and Pettie gave an algorithm with similar improvements for the 3-sum problem in the real RAM model.

A large class of problems in computational geometry are 3-sum-hard. Examples include ThreePointsOnALine [61], HoleInUnion [61], and PolygonContainment [20]. Apart from computational geometry, 3-sum-hard problems include diverse problems such as TriangleEnumeration [122], LocalAlignment [5], as well as a number of dynamic graph problems [122].

More generally, the k-sum conjecture says that there is no algorithm for k-sum with time $\mathrm{O}(n^{\lceil \frac{k}{2} \rceil - \varepsilon})$ for any $\varepsilon > 0$.

2.8.2 APSP Conjecture

The All-pairs shortest path problem (APSP) is defined as follows:

Problem 4 (All-Pairs Shortest Path (APSP))**.** *Given an undirected, weighted graph $G = (V, E)$ with weights $w : E \to [-W, W]$ for some $W = \mathsf{poly}(|V|)$, compute for every pair $v_1, v_2 \in V$, compute the length of the shortest path from v_1 to v_2.*

The APSP conjecture [150] is that there is no algorithm for APSP with time $\mathrm{O}(n^{3-\varepsilon})$ for any $\varepsilon > 0$.

Vassilevska Williams and Williams [150] give a number of problems that are subcubic equivalent to APSP, including $(\min, +)$-Product, NegativeWeightTriangle, and SecondShortestPath. Other problems known to be subcubic equivalent include GraphDiameter and GraphRadius [3].

The fine-grained equivalences with APSP are a feature of the APSP conjecture not present in the fine-grained complexity based on SETH and 3-sum. While the study of these conjectures also resulted in some fine-grained equivalences, they are rare and often trivial examples. Fine-grained equivalences draw a very clear picture of the complexity landscape. Both from a lower and an upper bound perspective, progress on one problem directly translates to progress to another problem. Note that fine-grained equivalences do not necessarily imply that the savings are preserved exactly, but it does imply that for example the APSP conjecture and the NegativeWeightTriangle conjecture are equivalent.

Some problems that are known to be APSP-hard, but not fine-grained equivalent to APSP include ZeroWeightTriangle [139] and dynamic graph problems such as SingleSourceShortestPath [125].

2.8.3 Min-Plus Convolution Conjecture

The $(\min,+)$-Convolution problem is defined as follows:

Problem 5 ($(\min,+)$-Convolution)**.** *Given n-dimensional vectors* $a = (a_0, \dots, a_{n-1})$*,* $b = (b_0, \dots, b_{n-1}) \in [-W, W]^n$ *for some* $W = \mathrm{poly}(n)$*, the* $(\min,+)$-Convolution $a * b$ *is defined by*

$$(a*b)_k = \min_{0 \le i,j<n : i+j=k} a_i + b_j \qquad \textit{for all } 0 \le k \le 2n-2.$$

The $(\min,+)$-Convolution conjecture is that, for any $\varepsilon > 0$, there is no algorithm with time $O(n^{2-\varepsilon})$ for $(\min,+)$-Convolution. This problem is of particular interest in the context of fine-grained complexity as the $(\min,+)$-Convolution conjecture implies both the 3-sum conjecture [18] and the APSP conjecture [30]. While the $(\min,+)$-Convolution conjecture has not been studied as extensively as other conjectures, a number of fine-grained equivalences with $(\min,+)$-Convolution have been established [49]. Chapter 5 discusses an equivalence between $(\min,+)$-Convolution and the CoinChange problem.

2.9 Lower Bounds from Algorithms

In this section we give an overview over some results that give a formal connection between the existence of fast satisfiability algorithm and circuit lower bounds. This section is adapted from [129].

Williams [143, 147] gives a formal connection between satisfiability algorithms for a circuit class and lower bounds for the same class. Given a satisfiability algorithm that improves over brute force by only a superpolynomial amount, he constructs a lower bound against NEXP (nondeterministic exponential time). Not only is the satisfiability algorithm used as a black box, the result applies to a large set of natural circuit classes. By giving a satisfiability algorithm for ACC^0, Williams completes an (unconditional) proof for $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$. Since the connection between satisfiability algorithms and circuit bounds is more general than just ACC^0 circuits, this result is a also a possible path to prove further lower bounds in the future.

Connections between satisfiability algorithms and circuit lower bounds have been observed on a more informal level before. Techniques such as the satisfiability coding lemma [117] and the switching lemma [74] are used to derive properties of circuits that lead to both satisfiability algorithms and lower bounds.

The technique by Williams achieves a similar goal, as the result is both a satisfiability algorithm for ACC^0 and a lower bound for the same circuit class. The satisfiability algorithms relies on properties of the circuit class. However, instead of deriving a circuit lower bound directly from the same properties, Williams adds another layer of abstraction. The proof of the circuit lower bound does not depend on the properties of the circuit directly, but only on the derived satisfiability algorithm. As a consequence of this abstraction, he is able to formalize a connection between algorithms and lower bounds. While it is difficult to characterize what properties of circuit classes lead to both satisfiability algorithms and lower bounds, the abstraction allows a quantitative statement on the required satisfiability algorithm.

In the first paper [143], Williams proves that if there is an algorithm for $\mathsf{P/poly\text{-}sat}$ that improves over exhaustive search by a superpolynomial amount, then $\mathsf{NEXP} \not\subseteq \mathsf{P/Poly}$. The proof is an *indirect diagonalization* argument. Assuming $\mathsf{NEXP} \subseteq \mathsf{P/Poly}$ and the existence of a fast satisfiability algorithm for general $\mathsf{P/Poly}$ circuits, it gives an algorithm to solve an arbitrary problem $L \in \mathsf{NTIME}(2^n)$ in nondeterministic time $O(2^n/\omega)$ for some superpolynomial ω. As a result, there are no problems in $\mathsf{NTIME}(2^n)$ that are not in $\mathsf{NTIME}(2^n/\omega)$, which contradicts the nondeterministic time hierarchy theorem [48, 133].

For a rough outline of the proof, suppose there is a satisfiability algorithm for general circuits that improves over exhaustive search by a superpolynomial factor and $\mathsf{NEXP} \subseteq \mathsf{P/Poly}$. Then pick an arbitrary problem L in $\mathsf{NTIME}(2^n)$ and reduce it to the $\mathsf{Succinct\text{-}3\text{-}sat}$ problem, which is NEXP-complete. The $\mathsf{Succinct\text{-}3\text{-}sat}$ problem is a variation on 3-sat for exponential formulas. Instead of having the 3-CNF as an direct input, the input is a polynomial size circuit, such that on input i in binary, the output is the ith bit of the encoding of a 3-CNF formula. The $\mathsf{Succinct\text{-}3\text{-}sat}$ problem is then to decide if the implied 3-CNF is satisfiable. By the NEXP-completeness of $\mathsf{Succinct\text{-}3\text{-}sat}$ we can, given an input x to L of length n, construct a polynomial size circuit C with $n + \mathrm{O}(\log n)$ inputs such that on input i in binary, the output is the ith bit of a 3-CNF that is satisfiable if and only if $x \in L$. The number of variables of this 3-CNF formula is exponential in n.

To test the satisfiability of this circuit without explicitly writing out the 3-CNF formula, we use the idea of a *universal witness*. Impagliazzo, Kabanets and Wigderson [79] show that if $\mathsf{NEXP} \subseteq \mathsf{P/Poly}$, then for every satisfiable instance of a $\mathsf{Succinct\text{-}3\text{-}sat}$ problem there is a polynomial size circuit such that on input i in binary, it outputs the value of the ith variable in a satisfying assignment.

The nondeterministic algorithm proceeds as follows. First nondeterministically guess the universal witness for the given $\mathsf{Succinct\text{-}3\text{-}sat}$ problem. Since the goal is to give an algorithm that runs in $\mathsf{NTIME}(2^n/\omega)$ the algorithm is free to use nondeterminism at this point. Let this circuit be called D. From the $\mathsf{Succinct\text{-}3\text{-}sat}$ instance C we can construct a circuit C' that takes as input a number i in binary, and outputs the ith clause, consisting of three variables in binary (requiring $n + \mathrm{O}(\log n)$ bits each) and three bits to indicate if the literals are negated. Each of these variables is then given as input to the circuit D. As a last step, we can check if the values that D assigns to the variables satisfies the clause.

The circuit D is a universal witness for the 3-CNF formula if and only if the constructed circuit is unsatisfiable, i.e. there is no input i such that the universal witness does not give an assignment that satisfies the ith clause. Using the assumed fast algorithm for circuit satisfiability, we can decide this in time $O(2^n/\omega)$, resulting in an overall algorithm in $\mathsf{NTIME}(2^n/\omega)$, contradicting the nondeterministic time hierarchy theorem.

In the second paper [147], Williams refines his result for restricted circuit classes. For any circuit class $\mathscr{C}$ that contains AC^0 and is closed under composition, if there is a satisfiability algorithm for $\mathscr{C}$ that improves over exhaustive search by a superpolynomial amount, then $\mathsf{NEXP} \not\subseteq \mathscr{C}$. The main part of the proof is ensuring that the circuit constructed for the proof is in the class $\mathscr{C}$ so that we can apply the supposed algorithm for $\mathscr{C}$-sat. In particular, the circuit C' that takes as input a value i and returns the ith clause is not necessarily in the class $\mathscr{C}$. The key idea to get around this is by guessing and checking an equivalent $\mathscr{C}$-circuit, and then building the whole circuit using the guessed component. By giving an algorithm for ACC^0-SAT in the same paper he completes the proof for $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$.

These techniques were pushed further to the following circuit classes, which are not closed under composition [89].

Theorem 2.9.1. *For each of the following circuit classes $\mathscr{C}$, if the satisfiability problems* *for circuits in $\mathscr{C}$ can be solved in time $2^n/n^{\omega(1)}$ then there is a problem $f \in \mathsf{E}^{\mathsf{NP}}$ that is not solvable by circuits in $\mathscr{C}$:*

1. *linear-size series-parallel circuits,*

2. *linear-size log-depth circuits*

The key part of the proof is to find reductions from any problem in $\mathsf{NTIME}[2^n]$ to Succinct-3-sat, such that the constructed circuit is in the circuit classes above.

The authors of [89] also relate the framework to (deterministic) ETH and SETH. Note that the failure of ETH or SETH are ultimately statements about the existence of efficient satisfiability algorithm. Given Williams' framework it is therefore not surprising that we can prove circuit lower bounds from them. In particular, we can use any of the following three assumptions to infer some lower bound for the circuit classes above:

1. The exponential time hypothesis (ETH) is false; i.e., for every $\varepsilon > 0$, 3-sat is in time $2^{\varepsilon n}$

2. The strong exponential time hypothesis (SETH) is false; i.e., there is a $\delta < 1$ such that for every k, k-sat is in time $2^{\delta n}$

3. There is $\alpha > 0$ such that n^{α}-sat is in time $2^{n-\omega(n/\log\log n)}$

The main idea is to use structural decomposition theorems for the circuit classes above to re-express the circuit as the disjunction of a (relatively) small number of CNF. Then, run the assumed faster k-sat algorithm on each CNF. The overall formula is unsatisfiable if and only if all of the CNF formulas are unsatisfiable.

The links between satisfiability algorithms and circuit lower bounds provide an interesting property of SETH and ETH that are not shared by other popular conjectures such as the 3-sum conjecture or the APSP conjecture. While all of these conjectures are supported by a long history of unsuccessful attempts to disprove them and fast algorithms for either of these problems would have immediate consequences, disproving SETH would also prove lower bounds. As such, we get lower bounds in both possible states of the world. If SETH is true, all SETH-hard problems become unconditionally hard, while if SETH is false, we get circuit lower bounds.

In Section 6.4 we will revisit these results with respect to nondeterministic variants of ETH and SETH.

2.10 Contributions

In this dissertation we discuss contributions to fine-grained complexity that involve upper bounds for restricted versions of otherwise hard problems, conditional lower bounds as well as meta-questions on the power of the fine-grained complexity framework.

In Chapter 3 we give satisfiability algorithms with constant savings for depth-two threshold circuits with a linear number of wires and related circuit classes. Our main technique is a fine-grained reduction from the SparseDepthTwoThr-sat problem to the VectorDomination problem, an extension of the fine-grained reduction from CNF-sat to OrthogonalVectors. In this chapter, fine-grained complexity plays a dual role. One one hand, SETH gives limits on when we can expect fast satisfiability algorithms, and the goal becomes to give fast algorithms for parameters not excluded by SETH. One the other hand, we use the fine-grained connections between exponential and polynomial time directly as an algorithmic technique, which highlights the dual use of fine-grained reductions both as a mean to prove conditional lower bounds as well as to find new algorithms.

In Chapters 4 and 5 we consider succinct versions of problems that have linear time algorithms (in the input size). In Chapter 4 we consider the StableMatching problem.

In the StableMatching problem, we are computing a matching in an $n \times n$ bipartite graph that respects the preference orders for each node. The problem allows an $O(n^2)$ algorithm [62], but since the input consists of $2n$ preference lists (i.e. permutations), the time complexity of this algorithm is in fact linear. We study succinct representations for the preferences including the attribute, list, or geometric models, and ask the question if there are subquadratic algorithms for these input models. We show that for these representations some questions such as finding and verifying stable matchings are SETH-hard at time n^2 where n is the size of the succinct input, while other questions have strongly subquadratic algorithms.

In Chapter 5 we consider the Least-Weight Subsequence (LWS) problem. In the LWS problem, we are given a sequence of length n and are looking for the minimum weight subsequence, where the weights are given for each pair of positions in the original sequence. This problem does have an $O(n^2)$ algorithm [77], but again similar to StableMatching, the input is given as an n by n matrix and the algorithms is therefore linear in the input size. We study a number of succinct representations, including the CoinChange problem and the NestedBoxes problem. We show subquadratic equivalences between these LWS instantiations and related problems including problems that are well-studied in the fine-grained complexity context, such as OrthogonalVectors, VectorDomination and $(\min,+)$-Convolution. We exploit those equivalences both for new condition lower bounds and for new upper bounds. A key insight of these equivalences is that while the generic algorithm for LWS is very sequential, the subquadratic equivalent problems are not. These problems therefore isolate the core properties of the LWS instantiations that explain the subquadratic hardness of the problem, and separate the hardness from the sequential nature of LWS.

Finally in Chapter 6 we give fine-grained non-reducibility results. We emulate in a fine-grained way one of the ways to prove that a problem is not NP-complete. In particular, if we assume $\mathsf{NP} \neq \mathsf{coNP}$ and show that some problem L is in $\mathsf{NP} \cap \mathsf{coNP}$, then L cannot be NP-complete. We consider a fine-grained version of $\mathsf{NP} \neq \mathsf{coNP}$ which we call the Nondeterministic Strong Exponential Time Hypothesis (NSETH) and show that a number of problems are not SETH-hard assuming NSETH. Among these problems are 3-sum and APSP, the problems the most popular fine-grained conjectures other than SETH are based on. These non-reducibility results indicate that it might be difficult to find a unified conjecture that explains the conditional hardness of all problems, and that the fine-grained picture is messier than the very clean theory of NP-completeness.

Chapter 3

Depth Two Threshold Circuits

In this chapter we give a nontrivial algorithm for the satisfiability problem for cn-wire threshold circuits of depth two ($\mathsf{SparseDepthTwoThr\text{-}sat}$) which is better than exhaustive search by a factor 2^{sn} where $s = 1/c^{\mathsf{O}(c^2)}$. For the independently interesting special case of feasibility of 0-1 integer linear programs ($\mathsf{0\text{-}1\text{-}ILP}$), we strengthen the result to $s = 1/\mathsf{poly}(c)$, where cn is the number of inequalities. The key idea is to reduce the satisfiability problem to the $\mathsf{VectorDomination}$ problem, the problem of checking whether there are two vectors in a given collection of vectors such that one dominates the other component-wise. We give the first subquadratic algorithm for the $\mathsf{VectorDomination}$ problem for $\mathsf{O}(\log n)$ dimensions.

We extend our result to depth two circuits with symmetric gates where the total weighted fan-in is at most cn, as well as constant depth formulas.

Satisfiability testing is both a canonical NP-complete problem [47, 102] and one of the most successful general approaches to solving real-world constraint satisfaction problems. In particular, optimized $\mathsf{CNF\text{-}sat}$ heuristics are used to address a variety of combinatorial search problems successfully in practice, such as circuit and protocol design verification. The exact complexity of the satisfiability problem is also central to complexity theory, as demonstrated by Williams [143], who has shown that any improvement (by even a superpolynomial factor compared to exhaustive search) for the satisfiability problem for general circuits implies circuit lower bounds. Furthermore he has successfully used the connection to prove superpolynomial size bounds for ACC^0 circuits using a novel nontrivial satisfiability algorithm for ACC^0 circuits, solving a long standing open problem [147].

This raises the questions: For which circuit models do nontrivial satisfiability algorithms exist? How does the amount of improvement over exhaustive search relate to the expressive power of the model (and hence to lower bounds)? Can satisfiability heuristics for stronger models than CNF be useful for real-world instances?

Both the connection to circuit lower bounds and to heuristic search algorithms point to threshold circuits as the model to study next. Bounded depth polynomial size threshold circuits TC^0 are the next natural circuit class stronger than ACC^0. TC^0 is a powerful bounded depth computational model. It has been shown that basic operations like addition, multiplication, division, and sorting can be performed by bounded depth polynomial size threshold circuits [40, 28]. In contrast, unbounded fan-in bounded depth polynomial size circuits over the standard basis (even when supplemented with mod p gates for prime p) cannot compute the majority function [28]. However, our understanding of the limitations of bounded depth threshold circuits is extremely weak. Exponential lower bounds for such circuits are only known for the special case of depth two and bounded weight [72]. For larger depth circuits, barely superlinear lower bounds are known on the number of wires [83].

On the other hand, satisfiability for depth two threshold circuits contains as special cases some well known problems of both theoretical and practical significance. CNF-sat is one such special case, since both conjunctions and disjunction are a special case of threshold gates. max-k-sat, the optimization form of k-CNF satisfiability, is another special case, since the top threshold gate can count the number of satisfied clauses for an assignment. Even for max-3-sat, no algorithms with a constant factor savings over exhaustive search are known (although such an algorithm is provided for max-2-sat in [142]). Another special case is the Boolean version of Integer Linear Programming (0-1-ILP), a problem that is very useful in expressing optimization problems both in theory and practice. Testing the feasibility for a 0-1-ILP is equivalent to testing the satisfiability of a circuit with two levels, the bottom consisting of threshold gates and the top level being a conjunction. So both theoretical and real-world motivation points us to trying to understand the satisfiability problem for depth two threshold circuits.

Santhanam [128] gives an algorithm with constant savings for linear size formulas of AND and OR gates with fan-in two. However, this does not directly give an algorithm for DepthTwoThr-sat, as converting a linear size threshold circuit into a formula over AND and OR gates gives quadratic size.

In all of these related problems, a key distinction is between the cases of *linear size* and *superlinear size* circuits. In particular, an algorithm with constant savings for DepthTwoThr-sat of superlinear size would refute the *Strong Exponential Time Hypothesis* (SETH) [82], since k-CNF for all k can be reduced (via sparsification lemma [85]) to superlinear size depth two threshold circuits [34]. However, for CNF-sat and max-sat, algorithms with constant savings are known when the formula is *linear size* [131, 51, 42]. So, short of refuting SETH, the best we could hope for is to extend such an improvement to the linear size DepthTwoThr-sat problem.

In this chapter, we give an algorithm which obtains constant savings in the exponent over exhaustive search for the satisfiability of cn-wire, depth two threshold circuits for every constant c (we refer to the satisfiability problem parametrized by the number of wires as opposed to gates as SparseDepthTwoThr-sat). For the independently interesting case of 0-1-ILP, we improve the constant savings to $1/\text{poly}(c)$, where cn is the number of inequalities. Under SETH, these results are qualitatively the best we could hope for, but we expect that further work will improve our results quantitatively. For example, our savings is exponentially small in c, whereas in, e.g., the satisfiability algorithm of [81] for AC^0 circuits, it is polylogarithmic in c. We consider this just a first step towards a real understanding of the satisfiability problem for threshold circuits, and hope that future work will get improvements both in depth and in savings. A first step is due to Chen and Santhanam [42], who improve the time bound for SparseDepthTwoThr-sat to savings $s(c) = 1/c^{O(c)}$.

A series of related work (e.g. [146, 14, 15, 43, 136]) explores fast satisfiability algorithms for threshold circuits and its implication to circuit lower bounds further. In contrast to our algorithms, the related work achieves algorithms with time bounds of the form $2^{n-n^{\varepsilon}}$ for circuits sizes significantly larger than linear. On the other hand, they do not get algorithms with time bounds of the form $2^{(1-s)n}$, where s is constant, for linear size circuits. Both the goal of optimizing constant savings for linear size circuits and optimizing the size requirement to achieve polynomially small savings are well-motivated. Constant savings provide a natural generalization to results on satisfiability algorithms on more restricted circuit classes such as CNF-sat [131] and max-sat [51, 42], while polynomially small savings are sufficient to exploit the connection between satisfiability algorithms and circuit lower bounds [143].

Our main sub-routine is an algorithm for the VectorDomination problem: given n vectors in $\mathbb{R}^d$, is there a pair of vectors so that the first is larger than the second in every coordinate? We show that, when $d = c \log n$ for a constant c, this problem can be solved in subquadratic time. In contrast, Williams [142] shows that solving even OrthogonalVectors, the Boolean special case of VectorDomination, with a subquadratic algorithm when $d = \omega(\log n)$ would refute SETH. Our algorithm is therefore closely related to conditional lower bounds inside P based on SETH and other popular conjectures. While fine-grained reductions from CNF-sat to problems in P are often used imply conditional lower bounds based on SETH (see Section 2.7 for an overview), we use the same fine-grained reductions to imply fast satisfiability algorithms.

3.1 Notation and Problems

We discuss and give algorithms for a number of satisfiability problems. Let V be a set of Boolean variables with $|V| = n$. An *assignment* on V is a function $V \to \{0,1\}$ that assigns every variable a Boolean value. A *restriction* is an assignment on a set $U \subseteq V$. For an assignment α and a variable x, $\alpha(x)$ denotes the value of x under the assignment α.

A *threshold gate* on n variables $x_1,\ldots,x_n$ is defined by *weights* $w_i \in \mathbb{R}$ for $1 \leq i \leq n$ and a *threshold* t. The output of the gate is 1, if $\sum_{i=1}^{n} w_i x_i \geq t$ and 0 otherwise. The *fan-in* of the threshold gate is the number of nonzero weights. We call a variable an input to a gate if the corresponding weight is nonzero. We define threshold gates with real weights and thresholds and assume the real RAM model as a matter of presentation. Without loss of generality, any threshold gate can be represented by integer weights bounded by $n^{\mathrm{O}(n)}$ [112], hence arithmetic operations are in time $\mathsf{poly}(n)$ even in the word RAM model. We also extend the definition of a threshold gate to d-ary gates whose inputs and outputs are d-ary.

For a collection of threshold gates, the *number of wires* is the sum of their fan-ins. A *depth two threshold circuit* consists of a collection of m threshold gates (called the *bottom-level gates*) on the same n variables and a threshold gate (called the *top-level gate*) on the outputs of the bottom-level gates plus the variables. The output of the circuit is the output of the top-level gate. We call a variable with nonzero weight at the top-level gate a *direct wire*. For a d-ary depth two threshold circuit, the gates are d-ary gates and the top-level gate only outputs Boolean values. The number of wires of a depth two threshold circuit is the number of wires of the bottom-level gates. We call a threshold circuit *sparse* if the the number of wires is linear in the number of variables.

We define two variant of the satisfiability problem on depth two threshold circuits, one parametrized by the number of gates, and one parametrized by the number of wires.

Problem 6 (DepthTwoThr-sat)**.** *Given a depth two threshold circuit on n variables and with cn bottom-level gates, decide if there is an input such that the circuit outputs* 1.

Problem 7 (SparseDepthTwoThr-sat)**.** *Given a depth two threshold circuit on n variables and with cn wires, decide if there is an input such that the circuit outputs* 1.

We also define a special case of DepthTwoThr-sat where the top-level gate is restricted to be a conjunction. This problem is a Boolean case of integer linear programming.

Problem 8 (0-1-ILP)**.** *Given a collection of cn threshold gates on the same n variables, decide if there is an input such that all gates output* 1.

We refer to the threshold gates of an integer linear program as the constraints or inequalities. Note that we define the problem as the feasibility version of integer linear programming. The more typical variant with a linear objective function is fine-grained equivalent, as we can do a binary search for the objective value.

The VectorDomination problem is defined as follows:

Problem 9 (VectorDomination)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ determine if there is i, j such that $a_i \leq b_j$ component-wise.*

3.2 Results and Techniques

The main contributions of this chapter are nontrivial algorithms for a number of related problems, SparseDepthTwoThr-sat, 0-1-ILP, and VectorDomination. Our algorithms are the first to achieve constant savings for SparseDepthTwoThr-sat with a linear number of wires, 0-1-ILP with a linear number of constraints, and VectorDomination for logarithmic dimensions.

For SparseDepthTwoThr-sat we prove the following:

Theorem 3.2.1. *There is a satisfiability algorithm for depth two threshold circuits on n variables with cn wires that runs in time $\tilde{O}\left(2^{(1-s)n}\right)$ where*

$$s = \frac{1}{c^{O(c^2)}}$$

While the proof in Section 3.4 assumes a Boolean inputs for simplicity, the proof easily extends to threshold circuits with d-ary inputs, yielding the following corollary.

Corollary 3.2.1. *There is a satisfiability algorithm for depth two threshold circuits on n d-ary variables with cn wires that runs in time $\tilde{O}\left(d^{(1-s)n}\right)$ where*

$$s = \frac{1}{c^{O(c^2)}}$$

The algorithm consists of a chain of fine-grained reductions. We first reduce from SparseDepthTwoThr-sat to 0-1-ILP, and then from 0-1-ILP to VectorDomination. The algorithm is then completed with a subquadratic algorithm for VectorDomination.

The results for the 0-1-ILP problem and the VectorDomination problem are as follows.

Lemma 3.2.1. *There is an algorithm for the* 0-1-ILP *problem on n variables and $d = cn$ constraints that runs in time $O\left(n^{2-s}\right)$, where $s = 1/\mathsf{poly}(c)$.*

Lemma 3.2.2. *There is an algorithm for the* VectorDomination *problem on n vectors and $d = c\log n$ dimensions that runs in time $O\left(n^{2-s}\right)$, where $s = 1/\mathsf{poly}(c)$.*

In the following, we provide a high level description of our fine-grained reduction from the SparseDepthTwoThr-sat problem to the 0-1-ILP problem. Intuitively, there are two extreme cases for the bottom layer of a linear size threshold circuits of depth two.

The first extreme case is when we have a linear number of gates each with bounded fan-in k. This case is almost equivalent to max-k-sat and can be handled in a way similar to [35]. Consider the family of k-sets of variables given by the support of each bottom-level gate. A probabilistic argument shows that, for some constant c, there exists a subset U of about $n - n/(ck)$ variables so that at most one element from each of the k-sets in the family is outside of U. Then for any assignment to the variables in U, each bottom-level gate becomes either constant or a single literal, and the top-level gate becomes a threshold function of the remaining inputs. To check if a threshold function is satisfiable, we set each variable according to the sign of its weight. A threshold gate is satisfiable if and only if this assignments satisfies the gate.

The second extreme case is when we have a relatively small number of bottom-level gates, say, at most εn for some small ε, but some of the gates might have a large fan-in. In this case we apply a series of fine-grained reductions from the satisfiability problem of such circuits first to 0-1-ILP and subsequently to VectorDomination. We reduce to 0-1-ILP by guessing the truth value of all bottom-level gates as well as the top gate, and then verifying the consistency of our guesses. Since we guess the output of threshold functions of the variables, testing consistency of our guesses is equivalent to testing whether the feasible region of about εn linear inequalities has a Boolean solution, that is testing for consistency is equivalent to the 0-1-ILP problem.

We then reduce the 0-1-ILP problem to the VectorDomination problem, in an extension of the fine-grained reduction from CNF-sat to OrthogonalVectors [142] (see Section 2.6) . To do this, we partition the variables arbitrarily into two equal size sets. For each assignment to the first set, we compute a vector where the ith component corresponds to the weighted sum contributed by the first set of variables to the ith threshold gate.

For the second set of variables, we do the same, but subtract the contribution from the threshold for the gate. It is easy to see that the vectors corresponding to a satisfying assignment are a dominating pair. Since there are $N = \mathrm{O}(2^{n/2})$ vectors in our set, and each vector is of dimension $d = \varepsilon n = 2\varepsilon \log N$, to get constant savings, we need an algorithm for the VectorDomination problem that is subquadratic when the dimension is in the order of the logarithm of the number of vectors. The last step is to give such an algorithm, using a simple but delicate divide-and-conquer strategy.

Finally, to put these pieces together, we need to reduce the arbitrary case to a convex combination of the two extreme cases mentioned above. To do this, we prove a fan-in separation lemma which asserts that there must be a relatively small value of k so that there are relatively few gates of fan-in between k and ka, for some constant a. We show that, as in the first extreme case, for a random subset U of variables, the gates with fan-in less or equal to k almost entirely simplify to constants or literals after setting the variables in U. Our selection of k ensures that the number of gates of fan-in greater than k is small relative to the number of remaining variables. So we can apply the method outlined for the second extreme case. The fan-in separation lemma is where the savings s becomes exponentially small as a function of c. Unfortunately, this lemma is essentially tight, so a new method of handling this step would be needed to make the savings polynomially small.

The following two sections contain the details of the proof. Section 3.3 introduces the VectorDomination problem and, for $\mathrm{O}(\log n)$ dimension, gives an algorithm faster than the trivial quadratic time. The feasibility of a 0-1-ILP with a small number of inequalities is then reduced to the VectorDomination problem, yielding an algorithm for such 0-1-ILP with constant savings. A reduction from the DepthTwoThr-sat problem with εn bottom-level gates for small ε to 0-1-ILP concludes that section. In Section 3.4, we show how to reduce the SparseDepthTwoThr-sat problem with cn wires to the special case with a small number of bottom-level gates relative to the number of variables. The remaining sections discuss generalizations of our result. Section 3.5 generalizes the result to constant depth formulas, and Section 3.6 discusses more general symmetric gates.

3.3 Vector Domination Problem

In this section we give an algorithm for VectorDomination that is faster than the trivial $\mathrm{O}(n^2)$ for small enough dimensions.

Recall the definition of VectorDomination.

Problem 76 (VectorDomination)**.** *Given $a_1,\dots,a_n,b_1,\dots,b_n \in \mathbb{R}^d$ determine if there is i,j such that $a_i \le b_j$ component-wise.*

Note that we define VectorDomination with arbitrary real values, we can simplify the problem to values in $[n]$ by replacing the coordinate values by their rank with respect to the same coordinate of the other vectors.

Our algorithm uses the weighted median of a collection of numbers. Here, every number has an associated weight and the weighted median is a number such the both the total weight of all numbers smaller than the median and the total weight of all numbers larger than the median are at most half of the total weight. Note that the weighted median can be computed in linear time.

The algorithm is a divide and conquer algorithm, splitting the two sets A and B into sets A^+,A^-, B^+ and B^- depending on if the first coordinate is larger or smaller than some weighted median. The weighted median is computed across both sets A and B.

Lemma 3.3.1. *Let $A,B \subseteq \mathbb{R}^d$ with $|A|\cdot|B| = n^2$ and $d = c\log n$. There is an algorithm for the* VectorDomination *problem that runs in time $\mathrm{O}\left(n^{2-s(c)}\right)$, where $s(c) = 1/\mathsf{poly}(c)$ for $c \ge 4$ and at least 2^{-66} otherwise.*

Proof. Let $c' = \max\{c, 4\}$, $\varepsilon = \frac{1}{c'^{15}}$, $\gamma = \frac{1}{15 \log c'}$ and $t = \gamma \log n$.

Furthermore, let a be the weighted median of the first coordinates of $A \cup B$, where all numbers from A have weight $|B|$ and all number from B have weight $|A|$. Then, let $A^+ \subseteq A$ consist of all vectors where the first coordinate is larger than a and A^- consist of all vectors where the first coordinate is smaller than a. Similarly, split B into two sets B^+ and B^-. For vectors where the first coordinate is exactly a, it is sufficient to split the vectors evenly between the two possible sets, as long as we do not at the same time add vectors to A^- and B^+. This rounding is equivalent to adding a small positive noise to all vectors in A and a small negative noise to all vectors in B.

Now a vector $u \in A$ can only dominate a vector $v \in B$ in one of three cases:

1. $u \in A^+$ and $v \in B^+$
2. $u \in A^-$ and $v \in B^-$
3. $u \in A^+$ and $v \in B^-$

Also, in the third case any vector in A^+ dominates any vector in B^- on the first coordinate, hence we can recurse on $d-1$ dimensions.

Let $\varepsilon' = \frac{|A^-|}{|A|}$. Since we split at a weighted median where both A and B have the same total weight, we have $\frac{|A^-|}{|A|} = \frac{|B^+|}{|B|} = \varepsilon'$. We distinguish two cases, a *balanced* case where $\varepsilon' \geq \varepsilon$ and an *unbalanced* case where $\varepsilon' < \varepsilon$. In the unbalanced case, we recurse on all three subcases. In the balanced case, we also recurse on the three subcases and further we decrement t. Once $t = 0$ we solve the Vector Domination problem on the remaining vectors by exhaustive search.

To bound the runtime of above algorithm, we consider the recursion tree. We first bound the time spent on exhaustive search in leaves where $t = 0$. Each of the n^2 possible pairs of dominating vectors appears in at most one of the subcases. Furthermore, in the balanced case there are at least $\varepsilon^2 n^2$ pairs where one vector is in A^- and the other is in B^+ that are not considered in any subcase. Since there are t balanced cases on the path from the root to any exhaustive leaf, the total time spent on exhaustive search is bounded by

$$(1-\varepsilon^2)^t n^2 \le e^{-\varepsilon^2\gamma\log n} n^2 = n^{2-\log(e)\varepsilon^2\gamma} = n^{2-\frac{\log(e)}{15c'^{30}\log(c')}} = n^{2-\mathrm{poly}(1/c')} \tag{3.1}$$

For $c' = 4$, we get the savings of 2^{-66} as claimed.

To bound the size of the recursion tree we bound the number of possible paths from the root to any leaf. On any path, there are at most d steps that decrease the dimension. Furthermore, there are at most t balanced cases on any path. For the unbalanced case, in both subcases where the dimension does not decrease the number of pairs of vertices is multiplied by a factor of at most $\varepsilon' < \varepsilon$, hence this can happen at most $\frac{\log(n^2)}{\log(1/\varepsilon)}$ times along any path. Let

$$r = t + \frac{\log(n^2)}{\log(1/\varepsilon)} = \left(\gamma + \frac{2}{\log(1/\varepsilon)}\right)\log n = \frac{1}{5\log(c')}\log(n) \tag{3.2}$$

be an upper bound for any path to the number of subproblems that do not decrease the dimension of the vectors. Taking into account that the time spent on computing the median in every node is linear, the total time is bounded by

$$\mathrm{O}(n)\binom{d+r}{r}2^r \le \mathrm{O}(n)\left(e\left(1+5c\log(c')\right)\right)^{\frac{1}{5\log(c')}\log n} n^{\frac{1}{5\log(c')}} \tag{3.3}$$

$$= \mathrm{O}\left(n^{1+\log(e(1+5c\log(c')))\frac{1}{5\log(c')}+\frac{1}{5\log(c')}}\right) \tag{3.4}$$

We distinguish two cases. If $c \le 4$, then $c' = 4$ and above bound simplifies to $\mathrm{O}\left(n^{1+\log(e(1+10c))\frac{1}{10}+\frac{1}{10}}\right)$, which is monotonely increasing in c and hence maximized at

$c = 4$ where the bound is $\mathrm{O}\left(n^{1.781}\right)$.

For $c \geq 4$, we have $c' = c$. The bound is monotonely decreasing for large enough c and maximized at $c = 4$, where the bound is again $\mathrm{O}\left(n^{1.781}\right)$.

Note that the overall runtime is dominated by the time spent on exhaustive search in the leaves. □

Our algorithm is strongly subquadratic for any dimension $d = \mathrm{O}(\log n)$. This behavior matches the algorithms for related problems, such as OrthogonalVectors and MaxInnProd, although the dependence on the constant may be different. The previously fastest known algorithm due to Chan [37] is only strongly subquadratic for $d = \delta \log n$ dimensions for sufficiently small δ.

Chan [38] gives an improved analysis of the same algorithm and gets a time bound $n^{2-1/\mathrm{O}(c \log^2 c)}$, which incidentally matches the known upper bound for MaxInnProd.

The reduction from 0-1-ILP to the VectorDomination problem is an immediate consequence of the split and list technique (see Section 2.6).

Corollary 3.3.1. *Consider a* 0-1-ILP *program on n variables and cn inequalities for some $c > 0$. Then we can find a solution in time $\mathrm{O}\left(2^{n-s(c)}\right)$, where $s(c) = 1/\mathsf{poly}(c)$ for $c \geq 4$ and at least 2^{-66} for all c.*

Proof. Separate the variable set into two sets S_1 and S_2 of equal size. We assign every assignment to the variables in S_1 and S_2 a cn-dimensional vector where every dimension corresponds to an inequality. Let α be an assignment to S_1 and let $\sum_{i=1}^{n} w_{i,j} x_i \geq t_j$ be the jth inequality for all j. Let $a \in \mathbb{R}^{cn}$ be the vector with $a_j = \sum_{x_i \in S_1} w_{i,j} \alpha(x_i)$ and let A be the set of $2^{n/2}$ such vectors. For an assignment β to S_2, let b be the vector with $b_j = t_j - \sum_{x_i \in S_2} w_{i,j} x_i(\beta)$ and let B be the set of all such vectors b.

An assignment to all variables corresponds to an assignment to S_1 and an assignment to S_2, and hence to a pair $a \in A$ and $b \in B$. The pair satisfies all inequalities if and only if a dominates b. Since $|A| * |B| = 2^n$ and the dimension is cn, we can solve the VectorDomination problem in time $2^{(1-1/\mathsf{poly}(c))n}$. □

We now reduce the satisfiability of a depth two threshold circuit with δn bottom-level gates and any number of direct wires to the union of $2^{\delta n}$ 0-1-ILP problems.

Corollary 3.3.2. *Consider a depth two threshold circuit on n variables and δn bottom-level gates for some $0 < \delta < 2^{-66}$. We allow an arbitrary number of direct wires to the top-level gate. Then there is a satisfiability algorithm that runs in time $\mathrm{O}\left(2^{n-s(\delta)}\right)$ where $s(\delta) > 0$.*

Proof. For every subset U of bottom-level gates, we solve the satisfiability problem under the condition that only the bottom-level gates of U are satisfied. For an assignment to satisfy both the circuit and the condition that only gates in U are satisfied, it must satisfy the following system of inequalities:

1. For gates in U with weights $w_1, \dots, w_n$ and threshold t, we have $\sum_{i=1}^{n} w_i x_i \geq t$.
2. For gates not in U we require $\sum_{i=1}^{n} w_i x_i < t$, which is equivalent to $\sum_{i=1}^{n} -w_i x_i \geq -t + \min_i w_i$.
3. Let $v_1, \dots, v_n$ be the weights of the direct wires and let s be the threshold of the top-level gate. Further let w_U be the sum of the weights of the gates in U. Then $\sum_{i=1}^{n} v_i x_i \geq s - w_U$.

Note that this system contains $\delta n + 1$ inequalities, and the additional dimension adds only a polynomial factor to the time.

Since we need to solve a system of inequalities for every possible subset of bottom-level gates to be satisfied, we have an additional factor of $2^{\delta n}$. For $\delta < 2^{-66}$, we get a positive constant $s(\delta)$. □

One interpretation of the corollary above is as a fine-grained reduction from the DepthTwoThr-sat problem to a weighted version of the MostDominantVectors problem which is defined as follows.

Problem 10 (MostDominantVectors)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ and $k \in \mathbb{N}$ determine if there is i, j such that $a_i \leq b_j$ in at least k dimensions.*

While no subquadratic algorithms for the MostDominantVectors problem is known for $d = c \log N$ dimensions for arbitrary c, we can reduce the problem to 2^d instances of VectorDomination. For c small enough, we get a subquadratic algorithm.

3.4 Fan-In Separation

In this section we reduce the SparseDepthTwoThr-sat problem with cn wires to the DepthTwoThr-sat problem with at most δn bottom-level gates by considering all possible assignments to a random subset U of variables. The goal of the restriction is to eliminate all but a small fraction of gates. U will consist of all but a $\mathrm{O}(1/(ck))$ fraction of the variables where k is chosen such that there are only a small number of gates of fan-in larger than k relative to the number of remaining variables. The fan-in separation lemma shows how to find such a k.

Lemma 3.4.1 (Fan-In Separation Lemma)**.** *Let $\mathscr{F}$ be a family of sets such that*

$$\sum_{F \in \mathscr{F}} |F| \leq cn \tag{3.5}$$

Further let $a > 1$ and $\varepsilon > 0$ be parameters. There is a k with $k \leq a^{c/\varepsilon}$ such that

$$\sum_{\substack{F \in \mathscr{F} \\ k < |F| \leq ka}} |F| \leq \varepsilon n \tag{3.6}$$

Proof. Assume otherwise for the sake of contradiction. For $0 \leq i \leq \frac{c}{\varepsilon}$, let f_i be the sum of $|F|$ where $a^i < |F| \leq a^{i+1}$. By assumption we have $f_i > \varepsilon n$ for all i. Hence $\sum_{i=0}^{c/\varepsilon} f_i > cn$, which is a contradiction. □

Lemma 3.4.2. *Consider a depth two threshold circuit with n variables and cn wires. Let $\delta > 0$ and let U be a random set of variables such that each variable is in U with some probability $1 - p$ independently. There exists a $p = \frac{1}{c^{O(c^2)}}$ such that the expected number of bottom-level gates that depend on at least two variables not in U is at most $3\delta pn$.*

Proof. Let $\varepsilon = \frac{\delta^2}{c}$ and $a = \frac{c^2}{\delta^2}$ and let k be the smallest value such that there are at most εn wires as inputs to gates with fan-in between k and ka. Further let $p = \frac{\delta}{ck}$.

Using the fan-in separation lemma we get $k \leq \left(\frac{c^2}{\delta^2}\right)^{c^2/\delta^2}$. We distinguish three types of bottom-level gates: Small gates, with fan-in at most k, medium gates with fan-in between k and ka, and large gates with fan-in at least ka. For each type of gates, we argue that the expected number of gates that depend on at least two variables not in U is bounded by δpn.

For medium gates, the total number of wires is bounded by $\frac{\delta^2}{c}n$ and each gate contains at least k wires. Hence the number of medium gates is bounded by $\frac{\delta}{ck}\delta n = \delta pn$.

Large gates contain at least ka wires, hence the number of large gates is bounded by $\frac{c}{ka}n = \frac{\delta}{ck}\delta n = \delta pn$.

For small gates, we argue as follows. Let m be the number of small gates and let $l_1, \ldots, l_m \leq k$ be their fan-ins. Let X_i denote the event that gate i depends on at least two variables not in U and let X be the number of such events that occur. We have $P(X_i) \leq \binom{l_i}{2} p^2 \leq l_i^2 p^2$ and therefore

$$E[X] = \sum_{i=1}^{m} P(X_i) \leq \sum_{i=1}^{m} l_i^2 p^2 \leq p^2 k \sum_{i=1}^{m} l_i \leq p^2 kcn = \delta pn$$

□

Lemma 3.4.3. *There is an algorithm for* SparseDepthTwoThr-sat *with cn wires that runs in time $2^{(1-s)n}$ for $E[s] = \frac{1}{c^{O(c^2)}}$.*

Proof. Let $\delta = 2^{-68}$ and U as well as other parameters be as above. For every assignment to U, we have a depth two threshold circuit with pn variables and $3\delta pn$ bottom-level gates in expectation. Since $3\delta < 2^{-66}$, we can decide the satisfiability of such a circuit using Corollary 3.3.2 with constant savings. Let s' be the savings with our parameters.

Let T be the time for carrying out the entire procedure. Since we are interested in the expected savings we consider the logarithm of the time and get

$$E[\log(T)] = (1-p)n + (1-s')pn = (1-s'p)n$$

and the lemma follows from $p = \frac{1}{c^{O(c^2)}}$. □

Since s is bounded above by 1, we can repeat the process an expected constant number of times until we find a restriction such that the savings is at least its expectation. This gives us our main result Theorem 3.2.1.

3.5 Generalization to Formulas

In this section we discuss an extension of our main result to linear size, constant depth *threshold formulas*. A formula is a circuit such that the output of every gate is an input to at most one other gate. A formula can be viewed as a tree where the internal nodes correspond to gates and the leaves to bottom variables. Note that a circuit of depth two is always a formula. The proof is a direct generalization of our main proof.

Corollary 3.5.1. *There is a satisfiability algorithm for depth d threshold formulas with* *cn wires that runs in time* $\tilde{O}\left(2^{(1-s)n}\right)$ *where*

$$s = \frac{1}{((d-1)c)^{O(((d-1)c)^2)}}$$

Proof sketch. The algorithm chooses a random restriction such that at most δn gates depend on more than one variable after restriction, where $\delta < 2^{-66}$ as before. As in the original proof, we take into account that there is only a single top-level gate, which does not simplify after restriction. The main difference to our main proof is the notion of the fan-in. Instead of considering the number of inputs to a gate, consider the *size* of a gate. The size of a gate is the size of the subtree rooted at that gate. It is also an upper bound to the number of variables the gate depends on.

For all $i \leq d$, the sum of sizes of all gates at depth i is at most cn, since the circuit is a tree with at most cn wires. Hence the sum of sizes of all gates (minus the top-level gate) is at most $(d-1)c$.

Using the fan-in separation lemma we can select a set U of size pn where $p = \frac{1}{((d-1)c)^{O(((d-1)c)^2)}}$ such that the number of gates that depend on at least two variables not in U is at most δn. We can then write each remaining gate as a linear inequality, as each input is either a variable, a negated variable or a constant, which allows us to apply Corollary 3.3.1. □

3.6 Generalization to Symmetric Gates

In this section we discuss a second extension, to *symmetric gates*. A gate is symmetric if the output depends only on the weighted sums of the inputs. In particular, threshold gates are symmetric. The proof of our main result does not directly generalize to symmetric gates, but we give a different algorithm to decide the satisfiability of depth two circuits consisting of symmetric gates that follows similar ideas as our main proof. For this algorithm we do however require that the weights are integer and small. Specifically, we define the *weighted fan-in* of a gate as the sum of the absolute weights and the *weighted number of wires* as the sum of the fan-ins of all the gates. The result applies to circuits with a weighted fan-in of cn. We denote the satisfiability problem on such circuits as the $\mathsf{DepthTwoSym\text{-}sat}$ problem.

The main difference between the two algorithms is the problem we reduce it to after applying a restriction. In our main result, we reduce the satisfiability of the simplified circuit to a (small) system of linear inequalities. Here, we reduce to a system of linear equations. We first give an algorithm for linear equations.

Lemma 3.6.1. *There is an algorithm to find a Boolean solution to a system of linear equations on variables* $\{x_1, \ldots, x_n\}$ *in time* $\tilde{O}(2^{n/2})$.

Proof. We first reduce the problem to subset sum. Let $w_{i,j}$ be the weight of x_i in the jth equation and let r_j be right-hand side of the jth equation. Further let $D = \max_{i,j}\{w_{i,j}, r_j\}$ be the largest such value. We define $s_i = \sum_j w_{i,j} D^j$ and $s = \sum_j r_j D^j$. Then there is a solution to the system of linear equations if and only if there is a subset of the s_i that sums to s.

To solve the subset sum problem, partition the set of s_i into two sets of equal size and list all $2^{n/2}$ possible subset sums each. We can then sort the lists in time $O(2^{n/2}n)$ and determine if there is a pair of numbers that sums to s. $\square$

While the algorithm above uses $\tilde{O}(2^{n/2})$ space, Ueno [138] shows that there is a time-space tradeoff for the problem of finding Boolean solutions of systems of linear equations.

We reduce the satisfiability problem of depth two threshold circuits with small integer weights to a system of linear equations to get the following result.

Theorem 3.6.1. *There is an algorithm for the* DepthTwoSym-sat *problem with weighted number of wires cn that runs in time* $\tilde{O}\left(2^{(1-s)n}\right)$ *where*

$$\mathbf{E}[s] = \frac{1}{c^{O(c^2)}}$$

As before, we pick a random restriction with some parameter p, such that most gates depend on at most one variable.

Given an assignment, we distinguish between the Boolean output of a gate and the *value*. The *value* is defined as the weighted sum of the inputs. Note that the value uniquely defines the output of a symmetric gate. Unlike the fine-grained reduction for SparseDepthTwoThr-sat, we guess the value of the remaining gates, including the top-level gate. Given a value for every gate, we can write a system of linear equation. We then solve the system of linear equations on n Boolean variables in time $\tilde{O}(2^{n/2})$ using Lemma 3.6.1.

We need the overhead for guessing the values to be smaller than the savings achieved with solving the system of linear equations. For this, it is crucial that both the number of remaining gates and the number of values they can obtain is small. Here we use the requirement that the weights are small. We defer the details of the calculation on how many systems of linear equations we need to solve until Section 3.6.1.

One possible approach would be to select p using a fan-in separation technique. However, we only achieved savings that are doubly exponentially small in c using this approach. To get better savings, it is useful to model the interplay between the parameter p and the circuit as an explicit zero-sum game, where the first player's (the algorithm designer) pure strategies are the values of p and the second player's (the circuit designer) pure strategies are the circuits where all the gates have the same fan-in. The payoff is the difference between the saving of solving the subset sum problem and the overhead of guessing the values of the gates.

The mixed strategies of the circuit designer are circuits of symmetric gates with a weighted number of wires of at most cn, where each such circuit is viewed as a distribution of the total number of wires among gates of different weighted fan-in. The mixed strategies of the algorithm designer are distributions on the values of p. We then apply the Min-Max theorem to lower bound the expected value of the game by exhibiting a distribution (with finite support) on the values of p. We search through the values in the support of the distribution to find a p that produces the expected value. This game-theoretic analysis yields an overall savings which is only single exponentially small in c. Section 3.6.2 contains the details of the Min-Max approach.

3.6.1 The Algorithm

We develop the algorithm of Theorem 3.6.1 in three stages. In this section, we consider p a parameter and present an algorithm for DepthTwoSym-sat with a weighted number of wires of cn. We further assume that all the bottom-level gates have the same weighted fan-in f. The algorithm achieves savings $s_{p,f}$ and for certain combinations of p and f the savings might be negative. In the second stage we extend the algorithm to circuits with varying fan-in and show that the savings of the algorithm is a convex combination of $s_{p,f}$. In the last stage, in Section 3.6.2 we show how to select a p such that the savings is at least $\frac{1}{c^{O(c^2)}}$ for any distribution on f.

As we are mainly interested in the savings, we look at the logarithm of the time complexity and bound its expectation.

Lemma 3.6.2. *Let $0 \leq p \leq 1$ be a parameter and C be a depth two circuit with symmetric gates, variables $V = \{x_1, \ldots, x_n\}$, a weighted number of wires of cn, and weighted fan-in f for all bottom-level gates. There is an algorithm that decides the satisfiability of such*

C with time complexity T such that $\mathbf{E}[\log(T)] = (1 - s_{p,f})n$ *for*

$$s_{p,f} = \begin{cases} \frac{p}{4} & \textit{if } pf < \frac{1}{4c} \\ \frac{p}{2} - \frac{c}{f}\log(8cpf) & \textit{otherwise} \end{cases}$$

Proof. We select a random subset $U \subseteq V$ such that a variable is in U with probability $(1-p)$ independently. We note that $\mathbf{E}[|U|] = (1-p)n$. For each of the $2^{|U|}$ assignments to U, we solve the satisfiability problem of the simplified circuit. Bottom-level gates where all inputs are in U are removed and the top-level gate is adjusted appropriately. Gates that only depend on one input are replaced by a direct wire to the top-level gate with an appropriate weight and adjustment to the top-level gate. For all gates with at least two remaining inputs, we guess the value of the gate and express the gate as a linear equation. Similarly, we guess the value of the top-level gate to get another linear equation. We then solve the resulting system of linear equations on $n' = n - |U|$ variables in time $\tilde{O}\left(2^{n'/2}\right)$ using Lemma 3.6.1.

The critical part of the analysis is bounding the overhead from guessing the values of the gates. We first bound the number of distinct values a gate can take. The top-level gate can only take polynomially many different values. Consider a bottom-level gate with fan-in $l \geq 2$ after applying an assignment to the variables in U. We bounds the number of distinct values that the gate can take in two different ways. The number of possible inputs, and hence the number of possible values is bounded by 2^l. On the other hand, since the value is an integer between $-l$ and l, the number of possible values for the gates is also upper bounded by $2l+1$. Hence, we use $\min\{2^l, 2l+1\}$ as an upper bound for the number of values of a bottom-level gate with fan-in l.

The overhead crucially depends on the number of exceptional gates, gates that depend on more than one variable after applying an assignment to the variables in U.

Intuition says that the number of exceptional gates should be small. If the fan-in of a gate is small, then we expect that it will simplify to depend on at most one variable after assigning values to the variables in U. On the other hand, there cannot be too many gates of large fan-in. While the intuition is simple, it is tricky to make it work for us in the general context. At this stage, our focus is on estimating the savings $s_{p,f}$ for the probability parameter p and weighted fan-in f.

Let H be a random variable denoting the number of possible values the remaining gates can obtain. Our estimation of H and $s_{p,f}$ involves two cases. Let $t = \frac{1}{4c}$. We first consider the case $pf < t$. Let $U' \subseteq V - U$ be the set of variables that appear in exceptional gates. Our goal is to upper bound $\mathbf{E}[\log(H)] \leq \mathbf{E}[U']$.

Consider a bottom-level gate. Let X be the random variable denoting the number of its inputs not in U. Let $f' \leq f$ be the number of variables the gate depends on, and let X be the random variable denoting the number of its inputs not in U. The distribution of X is $\mathrm{Bin}(f', p)$, hence we have $\mathbf{E}[X] = f'p$. Let the random variable Y denote the number of variables that the gate can contribute to U'. Since U' is the set of variables appearing in exceptional gates, we have $Y = X$ for $X \geq 2$ and $Y = 0$ otherwise. Hence

$$\begin{aligned}\mathbf{E}[Y] = \mathbf{E}[X] - \mathbf{P}[X=1] &\leq f'p - f'p(1-p)^{(f'-1)}\\ &\leq f'p(1-(1-p)^{f'}) \leq f'p(1-(1-f'p))\\ &= (f'p)^2 \leq (fp)^2\end{aligned}$$

by Bernoulli's inequality. Hence, for any variable x which is an input to the gate, the probability x belongs to U' is at most $\frac{\mathbf{E}[Y]}{f} \leq p^2 f \leq \frac{p}{4c}$. Since the total number of wires is bounded by cn, we have $\mathbf{E}[\log(H)] \leq \mathbf{E}[|U'|] = cn\frac{p}{4c} = \frac{p}{4}n$.

For the logarithm of the time complexity this yields

$$\mathbf{E}[\log(T)] = \mathbf{E}[|U|] + \mathbf{E}\left[\frac{1}{2}(n-|U|)\right] + \frac{p}{4}n + \mathrm{O}(\log n) \le n\left(1-\frac{p}{4}\right) + \mathrm{O}(\log n)$$

where the logarithmic summand stems from guessing the value of the top-level gate. We have $s_{p,f} = \frac{p}{4}$.

We now consider the case $pf \ge t$. Suppose the ith gate has l_i inputs that are not in U. The expected value of l_i is pf. There are at most $2l_i + 1$ possible values for the gate. Since all the bottom-level gates have the same weighted fan-in f, the number of bottom-level gates is at most cn/f and $\mathbf{E}[\sum_{i=1}^{cn/f} l_i] = pcn$. We bound the expected logarithm of the number of possible values of all gates by

$$\begin{aligned}\mathbf{E}\left[\log\left(\prod_{i=1}^{cn/f}\right)(2l_i+1)\right] &= (cn/f)\sum_{i=1}^{cn/f}\mathbf{E}\left[(\log(2l_i+1)f/cn)\right] \le (cn/f)\log(2pf+1)\\ &\le (cn/f)\log(8cpf)\end{aligned}$$

where we use the concavity of the logarithm function in the penultimate step and the fact $pf \ge \frac{1}{4c}$ in the last step.

For the logarithm of the time complexity we get,

$$\begin{aligned}\mathbf{E}[\log(T)] &= \mathbf{E}[|U|] + \mathbf{E}\left[\frac{1}{2}(n-|U|)\right] + cn/f\log(8cpf) + \mathrm{O}(\log n)\\ &\le n\left(1-\left(\frac{p}{2}-\frac{c}{f}\log(8cpf)\right)\right) + \mathrm{O}(\log n)\end{aligned}$$

with savings $s_{p,f} = \frac{p}{2} - \frac{c}{f}\log(8cpf)$. $\square$

We now extend the algorithm to circuits with varying fan-in and show that the logarithm of the time complexities is lower bounded by a convex combination of the savings $s_{p,f}$. We model the cn-wire circuits of varying weighted fan-in by a distribution

$\mathscr{F}$ on fan-ins. For each weighted fan-in f, the wire distribution $\mathscr{F}$ specifies the number $c_f n$ of wires of bottom-level gates of weighted fan-in f. We denote the savings of our algorithm on circuits with wire distribution $\mathscr{F}$ by $s_{p,\mathscr{F}}$.

Lemma 3.6.3. *Let $0 \le p \le 1$ be a parameter and C be a depth two circuit with symmetric gates, n variables and a weighted number of wires of cn, where the wires are distributed according to $\mathscr{F}$. There is a satisfiability algorithm for such C with time complexity T such that $\mathbf{E}[\log(T)] = (1 - s_{p,\mathscr{F}})n$ for*

$$s_{p,\mathscr{F}} \ge \sum_{f=1}^{n} \frac{c_f}{c} s_{p,f}$$

Proof. The algorithm is the same as above. The logarithm of the overhead of guessing the values for all bottom-level gates with fan-in f is $\log(H_f) = \frac{c_f n}{f} \log(8cpf)$ if $pf \ge t$ and $\log(H_f) = \frac{c_f}{c}\frac{p}{4}n$ otherwise. Solving the system of linear equations and using linearity of expectation then yields the savings as claimed. □

3.6.2 The Algorithm as a Zero-Sum Game

The time complexity of the algorithm in Section 3.6.1 depends crucially on choosing a suitable parameter p. Instead of trying to directly determine a good parameter p by analyzing the wire distribution of the circuit, we apply a trick from game theory.

A zero-sum game with two players A and C is a game where both players pick a strategy and the outcome is determined by a function of the two strategies. Player A tries to maximize the outcome, while player C tries to minimize it. The Min-Max Theorem states that it does not matter which player moves first, as long as we allow mixed strategies for the players.

We model the task of choosing the parameter p as the following zero-sum game: Player A, the algorithm designer, picks some probability p, and player C, the circuit designer, picks a value f. The outcome is $s_{p,f}$, the savings of the algorithm. The algorithm designer tries to maximize the savings, and the circuit designer tries to minimize it. The wire distribution of a circuit is a mixed strategy for the circuit designer. A mixed strategy for the algorithm designer A would be a distribution on the probabilities.

A direct approach for designing the algorithm would be to select the parameter p depending on the circuit so that we obtain large savings. Specifically, given the wire distribution of the circuit $\mathscr{F}$, the algorithm designer picks a p and and the outcome $s_{p,\mathscr{F}}$ is a convex combination of the values $s_{p,f}$. Using the Min-Max Theorem we turn this game around: The algorithm designer picks a mixed strategy and the circuit designer responds with a pure strategy f, a circuit where all bottom-level gates have weighted fan-in f. The following lemma shows that there is a good strategy for the algorithm designer.

Lemma 3.6.4. *There is a distribution $\mathscr{D}$ on parameters p such that for all f,*

$$\mathbf{E}_{p\sim\mathscr{D}}[s_{p,f}] \geq \frac{1}{c^{\mathrm{O}\left(c^2\right)}}$$

Proof. Let $\mathscr{D}$ be the following distribution on p: For $I = \mathrm{O}\left(c^2\log(c)\right)$ with suitable constants, and $1 \leq i \leq I$, we set $p = 2^{-i}$ with probability $A \cdot 2^{-(I-i+1)}$, where $A = \frac{1}{\sum_{i=1}^{I} 2^{-(I-i+1)}}$ is the normalization factor. We know that $1 \leq A \leq 2$. The expectation of p is $\mathbf{E}[p] = AI2^{-I-1}$.

Let f be any pure strategy of the circuit designer and $J = \log(f)$. The expected outcome of the game for these strategies is

$$\mathbf{E}_{p\sim\mathscr{D}}[s_{p,f}] = \sum_{i=1}^{I} 2^{-(I-i+1)} s_{2^{-i},2^J}.$$

To lower bound the expected outcome, we use a case analysis on the savings similar to the one in Section 3.6.1. Let $t = \frac{1}{4c}$ as defined in the previous section. Let $I' \leq I$ be the largest value such that for $i \leq I'$, we have $2^{J-i} \geq t$ and for $I' < i \leq I$ we have $2^{J-i} < t$.

Using the savings from Lemma 3.6.2, we have $s_{2^{-i},2^J} = 2^{-i-1} - \frac{c}{2^J} \log\left(c2^{J-i+1}\right)$ for $2^{J-i} \geq t$ and $s_{2^{-i},2^J} = 2^{-i-2}$ otherwise. The expected savings is then

$$\begin{aligned}
\mathbf{E}_{p\sim\mathscr{D}}[s_{p,f}] &= \sum_{i=1}^{I} 2^{-(I-i+1)} s_{2^{-i},2^J} \\
&= \sum_{i=1}^{I'} 2^{-(I-i+1)} \left(2^{-i-1} - \frac{c}{2^J}\log\left(c2^{J-i+3}\right)\right) + \sum_{i=I'+1}^{I} 2^{-(I-i+1)} 2^{-i-2} \\
&\geq \sum_{i=1}^{I} 2^{-(I+3)} - \sum_{i=1}^{I'} 2^{-(I-i+1)} \frac{c}{2^J} \log\left(c2^{J-i+3}\right) \\
&= \frac{1}{2^{I+1}} \left(\frac{I}{4} - c\sum_{i=1}^{I'} 2^{-(J-i)} \log\left(c2^{J-i+3}\right)\right)
\end{aligned}$$

Let $j = \lceil (J-i) \rceil$. By the definition of I' we have $j \geq \log(t) = -\log(c) - 2$. Hence

$$\begin{aligned}
\sum_{i=1}^{I'} 2^{-(J-i)} \log\left(c2^{J-i+3}\right) &\leq \sum_{j=\log(t)}^{\infty} 2^{-j}\left(j + \log(8c)\right) \\
&\leq 8c\log(8c) + \sum_{j=1}^{\infty} j2^{-j} + \log(8c) \\
&= \mathrm{O}\left(c\log(c)\right)
\end{aligned}$$

Hence for $I = \mathrm{O}\left(c^2\log(c)\right)$ we get

$$\mathbf{E}_{p\sim\mathscr{D}}\, s_{p,f} = \frac{1}{c^{\mathrm{O}\left(c^2\right)}}$$

□

We now conclude that for every f there is a $p = 2^{-i}$ for $1 \le i \le I$, such that $s_{p,f} \ge \frac{1}{c^{O(c^2)}}$. Using that for every mixed strategy for f, the savings is a convex combination of the savings for pure strategies, we conclude the same for any strategy on f.

This gives us the final algorithm: Given a circuit C with wire distribution $\mathscr{F}$, evaluate $\mathbf{E}_{f\sim\mathscr{F}}[s_{p,f}]$ with $p = 2^{-i}$ for each $1 \le i \le I$ as above and use the optimal p for the random restriction.

The savings is tight in the sense that there is a mixed strategy on f such that the expected savings is at most $1/2^{\Omega(c)}$.

Lemma 3.6.5. *There is a wire distribution $\mathscr{F}$ such that for any p*

$$\mathbf{E}_{f\sim\mathscr{F}}[s_{p,f}] \le \frac{1}{2^{\Omega(c)}}$$

Proof. Let p be the strategy of the algorithm designer and let $\mathscr{F}$ be the distribution such that for $1 \le j \le c$, $c_{2^j} = 1$ and $c_f = 0$ for any other f. By Lemma 3.6.3 we have

$$\mathbf{E}_{f\sim\mathscr{F}}[s_{p,f}] = \sum_{j=1}^{c} \frac{1}{c} s_{p,2^j}$$

We argue that for large c and $p \ge \frac{1}{2^c}$, the savings is negative. Assume $p \ge \frac{1}{2^c}$. There is some $j^* \le c$ such that for $f = 2^{j^*}$, $1 \le pf \le 2$. Using that for any p and f, the savings $s_{p,f}$ is upper bounded by $\frac{p}{2}$ we get

$$\begin{aligned}
\mathbf{E}_{f\sim\mathscr{F}}[s_{p,f}] &= \sum_{j=1}^{c} \frac{1}{c} s_{p,2^j} \\
&\le \frac{p}{2} - \frac{1}{c} s_{p,2^{j^*}} \\
&= \frac{p}{2} - \frac{1}{c}\frac{c}{2^{j^*}} \log\left(cp2^{j^*+3}\right) \\
&\le \frac{p}{2}\left(1 - (\log(8c) + 1)\right)
\end{aligned}$$

For large c, the expectation is therefore negative. On the other hand, if $p \leq \frac{1}{2^c}$, then $\mathbf{E}_{f \sim \mathscr{F}}[s_{p,f}] \leq \frac{1}{2^{c-1}}$. $\square$

3.7 Conclusion

In this chapter, we present the first nontrivial algorithm for deciding the satisfiability of cn-wire threshold circuits of depth two. The algorithm improves over exhaustive search by a factor 2^{sn} where $s = 1/c^{O(c^2)}$. We also give the first algorithms with constant savings for the 0-1-ILP problem with a linear number of constraints and the VectorDomination problem with a logarithmic number of dimensions.

Several straightforward open questions remain. Can we further improve the savings? The savings in our algorithm is exponentially small in c, while the best known savings for cn-size AC^0 circuits is only polylogarithmically small in c [81]. Can we decrease this gap? If not, can we explain it in terms of the expressive power of the circuits?

Our algorithm handles only linear size threshold circuits of depth two. Can we obtain nontrivial satisfiability algorithms for slightly more relaxed models? For example, it would be very interesting to extend the result to larger depth circuits. It would also be nice to generalize the algorithm to deal with depth two threshold circuits with linearly many gates.

It would also be interesting to relax the restriction on the number of wires. Unfortunately, as discussed earlier, it is not be possible to obtain a constant savings algorithm for depth two threshold circuits of superlinearly many wires under SETH. However, it might be possible to get an algorithm that depends on the number of bottom-level gates, as opposed to number of wires.

The VectorDomination problem also raises a number of interesting questions, such as its relationship to other SETH-hard problems. For example, $\mathsf{OrthogonalVectors}$ admits an algorithm with time $n^{2-1/\mathsf{O}(\log c)}$ [7], where $c \log n$ is the dimension of the vectors. The savings for the $\mathsf{VectorDomination}$ problem are an order of magnitude smaller. Techniques to argue whether this discrepancy is necessary could prove to be very insightful. Similarly, both the $\mathsf{VectorDomination}$ problem and the $\mathsf{MinInnProd}$ problem of finding two vectors that minimize the inner product have algorithms with time $n^{2-1/\mathsf{poly}(c)}$, but while there are fine-grained reductions from $\mathsf{OrthogonalVectors}$ to either problem, there is no fine-grained reduction in either direction.

Our algorithm is a Split and List algorithm [142], split the variable set into subsets and list all assignments to the subsets. As such, it inherently takes exponential space. Can we reduce the space requirement to polynomial space?

Chapter 3 is based on material as it appears in the following publications: Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. “A satisfiability algorithm for sparse depth two threshold circuits.” In Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on, pp. 479-488. IEEE, 2013. [84] The author of this dissertation was a principal author of this publication. Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. “0-1 integer linear programming with a linear number of constraints.” arXiv preprint arXiv:1401.5512 (2014). [80] The author of this dissertation was a principal author of this publication. Material from Chapter 3 is currently in preparation for submission for publication, by Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We thank Dominik Scheder and Ryan Williams for the fruitful discussions on the material in Chapter 3.

Chapter 4

Stable Matching

In this chapter we consider the StableMatching problem when the preference lists are not given explicitly but are represented in a succinct way and ask whether the problem becomes computationally easier and investigate other implications. We give subquadratic algorithms for finding a stable matching in special cases of natural succinct representations of the problem, the d-attribute, d-list, geometric, and single-peaked models. We also present algorithms for verifying a stable matching in the same models. We further show that for $d = \omega(\log n)$ both finding and verifying a stable matching in the d-attribute and d-dimensional geometric models requires quadratic time assuming the SETH. This suggests that these succinct models are not significantly simpler computationally than the general case for sufficiently large d.

The StableMatching problem has applications that vary from coordinating buyers and sellers to assigning students to public schools and residents to hospitals [71, 96, 127]. Gale and Shapley [62] proposed a quadratic time *deferred acceptance* algorithm for this problem which has helped clear matching markets in many real-world settings. For arbitrary preferences, the deferred acceptance algorithm is optimal and even verifying that a given matching is stable requires quadratic time [114, 132, 68]. This is reasonable since representing all participants' preferences requires quadratic space. However, in many applications the preferences are not arbitrary and can have more structure. For example, top doctors are likely to be universally desired by residency programs and students typically seek highly ranked schools. In these cases participants can represent their preferences succinctly. It is natural to ask whether the same quadratic time bounds apply with compact and structured preference models that have subquadratic representations. This will provide a more nuanced understanding of where the complexity lies: Is StableMatching inherently complex, or is the complexity merely a result of the large variety of possible preferences? To this end, we examine several restricted preference models with a particular focus on two originally proposed by Bhatnagar et al. [24], the d-attribute and d-list models. Using a wide range of techniques we provide algorithms and conditional hardness results for several settings of these models.

In the *d-attribute* model (AttributeMatching), we assume that there are d different attributes (e.g. income, height, sense of humor, etc.) with a fixed, possibly objective, ranking of the men for each attribute. Each woman's preference list is based on a linear combination of the attributes of the men, where each woman can have different weights for each attribute. Some women may care more about, say, height whereas others care more about sense of humor. Men's preferences are defined analogously. This model is applicable in large settings, such as online dating systems, where participants lack the resources to form an opinion of every other participant. Instead the system can rank the members of each gender according to the d attributes and each participant simply needs to provide personalized weights for the attributes. The combination of attribute values and weights implicitly represents the entire preference matrix. Bogomolnaia and Laslier [27] show that representing all possible $n \times n$ preference matrices requires $n-1$ attributes. Therefore it is reasonable to expect that when $d \ll n-1$, we could beat the worst case quadratic lower bounds for the general StableMatching problem.

In the *d-list* model (ListMatching), we assume that there are d different rankings of the men. Each woman selects one of the d lists as her preference list. Similarly, each man chooses one of d lists of women as his preference list. This model captures the setting where members of one group (i.e. student athletes, sorority members, engineering majors) may all have identical preference lists. Mathematically, this model is actually a special case of the d-attribute model where each participant places a positive weight on exactly one attribute. However, its motivation is distinct and we can achieve improved results for this model.

Chebolu et al. [41] prove that approximately counting stable matchings in the d-attribute model for $d \geq 3$ is as hard as the general case. Bhatnagar et al. [24] showed that sampling stable matchings using random walks can take exponential time even for a small number of attributes or lists but left it as an open question whether subquadratic algorithms exist for these models.

We show that faster algorithms exist for finding a stable matching in some special cases of these models. In particular, we provide subquadratic algorithms for the following models: the d-attribute model, where all values and weights are from a small set (BoundedAttributeMatching), and the one-sided d-attribute model, where one side of the market has only one attribute (OneSidedAttributeMatching). These results show we can achieve meaningful improvement over the general setting for some restricted preferences.

While we only provide subquadratic algorithms to find stable matchings in special cases of the attribute model, we have stronger results concerning verification of stable matchings (VerifyStableMatching). We demonstrate optimal subquadratic stability testing algorithms for the d-list (VerifyListMatching) and Boolean d-attribute (VerifyBooleanAttributeMatching) settings as well as a subquadratic algorithm for the general d-attribute model with constant d (VerifyAttributeMatching). These algorithms provide a clear distinction between the attribute model and the general setting. Moreover, these results raise the question of whether verifying and finding a stable matching are equally hard problems for these restricted models, as both require quadratic time in the general case.

Additionally, we show that the StableMatching problem in the d-attribute model for $d = \omega(\log n)$ cannot be solved in subquadratic time under the Strong Exponential Time Hypothesis (SETH) [82, 85]. We show SETH-hardness for BooleanAttributeMatching, the VerifyBooleanAttributeMatching problem and for checking if a given pair is in any or all stable matchings (BooleanAttributeStablePair). These lower bounds apply even when the weights and attributes are Boolean. This adds the StableMatching problem to a growing list of SETH-hard problems (see Section 2.7 for an overview). Thus the quadratic time hardness of the StableMatching problem in the general case extends to the more restricted and succinct d-attribute model. This limits the space of models where we can hope to find subquadratic algorithms.

We further present several results in related succinct preference areas. Single-peaked preferences are commonly used to model preferences in social choice theory because of their simplicity and because they often approximate reality. Essentially, single-peaked preferences require that everyone agree on a common spectrum along which all alternatives can be ranked. However, each individual may have a different ideal choice and prefers the *closest* alternatives. A typical example is the political spectrum where candidates fall somewhere between liberal and conservative. In this setting, voters tend to prefer the candidates that are closer to their own ideals. As explained below, these preferences can be succinctly represented. Bartholdi and Trick [21] present a subquadratic time algorithm for StableRoommate (and StableMatching) with narcissistic, single-peaked preferences. In the narcissistic case, the participants are located at their own ideals. This makes sense in some applications but is not always realistic. We provide a subquadratic algorithm to verify if a given matching is stable in the general single-peaked preference model. Chung [45] uses a slightly different model of single-peaked preferences where a stable roommate matching always exists. In this model the participants would rather be unmatched than matched with someone further away from their ideal than they are themselves, leading to incomplete preference lists.

We extend our algorithms and lower bounds for the attribute model to the geometric model where preference orders are formed according to euclidean distances among a set of points in multi-dimensional space. Arkin et al. [16] derive a subquadratic algorithm for stable roommates with narcissistic geometric preferences in constant dimensions. Our algorithms do not require the preferences to be narcissistic.

It is worth noting that all of our verification and hardness results apply to the StableRoommate problem as well. This problem is identical to stable matching except we remove the bipartite distinction between the participants [71]. Unlike with (bipartite) StableMatching, there need not always exist a stable roommate matching. However, Irving [86] discovered an algorithm that produces a stable matching or identifies that none exists in quadratic time. Since finding a stable roommate matching is strictly harder than finding a stable matching, this is also optimal. Likewise, verification is equally hard for both StableRoommate and StableMatching, as we can simply duplicate every participant and treat the roommate matching as bipartite. Therefore, our results show that verification can be done more efficiently for the StableRoommate problem when the preferences are succinct.

Finally, we address the issue of strategic behavior in these restricted models. It is often preferable for a market-clearing mechanism to incentivize truthful behavior from the participants so that the outcome faithfully captures the optimal solution. Particularly in matching markets, this objective complements the desire for a stable matching where participants have incentives to cooperate with the outcome. Roth [126] showed that there is no strategy proof mechanism to find a stable matching in the general preferences setting. Additionally, if a mechanism outputs the man-optimal stable matching, the women can manipulate it to obtain the woman-optimal solution by truncating their preference lists [126, 63]. Even if the women are required to rank all men, they can still achieve more preferable outcomes in some instances [137, 98]. However, in the d-attribute, d-list, single-peaked, and geometric preference models, there are considerably fewer degrees of freedom for preference misrepresentation. Nevertheless, we show that there is still no strategy proof mechanism to find a stable matching for any of these models with $d \geq 2$ and non-narcissistic preferences.

Dabney and Dean [50] study an alternative succinct preference representation where there is a canonical preference list for each side and individual deviations from this list are specified separately. They provide an adaptive $O(n+k)$ time algorithm for the special one-sided case, where k is the number of deviations.

4.1 Summary of Results

Section 4.3.1 gives an $O(C^{2d}n(d+\log n))$ time algorithm for finding a stable matching in the d-attribute model if both the attributes and weights are from a set of size at most C (BoundedAttributeMatching). This gives a strongly subquadratic algorithm (i.e. $O(n^{2-\varepsilon})$ for $\varepsilon > 0$) if $d < \frac{1}{2\log C}\log n$.

Section 4.3.2 considers the OneSidedAttributeMatching case, where one side of the matching market has d attributes, while the other side has a single attribute. We allow both the weights and attributes to be arbitrary real values. Our algorithm for finding a stable matching in this model has time complexity $\tilde{O}(n^{2-1/\lfloor d/2 \rfloor})$, which is strongly subquadratic for constant d.

In Section 4.4.1 we consider the problem of verifying that a given matching is stable in the d-attribute model with real attributes and weights (VerifyAttributeMatching). The time complexity of our algorithm is $\tilde{O}(n^{2-1/2d})$, which is again strongly subquadratic for constant d.

Section 4.4.2 gives an $\mathrm{O}(dn)$ time algorithm for verifying a stable matching in the d-list model (VerifyListMatching). This is linear in its input size and is therefore optimal.

In Section 4.4.3 we give a randomized $\tilde{\mathrm{O}}(n^{2-1/\mathrm{O}(c\log^2(c))})$ time algorithm for $d = c\log n$ for verifying a stable matching in the d-attribute model when both the weights and attributes are Boolean (VerifyBooleanAttributeMatching). This algorithm is strongly subquadratic for $d = \mathrm{O}(\log n)$.

In Section 4.5 we give a conditional lower bound for the three problems of BooleanAttributeMatching, the VerifyBooleanAttributeMatching problem, as well as the BooleanAttributeStablePair problem. We show that there is no strongly subquadratic algorithm for any of these problems when $d = \omega(\log n)$ assuming the SETH. For the StablePair problem we give further evidence that even nondeterminism does not give a subquadratic algorithm.

Finally in Section 4.6 we consider the related preference models of single-peaked and geometric preferences. We extend our algorithms and lower bounds for the attribute model to the geometric model and give an $\mathrm{O}(n\log n)$ algorithm for verifying a stable matching with single-peaked preferences.

4.2 Preliminaries

A *matching market* consists of a set of men M and a set of women W with $|M| = |W| = n$. We further have a permutation of W for every $m \in M$, and a permutation of M for every $w \in W$, called *preference lists*. Note that representing a general matching market requires size $\Omega(n^2)$.

For a perfect bipartite matching μ, a *blocking pair* with respect to μ is a pair $(m,w) \notin \mu$ where $m \in M$ and $w \in W$, such that w appears before $\mu(m)$ in m's preference list and m appears before $\mu(w)$ in w's preference list. A perfect bipartite matching is called *stable* if there are no blocking pairs. In settings where ties in the preference lists are possible, we consider weakly stable matchings where (m,w) is a blocking pair if and only if both strictly prefer each other to their partner. For simplicity, we assume all preference lists are complete though our results trivially extend to cases with incomplete lists.

Gale's and Shapley's deferred acceptance algorithm [62] works as follows. While there is an unmatched man m, have m *propose* to his most preferred woman who has not already rejected him. A woman accepts a proposal if she is unmatched or if she prefers the proposing man to her current partner, leaving her current partner unmatched. Otherwise, she rejects the proposal. This process finds a stable matching in time $\mathrm{O}(n^2)$.

A matching market in the d-attribute model consists of n men and n women as before. A participant p has attributes $A_i(p)$ for $1 \leq i \leq d$ and weights $\alpha_i(p)$ for $1 \leq i \leq d$. For a man m and woman w, m's *value* of w is given by $\mathrm{val}_m(w) = \langle \alpha(m), A(w) \rangle = \sum_{i=1}^{d} \alpha_i(m) A_i(w)$. m ranks the women in decreasing order of value. Symmetrically, w's value of m is $\mathrm{val}_w(m) = \sum_{i=1}^{d} \alpha_i(w) A_i(m)$. Note that representing a matching market in the d-attribute model requires size $\mathrm{O}(dn)$. Unless otherwise specified, both attributes and weights can be negative.

A matching market in the d-list model is a matching market where both sides have at most d distinct preference lists. Describing a matching market in this model requires $\mathrm{O}(dn)$ numbers.

4.3 Finding Stable Matchings

4.3.1 Small Set of Attributes and Weights

We first present a stable matching algorithm for the d-attribute model when the attribute and weight values are limited to a set of constant size. In particular, we assume that the number of possible values for each attribute and weight for all participants is bounded by a constant C.

Algorithm 1: Small Constant Attributes and Weights

```
Group the women into sets $S_i$ with a set for each of the $C' = O(C^{2d})$ types of
  women ($O(C^d)$ possible attribute values and $O(C^d)$ possible weight vectors)
Associate an empty min-heap $h_i$ with each set $S_i$
for each man $m$ do
    Create $m$'s preference list of sets $S_i$
    index$(m) \leftarrow 1$
while there is a man $m$ who is not in any heap do
    Let $S_i$ be the index$(m)$ set on $m$'s list
    if $|h_i| < |S_i|$ then
        $h_i$.insert$(m)$
    else
        if val$_{S_i}(m) >$ val$_{S_i}(h_i.min)$ then
            $h_i$.delete_min()
            $h_i$.insert$(m)$
    index$(m) \leftarrow$ index$(m) + 1$
for $i = 1$ to $C'$ do
    $\mu \leftarrow \mu \bigcup$ Arbitrarily pair women in $S_i$ with men in $h_i$
return $\mu$
```

Theorem 4.3.1. *There is an algorithm to find a stable matching in the d-attribute model with at most a constant C distinct attribute and weight values in time* $O(C^{2d}n(d + \log n))$.

Proof. Consider Algorithm 1. First observe that each man is indifferent between the women in a given set S_i because each woman has identical attribute values. Moreover, the women in a set S_i share the same ranking of the men, since they have identical weight vectors. Therefore, since we are looking for a stable matching, we can treat each set of women S_i as an individual entity in a many to one matching where the capacity for each S_i is the number of women it contains.

With these observations, the stability follows directly from the stability of the standard deferred acceptance algorithm for many-one stable matching. Indeed, each man proposes to the sets of women in the order of his preferences and each set of women tentatively accepts the best proposals, holding onto no more than the available capacity.

The grouping of the women requires $O(C^{2d} + dn)$ time to initialize the groups and place each woman in the appropriate group. Creating the men's preference lists requires $O(dC^{2d}n)$ time to evaluate and sort the groups of women for every man. The while loop requires $O(C^{2d}n(d+\log n))$ time since each man will propose to at most C^{2d} sets of women and each proposal requires $O(d+\log n)$ time to evaluate and update the heap. This results in an overall running time of $O(C^{2d}n(d+\log n))$. $\square$

As long as $d < \frac{1}{2\log C}\log n$, the time complexity in Theorem 4.3.1 will be subquadratic. It is worth noting that the algorithm and proof actually do not rely on any restriction of the men's attribute and weight values. Thus, this result holds whenever one side's attributes and weight values come from a set of constant size.

4.3.2 One-Sided Real Attributes

In this section we consider a one-sided attribute model with real attributes and weights. In this model, women have d attributes and men have d weights, and the preference list of a man is given by the weighted sum of the women's attributes as in the two-sided attribute model. On the other hand there is only one attribute for the men. The women's preferences are thus determined by whether they have a positive or negative weight on this attribute. For simplicity, we first assume that all women have a positive weight on the men's attribute and show a subquadratic algorithm for this case. Then we extend it to allow for negative weights.

To find a stable matching when the women have a global preference list over the men, we use a greedy approach: process the men from the most preferred to the least preferred and match each man with the highest unmatched woman in his preference list. This general technique is not specific to the attribute model but actually works for any market where one side has a single global preference list. (e.g. [50] uses a similar approach for their algorithm.) The complexity lies in repeatedly finding which of the available women is most preferred by the current top man.

This leads us to the following algorithm: for every woman w consider a point with $A(w)$ as its coordinates and organize the set of points into a data structure. Then, for the men in order of preference, query the set of points against a direction vector consisting of the man's weight and find the point with the largest distance along this direction. Remove that point and repeat.

The problem of finding a maximal point along a direction is typically considered in its dual setting, where it is called the RayShooting problem. In the RayShooting problem we are given n hyperplanes and must maintain a data structure to answer queries. Each query consists of a vertical ray and the data structure returns the first hyperplane hit by that ray.

The relevant results are in Lemma 4.3.1 which follows from several papers for different values of d. For an overview of the RayShooting problem and related range query problems, see [11].

Lemma 4.3.1 ([75, 54, 9, 107])**.** *Given an n point set in $\mathbb{R}^d$ for $d \geq 2$, there is a data structure for ray shooting queries with preprocessing time $\tilde{O}(n)$ and query time $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor})$. The structure supports deletions with amortized update time $\tilde{O}(1)$.*

For $d = 1$, queries can trivially be answered in constant time. We use this data structure to provide an algorithm when there is a global list for one side of the market.

Lemma 4.3.2. *For $d \geq 2$ there is an algorithm to find a stable matching in the one-sided d-attribute model with real-valued attributes and weights in time $\tilde{O}(n^{2-1/\lfloor d/2 \rfloor})$ when there is a single preference list for the other side of the market.*

Proof. For a man m, let $\dim(m)$ denote the index of the last non-zero weight, i.e.

$$\alpha_{\dim(m)+1}(m) = \cdots = \alpha_d(m) = 0$$

We assume $\dim(m) > 0$, as otherwise m is indifferent among all women and we can pick any woman as $\mu(m)$. We assume without loss of generality $\alpha_{\dim(m)}(m) \in \{-1, 1\}$. For each d' such that $1 \leq d' \leq d$ we build a data structure consisting of n hyperplanes in $\mathbb{R}^{d'}$. For each woman w, consider the hyperplanes

$$H_{d'}(w) = \left\{ x_{d'} = \sum_{i=1}^{d'-1} A_i(w)x_i - A_{d'}(w) \right\} \tag{4.1}$$

and for each d' preprocess the set of all hyperplanes according to Lemma 4.3.1. Note that $H_{d'}(w)$ is the dual of the point $(A_1(w), \ldots, A_{d'}(w))$.

For a man m we can find his most preferred partner by querying the $\dim(m)$-dimensional data structure. Let $s = \alpha_{\dim(m)}(m)$. Consider a ray $r(m) \in \mathbb{R}^{\dim(m)}$ originating at

$$(-\frac{\alpha_1(m)}{s}, \ldots, -\frac{\alpha_{\dim(m)-1}(m)}{s}, -s \cdot \infty) \tag{4.2}$$

in the direction $(0, \ldots, 0, s)$. If $\alpha_{\dim(m)} = 1$ we find the lowest hyperplane intersecting the ray, and if $\alpha_{\dim(m)} = -1$ we find the highest hyperplane. We claim that the first hyperplane $r(m)$ hits corresponds to m's most preferred woman. Let woman w be preferred over woman w', i.e. $\text{val}_m(w) = \sum_{i=1}^{\dim(m)} A_i(w)\alpha_i(m) \geq \sum_{i=1}^{\dim(m)} A_i(w')\alpha_i(m) = \text{val}_m(w')$. Since the ray $r(m)$ is vertical in coordinate $x_{d'}$, it is sufficient to evaluate the right-hand side of the definition in equation 4.1. Indeed we have $\mathrm{val}_m(w) \geq \mathrm{val}_m(w')$ if and only if

$$\sum_{i=1}^{\dim(m)-1} -A_i(w)\frac{\alpha_i(m)}{s} - A_{\dim(m)}(w) \leq \sum_{i=1}^{\dim(m)-1} -A_i(w')\frac{\alpha_i(m)}{s} - A_{\dim(m)}(w') \quad (4.3)$$

when $s = 1$ and

$$\sum_{i=1}^{\dim(m)-1} -A_i(w)\frac{\alpha_i(m)}{s} - A_{\dim(m)}(w) \geq \sum_{i=1}^{\dim(m)-1} -A_i(w')\frac{\alpha_i(m)}{s} - A_{\dim(m)}(w') \quad (4.4)$$

when $s = -1$.

Note that the query ray is dual to the set of hyperplanes with normal vector $(\alpha_1(m), \dots, \alpha_d(m))$.

Now we pick the highest man m in the (global) preference list, consider the ray as above and find the first hyperplane $H_{\dim(m)}(w)$ hit by the ray. We then match the pair (m, w), remove $H(w)$ from all data structures and repeat. Correctness follows from the correctness of the greedy approach when all women share the same preference list and the properties of the halfspaces proved above.

The algorithm preprocesses d data structures, then makes n queries and dn deletions. The time is dominated by the n ray queries each requiring time $\tilde{O}(n^{1-1/\lfloor d/2 \rfloor})$. Thus the total time complexity is bounded by $\tilde{O}(n^{2-1/\lfloor d/2 \rfloor})$, as claimed. $\square$

Algorithm 2: One-Sided Stable Matching

```
// For points in $P \in \mathbb{R}^d$ we use the notation $(x_1,\dots,x_d)$ to
   refer to its coordinates
input: matching $\mu$
for $d' = 1$ to $d$ do
    for each woman $w$ do
        $H(w) \leftarrow \{x_d = \sum_{i=1}^{d-1} A_i(w)x_i - A_{d'}(w)\}$
        $H_{d'} \leftarrow H_{d'} \cup H(w)$
    $H_{d'}$.preprocess()
for each man $m$ in order of preference do
    $s \leftarrow \alpha_{\dim(m)}(m)$
    $r(m) \leftarrow (-\frac{\alpha_1(m)}{s},\dots,-\frac{\alpha_{\dim(m)-1}(m)}{s},\infty \cdot s) + t \cdot (0,\dots,0,-s)$
    $H(w) \leftarrow \mathrm{Query}(H_{\dim(m)}, r(m))$
    $\mu \leftarrow \mu \cup (m,w)$
    for $d' = 1$ to $d$ do
        $H_{d'} \leftarrow H_{d'} - H_{d'}(w)$
return $\mu$
```

Note that for $d = 1$ there is a trivial linear time algorithm for the problem.

We use the following lemma to extend the above algorithm to account for positive and negative weights for the women. It deals with settings where the women choose one of two lists (σ_1, σ_2) as their preference lists over the men while the men's preferences can be arbitrary.

Lemma 4.3.3. *Suppose there are k women who use σ_1. If the top k men in σ_1 are in the bottom k places in σ_2, then the women using σ_1 will only match with those men and the $n-k$ women using σ_2 will only match with the other $n-k$ men in the woman-optimal* *stable matching.*

Proof. Consider the operation of the woman-proposing deferred acceptance algorithm for finding the woman-optimal stable matching. Suppose the lemma is false so that at some point a woman using σ_1 proposed to one of the last $n-k$ men in σ_1. Let w be the first such woman. w must have been rejected by all of the top k, so at least one of those men received a proposal from a woman, w', using σ_2. However, since the top k men in σ_1 are the bottom k men in σ_2, w' must have been rejected by all of the top $n-k$ men in σ_2. But there are only $n-k$ women using σ_2, so one of the top $n-k$ men in σ_2 must have already received a proposal from a woman using σ_1. This is a contradiction because w was the first woman using σ_1 to propose to one of the bottom $n-k$ men in σ_1 (which are the top $n-k$ men in σ_2). □

We can now prove the following theorem where negative values are allowed for the women's weights.

Theorem 4.3.2. *For $d \geq 2$ there is an algorithm to find a stable matching in the one-sided d-attribute model with real-valued attributes and weights in time $\tilde{O}(n^{2-1/\lfloor d/2 \rfloor})$.*

Proof. Suppose there are k women who have a positive weight on the men's attribute. Since the remaining $n-k$ women's preference list is the reverse, we can use Lemma 4.3.3 to split the problem into two subproblems. Namely, in the woman-optimal stable matching the k women with a positive weight will match with the top k men, and the $n-k$ women with a negative weight will match with the bottom $n-k$ men. Now the women in each of these subproblems all have the same list. Therefore we can use Lemma 4.3.2 to solve each subproblem. Splitting the problem into subproblems can be done in time $O(n)$ so the running time follows immediately from Lemma 4.3.2. □

Table 4.1. Two-list preferences where no participant receives their top choice in the stable matching

σ_1	σ_2	π_1	π_2	Man	List	Woman	List
m_1	m_3	w_1	w_3	m_1	π_1	w_1	σ_2
m_2	m_5	w_2	w_5	m_2	π_1	w_2	σ_2
m_3	m_1	w_3	w_1	m_3	π_2	w_3	σ_1
m_4	m_4	w_4	w_4	m_4	π_1	w_4	σ_2
m_5	m_2	w_5	w_2	m_5	π_2	w_5	σ_1

As a remark, this "greedy" approach where we select a man, find his most preferred available woman, and permanently match him to her will not work in general. Table 4.1 describes a simple 2-list example where the unique stable matching is $\{(m_1,w_2),(m_2,w_3),(m_3,w_5),(m_4,w_4),(m_5,w_1)\}$. In this instance, no participant is matched with their top choice. Therefore, the above approach cannot work for this instance. This illustrates to some extent why the general case seems more difficult than the one-sided case.

An alternative model of a greedy approach that is based on work by Davis and Impagliazzo in [52] also will not work. In this model, an algorithm can view each of the lists and the preferences of the women. It can then (adaptively) choose an order in which to process the men. When processing a man, he must be assigned a partner (not necessarily his favorite available woman) once and for all, based only on his choice of preference list and the preferences of the previously processed men. This model is similar to online stable matching [93] except that it allows the algorithm to choose the processing order of the men. Using the preferences in Table 4.2 and minor modifications to them, we can show that no greedy algorithm of this type can successfully produce a stable matching. Indeed, the unique stable matching of the preference scheme below is $\mu = \{(m_1,w_3),(m_2,w_1),(m_3,w_2)\}$. However, changing the preference list for whichever of m_1 or m_2 is processed later will form a blocking pair with the stable partner of the other. If m_1 uses π_1, (m_1,w_1) blocks μ and if m_2 uses π_2, (m_2,w_3) blocks μ. Therefore, no algorithm can succeed in assigning stable partners to these men without first knowing the preference list choice of all three.

Table 4.2. Two-list preferences where a greedy approach will not work

σ_1	σ_2	π_1	π_2	Man	List	Woman	List
m_1	m_2	w_1	w_3	m_1	π_2	w_1	σ_1
m_2	m_1	w_2	w_2	m_2	π_1	w_2	σ_2
m_3	m_3	w_3	w_1	m_3	π_1	w_3	σ_2

4.3.3 Strategic Behavior

As mentioned earlier, strategic behavior in the general preference setting allows for participants to truncate or rearrange their lists. However, in the d-attribute and d-list models, we assume that the attributes or lists are fixed, so that the only manipulation the participants are allowed is to misrepresent their weight vectors or which list they choose. Despite this limitation, there is still no strategy proof mechanism for finding a stable matching when $d \geq 2$.

Theorem 4.3.3. *For $d \geq 2$ there is no strategy proof algorithm to find a stable matching in the d-list model.*

Proof. Table 4.3 describes true preferences that can be manipulated by the women.

Observe that there are two stable matchings: the man-optimal matching

$$\{(m_1,w_1),(m_2,w_2),(m_3,w_3),(m_4,w_4)\}$$

and the woman-optimal matching

$$\{(m_1,w_2),(m_2,w_3),(m_3,w_1),(m_4,w_4)\}$$

However, if w_2 used list σ_2 instead of σ_1, then there is a unique stable matching which is $\{(m_1,w_2),(m_2,w_3),(m_3,w_1),(m_4,w_4)\}$, the woman-optimal stable matching from the original preferences. Therefore, any mechanism that does not always output the woman optimal stable matching can be manipulated by the women to their advantage. By symmetry, any mechanism that does not always output the man-optimal matching could be manipulated by the men. Thus there is no strategy-proof mechanism for the d-list setting with $d \geq 2$. □

Table 4.3. Two-list preferences that can be manipulated

σ_1	σ_2	π_1	π_2	Man	List	Woman	List
m_1	m_3	w_1	w_3	m_1	π_1	w_1	σ_2
m_2	m_1	w_2	w_1	m_2	π_1	w_2	σ_1
m_3	m_4	w_3	w_2	m_3	π_2	w_3	σ_1
m_4	m_2	w_4	w_4	m_4	π_2	w_4	σ_1

Since the d-list model is a special case of the d-attribute model, we immediately have the following result from Theorem 4.3.3.

Corollary 4.3.1. *For $d \geq 2$ there is no strategy proof algorithm to find a stable matching in the d-attribute model.*

Of course in the 1-list setting there is a trivial unique stable matching. Moreover, in the one-sided d-attribute model our algorithm is strategy proof since the women are receiving the woman-optimal matching and each man receives his best available woman, so misrepresentation would only give him a worse partner.

4.4 Verification

We now turn to the problem of verifying whether a given matching is stable. While this is as hard as finding a stable matching in the general setting, the verification algorithms we present here are more efficient than our algorithms for finding stable matchings in the attribute model.

4.4.1 Real Attributes and Weights

In this section we adapt the geometric approach for finding a stable matching in the one-sided d-attribute model to the problem of verifying a stable matching in the (two-sided) d-attribute model. We express the verification problem as a *simplex range searching problem* in $\mathbb{R}^{2d}$, which is the dual of the RayShooting problem. In simplex range searching we are given n points and answer queries that ask for the number of points inside a simplex. In our case we only need degenerate simplices consisting of the intersection of two halfspaces. Simplex range searching queries can be done in sublinear time for constant d.

Lemma 4.4.1 ([106])**.** *Given a set of n points in $\mathbb{R}^d$, one can process it for simplex range searching in time $\mathrm{O}(n \log n)$, and then answer queries in time $\tilde{\mathrm{O}}(n^{1-\frac{1}{d}})$.*

For $1 \leq d' \leq d$ we use the notation $(x_1, \ldots, x_d, y_1, \ldots, y_{d'-1}, z)$ for points in $\mathbb{R}^{d+d'}$.

We again let $\dim(w)$ be the index of w's last non-zero weight, assume without loss of generality $\alpha_{\dim(w)} \in \{-1,1\}$, and let $\operatorname{sgn}(w) = \operatorname{sgn}(\alpha_{\dim(w)})$. We partition the set of women into $2d$ sets $W_{d',s}$ for $1 \le d' \le d$ and $s \in \{-1,1\}$ based on $\dim(w)$ and $\operatorname{sgn}(w)$. Note that if $\dim(w) = 0$, then w is indifferent among all men and can therefore not be part of a blocking pair. We can ignore such women.

For a woman w, consider the point

$$P(w) = (A_1(w),\ldots,A_d(w),\alpha_1(w),\ldots,\alpha_{\dim(w)-1}(w),\mathrm{val}_w(m)) \tag{4.5}$$

where $m = \mu(w)$ is the partner of w in the input matching μ. For a set $W_{d',s}$ we let $P_{d',s}$ be the set of points $P(w)$ for $w \in W_{d',s}$. The basic idea is to construct a simplex for every man and query it against all sets $P_{d',s}$.

Given d',s, and a man m, let $H_1(m)$ be the halfspace $\left\{\sum_{i=1}^{d} \alpha_i(m)x_i > \mathrm{val}_m(w)\right\}$ where $w = \mu(m)$. For $w' \in W_{d',s}$ we have $P(w') \in H_1(m)$ if and only if m strictly prefers w' to w. Further let $H_2(m)$ be the halfspace $\left\{\sum_{i=1}^{d'-1} A_i(m)y_i + A_{d'}(m)s > z\right\}$. For $w' \in W_{d',s}$ we have $P(w') \in H_2(m)$ if and only if w' strictly prefers m to $\mu(w')$. Hence (m,w') is a blocking pair if and only if $P(w') \in H_1(m) \cap H_2(m)$.

Using Lemma 4.4.1 we immediately have an algorithm to verify a stable matching.

Theorem 4.4.1. *There is an algorithm to verify a stable matching in the d-attribute model with real-valued attributes and weights in time* $\tilde{O}(n^{2-1/2d})$

Proof. Partition the set of women into sets $W_{d',s}$ for $1 \le d' \le d$ and $s \in \{-1,1\}$ and for $w \in W_{d',s}$ construct $P(w) \in \mathbb{R}^{d+d'}$ as above. Then preprocess the sets according to Lemma 4.4.1. For each man m query $H_1(m) \cap H_2(m)$ against the points in all sets. By the definitions of $H_1(m)$ and $H_2(m)$, there is a blocking pair if and only if for some man m there is a point $P(w) \in H_1(m) \cap H_2(m)$ in one of the sets $P_{d',s}$.

The time to preprocess is $\mathrm{O}(n \log n)$. There are $2dn$ queries of time $\tilde{\mathrm{O}}(n^{1-1/2d})$. Hence the whole process requires time $\tilde{\mathrm{O}}(n^{2-1/2d})$ as claimed. □

Algorithm 3: Verify Stable Matching with Reals

```
// For points in $P \in \mathbb{R}^{d+d'}$ we use the notation
   $(x_1, \ldots, x_d, y_1, \ldots, y_{d'-1}, z)$ to refer to its coordinates
input: matching $\mu$
for each woman $w$ do
    $m \leftarrow \mu(w)$
    $P(w) \leftarrow (A_1(w), \ldots, A_d(w), \alpha_1(w), \ldots, \alpha_d(w), \mathrm{val}_w(m))$
    $P_{\dim(w),\mathrm{sgn}(w)} \leftarrow W_{\dim(w),\mathrm{sgn}(w)} \cup P(w)$
for $d' = 1$ to $d$ and $s \in \{-1, 1\}$ do
    $P_{d',s}$.preprocess()
    for each man $m$ do
        $w \leftarrow \mu(m)$
        $H_1(m) \leftarrow \left\{\sum_{i=1}^{d} \alpha_i(m) x_i > \mathrm{val}_m(w)\right\}$
        $H_2(m) \leftarrow \left\{\sum_{i=1}^{d'-1} A_i(m) y_i + A_{d'}(m) \cdot s > z\right\}$
        if Query$(P_{d',s}, H_1(m) \cap H_2(m)) > 0$ then
            return $\mu$ is not stable
return $\mu$ is stable
```

4.4.2 Lists

When there are d preference orders for each side, and each participant uses one of the d lists, we provide a more efficient algorithm. Here, assume μ is the given matching between M and W. Let $\{\pi_i\}_{i=1}^{d}$ be the set of d permutations on the women and $\{\sigma_i\}_{i=1}^{d}$ be the set of d permutations on the men. Define $\mathrm{rank}(w,i)$ to be the position of w in permutation π_i. This can be determined in constant time after $\mathrm{O}(dn)$ preprocessing of the permutations. Let head(π_i, j) be the first woman in π_i who uses permutation σ_j and next(w,i) be the next highest ranked woman after w in permutation π_i who uses the same permutation as w or $\perp$ if no such woman exists. These can also be determined in constant time after $\mathrm{O}(dn)$ preprocessing by splitting the lists into sublists, with one sublist for the women using each permutation of men. The functions rank, head, and next are defined analogously for the men.

Algorithm 4: Verify d-List Stable Matching

```
for $i = 1$ to $d$ do
    for $j = 1$ to $d$ do
        $w \leftarrow \mathrm{head}(\pi_i, j)$
        $m \leftarrow \mathrm{head}(\sigma_j, i)$
        while $m \neq \perp$ and $w \neq \perp$ do
            if $\mathrm{rank}(w,i) > \mathrm{rank}(\mu(m),i)$ then
                $m \leftarrow \mathrm{next}(m,j)$
            else
                if $\mathrm{rank}(m,j) > \mathrm{rank}(\mu(w),j)$ then
                    $w \leftarrow \mathrm{next}(w,i)$
                else
                    return $(m,w)$ is a blocking pair
return $\mu$ is stable.
```

Theorem 4.4.2. *There is an algorithm to verify a stable matching in the d-list model in* $\mathrm{O}(dn)$ *time.*

Proof. We claim that Algorithm 4 satisfies the theorem. Indeed, if the algorithm returns a pair (m,w) where m uses π_i and w uses σ_j, then (m,w) is a blocking pair because w appears earlier in π_i than $\mu(m)$ and m appears earlier in σ_j than $\mu(w)$.

On the other hand, suppose the algorithm returns that μ is stable but there is a blocking pair, (m,w), where m uses π_i and w uses σ_j. The algorithm considers permutations π_i and σ_j since it does not terminate early. Clearly if the algorithm evaluates m and w simultaneously when considering permutations π_i and σ_j, it will detect that (m,w) is a blocking pair. Therefore, the algorithm either moves from m to $\text{next}(m,j)$ before considering w or it moves from w to $\text{next}(w,i)$ before considering m. In the former case, $\text{rank}(\mu(m),i) < \text{rank}(w',i)$ for some w' that comes before w in π_i. Therefore m prefers $\mu(m)$ to w. Similarly, in the latter case, $\text{rank}(\mu(w),j) < \text{rank}(m',i)$ for some m' that comes before m in σ_j so w prefers $\mu(w)$ to m. Thus (m,w) is not a blocking pair and we have a contradiction.

The **for** and **while** loops proceed through all men and women once for each of the d lists in which they appear. Since at each step we are either proceeding to the next man or the next woman unless we find a blocking pair, the algorithm requires time $\mathrm{O}(dn)$. This is optimal since the input size is dn. □

4.4.3 Boolean Attributes and Weights

In this section we consider the problem of verifying a stable matching when the d attributes and weights are restricted to Boolean values and $d = c\log n$. The algorithm closely follows an algorithm for the MaxInnProd problem by Alman and Williams [15]. The idea is to express the existence of a blocking pair as a probabilistic polynomial with a bounded number of monomials and use fast rectangular matrix multiplication to evaluate it. A probabilistic polynomial for a function f is a polynomial p such that for every input x

$$\Pr[f(x) \neq p(x)] \leq \frac{1}{3} \tag{4.6}$$

We use the following tools in our algorithm. THR_d is the threshold function that outputs 1 if at least d of its inputs are 1.

Lemma 4.4.2 ([15])**.** *There is a probabilistic polynomial for* THR_d *on n variables and error* ε *with degree* $\mathrm{O}(\sqrt{n\log(1/\varepsilon)})$.

Lemma 4.4.3 ([124, 135])**.** *There is a probabilistic polynomial for the disjunction of n variables and error* ε *with degree* $\mathrm{O}(\log(1/\varepsilon))$

Lemma 4.4.4 ([144])**.** *Given a polynomial* $P(x_1,\ldots,x_m,y_1,\ldots y_m)$ *with at most* $n^{0.17}$ *monomials and two sets* $X,Y \subseteq \{0,1\}^m$ *with* $|X| = |Y| = n$*, we can evaluate P on all pairs* $(x,y) \in X \times Y$ *in time* $\tilde{\mathrm{O}}(n^2 + m\cdot n^{1.17})$.

We construct a probabilistic polynomial that outputs 1 if there is a blocking pair. To minimize the degree of the polynomial, we pick a parameter s and divide the men and women into sets of size at most s. The polynomial takes the description of s men $m_1,\ldots,m_s$ and s women $w_1,\ldots,w_s$ along with their respective partners as input, and outputs 1 if and only if there is a blocking pair (m_i,w_j) among the s^2 pairs of nodes with high probability.

Lemma 4.4.5. *Let u be a large constant and* $s = n^{1/uc\log^2 c}$*. There is a probabilistic polynomial with the following inputs:*

- *The attributes and weights of s men*

$$A(m_1),\ldots,A(m_s),\alpha(m_1),\ldots,\alpha(m_s)$$

- *The attributes of the s women that are matched with these men*

$$A(\mu(m_1)),\ldots,A(\mu(m_s))$$

- *The attributes and weights of s women*

$$A(w_1),\ldots,A(w_s),\alpha(w_1),\ldots,\alpha(w_s)$$

- *The attributes of the s men that are matched with these women*

$$A(\mu(w_1)),\ldots,A(\mu(w_s))$$

The output of the polynomial is 1 *if and only if there is a blocking pair with respect to the matching μ among the s^2 pairs in the input. The number of monomials is at most $n^{0.17}$ and the polynomial can be constructed efficiently.*

Proof. A pair (m_i, w_j) is a blocking pair if and only if

$$\langle \alpha(m_i), A(\mu(m_i))\rangle < \langle \alpha(m_i), A(w_j)\rangle$$

and

$$\langle \alpha(w_j), A(\mu(w_j))\rangle < \langle \alpha(w_j), A(m_i)\rangle$$

Rewriting

$$F(x,y,a,b) := \langle x,y\rangle < \langle a,b\rangle = \mathrm{THR}_{d+1}\left(\neg(x_1\wedge y_1),\ldots,\neg(x_d\wedge y_d),a_1\wedge b_1,\ldots,a_d\wedge b_d\right) \tag{4.7}$$

we have a blocking pair if and only if

$$\bigvee_{\substack{i\in[1,s]\\ j\in[1,s]}} \big(F(\alpha(m_i),A(\mu(m_i)),\alpha(m_i),A(w_j))\wedge F(\alpha(w_j),A(\mu(w_j)),\alpha(w_j),A(m_i))\big) \tag{4.8}$$

Note that we can easily adapt this algorithm to finding strongly blocking pairs by defining $F(x,y,a,b)$ as $\langle x,y\rangle \leq \langle a,b\rangle$.

Using Lemma 4.4.2 with $\varepsilon = \frac{1}{s^3}$ and Lemma 4.4.3 with $\varepsilon = 1/4$ we get a probabilistic polynomial of degree $a\sqrt{d \log s}$ for some constant a and error $1/4 + 1/s < 1/3$. Furthermore, since we are only interested in Boolean inputs we can assume the polynomial to be multilinear. For large enough u we have $2d > a\sqrt{d\log(s)}$ (i.e. the degree is at most half of the number of variables) and the number of monomials is then bounded by $O\left(\left(s^2\binom{4d}{a\sqrt{d\log(s)}}\right)^2\right)$.

Simplifying the binomial coefficient we have

$$\binom{4d}{a\sqrt{d\log s}} = \binom{4c\log n}{a\sqrt{(\log^2 n)/u\log^2 c}} = \binom{4c\log n}{a\log n/\sqrt{u}\log c}$$

Setting $\delta = a/(\sqrt{u}\log(c))$ we can upper bound this using Stirling's inequality by

$$\binom{4c\log n}{\delta\log n} \leq \left(\frac{(4c\log n)\cdot e}{\delta}\right)^{\delta\log n} = n^{\delta\log(4ce/\delta)}$$

By choosing u to be a large enough constant, we can make δ and the exponent arbitrarily small. The factor of s^2 only contributes a trivial constant to the exponent. Therefore we can bound the number of monomials by $n^{0.17}$. $\square$

Theorem 4.4.3. *In the d-attribute model with n men and women, and $d = c\log n$ Boolean attributes and weights, there is a randomized algorithm to decide if a given matching is stable in time $\tilde{O}(n^{2-1/O(c\log^2(c))})$ with error probability at most $1/3$.*

Proof. We again choose $s = n^{1/uc\log^2 c}$ and construct the probabilistic polynomial as in Lemma 4.4.5. We then divide the men and women into $\lceil \frac{n}{s} \rceil$ groups of size at most s.

For a group of men $m_1,\ldots,m_s$ we let the corresponding input vector be

$$A(m_1),\ldots,A(m_s),\alpha(m_1),\ldots,\alpha(m_s),A(\mu(m_1)),\ldots,A(\mu(m_s))$$

We set X as the set of all input vectors for the $\lceil \frac{n}{s} \rceil$ groups. We define the set Y symmetrically for the input vectors corresponding to the $\lceil \frac{n}{s} \rceil$ groups of women.

Using Lemma 4.4.4 we evaluate the polynomial on all pairs $x \in X$, $y \in Y$ in time

$$\tilde{O}\left(\left(\frac{n}{s}\right)^2 + O(sd)\left(\frac{n}{s}\right)^{1.17}\right) = \tilde{O}\left(\left(\frac{n}{s}\right)^2\right) = \tilde{O}(n^{2-1/O(c\log^2(c))}) \tag{4.9}$$

The probability that the output is wrong for any fixed input pair is at most $1/3$. We repeat this process $O(\log n)$ times and take the threshold output for every pair of inputs, such that the error probability is at most $O\left(\frac{1}{n^2}\right)$ for any fixed pair of inputs. Using a union bound we can make the probability of error at most $1/3$ on any input. $\square$

4.5 Conditional Hardness

4.5.1 Background

The Strong Exponential Time Hypothesis has proved useful in arguing conditional hardness for a large number of problems. We show SETH-hardness for both verifying and finding a stable matching in the d-attribute model, even if the weights and attributes are Boolean. The main step of the proof is a reduction from the MaxInnProd problem to variants of the stable matching problem. Recall the definition and the fact that the problem is SETH-hard (see Section 2.7 for a proof).

Problem 2 (MaxInnProd). *Given vectors $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0,1\}^d$ and $k \in nats$, determine if there is $i, j \in [n]$ satisfying $\langle a_i, b_j \rangle \geq k$.*

Lemma 4.5.1 ([15]). *Assuming* SETH*, for any $\varepsilon > 0$, there is a c such that solving the* MaxInnProd *problem on $d = c \log n$ dimensions requires time $\Omega(n^{2-\varepsilon})$.*

4.5.2 Finding Stable Matchings

In this subsection we give a fine-grained reduction from the MaxInnProd problem to the problem of finding a stable matching in the Boolean d-attribute model. This shows that the StableMatching problem in the d-attribute model is SETH-hard, even if we restrict the attributes and weights to Booleans.

Theorem 4.5.1. *Assuming* SETH*, for any* $\varepsilon > 0$*, there is a* c *such that finding a stable matching in the Boolean* d*-attribute model with* $d = c\log n$ *dimensions requires time* $\Omega(n^{2-\varepsilon})$.

Proof. The proof is a reduction from MaxInnProd to finding a stable matching. Given an instance of the MaxInnProd problem with sets $U, V \subseteq \{0,1\}^d$ where $|U| = |V| = n$ and threshold l, we construct a matching market with n men and n women. For every $u \in U$ we have a man m_u with $A(m_u) = u$ and $\alpha(m_u) = u$. Similarly, for vectors $v \in V$ we have women w_v with $A(w_v) = v$ and $\alpha(w_v) = v$. This matching market is symmetric in the sense that for m_u and w_v, $\text{val}_{m_u}(w_v) = \text{val}_{w_v}(m_u) = \langle u, v\rangle$.

We claim that any stable matching contains a pair (m_u, w_v) such that the inner product $\langle u, v\rangle$ is maximized. Indeed, suppose there are vectors $u \in U$, $v \in V$ with $\langle u, v\rangle \geq l$ but there exists a stable matching μ with $\langle u', v'\rangle < l$ for all pairs $(m_{u'}, w_{v'}) \in \mu$. Then (m_u, w_v) is clearly a blocking pair for μ which is a contradiction. $\square$

4.5.3 Verifying Stable Matchings

In this section we give a reduction from the MaxInnProd problem to the problem of verifying a stable matching, showing that this problem is also SETH-hard.

Theorem 4.5.2. *Assuming* SETH*, for any* $\varepsilon > 0$*, there is a* c *such that verifying a stable matching in the Boolean* d*-attribute model with* $d = c\log n$ *dimensions requires time* $\Omega(n^{2-\varepsilon})$.

Proof. We give a reduction from MaxInnProd with sets $U, V \subseteq \{0,1\}^d$ where $|U| = |V| = n$ and threshold l. We construct a matching market with $2n$ men and women in the d'-attribute model with $d' = d + 2(l-1)$. Since $d' < 3d$ the theorem then follows immediately from the SETH-hardness of MaxInnProd.

For $u \in U$, let m_u be a man in the matching market with attributes and weights $A(m_u) = \alpha(m_u) = u \circ 1^{l-1} \circ 0^{l-1}$ where we use $\circ$ for concatenation. Similarly, for $v \in V$ we have a woman w_v with $A(w_v) = \alpha(w_v) = v \circ 0^{l-1} \circ 1^{l-1}$. We further introduce *dummy women* w'_u for $u \in U$ with $A(w'_u) = \alpha(w'_u) = 0^d \circ 1^{l-1} \circ 0^{l-1}$ and *dummy men* m'_v for $v \in V$ with $A(m'_v) = \alpha(m'_v) = 0^d \circ 0^{l-1} \circ 1^{l-1}$.

We claim that the matching consisting of pairs (m_u, w'_u) for all $u \in U$ and (m'_v, w_v) for all $v \in V$ is stable if and only if there is no pair $u \in U$, $v \in V$ with $\langle u, v \rangle \geq l$. For $u, u' \in U$ we have $\mathrm{val}_{m_u}(w'_{u'}) = \mathrm{val}_{w'_{u'}}(m_u) = l-1$, and for $v, v' \in V$ we have $\mathrm{val}_{w_v}(m'_{v'}) = \mathrm{val}_{m'_{v'}}(w_v) = l-1$. In particular, any pair in μ has (symmetric) value $l-1$. Hence there is a blocking pair with respect to μ if and only if there is a pair with value at least l. For $u \neq u'$ and $v \neq v'$ the pairs $(m_u, w'_{u'})$ and $(w_v, m'_{v'})$ can never be blocking pairs as their value is $l-1$. Furthermore for any pair of dummy nodes w'_u and m'_v we have $\mathrm{val}_{m'_v}(w'_u) = \mathrm{val}_{w'_u}(m'_v) = 0$, thus no such pair can be a blocking pair either. This leaves pairs of real nodes as the only candidates for blocking pairs. For non-dummy nodes m_u and w_v we have $\mathrm{val}_{m_u}(w_v) = \mathrm{val}_{w_v}(m_u) = \langle u, v \rangle$ so (m_u, w_v) is a blocking pair if and only if $\langle u, v \rangle \geq l$. $\square$

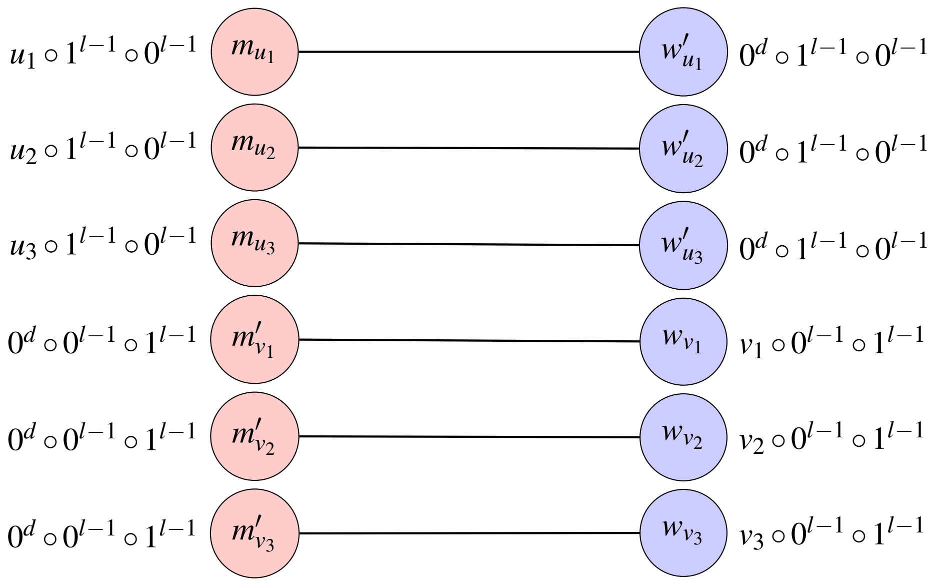

$u_1 \circ 1^{l-1} \circ 0^{l-1}$
m_{u_1}
w'_{u_1}
$0^d \circ 1^{l-1} \circ 0^{l-1}$
$u_2 \circ 1^{l-1} \circ 0^{l-1}$
m_{u_2}
w'_{u_2}
$0^d \circ 1^{l-1} \circ 0^{l-1}$
$u_3 \circ 1^{l-1} \circ 0^{l-1}$
m_{u_3}
w'_{u_3}
$0^d \circ 1^{l-1} \circ 0^{l-1}$
$0^d \circ 0^{l-1} \circ 1^{l-1}$
m'_{v_1}
w_{v_1}
$v_1 \circ 0^{l-1} \circ 1^{l-1}$
$0^d \circ 0^{l-1} \circ 1^{l-1}$
m'_{v_2}
w_{v_2}
$v_2 \circ 0^{l-1} \circ 1^{l-1}$
$0^d \circ 0^{l-1} \circ 1^{l-1}$
m'_{v_3}
w_{v_3}
$v_3 \circ 0^{l-1} \circ 1^{l-1}$

Figure 4.1. A representation of the reduction from maximum inner product to verifying a stable matching

4.5.4 Checking a Stable Pair

In this section we give a reduction from the MaxInnProd problem to the problem of checking whether a given pair is part of any or all stable matchings (StablePair), showing that these questions are SETH-hard when $d = c \log n$ for some constant c. For general preferences, both questions can be solved in time $O(n^2)$ [87, 70] and are known to require quadratic time [114, 132, 68].

Theorem 4.5.3. *Assuming* SETH*, for any $\varepsilon > 0$, there is a c such that determining whether a given pair is part of any or all stable matchings in the Boolean d-attribute model (*BooleanAttributeStablePair*) with $d = c \log n$ dimensions requires time $\Omega(n^{2-\varepsilon})$.*

Proof. We again give a reduction from MaxInnProd with sets $U, V \subseteq \{0,1\}^d$ where $|U| = |V| = n$ and threshold l. We construct a matching market with $2n$ men and women in the d'-attribute model with $d' = 7d + 7(l-1) + 18$. Since $d' < 15d$ the theorem then follows immediately from the SETH-hardness of MaxInnProd.

For simplicity, we will first describe the preference scheme, then provide weight and attribute vectors that result in those preferences. For $u \in U$, let m_u be a man in the matching market and for $v \in V$ we have a woman w_v. We also have $n-1$ *dummy men* $m_i : i = 1 \dots n-1$ and $n-1$ *dummy women* $w_j : j = 1 \dots n-1$. Finally, we have a *special man* m^* and *special woman* w^*. This special pair is the one we will test for stability. Let the preferences be

$$
\begin{aligned}
m_u :& \{w_v : \langle u,v\rangle \geq l\} \succ \{w_j\}_{j=1}^{n-1} \succ w^* \succ \{w_v : \langle u,v\rangle < l\} && \forall u \in U \\
m_i :& \{w_v\} \succ \{w_j\}_{j=1}^{n-1} \succ w^* && \forall i \in \{1 \dots n-1\} \\
m^* :& w^* \succ \{w_v\} \succ \{w_j\}_{j=1}^{n-1} && \\
w_v :& \{m_u : \langle u,v\rangle \geq l\} \succ \{m_i\}_{i=1}^{n-1} \succ m^* \succ \{m_u : \langle u,v\rangle < l\} && \forall v \in V \\
w_j :& \{m_u\} \succ \{m_i\}_{i=1}^{n-1} \succ m^* && \forall j \in \{1 \dots n-1\} \\
w^* :& \{m_i\}_{i=1}^{n-1} \succ \{m_u\} \succ m^* &&
\end{aligned}
$$

so that, for example, man m_u corresponding to $u \in U$ will most prefer women w_v for some $v \in V$ with $\langle u,v\rangle \geq l$ (in decreasing order of $\langle u,v\rangle$), then all of the dummy women (equally), then the special woman w^*, and finally the remaining women w_v (in decreasing order of $\langle u,v\rangle$).

First suppose for some $\hat{u} \in U$ and $\hat{v} \in V$ we have $\langle \hat{u},\hat{v}\rangle \geq l$ and let this be the pair with largest inner product. Now consider the deferred acceptance algorithm for finding the woman-optimal stable matching. First, $w_{\hat{v}}$ will propose to $m_{\hat{u}}$ and will be accepted. The dummy women will propose to the remaining men corresponding to U. Then any other woman w_v will be accepted by either a dummy man or a man m_u, causing the dummy woman matched with him to move to a dummy man. In any case, all men besides m^* are matched to a woman they prefer over w^*, so when she proposes to them, they will reject her. Thus w^* will match with m^*. Since w^* receives her least preferred choice in the woman optimal stable matching, (m^*, w^*) is a pair in every stable matching.

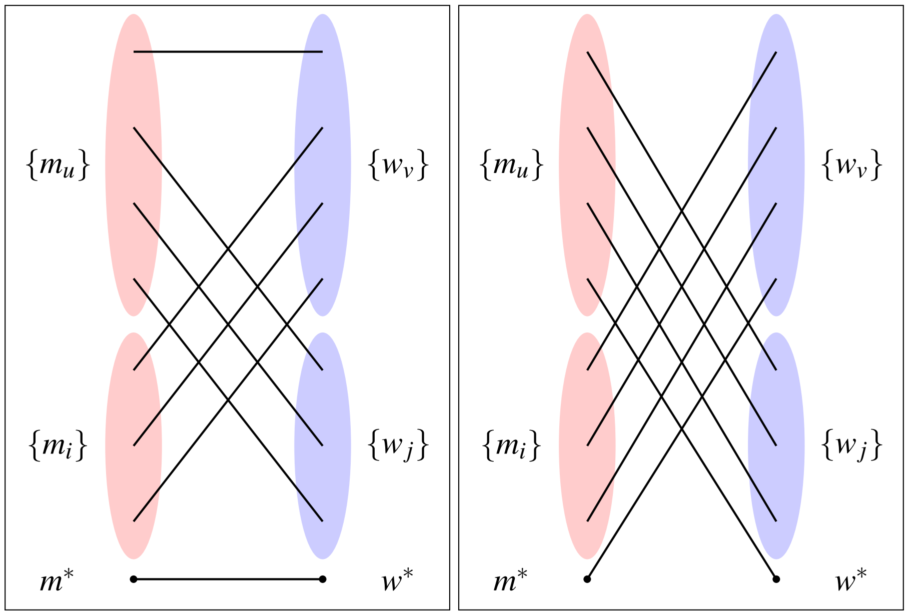
Now suppose $\langle u, v\rangle < l$ for every $u \in U, v \in V$. Consider the deferred acceptance algorithm for finding the man-optimal stable matching. First, the dummy men will propose to the women corresponding to V and will be accepted. Then every man m_u will propose to the dummy women, but only $n-1$ of them can be accepted. The remaining one will propose to w^*. When m^* proposes to w^*, she rejects him, causing him to eventually be accepted by the available woman w_v. Thus m^* will not match with w^* in any stable matching since she is his most preferred choice but he is not matched with her in the man-optimal stable matching, so (m^*, w^*) is not a pair in any stable matching. Section 4.5.4 demonstrates each of these cases.

Since the stable pair questions for whether (m^*, w^*) are a stable pair in any or all stable matchings are equivalent with these preferences, this reduction works for both.

Finally, we claim the following vectors realize the preferences above for the attribute model. We leave it to the reader to verify this. As in our other hardness reductions, the weight and attribute vectors are identical for each participant.

$$\begin{array}{rccccccccccc}
m_u: & u^7 & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & 1^6 & \circ & 0^6 & \circ & 0^6 \\
m_i: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 1^6 & \circ & 0^6 \\
m^*: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 0^6 & \circ & 1^6 \\
\hline
w_v: & v^7 & \circ & 0^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 1^6 & \circ & (1 \circ 0^5) \\
w_j: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & 1^6 & \circ & (1^5 \circ 0) & \circ & 0^6 \\
w^*: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & (1^3 \circ 0^3) & \circ & (1^4 \circ 0^2) & \circ & (1^2 \circ 0^4)
\end{array}$$

□



(a) A representative stable matching when there is a pair (u,v) with $\langle u,v\rangle \geq l$

(b) A representative stable matching when $\langle u,v\rangle < l$ for every pair (u,v)

Figure 4.2. A representation of the reduction from maximum inner product to checking a stable pair

This reduction also has consequences on the existence of nondeterministic algorithms for the StablePair problem assuming the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH)

NSETH stipulates that for CNF-sat there is no proof of unsatisfiability that can be checked deterministically in time $\Omega(2^{(1-\varepsilon)n})$. See Chapter 6 for an extensive treatment of NSETH.

Assuming NSETH, any problem that is SETH-hard at time $T(n)$ under deterministic reductions either require $T(n)$ time nondeterministically or co-nondeterministically, i.e. either there is no proof that an instance is true or there is no proof that an instance is false that can be checked in time faster than $T(n)$. Note that all reductions in this chapter are deterministic. In particular, the MaxInnProd problem does not have a $O(N^{2-\varepsilon})$ co-nondeterministic time algorithm for any $\varepsilon > 0$ assuming NSETH, since it has a simple linear time nondeterministic algorithm.

Since the reduction of Theorem 4.5.3 is a simple reduction that maps a true instance of MaxInnProd to a true instance of the StablePair problem, we can conclude that the StablePair problem is also hard co-nondeterministically.

Corollary 4.5.1. *Assuming* NSETH, *for any $\varepsilon > 0$, there is a c such that determining whether a given pair is part of any or all stable matchings in the Boolean d-attribute model with $d = c\log n$ dimensions requires co-nondeterministic time $\Omega(n^{2-\varepsilon})$.*

We also have a reduction so that the given pair is stable in any or all stable matchings if and only if there is not a pair of vectors with large inner product. This shows that the StablePair problem is also hard nondeterministically.

Theorem 4.5.4. *Assuming* NSETH, *for any $\varepsilon > 0$, there is a c such that determining whether a given pair is part of any or all stable matchings in the Boolean d-attribute model with $d = c\log n$ dimensions requires nondeterministic time $\Omega(n^{2-\varepsilon})$.*

Proof. This reduction uses the same setup as the one in Theorem 4.5.3 except that we now have n dummy men and women instead of $n-1$ and we slightly change the preferences as follows:

$$m_u : \{w_v : \langle u,v\rangle \geq l\} \succ \{w_j\}_{j=1}^n \succ w^* \succ \{w_v : \langle u,v\rangle < l\} \qquad \forall u \in U$$

$$m_i : \{w_v\} \succ \mathbf{w}^* \succ \{\mathbf{w_j}\}_{\mathbf{j=1}}^{\mathbf{n}} \qquad \forall i \in \{1\ldots n\}$$

$$m^* : w^* \succ \{w_v\} \succ \{w_j\}_{j=1}^n$$

$$w_v : \{m_u : \langle u,v\rangle \geq l\} \succ \{m_i\}_{i=1}^n \succ m^* \succ \{m_u : \langle u,v\rangle < l\} \qquad \forall v \in V$$

$$w_j : \{m_u\} \succ \{m_i\}_{i=1}^n \succ m^* \qquad \forall j \in \{1\ldots n\}$$

$$w^* : \{m_i\}_{i=1}^n \succ \{m_u\} \succ m^*$$

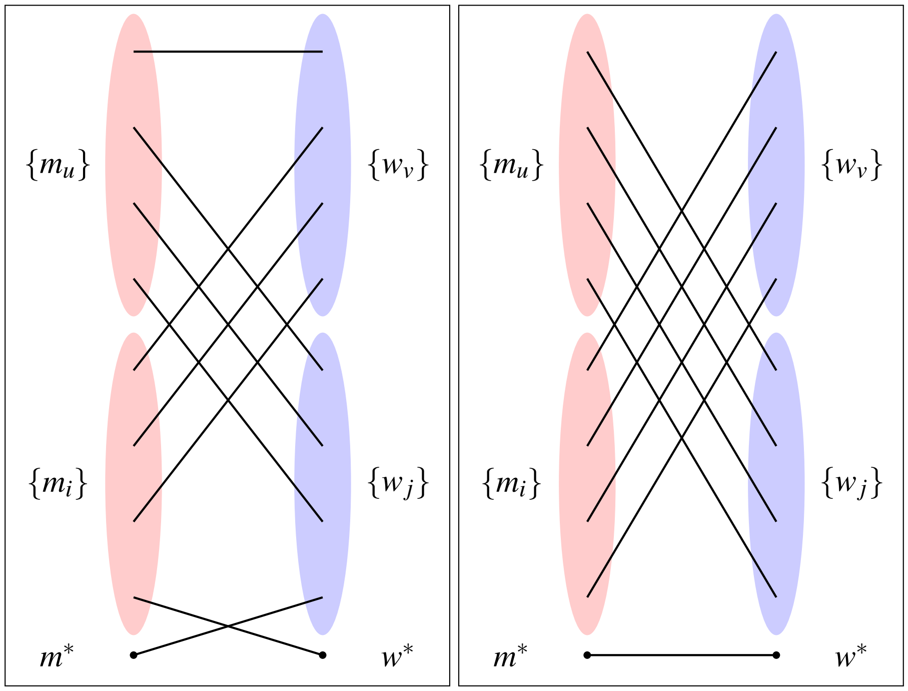
First suppose for some $\hat{u} \in U$ and $\hat{v} \in V$ we have $\langle \hat{u},\hat{v}\rangle \geq l$ and let this be the pair with largest inner product. Consider the deferred acceptance algorithm for finding the man-optimal stable matching. First, some of the men corresponding to U will propose to the women corresponding to V and at least $m_{\hat{u}}$ will be accepted by $w_{\hat{v}}$. The remaining men corresponding to U will be accepted by dummy women. The dummy men will propose to the women corresponding to V but not all can be accepted. These rejected dummy men will propose to w^* who will accept one. Then when m^* proposes to w^* she will reject him, as will the women corresponding to V, so he will be matched with a dummy woman. Since m^* and w^* are not matched in the man optimal stable matching, (m^*,w^*) is not a pair in any stable matching.

Now suppose $\langle u,v\rangle < l$ for every $u \in U$,$v \in V$ and consider the deferred acceptance algorithm for finding the woman-optimal stable matching. First, the dummy women will propose to the men corresponding to U and will be accepted. Then every woman w_v will propose to the dummy men and be accepted. Since every man besides m^* is matched with a woman he prefers to w^*, when she proposes to them, she will be rejected, so she will pair with m^*. Since w^* receives her least preferred choice in the woman optimal stable matching, (m^*, w^*) is a pair in every stable matching. Section 4.5.4 demonstrates each of these cases.

We can amend the vectors from Theorem 4.5.3 as follows so that they realize the changed preferences with the attribute model.

$$\begin{array}{rccccccccccccc}
m_u: & u^7 & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & 1^6 & \circ & 0^6 & \circ & 0^6 \\
m_i: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 1^6 & \circ & 0^6 \\
m^*: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 0^6 & \circ & 1^6 \\
\hline
w_v: & v^7 & \circ & 0^{7(l-1)} & \circ & 1^{7(l-1)} & \circ & 0^6 & \circ & 1^6 & \circ & (1 \circ 0^5) \\
w_j: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & 1^6 & \circ & (\mathbf{1^3} \circ \mathbf{0^3}) & \circ & 0^6 \\
w^*: & 0^{7d} & \circ & 1^{7(l-1)} & \circ & 0^{7(l-1)} & \circ & (1^3 \circ 0^3) & \circ & (1^4 \circ 0^2) & \circ & (1^2 \circ 0^4)
\end{array}$$

□



(a) A representative stable matching when there is a pair (u,v) with $\langle u,v\rangle \geq l$

(b) A representative stable matching when $\langle u,v\rangle < l$ for every pair (u,v)

Figure 4.3. A representation of the reduction from maximum inner product to checking a stable pair such that a true maximum inner product instance maps to a false stable pair instance

We would like to point out that the results above on the hardness for (co-)nondeterministic algorithms do not apply to Merlin-Arthur (MA) algorithms, i.e. algorithms with access to both nondeterministic bits and randomness. Williams [148] gives fast MA algorithms for a number of SETH-hard problems, and the same techniques also yield a $O(dn)$ time MA algorithm for the verification of a stable matching in the Boolean attribute model with d attributes. We can obtain MA algorithms with time $O(dn)$ for finding stable matchings and certifying that a pair is in at least one stable matching by first nondeterministically guessing a stable matching.

4.6 Other Succinct Preference Models

In this section, we provide subquadratic algorithms for other succinct preference models, single-peaked and geometric, which are motivated by economics.

4.6.1 One Dimensional Single-Peaked Preferences

Formally, we say the men's preferences over the women in a matching market are *single-peaked* if the women can be ordered as points along a line ($p(w_1) < p(w_2) < \cdots < p(w_n)$) and for each man m there is a point $q(m)$ and a binary preference relation $\succ_m$ such that if $p(w_i) \leq q(m)$ then $p(w_i) \succ_m p(w_j)$ for $j < i$ and if $p(w_i) \geq q(m)$ then $p(w_i) \succ_m p(w_j)$ for $j > i$. Essentially, each man prefers the women that are closest to his ideal point $q(m)$. One example of a preference relation for m would be the distance from $q(m)$. If the women's preferences are also single-peaked then we say the matching market has single peaked preferences. Since these preferences only consist of the p and q values and the preference relations for the participants, they can be represented succinctly as long as the relations require subquadratic space.

Verifying a Stable Matching for Single-Peaked Preferences

Here we demonstrate a subquadratic algorithm for verifying if a given matching is stable when the preferences of the matching market are single-peaked. We assume that the preference relations can be computed in constant time.

Theorem 4.6.1. *There is an algorithm to verify a stable matching in the single-peaked preference model in* $\mathrm{O}(n \log n)$ *time.*

Proof. Let $p(m_i)$ be the point associated with man m_i, $q(m_i)$ be m_i's preference point, and $\succ_{m_i}$ be m_i's preference relation. The women's points are denoted analogously. We assume that $p(m_i) < p(m_j)$ if and only if $i < j$ and the same for the women. Let μ be the given matching we are to check for stability.

First, for each man m, we compute the intervals along the line of women which includes all women m strictly prefers to $\mu(m)$. If this interval is empty, m is with his most preferred woman and cannot be involved in any blocking pairs so we can ignore him. For all nonempty intervals each endpoint is $p(w)$ for some woman w. We also compute these intervals for the women. Note that for any man m and woman w, (m,w) is a blocking pair for μ if and only if m is in w's interval and w is in m's interval.

We will process each of the women in order from w_1 to w_n maintaining a balanced binary search tree of the men who prefer that woman to their partners. This will allow us to easily check if she prefers any of them by seeing if any elements in the tree are between the endpoints of her interval. Initially this tree is empty. When processing a woman w, we first add any man m whose interval begins with w to the search tree. Then we check to see if w prefers any men in the tree. If so, we know the matching is not stable. Otherwise, we remove any man m from the tree whose interval ends with w and proceed to the next woman. Algorithm 5 provides pseudocode for this algorithm.

Computing the intervals requires $O(n \log n)$. Since we only insert each man into the tree at most once, maintaining the tree requires $O(n \log n)$. The queries also require $O(\log n)$ for each woman so the total time is $O(n \log n)$. $\square$

Algorithm 5: Single-Peaked Stable Matching Verification

```
for each woman $w$ do
    Create two empty lists $w.begin$ and $w.end$
    Use binary search to find the leftmost man $m$ and rightmost man $m'$ that
      $w$ prefers to $\mu(w)$ if any (otherwise remove $w$)
    Let $w.s = p(m)$ and $w.t = p(m')$
for each man $m$ do
    Use binary search to find the leftmost woman $w$ and rightmost woman $w'$
      that $m$ prefers to $\mu(m)$ if any (otherwise ignore $m$)
    Add $m$ to $w.begin$ and $w'.end$
Initialize an empty balanced binary search tree $T$
for $i = 1$ to $n$ do
    for $m \in w_i.begin$ do
        $T.\text{insert}(p(m))$
    if there are any points $p(m)$ in $T$ between $w_i.s$ and $w_i.t$ then
        return $(m, w_i)$ is a blocking pair
    for $m \in w_i.end$ do
        $T.\text{delete}(p(m))$
return $\mu$ is stable
```

Remarks on Finding a Stable Matching for Single-Peaked Preferences

The algorithm in [21] relies on the observation that there will always be a pair or participants who are each other's first choice with narcissistic single-peaked preferences. Thus a greedy approach where one such pair is selected and then removed works well. However, this is not the case when we remove the narcissistic assumption. In fact, as with the two-list case, Table 4.4 presents an example where no participant is matched with their top choice in the unique stable matching. Note that the preferences for the men and women are symmetric. The reader can verify that these preferences can be realized in the single-peaked preference model using the orderings $p(m_1) < p(m_2) < p(m_3) < p(m_4)$ and $p(w_1) < p(w_2) < p(w_3) < p(w_4)$ and that the unique stable matching is $\{(m_1,w_4),(m_2,w_2),(m_3,w_3),(m_4,w_1)\}$ where no participant receives their first choice.

Table 4.4. Single-peaked preferences where no participant receives their top choice in the stable matching

Man	Preference List	Woman	Preference List
m_1	$w_3 \succ w_2 \succ w_4 \succ w_1$	w_1	$m_3 \succ m_2 \succ m_4 \succ m_1$
m_2	$w_3 \succ w_2 \succ w_4 \succ w_1$	w_2	$m_3 \succ m_2 \succ m_4 \succ m_1$
m_3	$w_4 \succ w_3 \succ w_2 \succ w_1$	w_3	$m_4 \succ m_3 \succ m_2 \succ m_1$
m_4	$w_2 \succ w_1 \succ w_3 \succ w_4$	w_4	$m_2 \succ m_1 \succ m_3 \succ m_4$

Also no greedy algorithm following the model inspired by [52] will succeed for single-peak preferences because the preferences in Table 4.2 can be realized in the single-peaked preference model using the orderings $p(m_1) < p(m_2) < p(m_3)$ and $p(w_1) < p(w_2) < p(w_3)$.

4.6.2 Geometric Preferences

We say the men's preferences over the women in a matching market are *geometric* in d dimensions if each women w is defined by a *location* $p(w)$ and for each man m there is an *ideal* $q(m)$ such that m prefers woman w_1 to w_2 if and only if $\|p(m)-q(w_1)\|_2^2 < \|p(m)-q(w_2)\|_2^2$, i.e. $p(w_1)$ has smaller euclidean distance from the man's ideal than $p(w_2)$. If the women's preferences are also geometric we call the matching market geometric. We further call the preferences *narcissistic* if $p(x) = q(x)$ for every participant

x. Our results for the attribute model extend to geometric preferences.

Note that one-dimensional geometric preferences are a special case of single-peaked preferences. As such, geometric preferences might be used to model preferences over political candidates who are given a score on several (linear) policy areas, e.g. protectionist vs. free trade and hawkish vs. dovish foreign policy.

Arkin et al. [16] also consider geometric preferences, but restrict themselves to the narcissistic case. Our algorithms do not require the preferences to be narcissistic, hence our model is more general. On the other hand, our lower bounds for large dimensions also apply to the narcissistic special case. While Arkin et al. take special care of different notions of stability in the presence of ties, we concentrate on weakly stable matchings. Although we restrict ourselves to the StableMatching problem for the sake of presentation, all lower bounds and verification algorithms naturally extend to the StableRoommate problem. Since all proofs in this section are closely related those for the attribute model, we restrict ourselves to proof sketches highlighting the main differences.

Theorem 4.3.1 extends immediately to the geometric case without any changes in the proof.

Corollary 4.6.1 (Geometric version of Theorem 4.3.1)**.** *There is an algorithm to find a stable matching in the d-dimensional geometric model with at most a constant C distinct values in time* $\mathrm{O}(C^{2d} n(d+\log n))$.

For the verification of a stable matching with real-valued vectors we use a standard lifting argument.

Corollary 4.6.2 (Geometric version of Theorem 4.4.1)**.** *There is an algorithm to verify a stable matching in the d-dimensional geometric model with real-valued locations and ideals in time* $\tilde{\mathrm{O}}(n^{2-1/2(d+1)})$

Proof. Let $q \in \mathbb{R}^d$ be an ideal and let $a, b \in \mathbb{R}^d$ be two locations. Define $q', a', b' \in \mathbb{R}^{d+1}$ as $q' = (q_1, \dots, q_d, -1/2)$, $a' = (a_1, \dots, a_d, \sum_{i=1}^d a_i^2)$ and $b' = (b_1, \dots, b_d, \sum_{i=1}^d b_i^2)$.

We have $\langle a', q \rangle = 1/2 \sum_{i=1}^d q_i - 1/2 \|q - a\|_2^2$. Hence we get $\|q - a\|_2^2 < \|q - b\|_2^2$ if and only if $\langle q', a' \rangle > \langle q', b' \rangle$, so we can reduce the StableMatching problem in the d-dimensional geometric model to the $d+1$-attribute model. □

For the Boolean case, we can adjust the proof of Theorem 4.4.3 by using a threshold of parities instead of a threshold of conjunctions. The degree of the resulting polynomial remains the same.

Corollary 4.6.3 (Geometric version of Theorem 4.4.3). *In the geometric model with n men and women, with locations and ideals in $\{0,1\}^d$ with $d = c \log n$, there is a randomized algorithm to decide if a given matching is stable in time $\tilde{O}(n^{2-1/O(c \log^2(c))})$ with error probability at most $1/3$.*

For lower bounds we reduce from the minimum Hamming distance problem (MinHammDistance) which is SETH-hard with the same parameters as the MaxInnProd problem [15]. The Hamming distance of two Boolean vectors is exactly their squared euclidean distance, hence a matching market where the preferences are defined by Hamming distances is geometric.

Problem 11 (MinHammDistance). *For any d and input l, the minimum Hamming distance problem is to decide if two input sets $U, V \subseteq \{0,1\}^d$ with $|U| = |V| = n$ have a pair $u \in U$, $v \in V$ such that $\|u - v\|_2^2 < l$.*

Lemma 4.6.1 ([15]). *Assuming* SETH, *for any $\varepsilon > 0$, there is a c such that solving the* MinHammDistance *problem on $d = c \log n$ dimensions requires time $\Omega(n^{2-\varepsilon})$.*

For the hardness of finding a stable matching, the construction from Theorem 4.5.1 works without adjustments.

Corollary 4.6.4 (Geometric version of Theorem 4.5.1)**.** *Assuming* SETH*, for any* $\varepsilon > 0$*, there is a* c *such that finding a stable matching in the (Boolean)* d*-dimensional geometric model with* $d = c\log n$ *dimensions requires time* $\Omega(n^{2-\varepsilon})$.

For the hardness of verifying a stable matching, the construction is as follows.

Corollary 4.6.5 (Geometric version of Theorem 4.5.2)**.** *Assuming* SETH*, for any* $\varepsilon > 0$*, there is a* c *such that verifying a stable matching in the (Boolean)* d*-dimensional geometric model with* $d = c\log n$ *dimensions requires time* $\Omega(n^{2-\varepsilon})$.

Proof. Let $U, V \subseteq \{0,1\}^d$ be the inputs to the $\mathsf{MinHammDistance}$ problem and let l be the threshold.

For every $u \in U$, define a real man m_u with both ideal and location as $u \circ 0^l$ and a dummy woman w'_u with ideal and location $u \circ 1^l$. Symmetrically for $v \in V$ define w_v with $v \circ 0^l$ and m'_v with $v \circ 1^l$. The matching (m_u, w'_u) for all $u \in U$ and (w_v, m'_v) for all $v \in V$ is stable if and only if there is there is no pair u, v with Hamming distance less than l. $\square$

The hardness results for checking a stable pair also translate to the geometric model. In particular, since both variants of the proof extend to the geometric model we have the same consequences for nondeterministic algorithms.

Corollary 4.6.6 (Geometric version of Theorem 4.5.3)**.** *Assuming* SETH*, for any* $\varepsilon > 0$*, there is a* c *such that determining whether a given pair is part of any or all stable matchings in the (Boolean)* d*-dimensional geometric model with* $d = c\log n$ *dimensions requires time* $\Omega(n^{2-\varepsilon})$.

Proof. We again reduce from the $\mathsf{MinHammDistance}$ problem. We assume without loss of generality that d is even and the threshold l is exactly $d/2+1$, i.e. the instance is true if and only if there are vectors u, v with Hamming distance at most $d/2$. We can reduce to this case from any other threshold by padding the vectors.

We use the same preference orders as in the d-attribute model. The following narcissistic instance realizes the preference order from Theorem 4.5.3. For a vector $u \in \{0,1\}^d$, $\overline{u}$ denotes its component-wise complement.

$$\begin{aligned}
m_u &: (u \circ \overline{u} \circ u \circ \overline{u})^3 \circ 000000000\\
m_i &: 0^{12d} \circ 100000000\\
m^* &: 0^{12d} \circ 001111111\\
w_v &: (v \circ \overline{v} \circ v \circ \overline{v})^3 \circ 000000000\\
w_j &: (0^{2d} \circ 1^{2d})^3 \circ 010000000\\
w^* &: (0^{2d} \circ 1^{2d})^3 \circ 101110000
\end{aligned}$$

Likewise the preference orders for Theorem 4.5.4 are achieved by the following vectors.

$$\begin{aligned}
m_u &: (u \circ \overline{u} \circ u \circ \overline{u})^3 \circ 000000000\mathbf{00}\\
m_i &: 0^{12d} \circ 100000000\mathbf{00}\\
m^* &: 0^{12d} \circ 001111111\mathbf{00}\\
w_v &: (v \circ \overline{v} \circ v \circ \overline{v})^3 \circ 000000000\mathbf{00}\\
w_j &: (0^{2d} \circ 1^{2d})^3 \circ 010000000\mathbf{11}\\
w^* &: (0^{2d} \circ 1^{2d})^3 \circ 101110000\mathbf{00}
\end{aligned}$$

□

4.6.3 Strategic Behavior

With geometric and single-peaked preferences, we assume that the participants are not allowed to misrepresent their location points. Rather they may only misrepresent their preference ideal. As such, the results of this section do not apply when preferences are narcissistic.

Theorem 4.6.2. *There is no strategy proof algorithm to find a stable matching in the geometric preference model.*

Proof. We consider one-dimensional geometric preferences. Let the preference points and ideals be as given in Table 4.5 which yield the depicted preference lists. As in the proof for Theorem 4.3.3, there are two stable matchings: the man-optimal matching

$$\{(m_1,w_3),(m_2,w_1),(m_3,w_2)\}$$

and the woman-optimal matching

$$\{(m_1,w_3),(m_2,w_2),(m_3,w_1)\}$$

. However, if w_2 changes her ideal to $5/3$ then her preference list is $m_2 \succ m_1 \succ m_3$. Now there is a unique stable matching which is $\{(m_1,w_3),(m_2,w_2),(m_3,w_1)\}$, the woman-optimal stable matching from the original preferences. Therefore, any mechanism that does not always output the woman optimal stable matching can be manipulated by the women to their advantage. Similarly, any mechanism that does not always output the man-optimal matching could be manipulated by the men in some instances. Thus there is no strategy-proof mechanism for geometric preferences. □

Table 4.5. Geometric preferences that can be manipulated

Man	Location (p)	Ideal (q)
m_1	1	7/3
m_2	2	1
m_3	3	5/3

Woman	Location (p)	Ideal (q)
w_1	1	3
w_2	2	7/3
w_3	3	3

Man	Preference List
m_1	$w_2 \succ w_3 \succ w_1$
m_2	$w_1 \succ w_2 \succ w_3$
m_3	$w_2 \succ w_1 \succ w_3$

Woman	Preference List
w_1	$m_3 \succ m_2 \succ m_1$
w_2	$m_2 \succ m_3 \succ m_1$
w_3	$m_3 \succ m_2 \succ m_1$

Since one-dimensional geometric preferences are a special case of single-peaked preferences the following corollary results directly from Theorem 4.6.2.

Corollary 4.6.7. *There is no strategy proof algorithm to find a stable matching in the single-peaked preference model.*

4.7 Conclusion and Open Problems

We give subquadratic algorithms for finding and verifying stable matchings in the d-attribute model and d-list model. We also show that, assuming SETH, one can only hope to find such algorithms if the number of attributes d is bounded by $O(\log n)$.

For a number of cases there is a gap between the conditional lower bound and the upper bound. Our algorithms with real attributes and weights are only subquadratic if the dimension is constant. Even for small constants our algorithm to find a stable matching is not tight, as it is not subquadratic for any $d = O(\log n)$. The techniques we use when the attributes and weights are small constants do not readily apply to the more general case.

There is also a gap between the time complexity of our algorithms for finding a stable matching and verifying a stable matching. It would be interesting to either close or explain this gap. On the one hand, subquadratic algorithms for finding a stable matching would demonstrate that the attribute and list models are computationally simpler than the general preference model. On the other hand, proving that there are no subquadratic algorithms would show a distinction between the problems of finding and verifying a stable matching in these settings which does not exist for the general preference model. Currently, we do not have a subquadratic algorithm for finding a stable matching even in the 2-list case, while we have an optimal algorithm for verifying a stable matching for d lists. This 2-list case seems to be a good starting place for further research.

Additionally it is worth considering succinct preference models for other computational problems that involve preferences to see if we can also develop improved algorithms for these problems. For example, the Top Trading Cycles algorithm [134] can be made to run in subquadratic time for d-attribute preferences (when d is constant) using the ray shooting techniques applied in this chapter to find participants' top choices.

Chapter 4 is based on material as it appears in the following publications: Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. "Subquadratic algorithms for succinct stable matching." In International Computer Science Symposium in Russia, pp. 294-308. Springer International Publishing, 2016. [110] The author of this dissertation was a principal author of this publication. Marvin Künnemann, Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. "Subquadratic Algorithms for Succinct Stable Matching." arXiv preprint arXiv:1510.06452v5 (2016). [101] The author of this dissertation was a principal author of this publication. Material from Chapter 4 is currently in submission for publication, by Marvin Künnemann, Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We

would like to thank Russell Impagliazzo, Vijay Vazirani, and the anonymous reviewers for helpful discussions and comments on the material in Chapter 4.

Chapter 5

One-Dimensional Dynamic Programming

In this chapter, we investigate the complexity of one-dimensional dynamic programming, or more specifically, of the Least-Weight Subsequence (LWS) problem: Given a sequence of n data items together with weights for every pair of the items, the task is to determine a subsequence S minimizing the total weight of the pairs adjacent in S. A large number of natural problems can be formulated as LWS problems, yielding obvious $O(n^2)$-time solutions.

In many interesting instances, the $O(n^2)$-many weights can be succinctly represented. Yet except for near-linear time algorithms for some specific special cases, little is known about when an LWS instantiation admits a subquadratic-time algorithm and when it does not. In particular, no lower bounds for LWS instantiations have been known before. In an attempt to remedy this situation, we provide a general approach to study the fine-grained complexity of succinct instantiations of the LWS problem. In particular, given an LWS instantiation we identify a highly parallel *core* problem that is subquadratically *equivalent*. This provides either an explanation for the apparent hardness of the problem or an avenue to find improved algorithms as the case may be.

More specifically, we prove subquadratic equivalences between the following pairs (an LWS instantiation and the corresponding core problem) of problems: a low-rank version of LWS and minimum inner product, finding the longest chain of nested boxes and vector domination, and a coin change problem which is closely related to the knapsack problem and $(\min, +)$-Convolution. Using these equivalences and known SETH-hardness results for some of the core problems, we deduce tight conditional lower bounds for the corresponding LWS instantiations. We also establish the $(\min, +)$-Convolution-hardness of the knapsack problem. Furthermore, we revisit some of the LWS instantiations which are known to be solvable in near-linear time and explain their easiness in terms of the easiness of the corresponding core problems.

Dynamic programming (DP) is one of the most fundamental paradigms for designing algorithms and a standard topic in textbooks on algorithms. Scientists from various disciplines have developed DP formulations for basic problems encountered in their applications. However, it is not clear whether the existing (often simple and straightforward) DP formulations are in fact optimal or nearly optimal. Our lack of understanding of the optimality of the DP formulations is particularly unsatisfactory since many of these problems are computational primitives.

Interestingly, there have been recent developments regarding the optimality of standard DP formulations for some specific problems, namely, conditional lower bounds assuming the Strong Exponential Time Hypothesis (SETH) [82]. The longest common subsequence (LongestCommonSubsequence) problem is one such problem for which almost tight conditional lower bounds have been obtained recently. The LongestCommonSubsequence problem is defined as follows:

Problem 12 (LongestCommonSubsequence)**.** *Given two strings x and y of length at most n, compute the length of the longest string z that is a subsequence of both x and y.*

The standard DP formulation for the LongestCommonSubsequence problem involves computing a two-dimensional table requiring $O(n^2)$ steps. This algorithm is only slower than the fastest known algorithm due to Masek and Paterson [105] by a polylogarithmic factor. However, there has been no progress in finding more efficient algorithms for this problem since the 1980s, which prompted attempts as early as in 1976 [13] to understand the barriers for efficient algorithms and to prove lower bounds. Unfortunately, there have not been any nontrivial unconditional lower bounds for this or any other problem in general models of computation. This state of affairs prompted researchers to consider *conditional* lower bounds based on conjectures such as 3-sum conjecture [61] and more recently based on ETH [85] and SETH [82]. Both ETH and SETH proved to be useful to explain the exact complexity of several NP-complete problems (see the survey paper [103]). Surprisingly, Ryan Williams [142] has found a simple reduction from the CNF-sat problem to the OrthogonalVectors problem which under SETH leads to a matching quadratic lower bound for the OrthogonalVectors problem. This in turn led to a number of conditional lower bound results for problems in P (including LongestCommonSubsequence and related problems) under SETH [17, 2, 33, 4, 67]. Also see [149] for a recent survey and Chapter 2 for more detail.

The DP formulation of the LongestCommonSubsequence problem is perhaps the conceptually simplest example of a *two-dimensional* DP formulation. In the standard formulation, each entry of an $n \times n$ table is computed in constant time. The LongestCommonSubsequence problem belongs to the class of alignment problems which, for example, are used to model similarity between gene or protein sequences. Conditional lower bounds have recently been extended to a number of alignment problems [31, 17, 2, 33, 5].

In contrast, there are many problems for which natural quadratic-time DP formulations compute a *one-dimensional* table of length n by spending $O(n)$-time per entry. In this work, we investigate the optimality of such DP formulations and obtain new (conditional) lower bounds which match the complexity of the standard DP formulations.

5.1 The Least-Weight Subsequence (LWS) Problem

In this paper, we investigate the optimality of the standard DP formulation of the LWS problem. A classic example of an LWS problem is AirplaneRefueling [77].

Problem 13 (AirplaneRefueling). *Given airport locations on a line, and a preferred distance per hop k (in miles), we define the penalty for flying k' miles as $(k-k')^2$. The goal is then to find a sequence of airports starting at the first airport and terminating at the last airport that minimizes the sum of the penalties.*

We now define the LWS problem formally.

Problem 14 (LWS). *We are given weights $w_{i,j} \in [-W,W] \cup \{\infty\}$ for every $0 \le i < j \le n$ and an arbitrary function $g : \mathbb{N} \to \mathbb{N}$. The* LWS *problem is to determine $F[n]$ which is defined by the following DP formulation.*

$$F[0] = 0,$$

$$F[j] = \min_{0 \le i < j} g(F[i]) + w_{i,j} \qquad \text{for } j = 1,\dots,n \tag{5.1}$$

We typically consider succinct instantiations of LWS, where the input has subquadratic size (typically $\tilde{O}(n)$) and the weights are a function of the input. In many cases the input is a list of data items $x_0,\dots,x_n$ and $w_{i,j}$ is a function of x_i and x_j.

To formulate AirplaneRefueling as an LWS problem, we let x_i be the location of the ith airport, g be the identity function, and $w_{i,j} = (x_j - x_i - k)^2$.

In the definition of the LWS problem, we did not specify the encoding of the problem (in particular, the type of data items and the representation of the weights

$w_{i,j}$) so we can capture a larger variety of problems: it not only encompasses classical problems such as the PrettyPrinting problem due to Knuth and Plass [97], the AirplaneRefueling problem [77] and the longest increasing subsequence (LIS) problem [60], but also the UnboundedSubsetSum problem [120, 32], which is a more general CoinChange problem that is effectively equivalent to the UnboundedKnapsack problem, the 1DKMeansClustering problem [69], finding longest R-chains for an arbitrary binary relation R (ChainLWS), and many others. For a more complete list of problems definitions, see Section 5.2 and Appendix A.

Under mild assumptions on the encoding of the data items and weights, any instantiation of the LWS problems can be solved in time $O(n^2)$ using the definition for determining the values $F[j], j = 1, \ldots, n$ in time $O(n)$ each. However, the best known algorithms for the LWS problems differ quite significantly in their time complexity. Some problems including PrettyPrinting, AirplaneRefueling and LIS turn out to be solvable in near-linear time, while no subquadratic algorithms are known for UnboundedKnapsack or the general ChainLWS problem.

The main goal of the paper is to investigate the optimality of the LWS DP formulation for various problems by proving conditional lower bounds.

5.1.1 Succinct LWS instantiations

In the extremely long presentation of an LWS problem, the weights $w_{i,j}$ are given explicitly. This is however not a very interesting case from a computational point of view, as the standard DP formulation takes linear time (in the size of the input) to compute $F[n]$. In the example of the airplane refueling problem the size of the input is only $O(n)$ assuming that the values of the data items are bounded by some polynomial in n. For such succinct representations, we ask if the quadratic-time algorithm based on the standard LWS DP formulation is optimal. Our approach is to study several natural succinct versions of the LWS problem (by specifying the type of data items and the weight function[1]) and determine their complexity. We refer to Section 5.2 for examples of succinct instantiations of the LWS problem.

5.1.2 Contributions and Results

The main contributions of our paper include a general framework for reducing succinct LWS instantiations to what we call the *core* problems and proving subquadratic equivalences between them. The subquadratic equivalences are interesting for two reasons. First, they allows us to conclude conditional lower bounds for certain LWS instantiations, where previously no lower bounds are known. Second, subquadratic (or more general fine-grained) equivalences are more useful since they let us translate hardness as well as easiness results.

Our results include tight (up to subpolynomial factors) conditional lower bounds for several LWS instantiations with succinct representations. These instantiations include the CoinChange problem, low rank versions of the LWS problem (LowRankLWS), and the ChainLWS problems. Our results are somewhat more general. We propose a *factorization* of the LWS problem into a *core* problem and a fine-grained reduction from the LWS problem to the core problem. The idea is that core problems (which are often well-know problems) capture the hardness of the LWS problem and act as a potential barrier for more efficient algorithms. While we do not formally define the notion of a core problem, we identify several core problems which share several interesting properties. For example, they do not admit natural DP formulations and are easy to parallelize. In contrast, the quadratic-time DP formulation of LWS problems requires the entries $F[i]$ to be computed in order, suggesting that the general problem might be inherently sequential.

The reductions between LWS problems and core problems involve a natural

[1]In all our applications, the function g is the trivial identity function.

intermediate problem, which we call the Static-LWS problem. We first reduce the LWS problem to the Static-LWS problem in a general way and then reduce the Static-LWS problem to a core problem. The first reduction is divide-and-conquer in nature and is inherently sequential. The latter reduction is specific to the instantiation of the LWS problem. The Static-LWS problem is easy to parallelize and does not have a natural DP formulation. However, the problem is not necessarily a natural problem. The Static-LWS problem can be thought of as a generic core problem, but it is output-intensive.

In the other direction, we show that many of the core problems can be reduced to the corresponding LWS instantiations thus establishing an equivalency between LWS instantiations and their core problems. This equivalence enables us to translate both the hardness and easiness results (i.e., the subquadratic-time algorithms) for the core problems to the corresponding LWS instantiations.

The first natural succinct representation of the LWS problem we consider is the LowRankLWS problem, where the weight matrix $\mathbf{W} = (w_{i,j})$ is of low rank and thus representable as $\mathbf{W} = L \cdot R$ where L and R^{T} are $(n \times n^{o(1)})$-matrices. For this low rank LWS problem, we identify the minimum inner product problem (IntegerMinInnProd) as a suitable core problem. It is only natural and not particularly surprising that IntegerMinInnProd can be reduced to the low-rank LWS problem which shows the SETH-hardness of the low-rank LWS problem. The other direction is more surprising: Inspired by an elegant trick of Vassilevska Williams and Williams [150], we are able to show a subquadratic-time reduction from the (highly sequential) low-rank LWS problem to the (highly parallel) IntegerMinInnProd problem. Thus, the very compact problem IntegerMinInnProd problem captures exactly the complexity of the LowRankLWS problem (under subquadratic reductions).

We also show that the coin change problem is subquadratically equivalent to the $(\min,+)$-Convolution problem. In the coin change problem, the weight matrix $\mathbf{W}$

is succinctly given as a Toeplitz matrix. At this point, the conditional hardness of the $(\min,+)$-Convolutionproblem is unknown. Only very recently and independent of our work, a detailed treatment of Cygan et al. [49] considers quadratic-complexity of $(\min,+)$-Convolution as a hardness assumption and discusses its relation to more established assumptions. The quadratic-time hardness of the $(\min,+)$-Convolution problem would be very interesting, since it is known that the $(\min,+)$-Convolution problem is reducible to the 3-sum problem and the APSP problem (see also [49]). However, recent results give surprising subquadratic-time algorithms for special cases of $(\min,+)$-Convolution [39]. If these subquadratic-time algorithms extend to the general $(\min,+)$-Convolution problem, our equivalence result also provides a subquadratic-time algorithm for the coin change problem and the closely related unbounded knapsack problem. Our reductions also give, as a corollary, a quadratic-time $(\min,+)$-Convolution-based lower bound for the bounded case of knapsack. We remark that independently of our results, [49] gave randomized subquadratic equivalences of $(\min,+)$-Convolution to UnboundedKnapsack (while we give deterministic reductions) and (bounded) Knapsack (where we only give a $(\min,+)$-Convolution-based lower bound).

We next consider the ChainLWS problem: here, we search for the longest subsequence (chain) in the input sequence such that all adjacent pairs in the subsequence are contained in some binary relation R. We show that for any binary relation R satisfying certain conditions the chaining problem is subquadratically equivalent to a corresponding (highly parallel) selection problem. As corollaries, we get equivalences between finding the longest chain of nested boxes (NestedBoxes) and VectorDomination as well as between finding the longest subset chain (SubsetChain) and the orthogonal vectors (OrthogonalVectors) problem. Interestingly, these results have algorithmic implications: known algorithms for low-dimensional VectorDomination and low-dimensional OrthogonalVectors translate to faster algorithms for low-dimensional NestedBoxes and

Table 5.1. Summary of our results on LWS

Name	Weights	Equivalent Core	Reference
CoinChange	Toeplitz matrix: $w_{i,j} = w_{j-i}$	$(\min,+)$-Convolution	Theorem 5.5.1
	Remark: Subquadratically equivalent to UnboundedKnapsack		
LowRankLWS	Low rank representation: $w_{i,j} = \langle \sigma_i, \mu_j \rangle$	IntegerMinInnProd	Theorem 5.4.1
ChainLWS(R)	matrix induced by R: $w_{i,j} = w_j$ if $R(x_i, x_j)$ and ∞ o/w	Selection(R)	Theorem 5.6.1, Theorem 5.6.2
	Remark: Results below are corollaries.		
NestedBoxes	$w_{i,j} = -1$ if $B_j \leq B_i$	VectorDomination	
SubsetChain	$w_{i,j} = -1$ if $S_i \subseteq S_j$	OrthogonalVectors	

Table 5.2. Near-linear time algorithms following from the proposed framework

Name	Weights	Reducible to	Reference
LIS	matrix induced by $R_<$: $w_{i,j} = -1$ if $x_i < x_j$	Sorting	[60], Observation 5.7.1
UnboundedSubsetSum	Toeplitz $\{0,\infty\}$ matrix: $w_{i,j} = w_{j-i} \in \{0,\infty\}$	Convolution	[32], Observation 5.7.2
ConcaveLWS	concave matrix: $w_{i,j} + w_{i',j'} \leq w_{i',j} + w_{i,j'}$ for $i \leq i' \leq j \leq j'$	SMAWK	[77, 65, 140], Observation 5.7.3

SubsetChain for small universe size.

Table 5.1 lists the LWS succinct instantiations (as discussed above) and their corresponding core problems. All LWS instantiations and core problems considered in this paper are formally defined in Section 5.2.

Finally, we revisit classic problems including the longest increasing subsequence problem, the unbounded subset sum problem and the concave LWS problem and analyze the Static-LWS instantiations to immediately infer that the corresponding core problem can be solved in near-linear time. Table 5.2 gives an overview of some of the problems we look at in this context.

5.1.3 Related Work

LWS has been introduced by Hirschberg and Lamore [77]. If the weight function satisfies the *quadrangle inequality*[2] formalized by Yao [151], one obtains the ConcaveLWS problem, for which they give an $O(n \log n)$-time algorithm. Subsequently, improved algorithms solving ConcaveLWS in time $O(n)$ were given [140, 65]. This yields a fairly large class of weight functions (including, e.g., the pretty printing and airplane refueling problems) for which linear-time solutions exist. To generalize this class of problems, further works address convex weight functions[3] [64, 109, 94] as well as certain combinations of convex and concave weight functions [57] and provide near-linear time algorithms. For a more comprehensive overview over these algorithms and further applications of the LWS problem, we refer the reader to Eppstein's PhD thesis [58].

Apart from these notions of concavity and convexity, results on the succinct LWS problems are typically more scattered and problem-specific (see, e.g., [60, 97, 32, 69]; furthermore, a closely related recurrence to LWS pops up when solving BitonicTSP [53]). An exception to this rule is a study of the parallel complexity of LWS [66].

5.1.4 Organization

Section 5.2 defines or redefines all the relevant LWS, core, and intermediate problems. Section 5.3 gives a general reduction from LWS instantiations to Static-LWS that is independent of the representation of the weight matrix. Section 5.4 contains the result on low-rank LWS. Section 5.5 proves the subquadratic equivalence of the coin change problem and $(\min,+)$-Convolution, while Section 5.6 discusses chaining problems and their corresponding selection (core) problem. Our results on near-linear

[2] See Section 5.2 for definitions.

[3] A weight function is convex if it satisfies the inverse of the quadrangle inequality.

time algorithms are given in Section 5.7.

5.2 Preliminaries

In this section, we state the notational conventions and list or recall the main problems considered in this chapter.

For a problem P, we write T^P for its time complexity. We generally assume the word-RAM model of computation with word size $w = \Theta(\log n)$. For most problems defined in this paper, we consider inputs to be integers in the range $[-W, W]$ where W fits in a constant number of words[4]. For vectors, we use d for the dimension and generally assume $d = n^{o(1)}$.

5.2.1 Succinct LWS Instantiations

In the definition of LWS (Appendix A.3) we did not fix the encoding of the problem (in particular the representation of the weights $w_{i,j}$ and the function g). Assuming that g and the weights can be determined in $\tilde{O}(1)$ and that $W = \mathsf{poly}(n)$, this problem can naturally be solved in time $\tilde{O}(n^2)$, by evaluating the central recurrence for each $j = 1, \ldots, n$ – this takes $\tilde{O}(n)$ time for each j, since we take the minimum over at most n expressions that can be evaluated in time $\tilde{O}(1)$ by accessing the previously computed entries $F[0], \ldots, F[j-1]$ as well as computing g. In all our applications, g will be the identity function, hence it will suffice to define the type of data items and the corresponding weight matrix. Throughout this paper, whenever we fix a representation of the weight matrix $\mathbf{W} = (w_{i,j})_{i,j}$, we denote the corresponding problem $\mathsf{LWS}(\mathbf{W})$.

We first list problems considered in this paper that can be expressed as an LWS instantiation. At this point, we typically give the most natural formulations of these problems – the corresponding definitions as LWS instantiations are given in the corresponding

[4]For the purposes of our reductions, even values up to $W = 2^{n^{o(1)}}$ would be fine.

sections.

We start off with a natural succinct low-rank version of LWS.

Problem 15 (LowRankLWS)**.** LowRankLWS *is the* LWS *problem where the weight matrix* $\mathbf{W}$ *is of rank* $d \ll n$*. The input is given succinctly as two matrices* A *and* B*, which are* $(n \times d)$*- and* $(d \times n)$*-matrices respectively, and* $\mathbf{W} = A \cdot B$*.*

Alternatively, LowRankLWS may be interpreted in the following way: There are places $0, 1, \ldots, n$, each of which is equipped with an in- and an out-vector. The cost of going from place i to j is then defined as the inner product of the out-vector of i with the in-vector of j, and the task is to compute the minimum-cost monotonically increasing path to reach place n starting from 0. In Section 5.4, we discuss the complexity of LowRankLWS.

We consider the following coin change problem and variations of Knapsack.

Problem 16 (CoinChange)**.** *Given a weight sequence* $x_1, \ldots, x_n$ *with* $x_i \in [-W, W] \cup \{\infty\}$*, that is the coin with value* i *has weight* x_i*. Find the weight of the multiset of denominations* I *such that* $\sum_{i \in I} i = n$ *and the sum of the weights* $\sum_{i \in I} x_i$ *is minimized.*

We can restate CoinChange as an LWS instantiation where $w_{i,j} = x_{j-i}$.

The unbounded Knapsack problem is a close relative of CoinChange.

Problem 17 (UnboundedKnapsack)**.** *We are given a sequence of profits* $p = (p_1, \ldots, p_n)$ *with* $p_i \in [0, W]$*, that is the item of size* i *has profit* p_i*. Find the total profit of the multiset of indices* I *such that* $\sum_{i \in I} i \leq n$ *and the total profit* $\sum_{i \in I} p_i$ *is maximized.*

Note that if we replace multiset by set in the above definition, we obtain the bounded version of the problem, which we denote by Knapsack.

We remark that our perspective on CoinChange and UnboundedKnapsack (as well as UnboundedSubsetSum below) using LWS is slightly different than many classical accounts of Knapsack: We define the problem size as the budget size instead of the number of items, thus our focus is on pseudo-polynomial time algorithms for the typical formulations of these problems.

Note that we state the coin change problem as allowing positive or negative weights, but UnboundedKnapsack only allows for positive profits. Furthermore, the CoinChange problem is a minimization problem, while UnboundedKnapsack is a maximization problem. For CoinChange, the maximization problem is trivially equivalent as we can negate all weights. Furthermore, we can freely translate the range of the weights in the coin change problem by defining $w'_i = i \cdot M + w_i$ for all i and sufficiently large or small M. The most significant difference between CoinChange and UnboundedKnapsack is that for CoinChangethe indices have to sum to exactly n, while for UnboundedKnapsack n is only an upper bound.

We will encounter an important generalization of the two problems above, defined as follows.

Problem 18 (oiCoinChange)**.** *The output-intensive version of* CoinChange *is to determine, given an input to* CoinChange*, the weight of the optimal multiset such that the denominations sum up to j* for all $1 \leq j \leq n$.

It is easy to see that oiCoinChange is equivalent to computing all values $F[j]$ for both CoinChange and UnboundedKnapsack, and is therefore at least as hard as either of them. We discuss the complexity of the Knapsack variants above in Section 5.5.

One Knapsack variant that turns out to be solvable in suquadratic time is the unbounded subset sum problm discussed in Section 5.7, we will revisit this problem together with other near-linear time algorithms.

Problem 19 (UnboundedSubsetSum)**.** *Given a subset $S \subseteq [n]$, determine whether there is a multiset of elements of S that sums up to exactly n.*

We also discuss problems where the goal is to find the longest chain among data items, where the notion of a chain is defined by some binary relation R. We first give the definition of the general problem which is parameterized by R.

Problem 20 (ChainLWS)**.** *Fix a set X of objects and a relation $R \subseteq X \times X$. The Weighted Chain Least-Weight Subsequence Problem for R, denoted* ChainLWS(R)*, is the following problem: Given data items $x_0, \ldots, x_n \in X$, weights $y_1, \ldots, y_{n-1} \in [-W, W]$, find the weight of the increasing sequence $i_0 = 0 < i_1 < i_2 < \ldots < i_k = n$ such that for all j with $1 \leq j \leq k$ the pair $(x_{i_{j-1}}, x_{i_j})$ is in the relation R and the weight $\sum_{j=1}^{k-1} y_{i_j}$ is minimized.*

ChainLWS is a subclass of LWS where we restrict the input to be a sequence of data items, and $w_{i,j} = y_j$ if $(x_i, x_j) \in R$ and $w_{i,j} = \infty$ otherwise.

The following problems are specializations of this problem for different relations.

Problem 21 (NestedBoxes)**.** *Given n boxes in d dimensions, given as non-negative, d-dimensional vectors $(b_1, \ldots, b_n)$, find the longest chain such that each box fits into the next (without rotation). We say box that box a fits into box b if for all dimensions $1 \leq i \leq d$, $a_i \leq b_i$.*

Problem 22 (SubsetChain)**.** *Given n sets from a universe U of size d, given as Boolean, d-dimensional vectors $(b_1, \ldots, b_n)$, find the longest chain such that each set is a subset of the next.*

Note that SubsetChain is a special case of NestedBoxes.

The complexity of ChainLWS problems is discussed in Section 5.6.

A number of ChainLWS instances have previously known near-linear time algorithms are are discussed in Section 5.7.

Problem 23 (LIS)**.** *Given a sequence of n integers $x_1, \ldots, x_n$, compute the length of the longest subsequence that is strictly increasing.*

We also briefly discuss the following LWS instatiation that allows for near-linear time algorithms.

Problem 24 (ConcaveLWS)**.** *Given an LWS instance in which the weights satisfy the quadrangle inequality*

$$w_{i,j} + w_{i',j'} \leq w_{i',j} + w_{i,j'} \qquad \textit{for } i \leq i' \leq j \leq j',$$

solve it. The weights are not explicitly given, but each $w_{i,j}$ can be queried in constant time.

5.2.2 Core Problems and Hypotheses

We characterize the complexity of LWS by showing subquadratic equivalences to *core* problems. These core problems are typically better understood than the LWS instantiations mentioned above, in particular from a conditional lower bound perspective. They also differ from LWS in that the problems are not inherently sequential. Even stronger, the core problems have a very small (subpolynomial) nondeterminstic complexity. Our key results will be subquadratic equivalences between succinct LWS instantiations and corresponing core problems. For an overview of known conditional lower bounds, including the core problems mentioned in this chapter, see Chapter 2.

Recall the definition of OrthogonalVectors.

Problem 25 (OrthogonalVectors)**.** *Given vectors $a_1, \dots, a_n, b_1, \dots, b_n \in \{0,1\}^d$, determine if there is $i, j \in [n]$ satisfying $\langle a_i, b_j \rangle = 0$.*

For OrthogonalVectors (and the related problems below) we assume $d = n^{o(1)}$. Thus the naive algorithm solves OrthogonalVectors in time $\mathrm{O}(n^2 \cdot d) = \mathrm{O}(n^{2+o(1)})$.

We will rely on the SETH-hardness of OrthogonalVectors proved in Section 2.6.

We consider the following generalizations of OrthogonalVectors and also consider them core problems. Some of these problems have already been discussed in Section 2.7.

Problem 26 (IntegerMinInnProd)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in [-W, , W]^d$ and a natural number $r \in \mathbb{N}$, determine if there are i, j satisfying $\langle a_i, b_j \rangle \leq r$.*

In Section 5.4 we show subquadratic equivalence between IntegerMinInnProd and LowRankLWS.

Problem 27 (SetContainment)**.** *Given sets $a_1, \dots, a_n, b_1, \dots, b_n \subseteq [d]$ given as vectors in $\{0,1\}^d$ determine if there are i, j such that $a_i \subseteq b_j$.*

We also recall the VectorDomination problem discussed in Chapter 3.

Problem 28 (VectorDomination)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ determine if there is i, j such that $a_i \leq b_j$ component-wise.*

Note that SetContainment is a special case of VectorDomination and computationally equivalent to OrthogonalVectors, as $\langle a, b \rangle = 0$ if and only if $a \subseteq \bar{b}$ (in this slight misuse of notation we think of the Boolean vectors a, b as sets and let $\bar{b}$ denote the complement of b).

In Section 5.6 we show subquadratic equivalences between VectorDomination and NestedBoxes, as well as between SetContainment and SubsetChain.

Since subquadratic solutions to any of these problems trivially give a subquadratic solution to OrthogonalVectors, these problems are also quadratic-time SETH-hard. However, the converse does not necessarily hold. In particular, the strongest currently known upper bounds differ: while for OrthogonalVectors and SetContainment for small dimension $d = c \cdot \log(n)$, an $n^{2-1/\mathrm{O}(\log c)}$-time algorithm is known [7], for VectorDomination the best known algorithm runs only in time $n^{2-1/\mathrm{O}(c \log^2 c)}$ [80, 38].

Another fundamental quadratic-time problem is $(\min,+)$-Convolution, discussed in Section 2.8.3.

Problem 29 ($(\min,+)$-Convolution)**.** *Given n-dimensional vectors* $a = (a_0,\dots,a_{n-1})$, $b = (b_0,\dots,b_{n-1}) \in [-W,W]^n$ *for some* $W = \mathsf{poly}(n)$*, the* $(\min,+)$-Convolution $a * b$ *is defined by*

$$(a*b)_k = \min_{0\le i,j<n: i+j=k} a_i + b_j \qquad \textit{for all } 0 \le k \le 2n-2.$$

As opposed to the classical convolution, which we denote as $a \circledast b$, solvable in time $\mathrm{O}(n\log n)$ using FFT, no strongly subquadratic algorithm for $(\min,+)$-Convolution is known. Compared to OrthogonalVectors, we have less support for believing that no $\mathrm{O}(n^{2-\varepsilon})$-time algorithm for $(\min,+)$-Convolutionexists. In particular, interesting special cases can be solved in subquadratic-time [39] and there are subquadratic-time co-nondeterministic and nondeterministic algorithms [30, 36]. At the same time, breaking this long-standing quadratic-time barrier is a prerequisite for progress on refuting the 3-sum and APSP conjectures. $(\min,+)$-Convolution is therefore an interesting target particularly for proving subquadratic *equivalences*, since both positive and negative resolutions of this open question appear to be reasonable possibilities.

For LWS instantiatiations where near-linear time algorithms are already known we also identify corresponding core problems in Section 5.7. None of the algorithmic results are new, but the core problems do give a new perspective on these problems.

5.2.3 Intermediate Problems

Our reductions from LWS instantiations to core problems go through intermediate problems that share some of the characteristics of core problems, as well as some of the characteristics of LWS. In particular, these problems are naturally parallelizable and their brute-force algorithm is already quadratic time, similar to core problems. On the other hand their definitions are closely related to the definition of LWS. Other than core problems, intermediate problems are not decision problems but ask to compute some linear sized output. In many instance, the intermediate problem is not a natural problem. Note that $(\min,+)$-Convolution shares many of the same properties as intermediate problems, with the exception that in our result it does not actually constitute an intermediate step but acts as the well-studied problem we relate the computation complexity of LWS to.

We define a generic intermediate problem called Static-LWS.

Problem 30 (Static-LWS($\mathbf{W}$))**.** *Fix an instance of* LWS($\mathbf{W}$). *Given intervals of indices* $I := \{a+1,\dots,a+N\}$ *and* $J := \{a+N+1,\dots,a+2N\}$ *with* a,N *such that* $I,J \subseteq [n]$, *together with the values* $F[a+1],\dots,F[a+N]$, *the Static Least-Weight Subsequence Problem (*Static-LWS*) asks to determine*

$$F'[j] := \min_{i\in I} F[i] + w_{i,j} \qquad\qquad \text{for all } j \in J.$$

Static-LWS has appeared implicitly in previous work. For example, the SMAWK problem [12] of finding the column minima in a totally monotone matrix is the instantiation of Static-LWS obtained from ConcaveLWS, and used in that capacity in [65].

Some of the instantiations of Static-LWS are natural problems. For example, the Static-LWS instantiation that corresponds to LowRankLWS can easily be seen to be equivalent to AllInnProd.

Problem 31 (AllInnProd)**.** *Given* $a_1,\dots,a_n \in [-W,W]^d$ *and* $b_1,\dots,b_n \in [-W,W]^d$, *determine* for all $j \in [n]$, *the value* $\min_{i\in[n]}\langle a_i,b_j\rangle$.

5.3 Static LWS

In this section we give a reduction from $\mathsf{LWS}(\mathbf{W})$ to $\mathsf{Static\text{-}LWS}(\mathbf{W})$ that is independent of the weight matrix $\mathbf{W}$ and therefore independent of the succinct LWS instantiations we consider throughout this paper. This reduction is a key step in our reductions from LWS to their corresponding core problems.

The reduction is a divide-and-conquer scheme that divides the LWS problem into two subproblems of half the size each and Static-LWS to combine the two. Crucially, the two subproblems have to be solved sequentially. The reduction therefore captures the sequential nature of the LWS problem, while Static-LWS captures a parallelizable part of the problem.

This reduction has appeared implicitly in previous work on LWS [77]. In particular, the reduction of ConcaveLWS to the SMAWK problem by Galil and Park [65] can be thought of as a variant of this reduction specialized to the concave case to avoid log-factors.

Lemma 5.3.1 ($\mathsf{LWS}(\mathbf{W}) \leq_2 \mathsf{Static\text{-}LWS}(\mathbf{W})$)**.** *For any choice of* $\mathbf{W}$*, if* $\mathsf{Static\text{-}LWS}(\mathbf{W})$ *can be solved in time* $\mathrm{O}(N^{2-\varepsilon})$ *for some* $\varepsilon > 0$*, then* $\mathsf{LWS}(\mathbf{W})$ *can be solved in time* $\tilde{\mathrm{O}}(n^{2-\varepsilon})$.

Proof. In what follows, we fix LWS as $\mathsf{LWS}(\mathbf{W})$ and Static-LWS as $\mathsf{Static\text{-}LWS}(\mathbf{W})$.

We define the subproblem $S(\{i,\dots,j\},(t_i,\dots,t_j))$ that given an interval spanned by $1 \leq i \leq j \leq n$ and values $t_k = \min_{0\leq k'<i} F[k'] + w_{k',k}$ for each point $k \in \{i,\dots,j\}$, computes all values $F[k]$ for $k \in \{i,\dots,j\}$. Note that a call to $S([n],(w_{0,1},\dots,w_{0,n}))$ solves the LWS problem, since $F[0] = 0$ and thus the values of $t_k, k \in [n]$ are correctly initialized.

We solve S using Algorithm 6.

Algorithm 6: Reducing LWS to Static-LWS, S

```
input: $\{i,\dots,j\},(t_i,\dots,t_j)$
if $i = j$ then
    return $F[i] \leftarrow t_i$
$m \leftarrow \lceil \frac{j-i}{2} \rceil$
$(F[i],\dots,F[i+m-1]) \leftarrow S(\{i,\dots,i+m-1\},(t_i,\dots,t_{i+m-1}))$
solve Static-LWS on the subinstance given by $I := \{i,\dots,i+m-1\}$ and
 $J := \{i+m,\dots,i+2m-1\}$
// obtains values $F'[k] = \min_{i \le k' < i+m} F[k'] + w_{k',k}$ for
    $k = i+m,\dots,i+2m-1$
$t'_k \leftarrow \min\{t_k, F'[k]\}$ for all $k = i+m,\dots,i+2m-1$
$(F[i+m],\dots,F[i+2m-1]) \leftarrow S(\{i+m,\dots,i+2m-1\},(t'_{i+m},\dots,t'_{i+2m-1}))$

if $j = i+2m$ then
    $F[j] := \min\{t_j, \min_{i \le k < j} F[k] + w_{k,j}\}$
return $(F[i],\dots,F[j])$
```

We briefly argue correctness, using the invariant that $t_k = \min_{0 \le k' < i} F[k'] + w_{k',k}$ in every call to S. If S is called with $i = j$, then the invariant yields $t_i = \min_{0 \le k' < i} F[k'] + w_{k',i} = F[i]$, thus $F[i]$ is computed correctly. For the first recursive call, the invariant is fulfilled by assumption, hence the values $(F[i],\dots,F[i+m-1])$ are correctly computed. For the second recursive call, we note that for $k = i+m,\dots,i+2m-1$, we have

$$t'_k = \min\{t_k, T'[k]\} = \min\{\min_{0 \le k' < i} F[k'] + w_{k',k}, \min_{i \le k' < i+m} F[k'] + w_{k',k}\} \tag{5.2}$$

$$= \min_{0 \le k' < i+m} F[k'] + w_{k',k} \tag{5.3}$$

Hence the invariant remains satisfied. Thus, the values $(F[i+m],\dots,F[i+2m-1])$ are correctly computed. Finally, if $j = i+2m$, we compute the remaining value $F[j]$ correctly, since $t_j = \min_{0 \le k < i} F[k] + w_{k,j}$ by assumption.

To analyze the running time $T^S(n)$ of S on an interval of length $n := j-i+1$, note that each call results in two recursive calls of interval lengths at most $n/2$. In each call, we need an additional overhead that is linear in n and $T^{\mathsf{Static\text{-}LWS}}(n/2)$. Solving the corresponding recursion $T^S(n) \leq 2T^S(n/2) + T^{\mathsf{Static\text{-}LWS}}(n/2) + O(n)$, we obtain that an $O(N^{2-\varepsilon})$-time algorithm Static-LWS, with $0 < \varepsilon < 1$ yields $T^{\mathsf{LWS}}(n) \leq T^S(n) = O(n^{2-\varepsilon})$. Similarly, an $O(N \log^c N)$-time algorithm for Static-LWS would result in an $O(n \log^{c+1} n)$-time algorithm for LWS. □

This reduction has appeared implicitly in previous work on LWS [77]. In particular, the reduction of ConcaveLWS to the SMAWK problem by Galil and Park [65] can be thought of as a variant of this reduction specialized to the concave case to avoid log-factors.

5.4 Low Rank LWS

In this section we prove the first equivalence between an instantiation of LWS and a core problem.

Let us first analyze the following canonical succinct representation of a low-rank weight matrix $\mathbf{W} = (w_{i,j})_{i,j}$: If $\mathbf{W}$ is of rank $d \ll n$, we can write it more succinctly as $\mathbf{W} = A \cdot B$, where A and B are $(n \times d)$- and $(d \times n)$ matrices, respectively. We can express the resulting natural LWS problem equivalently as follows.

Problem 32 (LowRankLWS)**.** *We define the following* LWS *instantiation* LowRankLWS $=$ LWS$(\mathbf{W}_{\text{LowRank}})$.

Data items: *out-vectors* $\mu_0, \ldots, \mu_{n-1} \in [-W, W]^d$*, in-vectors* $\sigma_1, \ldots, \sigma_n \in [-W, W]^d$

Weights: $w(i, j) = \langle \mu_i, \sigma_j \rangle$ *for* $0 \leq i < j \leq n$

In this section, we show that this problem is equivalent, under subquadratic reductions, to the following *non-sequential* problem.

Problem 33 (IntegerMinInnProd)**.** *Given* $a_1, \ldots, a_n, b_1, \ldots, b_n \in [-W, W]^d$ *and a natural number* $r \in \mathbb{N}$*, determine if there is a pair* i, j *satisfying* $\langle a_i, b_j \rangle \leq r$.

This is interesting for a number of reasons. For one, IntegerMinInnProd is a fairly natural problem and, as opposed to LowRankLWS it is not inherently sequential in its definition. We understand IntegerMinInnProd comparably well both from an upper and from a lower bound perspective. Using ray shooting data structures [106] we can solve IntegerMinInnProd in strongly subquadratic time if d in constant. At the same time, if $d = \omega(\log n)$, the problem is SETH-hard at time n^2. By showing subquadratic equivalence between IntegerMinInnProd and LowRankLWS, we can conclude both these results, as well as any future improvements, for LowRankLWS.

We first give a simple reduction from IntegerMinInnProd that along the way proves quadratic-time SETH-hardness of LowRankLWS.

Lemma 5.4.1. *It holds that*

$$T^{\mathsf{IntegerMinInnProd}}(n,d,W) \leq T^{\mathsf{LowRankLWS}}(2n+1,d+2,dW) + \mathrm{O}(nd)$$

Proof. Given $a_1,\dots,a_n,b_1,\dots,b_n \in [-W,W]^d$, let $\mathbf{O} = (0,\dots,0) \in \mathbb{N}^d$ be the all-zeroes vector and define the following in- and out-vectors

$$\begin{aligned} \mu_0 &= (dW,0,\mathbf{O}), & \sigma_{2n+1} &= (dW,dW,\mathbf{O}), & & \\ \mu_i &= (0,dW,a_i), & \sigma_i &= (0,0,\mathbf{O}), & & \text{for } i=1,\dots,n, \\ \mu_{n+j} &= (0,0,\mathbf{O}), & \sigma_{n+j} &= (dW,0,b_j), & & \text{for } j=1,\dots,n. \end{aligned}$$

To prove correctness, we show that in the constructed LowRankLWS instance, we have $F[2n+1] = \min_{i,j}\langle a_i,b_j\rangle$, from which the results follows immediately.

We use the following observations:

- $\langle \mu_0, \sigma_{n+j}\rangle = (dW)^2 \geq \max_{i,j}\langle a_i,b_j\rangle$
- $\langle \mu_i, \sigma_{n+j}\rangle = \langle a_i,b_j\rangle$

- $\langle \mu_{n+j'}, \sigma_{n+j} \rangle = 0$

for all $1 \le i, j \le n$ and $j' \le j$.

Inductively, we have $F[i] = 0$ for $i = 1, \dots, n$, since $\langle \mu_{i'}, \sigma_i \rangle = 0$ for all $0 \le i' < i \le n$. Similarly, for $j = 1, \dots, n$ one can inductively show that

$$F[n+j] = \min_{1 \le i \le n, j' \le j} \langle a_i, b_{j'} \rangle$$

Finally, using

- $\langle \mu_0, \sigma_{2n+1} \rangle = (dW)^2 \ge \max_{i,j} \langle a_i, b_j \rangle$ and $F[0] = 0$
- $\langle \mu_i, \sigma_{2n+1} \rangle = (dW)^2 \ge \max_{i,j} \langle a_i, b_j \rangle$ and $F[i] = 0$ for $i = 1, \dots, n$
- $\langle \mu_{n+j}, \sigma_{2n+1} \rangle = 0$ and $F[n+j] = \min_{1 \le i \le n, 1 \le j' \le j} \langle a_i, b_{j'} \rangle$

for all $j = 1, \dots, n$, we can finally determine $F[2n+1] = \min_{i,j} \langle a_i, b_j \rangle$. □

To prove the other direction, we will use the quite general approach to compute the sequential LWS problem by reducing to Static-LWS (Lemma 5.3.1)

For the special case of LowRankLWS, it is straightforward to see that the static version boils down to the following natural reformulation.

Problem 34 (AllInnProd)**.** *Given* $a_1, \dots, a_n \in [-W, W]^d$ *and* $b_1, \dots, b_n \in [-W, W]^d$*, determine* for all $j \in [n]$*, the value* $\min_{i \in [n]} \langle a_i, b_j \rangle$*. (Again, we typically assume that* $d = n^{o(1)}$ *and* $W = 2^{n^{o(1)}}$*.)*

Lemma 5.4.2 (Static-LWS($\mathbf{W}_{\text{LowRank}}$) $\le_2$ AllInnProd)**.** *We have*

$$T^{\text{Static-LWS}(\mathbf{W}_{\text{LowRank}})}(n, d, W) \le T^{\text{AllInnProd}}(n, d+1, nW) + O(nd).$$

Proof. Consider Static-LWS($\mathbf{W}_{\text{LowRank}}$). Let $I = \{a+1, \ldots, a+N\}$, $J = \{a+N+1, \ldots, a+2N\}$ and values $F[a+1], \ldots, T[a+N]$ be given. To determine

$$F'[j] = \min_{i \in I} F[i] + w_{i,j}$$

for all $j \in J$, it is sufficient to solve the AllInnProd problem where the input vectors are given by $a_{a+1}, \ldots, a_{a+N} \in [nW, nW]^{d+1}$ and $b_{a+N+1}, \ldots, b_{a+2N} \in [nW, nW]^{d+1}$ defined by

$$a_i := (\mu_i, F[i]) \qquad b_j = (\sigma_j, 1), \qquad \text{for all } i \in I, j \in J,$$

since then

$$\langle a_i, b_j \rangle = F[i] + \langle \mu_i, \sigma_j \rangle = F[i] + w_{i,j}$$

The claim immediately follows (note that $|F[i]| \leq nW$). □

Finally, inspired by an elegant trick of [150], we reduce the AllInnProd problem to the IntegerMinInnProd problem.

Lemma 5.4.3 (AllInnProd $\leq_2$ IntegerMinInnProd)**.** *We have*

$$T^{\mathsf{AllInnProd}}(n, d, W) \leq \mathrm{O}(n \cdot T^{\mathsf{IntegerMinInnProd}}(\sqrt{n}, d+3, ndW^2) \cdot \log^2 nW).$$

Proof. We first observe that we can tune IntegerMinInnProd to also return a witness (i, j) with $\langle a_i, b_j \rangle \leq r$, if it exists. To do so, we replace each a_i by the $(d+2)$-dimensional vector $a'_i = (a_i \cdot n, (i-1)n, -1)$ and similarly, each b_j by the $(d+2)$-dimensional vector $b'_j = (b_j \cdot n, -1, j-1)$. Clearly, we have $\langle a'_i, b'_j \rangle = \langle a_i, b_j \rangle n^2 - (i-1)n - (j-1)$. Thus $\langle a'_i, b'_j \rangle \leq rn^2$ if and only if $\langle a_i, b_j \rangle \leq r$ since $i, j \in [n]$. Using a binary search over r, we can find $\min_{i,j} \langle a'_i, b'_j \rangle$, from whose precise value we can determine also a witness, if it exists. Thus the running time $\mathrm{wit}(n,d,W)$ for finding such a witness is bounded by $\mathrm{O}(\log nW)\cdot T^{\mathsf{IntegerMinInnProd}}(n,d+2,nW)$.

To solve AllInnProd, i.e., to compute $p_j := \min_{i\in[n]}\langle a_i,b_j\rangle$ for all $j\in[n]$, we employ a parallel binary search. Consider in particular the following problem $\mathscr{P}$: Given arbitrary $r_1,\dots,r_n$, determine for all $j\in[n]$ whether there exists $i\in[n]$ such that $\langle a_i,b_j\rangle \le r_j$. We will show below that this problem can be solved in time $\mathrm{O}(n\cdot \mathrm{wit}(\sqrt{n},d+1,dW^2))$. The claim then follows, since starting from feasible intervals $\mathscr{R}_1=\dots=\mathscr{R}_n=[-dW^2,dW^2]$ satisfying $p_j\in\mathscr{R}_j$, we can halve the sizes of each interval simultaneously by a single call to $\mathscr{P}$. Thus, after $\mathrm{O}(\log(dW))$ calls, the true values p_j can be determined, resulting in the time guarantee $T^{\mathsf{AllInnProd}}(n,d,w)=\mathrm{O}(n\cdot \mathrm{wit}(\sqrt{n},d+1,dW^2)\cdot\log(dW))=\mathrm{O}(n\cdot T^{\mathsf{IntegerMinInnProd}}(\sqrt{n},d+3,ndW^2)\log^2(nW))$, as desired.

We complete the proof of the claim by showing how to solve $\mathscr{P}$. Without loss of generality, we can assume that $r_j\le dW^2$ for every j, since no larger inner product may exist. We group the vectors $a_1,\dots,a_n$ in $g:=\lceil\sqrt{n}\rceil$ groups $A_1,\dots,A_g$ of size at most $\sqrt{n}$ each, and do the same for the vectors $b_1,\dots,b_n$ to obtain $B_1,\dots,B_g$. Now, we iterate over all pairs of groups A_k,B_ℓ, $k,\ell\in[g]$: For each such choice of pairs, we do the following process. For each vector $a_i\in A_k$, we define the $(d+1)$-dimensional vector $\tilde{a}_i:=(a_i,-1)$ and for every vector $b_j\in B_\ell$, we define $\tilde{b}_j:=(b_j,r_j)$. In the obtained instance $\{\tilde{a}_i\}_{a\in A_k},\{\tilde{b}_j\}_{b\in B_\ell}$, we try to find some i,j such that $\langle\tilde{a}_i,\tilde{b}_j\rangle\le 0$, which is equivalent to $\langle a_i,b_j\rangle\le r_j$. If we succeed in finding such a witness, we delete b_j and $\tilde{b}_j$ (but remember its witness) and repeat finding witnesses (an deleting the witnessed b_j) until we cannot find any. The process then ends and we turn to the next pair of groups.

It is easy to see that for all $j\in[n]$, we have $\langle a_i,b_j\rangle\le r_j$ for some $i\in[n]$ if and only if the above process finds a witness for b_j at some point. To argue about the running time, we charge the running time of every call to witness finding to either (1) the pair A_k,B_ℓ, if the call is the first call in the process for A_k, B_ℓ, or (2) to b_j, if the call resulted from finding a witness for b_j in the previous call. Note that every pair A_k, B_ℓ is charged by exactly one call and every b_j is charged by at most one call (since in after a witness for b_j is found, we delete b_j and no further witness for b_j can be found). Thus in total, we obtain a running time of at most $(g^2+n)\cdot \mathrm{wit}(\sqrt{n}, d+1, dW^2) + \mathrm{O}(n) = \mathrm{O}(n \cdot \mathrm{wit}(\sqrt{n}, d+1, dW^2))$. □

Theorem 5.4.1. *We have* LowRankLWS $\equiv_2$ IntegerMinInnProd.

Proof. In Lemmas 5.4.1, 5.3.1, 5.4.2, and 5.4.3, we have proven

$$\begin{aligned}\mathsf{IntegerMinInnProd} &\leq_2 \mathsf{LowRankLWS} = \mathsf{LWS}(\mathbf{W}_{\mathrm{LowRank}}) \\ &\leq_2 \mathsf{Static\text{-}LWS}(\mathbf{W}_{\mathrm{LowRank}}) \leq_2 \mathsf{AllInnProd} \leq_2 \mathsf{IntegerMinInnProd},\end{aligned}$$

proving the claim. □

5.5 Coin Change and Knapsack Problems

In this section, we focus on the following problem related to Knapsack: Assume we are given coins of denominations $d_1, \dots, d_m$ with corresponding weights $w_1, \dots, w_m$ and a target value n, determine a way to represent n using these coins (where each coin can be used arbitrarily often) minimizing the total sum of weights of the coins used. Since without loss of generality $d_i \leq n$ for all i, we can assume that $m \leq n$ and think of n as our problem size. In particular, we describe the input by weights $w_1, \dots, w_n$ where w_i denotes the weight of the coin of denomination i (if no coin with denomination i exists, we set $w_i = \infty$). It is straightforward to see that this problem is an LWS instance LWS$(\mathbf{W}_{\mathsf{CoinChange}})$, where the weight matrix $\mathbf{W}_{\mathsf{CoinChange}}$ is a Toeplitz matrix.

Problem 35 (CoinChange)**.** *We define the following* LWS *instantiation* CoinChange $=$ LWS$(\mathbf{W}_{\mathsf{CoinChange}})$.

Data items: *weight sequence* $w = (w_1, \dots, w_n)$ *with* $w_i \in [-W, W] \cup \{\infty\}$

Weights: $w_{i,j} = w_{j-i}$ *for* $0 \leq i < j \leq n$

Translated into a Knapsack-type formulation (i.e., denominations are weights, weights are profits, and the objective becomes to maximize the profit), the problem differs from UnboundedKnapsack only in that it searches for the most profitable multiset of items of weight *exactly n*, instead of *at most n*.

Problem 36 (UnboundedKnapsack)**.** *We are given a sequence of profits* $p = (p_1, \dots, p_n)$ *with* $p_i \in [0, W]$*, that is the item of size i has profit* p_i*. Find the total profit of the multiset of indices I such that* $\sum_{i \in I} i \leq n$ *and the total profit* $\sum_{i \in I} p_i$ *is maximized.*

The purpose of this section is to show that both the CoinChange problem and the UnboundedKnapsack problem are subquadratically equivalent to $(\min, +)$-Convolution. Along the way, we also prove quadratic-time $(\min, +)$-Convolution-hardness of the Knapsack problem. Recall the definition of $(\min, +)$-Convolution.

Problem 37 ($(\min, +)$-Convolution)**.** *Given n-dimensional vectors* $a = (a_0, \dots, a_{n-1})$*,* $b = (b_0, \dots, b_{n-1}) \in [-W, W]^n$ *for some* $W = \mathsf{poly}(n)$*, the* $(\min, +)$-Convolution $a * b$ *is defined by*

$$(a * b)_k = \min_{0 \leq i,j < n : i+j=k} a_i + b_j \qquad \textit{for all } 0 \leq k \leq 2n-2.$$

As opposed to the classical Convolution, which we denote as $a \circledast b$, solvable in time $O(n \log n)$ using FFT, no strongly subquadratic algorithm for $(\min, +)$-Convolution is known. Compared to the popular orthogonal vectors problem, we have less support for believing that no $O(n^{2-\varepsilon})$-time algorithm for $(\min, +)$-Convolution exists. In particular, interesting special cases can be solved in subquadratic time [39] and there are subquadratic-time co-nondeterministic and nondeterministic algorithms [30, 36]. At the same time, breaking this long-standing quadratic-time barrier is a prerequisite for progress on refuting the 3-sum and APSP conjectures. This makes it an interesting target particularly for proving subquadratic *equivalences*, since both positive and negative resolutions of this open question appear to be reasonable possibilities.

To obtain our result, we address two issues: (1) We show an equivalence between the problem of determining only the value $F[n]$, i.e., the best way to give change only for the target value n, and to determine *all values* $F[1],\dots,F[n]$, which we call the *output-intensive version*. (2) We show that the output-intensive version is subquadratic equivalent to $(\min,+)$-Convolution.

Problem 38 (oiCoinChange)**.** *The output-intensive version of* CoinChange *is to determine, given an input to* CoinChange*, all values* $F[1],\dots,F[n]$.

We first consider the second issue and provide a $(\min,+)$-Convolution-based lower bound for oiCoinChange.

Lemma 5.5.1 ($(\min,+)$-Convolution $\leq_2$ oiCoinChange)**.** *We have*

$$T^{(\min,+)\text{-Convolution}}(n,W) \leq T^{\text{oiCoinChange}}(6n,4(2W+1)) + \mathrm{O}(n)$$

Proof. We first do a translation of the input. Note that for any scalars α,β, we have $(a+\alpha)*(b+\beta) = (a*b)+\alpha+\beta$. Let $M := 2W+1$. Without loss of generality, we may assume that

$$\begin{aligned} 2M \leq a_i &\leq 3M && \text{for all } i = 0,\dots,n-1,\\ 0 \leq b_j &\leq M && \text{for all } j = 0,\dots,n-1. \end{aligned}$$

We now define a CoinChange instance with a problem size $n' = 6n$ and $W' = 4M$ by defining

$$w = (4M)^n \circ (a_{n-1}, \dots, a_0) \circ (4M)^n \circ (b_{n-1}, \dots, b_0) \circ (4M)^{2n}.$$

We now claim that $F[4n+i] = (a*b)_{2n-i}$ for $i = 1, \dots, 2n$, which immediately yields the lemma. To do so, we will prove the following sequence of identities.

$$F[i] = 4M \qquad \text{for } i \in [n], \tag{5.4}$$

$$F[n+i] = a_{n-i} \qquad \text{for } i \in [n], \tag{5.5}$$

$$F[2n+i] = 4M \qquad \text{for } i \in [n], \tag{5.6}$$

$$F[3n+i] = b_{n-i} \qquad \text{for } i \in [n], \tag{5.7}$$

$$F[4n+i] = (a*b)_{2n-i} \qquad \text{for } i \in [2n], \tag{5.8}$$

In the last line, we define, for our convenience, $(a*b)_{2n-1} = 4M$ (note that before, we defined only the entries $(a*b)_k$ with $k \leq 2n-2$).

For later convenience, observe that $0 \leq w_i \leq 4M$ for all $i \in [n']$. It is easy to see that this implies $0 \leq F[i] \leq 4M$ for $i \in [n']$.

The identities in (5.4) are obvious.

To prove the identities in (5.5) inductively over i, recall that

$$F[n+i] = \min_{j=1,\dots,n+i} \{F[n+i-j] + w_j\}$$

We observe that $F[n+i-j] + w_j < 4M$ can only occur if $j \geq n+1$ (since otherwise $w_j = 4M$), which implies $n+i-j \leq n$ and $F[n+i-j] = 4M$ except for the case $j = n+i$. In this case, we have $F[n+i-j] + w_j = F[0] + w_{n+i} = a_{n-i} \leq 4M$.

To prove the identities in (5.6), observe that for $1 \leq j \leq 3n$, we have $w_j \geq 2M$ by assumption $\min_i a_i \geq 2M$. Similarly, we have already argued that $F[i'] \geq 2M$ for

$1 \leq i' \leq 2n$. Thus, we can inductively show that

$$F[2n+i] = \min\{F[0] + w_{2n+i}, \min_{j=1,\dots,2n+i-1} F[2n+i-j] + w_j\} = 4M$$

using $w_{2n+i} = 4M$ and that every sum in the inner minimum expression is at least $4M$.

To prove the identities in (5.7), note that for $F[3n+i-j] + w_j < 4M$ to hold, we must have either $n+1 \leq j \leq 2n$ or $3n+1 \leq j \leq 3n+i$, since otherwise $w_j = 4M$. We observe that for $n+1 \leq j \leq 2n$, we have $w_j \geq \min_i a_i \geq 2M$ and $F[3n+i-j] \geq \min_i a_i \geq 2M$. Thus, we may assume that $3n+1 \leq j \leq 3n+i$. Note that in this case, we have $F[3n+i-j] = 4M$ except for the case $j = 3n+i$, where we have $F[3n+i-j] + w_j = F[0] + w_{3n+i} = b_{n-i} < 4M$.

Finally, for the identities in (5.8), we might have $F[4n+i] + w_j < 4M$ only if $n+1 \leq j \leq 2n$ or $3n+1 \leq j \leq 4n$. First consider the case that $i = 1$. We have

$$F[4n+1] = \min\{w_{4n+1}, \min_{n+1 \leq j \leq 2n} \underbrace{F[4n+1-j]}_{=4M} + w_j, \min_{3n+1 \leq j \leq 4n} \underbrace{F[4n+1-j]}_{=4M} + w_j\} \tag{5.9}$$

$$= 4M \tag{5.10}$$

Inductively over $1 < i \leq 2n$, we will prove $F[4n+i] = (a * b)_{2n-i}$. By definition,

$$\begin{aligned} F[4n+i] &= \min\{w_{4n+i}, \min_{n+1 \leq j \leq 2n} F[4n+i-j] + w_j, \min_{3n+1 \leq j \leq 4n} F[4n+i-j] + w_j\} \\ &= \min\{w_{4n+i}, \min_{1 \leq j' \leq n} F[3n+i-j'] + a_{n-j'}, \min_{1 \leq j' \leq n} F[n+i-j'] + b_{n-j'}\} \end{aligned} \tag{5.11}$$

Note that

$$\min_{1\le j'\le n} \underbrace{F[n+i-j']}_{=4M \text{ for } j'\ge i \text{ or } j'<i-n} + b_{n-j'} = \min_{\max\{1,i-n\}\le j'\le \min\{i-1,n\}} a_{n-(i-j')} + b_{n-j'} \tag{5.12}$$

$$= (a*b)_{2n-i} \tag{5.13}$$

where the last equation follows from noting that the choice of j' lets $n-j'$ and $n-(i-j')$ range over all admissible pairs of values in $[0,n-1]$ summing up to $2n-i$. Similarly, we inductively prove that

$$\min_{1\le j'\le n} F[3n+i-j'] + a_{n-j'} = \min_{\max\{1,i-n\}\le j'\le \min\{i-1,n\}} a_{n-(i-j')} + b_{n-j'} = (a*b)_{2n-i}$$

since $a_{n-j'} \ge 2M$ and $F[3n+i-j'] \ge 2M$ whenever $j' \ge i$ or $j' < i-n$ (where the last regime uses $F[4n+i'] = (a*b)_{2n-i'} \ge 2M$ inductively for $i' < i$). Finally, since $(a*b)_{2n-i} \le (\max_i a_i) + (\max_j b_j) \le 4M$, we can simplify (5.11) to $F[4n+i] = (a*b)_{2n-i}$. $\square$

Using the notion of Static-LWS, the other direction is straight-forward.

Lemma 5.5.2. *We have*

$$\mathsf{oiCoinChange} \le_2 \mathsf{Static\text{-}LWS}(\mathbf{W}_{\mathsf{CoinChange}}) \le_2 (\min,+)\text{-}\mathsf{Convolution}$$

Proof. In Lemma 5.3.1, we have in fact reduced the output-intensive version of $\mathsf{LWS}(\mathbf{W})$ to our static problem $\mathsf{Static\text{-}LWS}(\mathbf{W})$, thus specialized to the coin change problem, we only need to show that $\mathsf{Static\text{-}LWS}(\mathbf{W}_{\mathsf{CoinChange}})$ subquadratically reduces to the $(\min,+)$-Convolution problem. Consider an input instance to Static-LWS given by $I = \{a+1,\dots,a+N\}$, $J = \{a+N+1,\dots,a+2N\}$ and values $F[i], i \in I$. Defining

$M := 2W + 1$ and the vectors

$$u := (nM, F[a+1], \ldots, F[a+N], \overbrace{nM, \ldots, nM}^{N \text{ times}}),$$
$$v := (nM, w_1, \ldots, w_{2N}),$$

we have $(u * v)_{N+k} = \min_{i=1,\ldots,N} F[a+i] + w_{N+k-i} = F'[a+N+k]$ for all $k = 1, \ldots, N$, thus computing the $(\min,+)$-Convolution of two $(2n+1)$-dimensional vectors solves the Static-LWS($\mathbf{W}_{\mathsf{CoinChange}}$) problem, yielding the claim. □

The last two lemmas resolve issue (2). We proceed to issue (1) and show that the output-intensive version is subquadratically equivalent to both CoinChange and UnboundedKnapsack that only ask to determine a single output number. We introduce the following notation for our convenience: Recall that weight w_i denotes the weight of a coin of denomination i. For a multiset $S \subseteq [n]$, we let $d(S) := \sum_{i \in S} i$ denote its *total denomination*, i.e., sum of the denomination of the coins in S (where multiples uses of the same coin is allowed, since S is a multiset). We let $w(S) := \sum_{i \in S} w_i$ denote the weight of the multiset. Analogously, when considering a Knapsack instance, $p(S) = \sum_i p_i$ denotes the total profit of the item (multi)set S.

It is trivial to see that UnboundedKnapsack $\leq_2$ oiCoinChange. Furthermore, we can give the following simple reduction from CoinChange to UnboundedKnapsack.

Observation 5.5.1 (CoinChange $\leq_2$ UnboundedKnapsack $\leq_2$ oiCoinChange)**.** *We have*

$$T^{\mathsf{CoinChange}}(n, W) \leq T^{\mathsf{UnboundedKnapsack}}(n, nW) + \mathrm{O}(n)$$

and

$$T^{\mathsf{UnboundedKnapsack}}(n, W) \leq T^{\mathsf{oiCoinChange}}(n, W) + \mathrm{O}(n)$$

Proof. Given a CoinChange instance, for every weight $w_i < \infty$, we create an item of size i and profit $p_i := i \cdot M - w_i$ in our resulting UnboundedKnapsack instance for a sufficiently large constant $M \geq nW$. This way, all profits are positive and every multiset S whose sizes sum up to B has a profit of $p(S) = B \cdot M - w(S)$. Since $M \geq nW \geq \max_{S,d(S) \leq n} |w(S)|$, this ensures that the maximum-profit multiset of total size/denomination at most n has a total size/denomination of exactly n. Thus, the optimal multiset S^* has profit $p(s^*) = n \cdot M - \min_{S:d(S)=n} w(S) = n \cdot M - F[n]$, from which we can derive $F[n]$, as desired.

Given an UnboundedKnapsack instance, we define for every item of size i and profit p_i the corresponding weight $w_i = -p_i$ in a corresponding CoinChange instance. It remains to compute all $F[1], \dots, F[n]$ in this instance and determining their minimum, concluding the reduction. □

The remaining part is similar in spirit to Lemma 5.4.3: Somewhat surprisingly, the same general approach works despite the much more sequential nature of the Knapsack/CoinChange problem – this sequentiality can be taken care of by a more careful treatment of appropriate subproblems that involves solving them in a particular order and feeding them with information gained during the process.

Lemma 5.5.3 (oiCoinChange $\leq_2$ CoinChange)**.** *We have that*

$$T^{\mathsf{oiCoinChange}}(n,W) \leq \mathrm{O}(\log(nW) \cdot n \cdot T^{\mathsf{CoinChange}}(24\sqrt{n}, 3n^2W))$$

Proof. Let $\mathscr{I}$ be an oiCoinChange instance. To define our subproblems, we set $N := \lceil \sqrt{n} \rceil$ and define N ranges $\mathbf{W}_1 := [1,N], \dots, \mathbf{W}_N := [(N-1)N+1, N^2]$. To determine all $F[i] = \min_{S:d(S)=i} w(S)$, we will compute $F[i]$ for all $i \in \mathbf{W}_j$ successively over all $j = 1, \dots, N$. The case of $j = 1$ and $j = 2$ can be computed by the naive algorithm in time $\mathrm{O}(N^2) = \mathrm{O}(n)$. Consider now any fixed $j \geq 3$ and assume that all values $F[i]$

for $i \in \mathbf{W}_{j'}$ with $j' < j$ have already been computed. We employ a parallel binary search. For every $i \in \mathbf{W}_j$, we set up a feasible range $\mathscr{R}_i$ initialized to $[-nW, nW]$. We will maintain the invariant that $F[i] \in \mathscr{R}_i$ and will halve the size of all feasible ranges $\mathscr{R}_i, i \in \mathbf{W}_j$ simultaneously using a small number of calls to the following problem $\mathbf{P}(M, \bar{W})$: Given an instance $\mathscr{J}$ for $\mathsf{CoinChange}$ specified by the weights $\tilde{w}_1, \dots, \tilde{w}_M$, as well as values $\tilde{r}_1, \dots, \tilde{r}_M \in [-\bar{W}, \bar{W}] \cup \{-\infty, \infty\}$, determine whether there exists an $i \in [M]$ with $T^{\mathscr{J}}[i] \le \tilde{r}_i$, and if so, also return a witness i. We will later prove that this problem can be solved in time $T^{\mathbf{P}}(M, \bar{W}) = \mathrm{O}(T^{\mathsf{CoinChange}}(2M, 3M^2\bar{W}))$. Clearly, after $\mathrm{O}(\log(nW))$ rounds of this parallel binary search, the feasible ranges consists of single values, thus determining the values of all $F[i]$ for $i \in \mathbf{W}_j$. Since we will show that halving all feasible ranges for range $\mathbf{W}_j$ takes $\mathrm{O}(N)$ calls to $\mathbf{P}(12N, nW)$, and we need to determine at most N ranges $\mathbf{W}_3, \dots, \mathbf{W}_N$, the total time for this process amounts to $\mathrm{O}(\log(nW)N^2 \cdot T^{\mathbf{P}}(12N, nW)) = \mathrm{O}(\log(nW)N^2 \cdot T^{\mathsf{CoinChange}}(24N, 3n^2W))$.

We now describe how to use $\mathbf{P}$ to halve the size of all feasible ranges $\mathscr{R}_i, i \in \mathbf{W}_j$: we set r_i to the median of $\mathscr{R}_i$ and aim to determine, for all $i \in \mathbf{W}_j$, whether $F[i] \le r_i$, i.e., whether some multiset S with $d(S) = i$ and $w(S) \le r_i$ exists. We achieve this by the following process: For every $k = 1, \dots, j$, we consider only two ranges, namely $\mathbf{W}_k = [(k-1)N+1, kN]$ and $\mathbf{W}_{j-k} \cup \mathbf{W}_{j-k+1} = [(j-k-1)N+1, (j-k+1)N]$. Let us first consider the case $k \ge 2$. Here, we can define the $2N$-dimensional vectors a, b with

$$a_\ell = \begin{cases} w_{(k-1)N+\ell} & \text{for } \ell \in [N], \\ \infty & \text{for } \ell > N, \end{cases}$$

$$b_\ell = F[(j-k-1)N+\ell] \qquad \text{for } \ell \in [2N].$$

(Note that all $F[i], i \in \mathbf{W}_{j-k} \cup \mathbf{W}_{j-k+1}$ for $k \ge 2$ have already been computed by assumption.) We are interested in all those values of $a * b$ of these vectors that correspond to summing up some $w_{(k-1)N+\ell}$ with some $F[(j-k-1)N+\ell']$ such that $(j-2)N+\ell+\ell' \in \mathbf{W}_j$. More specifically, we aim to determine whether there is some ℓ with $(a*b)_{N+\ell} \le r_{(j-1)N+\ell}$. To do so, we use the reduction from $(\min,+)$-Convolution to oiCoinChange given in Lemma 5.5.1 to create an oiCoinChange instance $\mathscr{J}$. From this instance of problem size $12N$ we can read off the values of $a*b$ as a certain interval in the corresponding $T^{\mathscr{J}}$-table. Thus, we can test whether $(a*b)_{N+\ell} \le r_{(j-1)N+\ell}$ for some ℓ using $\mathbf{P}(12N, nW)$: for every ℓ, we let i be the unique index in the $T^{\mathscr{J}}$-table representing the entry $(a*b)_{N+\ell}$ and set $\tilde{r}_i := r_{(j-1)N+\ell}$. For all other i', we set $\tilde{r}_{i'} = -\infty$, thus enforcing that those indices will never be reported.

For the special case $k = 1$, we proceed slightly differently: Here, we define the $2N$-dimensional vectors a, b with

$$a_\ell = F[\ell] \qquad \text{for } \ell \in [2N]$$
$$b_\ell = \begin{cases} F[(j-2)N+\ell] & \text{for } \ell \in [N] \\ \infty & \text{for } \ell > N. \end{cases}$$

(Note that all necessary $F[i], i \in \mathbf{W}_1 \cup \mathbf{W}_2$ and $F[i], i \in \mathbf{W}_{j-1}$ have already been computed by assumption.) Analogously to above, we use $\mathbf{P}(12N, nW)$ to test whether $(a*b)_{N+\ell} \le r_{(j-1)N+\ell}$ using the reduction from $(\min,+)$-Convolution to oiCoinChange given in Lemma 5.5.1.

Once an $i \in \mathbf{W}_j$ has been reported to satisfy $F[i] \le r_i$ for some witnessing subproblem given by the ranges $\mathbf{W}_k$ and $\mathbf{W}_{j-k} \cup \mathbf{W}_{j-k+1}$ for some k, we set $r_i := -\infty$ and repeat on the same subproblem k (analogously to the approach of Lemma 5.4.3). Note that for every j, we have $j \le N$ subproblems and at most N many indices $i \in \mathbf{W}_j$ that can be reported. Thus, we use at most $O(N)$ many calls to the subproblem $\mathbf{P}$.

To briefly argue correctness, note that by construction, we only determine some

i with $F[i] \leq r_i$ if we have found a witness. For the converse, let k be the largest index such that the optimal multiset for i includes a coin in $\mathbf{W}_k$. Then the subproblem given by the ranges $\mathbf{W}_k$ and $\mathbf{W}_{j-k} \cup \mathbf{W}_{j-k+1}$ will give a witness. This is obvious for $k \geq 2$. For $k = 1$, note that no weight in $\mathbf{W}_{k'}$ with $k' > 1$ is used in an optimal multiset for $F[i] \in \mathbf{W}_j$. In particular, the optimal multiset S can be represented as $S = S' \cup S''$, where S' is a multiset of total denomination $i' \in \mathbf{W}_{j-1}$ and S'' is a multiset of total denomination $i - i' \in \mathbf{W}_1 \cup \mathbf{W}_2$. Thus, in the instance constructed from a, b, we will find the witness $F[i] \leq F[i'] + F[i - i'] \leq r_i$.

We finally describe how to solve $\mathbf{P}(M, \bar{W})$ in time $T^{\mathsf{CoinChange}}(2M, 3M^2\bar{W})$. First consider the problem *without finding a witnessing i*. Let $\tilde{w}_1, \ldots, \tilde{w}_M, \tilde{r}_1, \ldots, \tilde{r}_M$ be an instance $\mathscr{J}$ of $\mathbf{P}(M, \bar{W})$. We define a CoinChange instance $\mathscr{K}$ of problem size $2M$ by giving the weights

$$
\begin{aligned}
w'_i &:= \tilde{w}_i && \text{for all } i \in [M], \\
w'_{2M-i} &:= -3M\bar{W} - \tilde{r}_i && \text{for all } i \in [M].
\end{aligned}
$$

We claim that $T^{\mathscr{K}}[2M] \leq -3M\bar{W}$ iff the input instance to $\mathbf{P}$ is a yes instance: First observe that $T^{\mathscr{K}}[1] = T^{\mathscr{J}}[1], \ldots, T^{\mathscr{K}}[M] = T^{\mathscr{J}}[M]$ since the first M weights agree for both $\mathscr{J}$ and $\mathscr{K}$. Consider the case that there is some $i \in [M]$ with $T^{\mathscr{J}}[i] \leq \tilde{r}_i$. Then we have $T^{\mathscr{K}}[2M] \leq T^{\mathscr{K}}[i] + w_{2M-i} = (T^{\mathscr{J}}[i] - \tilde{r}_i) - 3M\bar{W} \leq -3M\bar{W}$, as desired. Conversely, assume that all $T^{\mathscr{J}}[i] > \tilde{r}_i$. We distinguish the cases whether the optimal subsequence S uses only weights among $\tilde{w}_1, \ldots, \tilde{w}_M$ or not. In the first case, since $|\tilde{w}_i| \leq W$ for $i \in [M]$, we have that $w(S) \geq 2M \cdot \min_{i \in [n]} |\tilde{w}_i| \geq -2M\bar{W} > -3M\bar{W}$. Otherwise, S uses exactly one weight among $\tilde{w}_{M+1}, \ldots, \tilde{w}_{2M}$. Let this weight be $\tilde{w}_{2M-i}$. Then $w(S) = T^{\mathscr{K}}[i] + \tilde{w}_{2M-i} = (T^{\mathscr{J}}[i] - \tilde{r}_i) - 3M\bar{W} > -3M\bar{W}$ since $T^{\mathscr{J}}[i] > \tilde{r}_i$, yielding the claim.

Very similar to Lemma 5.4.3, we can now tune the above reduction to also produce a witness i such that $T^{\mathscr{I}}[i] \leq \tilde{r}_i$. For this, we scale all weights $w'_i, i \in [2M]$ by a factor of M and subtract a value of $i-1$ for every $w'_i, i \in [M]$. It is easy to see that a yes instance $\mathscr{K}$ attains some value $T^{\mathscr{K}}[2M] = -\kappa \cdot M - i$ for some integers $\kappa \geq 3$ and $0 \leq i < n$, where $i+1$ is a witness for $T^{\mathscr{I}}[i+1] \leq \tilde{r}_{i+1}$, thus computing $T^{\mathscr{K}}[2M]$ lets us derive a witness as well. Thus, problem **P** can be solved by a single call to $T^{\mathsf{CoinChange}}(2M, 3M^2\bar{W})$. □

The results above prove the following theorem.

Theorem 5.5.1. *We have*

$$(\min,+)\text{-}\mathsf{Convolution} \equiv_2 \mathsf{CoinChange} \equiv_2 \mathsf{UnboundedKnapsack}$$

Furthermore, the bounded version of Knapsack *admits no strongly subquadratic-time algorithm unless* $(\min,+)$-Convolution *can be solved in strongly subquadratic time.*

Proof. Lemma 5.5.1 and Lemma 5.5.2 prove $(\min,+)$-Convolution $\equiv_2$ oiCoinChange, while Observation 5.5.1 and Lemma 5.5.3 establish oiCoinChange $\equiv_2$ CoinChange $\equiv_2$ UnboundedKnapsack, yielding the first claim.

The second claim follows directly from inspecting the proofs of Lemma 5.5.1, Lemma 5.5.3 and the first claim of Observation 5.5.1 and observing that we only reduce to CoinChange/Knapsack instances in which the optimal multiset (for each total size) is always a set, i.e., uses each element at most once. □

5.6 Chain LWS

In this section we consider a special case of of Least-Weight Subsequence problems called the Chain Least-Weight Subsequence (ChainLWS) problem. This captures problems in which edge weights are given implicitly by a relation R that determines which pairs of data items we are allowed to chain. The aim is to find the longest chain.

An example of a Chain Least-Weight Subsequence problem is the NestedBoxes problem. Given n boxes in d dimensions, given as non-negative, d-dimensional vectors $b_1, \dots, b_n$, find the longest chain such that each box fits into the next (without rotation). We say box that box a fits into box b if for all dimensions $1 \leq i \leq d$, $a_i \leq b_i$.

NestedBoxes is not immediately a least-weight subsequence problem, as for least weight subsequence problems we are given a sequence of data items, and require any sequence to start at the first item and end at the last. We can easily convert NestedBoxes into a LWS problem by sorting the vectors by the sum of the entries and introducing two special boxes, one very small box $\perp$ such that $\perp$ fits into any box b_i and one very large box $\top$ such that any b_i fits into $\top$.

We define the chain least-weight subsequence problem with respect to any relation R and consider a weighted version where data items are given weights. To make the definition consistent with the definition of LWS the output is the weight of the sequence that minimizes the sum of the weights.

Problem 39 (ChainLWS)**.** *Fix a set of objects D and a relation $R \subseteq D \times D$. We define the following* LWS *instantiation* $\mathsf{ChainLWS}(R) = \mathsf{LWS}(\mathbf{W}_{\mathsf{ChainLWS}(R)})$.

Data items: *sequence of objects $d_0, \dots, d_n \in D$ with weights $w_1, \dots, w_n \in [-W, W]$.*

Weights: $w_{i,j} = \begin{cases} w_j & \text{if } (x_i, x_j) \in R, \\ \infty & \text{otherwise}, \end{cases}$ *for* $0 \leq i < j \leq n$.

The input to the (weighted) chain least-weight subsequence problem is a sequence of data items, and not a set. Finding the longest chain in a set of data items is NP-complete in general. For example, consider the box overlap problem: The input is a set of boxes in two dimensions, given by the top left corner and the bottom right corner, and the relation consists of all pairs such that the two boxes overlap. This problem is a generalization of the Hamiltonian path problem on induced subgraphs of the two-dimensional grid, which is an NP-complete problem [88].

We relate ChainLWS(R) to the class of Selection problems with respect to the same relation R.

Problem 40 (Selection)**.** *Let D be a set of objects, and let $D_1, D_2 \subseteq D^n$. Given two sequences of inputs $(a_1, \ldots, a_n) \in D_1$ and $(b_1, \ldots, b_n) \in D_2$ and a relation $R \subseteq D \times D$, determine if there is i, j satisfying $R(a_i, b_j)$. We denote this selection problem with respect to a relation R and sets D_1, D_2 by* Selection(R^{D_1,D_2})*. If $D_1 = D_2 = D^n$, we denote the problem by* Selection(R).

The class of Selection problems includes several well studied problems including MinInnProd, OrthogonalVectors [142, 7] and VectorDomination [80].

We will use the Selection problems in the search variant, where we find a pair satisfying the R if such a pair exists. To reduce the the search variant to the decision variants in a fine-grained way, we can use a simple, binary search type reduction from the decision problem to the search problem: If there is a pair (a_i, b_j) satisfying R, subdivide the input into subsets $A_1 = \{a_1, \ldots, a_{n/2}\}$, $A_2 = \{a_{n/2+1}, \ldots, a_n\}$, $B_1 = \{b_1, \ldots, b_{n/2}\}$, $B_2 = \{b_{n/2+1}, \ldots, b_n\}$, and decide if which of the four instances $(A_1, B_1), (A_1, B_2), (A_2, B_1), (A_2, B_2)$ contains a solution. We then recurse on the subproblem where there is a solution. If the decision version has time complexity $\mathrm{O}(n^c)$ for some constant c, then this algorithm for finding a pair has time complexity $\tilde{\mathrm{O}}(n^c)$.

We give a subquadratic reduction from ChainLWS(R) to Selection(R) that is independent of R.

Theorem 5.6.1. *For all relations R such that R can be computed in time subpolynomial in the number of data items n,* ChainLWS$(R) \leq_2$ Selection(R).

The proof is again based on Static-LWS and a variation on a trick of [150].

As an intermediate step, we define the Static-ChainLWS problem as the equivalent of the Static-LWS problem in the special case for chains.

Problem 41 (Static-ChainLWS)**.** *Fix an instance of* ChainLWS(R). *Given intervals* $I := \{a+1,\dots,a+N\}$ *and* $J := \{a+N+1,\dots,a+2N\}$ *for some* a *and* N*, together with the correctly computed values* $F[a+1],\dots,F[a+N]$*, the Static Chain Least-Weight Subsequence Problem (*Static-ChainLWS*) asks to determine*

$$F'[j] := \min_{i \in I : R(i,j)} F[i] + w_j \qquad\qquad \text{for all } j \in J.$$

Similar to the definition of ChainLWS, Static-ChainLWS is the special case of Static-LWS where the the weights $w_{i,j}$ are restricted to be either w_j or ∞, depending on R. As a result, Lemma 5.3.1 applies directly.

Corollary 5.6.1 (ChainLWS(R) $\leq_2$ Static-LWS(R))**.** *For any* R*, if* Static-ChainLWS(R) *can be solved in time* $\mathrm{O}(n^{2-\varepsilon})$ *for some* $\varepsilon > 0$*, then* ChainLWS(R) *can be solved in time* $\tilde{\mathrm{O}}(n^{2-\varepsilon})$.

We now reduce Static-ChainLWS(R) to Selection(R) with a variation on the trick by [150].

Lemma 5.6.1 (Static-ChainLWS(R) $\leq_2$ Selection(R))**.** *For all relations* R *such that* R *can be computed in time subpolynomial in the number of data items* n*,*

$$\text{Static-ChainLWS}(R) \leq_2 \text{Selection}(R)$$

Proof. As a first step, we sort the data items $a_i, i \in I = \{a+1,\dots,a+N\}$ by $F[i]$ in increasing order and we will assume for the remainder of the proof that for all $a+1 \leq i <$

$a+N$ we have $F[i] \leq F[i+1]$. We then split the set $a_{a+1},\dots,a_{a+N}$ into $g := \lceil\sqrt{N}\rceil$ groups $A_1,\dots,A_g$ with $A_i = \{a_{(i-1)\lceil N/g\rceil},\dots,a_{i\lceil N/g\rceil-1}\}$. We split the set $b_{a+N+1},\dots,b_{a+2N}$ into $B_1,\dots,B_g$ in a similar fashion. We then iterate over all pairs A_k,B_l with $k,l \in [g]$ in lexicographic order, and for each pair we do the following. Call the oracle for $\mathsf{Selection}(R)$ on the input A_k,B_l to find a pair a_i,b_j such that the relation R is satisfied on the pair. If there is no such pair, move to the next pair A_{k^*},B_{l^*} of sets of data items. If there is such a pair, find the first element $a_{i^*} \in A_k$ such that $R(a_{i^*},b_j)$ using a simple linear scan. As we first sorted A and iterate over sets A_k,B_l in lexicographic order, we have $F'[j] = F[i^*]+w_j$. We then remove b_j from B_l and repeat.

For the runtime analysis, we observe, that the oracle can find a pair of elements at most $\mathrm{O}(N)$ times, as each time we find a pair we remove an element from the input. In the case where we do find a pair of elements we do a linear scan that takes $\mathrm{O}(N/g)$ time. Furthermore, each pair of sets A_k,B_l can fail to find a pair at most once. Hence, if $T^{\mathsf{Selection}}$ is the time to solve the Selection problem and using $g = \sqrt{N}$ we get a time of

$$T(N) = NT^{\mathsf{Selection}}(\sqrt{N}) + N(T^{\mathsf{Selection}}(\sqrt{N}) + \sqrt{N}) = NT^{\mathsf{Selection}}(\sqrt{N}) \tag{5.14}$$

which is subquadratic if $T^{\mathsf{Selection}}(N)$ is subquadratic. □

For the other direction, we do not have a reduction that is independent of the relation R. Instead, we give sufficient conditions for the existence of such subquadratic reductions.

Theorem 5.6.2. *Let D be a set of objects and $D_1,D_2 \subseteq D^n$ be a set of possible sequences. For any relation $R \subseteq D \times D$ such that*

- *There is a data item $\bot$ such that $(\bot,d) \in R$ for all $d \in D$.*
- *There is a data item $\top$ such that $(d,\top) \in R$ for all $d \in D$.*

- *For all $a \in \{1,2\}$ and any set of data items $(d_1,\ldots,d_n) \in D_a$ there is a permutation of indices $i_1,\ldots,i_n$ such that for any $j < k$, $(d_{i_j}, d_{i_k}) \notin R$. This ordering can be computed in time $\mathrm{O}(n^{2-\delta})$ for $\delta > 0$. We call this ordering the natural ordering.*

Then $\mathsf{Selection}(R^{D_1,D_2}) \leq_2 \mathsf{ChainLWS}(R)$.

We call a relation satisfying the conditions above a *topological* relation.

Proof. We construct an unweighted ChainLWS problem with all weights set to -1, so that the problem is to find the longest chain. Let $(a_1,\ldots a_n) \in D_1$ and $(b_1,\ldots,b_n) \in D_2$ be the data items of $\mathsf{Selection}(R^{D_1,D_2})$ and sort both sets according to the natural ordering. We claim that for the sequence of data items $\bot, a_1,\ldots a_n, b_1,\ldots,b_n, \top$ the weight of the least weight subsequence is -3 exactly if there is a pair $(a_i,b_j) \in R$. Because of the property of the natural orderings, any valid subsequence starting at $\bot$ and ending at $\top$ contains at most one element a_i and at most one element b_j. If there is a pair $(a_i,b_j) \in R$, then the sequence $\bot, a_i, b_j, \top$ will have value -3. If there is no such pair, any valid sequence contains at most one element other than $\bot$ and $\top$ and its value is therefore at least -2. □

We call a relation satisfying the conditions above a *topological* relation.

Proof. We construct an unweighted ChainLWS problem with all weights set to -1, so that the problem is to find the longest chain. Let $(a_1,\ldots a_n) \in D_1$ and $(b_1,\ldots,b_n) \in D_2$ be the data items of $\mathsf{Selection}(R^{D_1,D_2})$ and sort both sets according to the natural ordering. We claim that for the sequence of data items $\bot, a_1,\ldots a_n, b_1,\ldots,b_n, \top$ the weight of the least weight subsequence is -3 exactly if there is a pair $(a_i,b_j) \in R$. Because of the property of the natural orderings, any valid subsequence starting at $\bot$ and ending at $\top$ contains at most one element a_i and at most one element b_j. If there is a pair $(a_i,b_j) \in R$, then the sequence $\bot, a_i, b_j, \top$ will have value -3. If there is no such pair, any valid

sequence contains at most one element other than $\perp$ and $\top$ and its value is therefore at least -2. □

In the remainder of this section we give some interesting instantiations of the subquadratic equivalence of Selection and ChainLWS.

Corollary 5.6.2 (NestedBoxes $\equiv_2$ VectorDomination)**.** *The weighted* NestedBoxes *problem on $d = c\log n$ dimensions can be solved in time $n^{2-(1/\mathrm{O}(c\log^2 c))}$. For $d = \omega(\log n)$, the (unweighted)* NestedBoxes *problem cannot be solved in time $\mathrm{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$ assuming* SETH.

Proof. Let R be the relation that contains all pairs of non-negative, d-dimensional vectors a, b such that $a_i \leq b_i$ for all i. Now Selection(R) is VectorDomination, and ChainLWS(R) is the NestedBoxes problem.

Using the reduction from Theorem 5.6.1 and the algorithms for vector domination of the stated runtime [80, 38] we immediately get an algorithm for NestedBoxes.

We apply Theorem 5.6.2 with $\top = W^d$ where W is the largest coordinate in all input vectors, $\perp = 0^d$ and use the sum of the coordinates of the boxes as the natural ordering. SETH-hardness of NestedBoxes then follows from the SETH-hardness of the VectorDomination problem [142]. □

If we restrict NestedBoxes and VectorDomination to Boolean vectors, then we get SubsetChain and SetContainment respectively. In this case the upper bound improves to $n^{2-1/\mathrm{O}(\log c)}$ [7]. Note that SetContainment$\equiv_2$ OrthogonalVectors, hence SubsetChain$\equiv_2$ OrthogonalVectors.

To give an example of an indirect use of Theorem 5.6.2, consider the space of all d-dimensional axis-aligned boxes in space (given by two corners as a pair of vectors $(v, \bar{v})$) and the relation R containing all boxes that overlap. While R is not a topological relation, we do the following reduction to the overlap problem on $d+2$ dimensions:

Let $(a_1,\overline{a}_1),\ldots,(a_n,\overline{a}_n),(b_1,\overline{b}_1),\ldots,(b_n,\overline{b}_n)$ be the input to $\mathsf{Selection}(R)$. Define $(a_i',\overline{a}_i')$ as $(a_i \circ i + 0.1 \circ 0, \overline{a}_i \circ i + 0.9, n)$ and $(b_i',\overline{b}_i')$ as $(b_i \circ 0 \circ i + 0.1, \overline{b}_i \circ n \circ i + 0.9)$ for all i. Note that $(a_i',\overline{a}_i')$ and $(a_j',\overline{a}_j')$ do not overlap for all $i \neq j$, $(b_i',\overline{b}_i')$ and $(b_j',\overline{b}_j')$ do not overlap for all $i \neq j$, but $(a_i',\overline{a}_i')$ and $(b_j',\overline{b}_j')$ overlap exactly if $(a_i,\overline{a}_i)$ and $(b_j,\overline{b}_j)$ overlap. We therefore reduced $\mathsf{Selection}(R)$ to $\mathsf{Selection}(R^{D_1,D_2})$ for some D_1,D_2 such that the relation is topological and can then apply Theorem 5.6.2.

We would also like to point out that the definition of ChainLWS requires the input to be a sequence of data items, and not a set. Consider the following definition:

Problem 42 (ChainSet). *Let $\{x_0,\ldots,x_n\}$ be a set of data items, weights $w_1,\ldots,w_{n-1} \in [-W,W]$ and a relation $R(x_i,x_j)$ be given. The chain set problem for R, denoted* ChainSet(R) *asks to find the sequence $i_0,i_1,i_2,\ldots,i_k$ such that for all j with $1 \leq j \leq k$ the pair $(x_{i_{j-1}},x_{i_j})$ is in the relation R and the weight $\sum_{j=1}^{k-1} w_{i_j}$ is minimized.*

The only difference between ChainLWS and ChainSet is that the input is given as a set of data items, as opposed to a sequence.

While ChainLWS can always be solved in quadratic time, the ChainSet problem is NP-complete. For example, if R is the box overlap relation (for $d = 2$) above, then this problem is a generalization of the Hamiltonian path problem on induced subgraphs of the two-dimensional grid, which is an NP-complete problem [88]. This is a formal barrier to a more general (black-box) reduction than Theorem 5.6.2, as any reduction from Selection to ChainLWS must impose an ordering on the data items.

5.7 Near-linear time algorithms

In this section, we classify problems to be solvable in near-linear time using the lens of our framework. Note that in these instances, near-linear time solutions have already been known, however, our focus on the static variants of LWS provides a simple, general approach to find fast algorithms by identifying a simple core problem. Since in this paper, we generally ignore subpolynomial factors in the running time, we concentrate here on the reduction from some LWS variant to its corresponding core problem and disregard reductions in the other direction.

5.7.1 Longest Increasing Subsequence

The longest increasing subsequence problem LIS has been first investigated by Fredman [60], who gave an $O(n \log n)$-time algorithm and gave a corresponding lower bound based on Sorting. The following LWS instantiation is equivalent to LIS.

Problem 43 (LIS)**.** *We define the following* LWS *instantiation* $\mathsf{LWS}(\mathbf{W}_{\mathsf{LIS}})$.

Data items: *integers* $x_1, \dots, x_n \in \{1, \dots, W\}$

Weights: $w_{i,j} = \begin{cases} -1 & \text{if } x_i < x_j \\ \infty & \text{ow.} \end{cases}$

It is straightforward to verify that $-F[n]$ yields the value of the longest increasing subsequence of $x_1, \dots, x_n$. Using the static variant of LWS introduced in Section 5.4, we observe that LIS effectively boils down to Sorting.

Observation 5.7.1. LIS *can be solved in time* $\tilde{O}(n)$.

Proof. By Lemma 5.3.1, we can reduce LIS to the static variant Static-LWS($\mathbf{W}_{\mathsf{LIS}}$). It is straight-forward to see that the latter can be reformulated as follows: Given $(a_1, F[1]), \dots, (a_N, F[N])$ and $b_1, \dots, b_N$, determine for every $j = 1, \dots, N$, the value $F'[j] = -1 + \min_{1 \le i \le N, a_i < b_j} F[i]$. For this purpose, it suffices to sort the first list $(a_{i_1}, F[i_1]), \dots, (a_{i_N}, F[i_N])$ such that $a_{i_1} \le \cdots \le a_{i_N}$ and the second as $b_{j_1}, \dots, b_{j_N}$ with $b_{j_1} \le \cdots \le b_{j_N}$. Finally, a single pass over both lists will do: For each $k = 1, \dots, N$, we search for the largest ℓ such that $a_{i_\ell} < b_{j_k}$, then the T'-value corresponding to b_{j_ℓ} is $-1 + \min_{1 \le \ell' \le \ell} F[i_{\ell'}]$. By this approach, it is easy to see that after sorting, these values can be computed in time $\mathrm{O}(N)$. For the exact running time, note that solving $\mathsf{Static\text{-}LWS}(\mathbf{W}_{\mathsf{LIS}})$ takes time $\mathrm{O}(N \log N)$ due to sorting, yielding a $\mathrm{O}(n \log^2 n)$-time algorithm for LIS by Lemma 5.3.1. $\square$

5.7.2 Unbounded Subset Sum

UnboundedSubsetSum is a variant of the classical SparseSubsetSum, in which repetitions of elements are allowed. While improved pseudo-polynomial-time algorithms for SparseSubsetSum could only recently be found [99, 32], there is a simple algorithm solving UnboundedSubsetSum in time $\mathrm{O}(n \log n)$ [32]. It can be cast into an LWS formulation as follows.

Problem 44 (UnboundedSubsetSum)**.** *The* UnboundedSubsetSum *problem is defined as the* LWS *instantiation* $\mathsf{LWS}(\mathbf{W}_{\mathsf{UnboundedSubsetSum}})$.

Data items: $S \subseteq [n]$

Weights: $w_{i,j} = \begin{cases} 0 & \text{if } j-i \in S \\ \infty & \text{ow.} \end{cases}$

Note that in this formulation, $F[n] = 0$ if and only if there is a multiset of numbers from S that sums up to n. It is a straightforward observation that the static variant of UnboundedSubsetSum can be solved by classical convolution, i.e., $(\cdot, +)$-convolution.

Observation 5.7.2. UnboundedSubsetSum *can be solved in time* $\tilde{\mathrm{O}}(n)$.

Proof. Noting that all weights $w_{i,j}$ are either 0 or ∞, it is easy to see that the static variant $\mathsf{Static\text{-}LWS}(\mathbf{W}_{\mathsf{UnboundedSubsetSum}})$ can be reformulated as follows: Given a subset $X \subseteq I = \{a+1, \dots, a+N\}$, determine, for all $j \in J = \{a+N+1, \dots, a+2N\}$, whether there exists some $i \in X$ such that $j - i \in S$. To do so, we do the following: We represent X as an N-bit vector $x = (x_1, \dots, x_N) \in \{0,1\}^N$ with $x_i = 1$ iff $a + i \in X$. Furthermore, we represent the relevant part of S by defining a $2N$-bit vector $s = (s_1, \dots, s_{2N}) \in \{0,1\}^{2N}$

with $s_i = 1$ iff. $i \in S$. Then the $(\cdot,+)$-convolution $r = x \circledast s$ of x and s allows us to determine $F'[a+N+j]$ for $j = 1,\dots,N$: this values is 0 iff $r_{N+j} > 0$ and ∞ otherwise. Correctness follows from the observation that $r_{N+j} > 0$ is equivalent to the existence of some $i \in [N]$ and $k \in [2N]$ with $i+k = N+j$ and $x_i = s_k = 1$. This in turn is equivalent to $a+i \in X$ and $(a+N+j)-(a+i) = N+j-i = k \in S$, as desired.

Thus Static-LWS($\mathbf{W}_{\mathsf{UnboundedSubsetSum}}$) can be solved by a single convolution computation, which can be performed in time $O(N \log N)$. Thus by Lemma 5.3.1, this gives rise to a $O(n \log^2 n)$-time algorithm for UnboundedSubsetSum. □

5.7.3 Concave LWS

The ConcaveLWS problem is a special case of LWS in which the weights satisfy the quadrangle inequality. Since a complete description of the input instance consists of $\Omega(n^2)$ weights, we use the standard assumption that each $w_{i,j}$ can be queried in constant time. This allows for sublinear solutions in the input description, in particular there exist $O(n)$-time algorithms [140, 65].

Problem 45 (ConcaveLWS)**.** *We define the* LWS *instantiation* LWS($\mathbf{W}_{\mathrm{CONC}}$).
Weights: *$w_{i,j}$ given by oracle access, satisfying $w_{i,j} + w_{i',j'} \le w_{i',j} + w_{i,j'}$ for $i \le i' \le j \le j'$.*

We revisit ConcaveLWS and its known connection to the problem of computing column (or row) minima in a totally monotone[5] $(n \times n)$-matrix, which we call the SMAWK problem because of its remarkable $O(n)$-time solution called the SMAWK algorithm [12].

Observation 5.7.3. ConcaveLWS *can be solved in time* $\tilde{O}(n)$.

Proof. The static variant of ConcaveLWS can be formulated as follows: Given intervals $I = \{a+1,\dots,a+N\}$ and $J = \{a+N+1,\dots,a+2N\}$, we define a matrix

[5]A matrix $M = (m_{i,j})_{i,j}$ is totally monotone if for all $i < i'$ and $j < j'$, we have that $m_{i,j} > m_{i',j}$ implies that $m_{i,j'} > m_{i',j'}$. For a more comprehensive treatment, we refer to [12, 65].

$M := (m_{i,j})_{i \in I, j \in J})$ with $m_{i,j} = F[i] + w_{i,j}$. It is easy to see that M is a totally monotone matrix since w satisfies the quadrangle inequality. Note that the minimum of column $j \in J$ in M is $\min_{i \in I} F[i] + w_{i,j} = F'[j]$ by definition. Thus, using the SMAWK algorithm we can determine all $F'[j]$ in simultaneously in time $O(N)$.

Thus by Lemma 5.3.1, we obtain an $O(n \log n)$-time algorithm for ConcaveLWS.

□

5.8 Open Problems

We discuss the complexity of some succinct LWS instantiations both from an upper bound and a lower bound perspective by proving equivalences with a number of comparably well-studied core problems. The succinct instantiations we study include natural problems such as LowRankLWS, CoinChange, ChainLWS including NestedBoxes and SubsetChain, as well as previously studied instantiations such as ConcaveLWS and LIS. A number of open questions remain. Our results do not generalize to arbitrary instantiations of LWS. In particular, Static-LWS does not reduce subquadratically to the problem of finding the minimum element in a succinctly described matrix. With LowRankLWS and CoinChange we do provide instances we can identify equivalent core problems, and it will be interesting to find further examples or even sufficient conditions for which we can reduce LWS to other problems and vice versa.

For the case of ChainLWS, we are able to generalize the reduction from LWS to Selection problems. However, the reduction, while preserving subquadratic algorithms, does not preserve near-linear algorithms. For some cases, such as LIS, we are able to reconstruct the near-linear time algorithm, which raises the question of what conditions are necessary to do that. Similarly, we give sufficient conditions to reduce from Selection to ChainLWS, and other sufficient or even necessary conditions should be explored for both black-box, as well as white-box reductions.

Chapter 5 is based on material as it appears in the following publications: Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. “On the Fine-grained Complexity of One-Dimensional Dynamic Programming”. To appear in the International Colloquium on Automata, Languages, and Programming, 2017. The author of this dissertation was a principal author of this publication. We would like to thank Karl Bringmann and Russell Impagliazzo for helpful discussions and comments on the material in Chapter 5.

Chapter 6

Fine-Grained Non-Reducibility

6.1 Introduction

In recent years, fine-grained complexity and conditional lower bounds in particular have been a fruitful area of study, with the number of problems where we can explain their complexity growing steadily. Unfortunately, as our understanding of the relationship between the exact complexities of problems grows, so does the complexity of the web of known reductions and the number of distinct conjectures these results are based on. Ideally, we would like to show that many of these conjectures are in fact equivalent, or that all follow from some basic unifying hypothesis, thereby improving our understanding and simplifying the state of knowledge. For example, it would be nice to show that the 3-sum conjecture or the APSP conjecture follows from SETH. A result like that would reduce the number of conjectures we rely on to explain the complexity of problems.

At the same time, many problems seem to be hard, but their hardness is not explained by any of the three most popular conjectures in fine-grained complexity, SETH, the 3-sum conjecture and the APSP conjecture. Among these questions is if HittingSet can be solved in subquadratic time or if MaxFlow has a linear time algorithm. For neither of these problems can we answer the question positively with an algorithm nor negatively with a conditional lower bound at this point.

In this chapter, we introduce a new technique which provides evidence that unifying hardness results under one unifying hypothesis such as SETH is unlikely, at least when restricted to deterministic reductions.

Our technique is a continuation of the idea to consider fine-grained versions of traditional complexity assumptions. The Strong Exponential Time Hypothesis is a fine-grained version of $\mathsf{P} \neq \mathsf{NP}$. In this chapter we consider a fine-grained version of $\mathsf{NP} \neq \mathsf{coNP}$,which we call the *Nondeterministic Strong Exponential Time Hypothesis* (NSETH).

Our goal is to get a fine-grained version of non-reducibility results based on $\mathsf{NP} \neq \mathsf{coNP}$. If any problem in $\mathsf{NP} \cap \mathsf{coNP}$ can be shown to be NP-complete, then $\mathsf{NP} = \mathsf{coNP}$. Hence, assuming $\mathsf{NP} \neq \mathsf{coNP}$, placing a problem L in $\mathsf{NP} \cap \mathsf{coNP}$ means that L cannot be NP-complete (or coNP-complete) and there is no polynomial time reduction from an NP-complete problem like 3-sat to L.

Similarly, by considering the nondeterministic and co-nondeterministic complexities of problems on a more fine-grained level, we deduce non-reducibility results based on NSETH. We define NSETH formally as follows.

Definition 6.1.1 (Nondeterministic Strong Exponential Time Hypothesis (NSETH))**.** *For every $\varepsilon > 0$, there exists a k so that* k-sat *is not in* $\mathsf{coNTIME}[2^{n(1-\varepsilon)}]$.

Equivalently, the k-taut problem, the tautology problem on k-DNF formulas, is not in $\mathsf{NTIME}[2^{n(1-\varepsilon)}]$ for sufficiently large k.

We feel that NSETH is plausible for many of the same reasons as SETH. We can think of NSETH as a statement about proof systems. Just as many algorithmic techniques have been developed for k-sat, all of which approach exhaustive search for large k, many proof systems have been considered for k-taut, and none have been shown to have significantly less than 2^n complexity for large k. In fact, the tree-like ([121]) and regular resolution ([22]) proof systems have been proved to require such sizes. Moreover, we observe that results of [89] (see Section 2.9) that obtain circuit lower bounds assuming SETH is false yield the same bounds assuming that NSETH is false. So disproving NSETH would be both a breakthrough in proof complexity and in circuit complexity.

We also consider the natural nondeterministic variant of ETH.

Definition 6.1.2 (Nondeterministic Exponential Time Hypothesis (NETH))**.** *The* 3-sat *problem on n variables requires is not in* coNTIME$[2^{\varepsilon n}]$ *for some* $\varepsilon > 0$.

We study the complexity of problems with respect to their natural complexity T. In particular, we show that if both the nondeterministic and co-nondeterministic problem is bounded by $T^{1-\varepsilon}$, then the problem cannot be SETH-hard under deterministic reductions at time T, assuming NSETH.

We show non-reducibility results for the following problems and time bounds.

- HittingSet for sets of total size m and time $T(m) = m^{1+\gamma}$ is not SETH-hard for any $\gamma > 0$, and no problem that is SETH-hard fine-grained reduces to HittingSet for any such time complexity.
- 3-sum for $T(n) = n^{1.5+\gamma}$ is not SETH-hard for any $\gamma > 0$.
- MaxFlow, MinCostMaxFlow, and MaximumMatching on a graph with m edges and $T(m) = m^{1+\gamma}$ are not SETH-hard.
- APSP on a graph with n vertices and $T(n) = n^{\frac{3+\omega}{2}+\gamma}$, where ω is the matrix multiplication exponent, is not SETH-hard.

All the results above are assuming NSETH and under deterministic reductions.

While there are many known SETH-hard problems, few are graph problems, and those few have the same logical structure. In addition to specific problems, our method can be used to explain why the structure of SETH-hard graph problems are all similar. In particular, we consider first-order definable graph properties on sparse graphs (where we view the input size as the number of edges m). Under SETH there are such first-order definable functions that require time $\Omega(m^{k-1-\varepsilon})$ for all $\varepsilon > 0$. At the same time, the problem can be solved in time $O(m^{k-1})$ for all properties. On the other hand, if NSETH, all SETH-hard properties at time m^{k-1} have the same logical structure: $k-1$ quantifiers of one type, followed by a single quantifier of the other type.

These results are only valid for deterministic or zero-error probabilistic fine-grained reductions. We introduce a non-uniform variant NUNSETH under which they also hold for randomized reductions with bounded error. However, some care should be used to evaluate whether this hypothesis is true, since it has not been the subject to previous study and Williams has shown related hypotheses about the Merlin-Arthur complexity of k-taut are false ([148]).

6.2 Outline

In Section 6.3, we take another look at fine-grained reductions with an emphasis on properties with regard to nondeterministic complexity and show how properties of fine-grained reductions can be used for non-reducibility results under NSETH. In Section 6.4, we consider the connection between satisfiability algorithms and circuit lower bounds (see Section 2.9) with respect to NSETH and give reasons why disproving NSETH might be difficult. In Section 6.5, we examine the nondeterministic and co-nondeterministic complexities of several problems within polynomial time whose exact complexities have been extensively studied, and show that, under NSETH, none of these problems are SETH-hard. In Section 6.6, we explain why all the known maximally hard SETH-hard first-order graph properties have the same logical structure.

Finally Section 6.7 discusses the implications of NSETH with respect to functional and verification problems.

6.3 Definitions and basic properties

In this section we consider fine-grained reductions, as defined in Section 2.4, with respect to nondeterminism. Remember the definition of a (deterministic) fine-grained reduction.

Definition 2.4.1 (Fine-Grained Reductions ($\leq_{FGR}$))**.** *Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1,T_1)* fine-grained reduces *to (L_2,T_2) (denoted $(L_1,T_1) \leq_{FGR} (L_2,T_2)$) if for all $\varepsilon > 0$, there is a $\delta > 0$ and a deterministic Turing reduction $\mathscr{M}^{L_2}$ from L_1 to L_2 satisfying the following conditions.*

(a) The time complexity of the Turing reduction without counting the oracle calls is bounded by $T_1^{1-\delta}$.

$$\mathsf{TIME}[\mathscr{M}] \leq T_1^{1-\delta} \tag{2.2}$$

(b) Let $\tilde{Q}(\mathscr{M},x)$ denote the set of queries made by $\mathscr{M}$ to the oracle on an input x of length n. The query lengths obey the following time bound.

$$\sum_{q \in \tilde{Q}(\mathscr{M},x)} (T_2(|q|))^{1-\varepsilon} \leq (T_1(n))^{1-\delta}$$

Given two problems L_1 and L_2, and $(L_1,T_1) \leq_{FGR} (L_2,T_2)$ for some time bounds T_1 and T_2, then $L_2 \in \mathsf{TIME}[T_2(n)^{1-\varepsilon}]$ for some $\varepsilon > 0$ implies $L_1 \in \mathsf{TIME}[T_1(n)^{1-\delta}]$ for some $\delta > 0$. We first argue that the same fine-grained reduction also implies that the existence of both a fast nondeterministic and co-nondeterministic algorithm for L_2 implies that there are also fast nondeterministic and co-nondeterministic algorithms for L_1.

Lemma 6.3.1 (Fine-grained reductions transfer savings for $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$)**.** *Let L_1, L_2 be languages and T_1, T_2 be time bounds. Further let $(L_1, T_1) \leq_{FGR} (L_2, T_2)$, and $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2(n)^{1-\varepsilon}]$ for some $\varepsilon > 0$. Then there exists a $\delta > 0$ such that*

$$L_1 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1(n)^{1-\delta}]$$

Proof. Both the nondeterministic algorithm for L_1 and $\neg L_1$ follow the same outline. We simulate the deterministic Turing reduction $\mathscr{M}^{L_2}$. For the oracle calls, we nondeterministically guess for instances $q \in \tilde{Q}(\mathscr{M}, x)$ if $q \in L_2$ or not and simulate either the nondeterministic machine for L_2 or $\neg L_2$. We can therefore simulate each oracle call q in $\mathsf{NTIME}[T_2(|q|)^{1-\varepsilon}]$ for some $\varepsilon > 0$. By the properties of the fine-grained reduction we have $\mathsf{TIME}[\mathscr{M}] \leq T_1^{1-\delta}$ and $\sum_{q \in \tilde{Q}(\mathscr{M},x)} (T_2(|q|))^{1-\varepsilon} \leq (T_1(n))^{1-\delta}$, and hence $L_1 \in \mathsf{NTIME}[T_1(n)^{1-\delta}]$. Similarly we have $L_1 \in \mathsf{coNTIME}[T_1(n)^{1-\delta}]$ as we can negate the output of the fine-grained reduction $\mathscr{M}^{L_2}$. $\square$

As an immediate consequence of Lemma 6.3.1 we get a way to show non-reducibility assuming NSETH.

Corollary 6.3.1. *Assuming* NSETH*, for any problem L and time bound T, if $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T(n)^{1-\delta}]$ for some $\delta > 0$, then L is not* SETH*-hard under deterministic reductions at time T.*

While Lemma 6.3.1 and Corollary 6.3.1 are formulated with respect to deterministic fine-grained reductions, we can extend the result to nondeterministic and zero-error reductions. On the other hand, the results do not extend to randomized reductions.

We define nondeterministic fine-grained reductions as follows.

Definition 6.3.1 (Nondeterministic Fine-Grained Reductions)**.** *Let L_1 and L_2 be languages, and let T_1 and T_2 be time bounds. We say that (L_1, T_1)* nondeterministically fine-grained reduces *to* (L_2, T_2) *if for all* $\varepsilon > 0$*, there is a* $\delta > 0$ *and two nondeterministic Turing reductions* $\mathscr{M}_1^{L_2}$ *and* $\mathscr{M}_2^{L_2}$ *satisfying the following conditions.*

(a) For all $x \in L_1$*, then there is a* y *with* $|y| \leq T_1^{1-\delta}$ *such that* $\mathscr{M}_1(x,y)$ *accepts.*

(b) For all $x \notin L_1$*, then there is a* y *with* $|y| \leq T_1^{1-\delta}$ *such that* $\mathscr{M}_2(x,y)$ *accepts.*

(c) The time complexity of both Turing reduction without counting the oracle calls is bounded by $T_1^{1-\delta}$*, that is, for* $c \in \{1,2\}$

$$\mathsf{TIME}[\mathscr{M}_c] \leq T_1^{1-\delta} \tag{6.1}$$

(d) Let $\tilde{Q}(\mathscr{M}, x)$ *denote the set of queries made by* $\mathscr{M}$ *to the oracle on an input* x *of length* n*. The query lengths obey the following time bound for* $c \in \{1,2\}$.

$$\sum_{q \in \tilde{Q}(\mathscr{M}_c, x)} (T_2(|q|))^{1-\varepsilon} \leq (T_1(n))^{1-\delta}$$

To prove Lemma 6.3.1 under nondeterministic reductions we can nondeterministically guess y and simulate $\mathscr{M}_1$ and $\mathscr{M}_2$ similar to the deterministic case to get nondeterministic Turing machines for L_1 and $\neq L_1$.

We use nondeterministic reductions in Section 6.5.6 to strengthen our result on the APSP problem.

We will not define zero-error fine-grained reductions formally but note that non-reducibility extends to zero-error reductions.

As we will show, many of the problems such as k-sum and HittingSet which have served as starting points for fine-grained reductions have substantially smaller nondeterministic complexities than their conjectured deterministic complexities. From the above closure properties, it will follow that if NSETH is true, none of these problems is SETH-hard under deterministic (or zero-error probabilistic) fine-grained reductions. This leaves a major loophole: these problems might still be SETH-hard under randomized reductions.

If we mimic the proof of Lemma 6.3.1 with randomized reductions, we get a statement on Merlin-Arthur protocols. In a Merlin-Arthur protocol for L, for any $x \in L$ there is a proof y that is accepted with high probability, and for any $x \in L$, any proof is rejected with high probability. We can show that $(L_1, T_1) \leq_{rFGR} (L_2, T_2)$ and $L_2 \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_2(n)^{1-\varepsilon}]$ for some $\varepsilon > 0$ implies that for some $\delta > 0$, there is a Merlin-Arthur protocol for L_1 with time $T_1(n)^{1-\delta}$. However, a fast Merlin-Arthur protocol for k-sat exists. In particular, Williams [148] shows that for any k, there is a Merlin-Arthur protocol for k-sat with time $\tilde{\mathsf{O}}(2^{n/2})$. A Merlin-Arthur version of SETH (MASETH) would therefore be false.

We outline a reason why even randomized SETH-hardness would still be somewhat surprising. We introduce a non-uniform version of NSETH, NUNSETH, and show that this hypothesis would imply the non-existence of even randomized SETH-hardness results.

Definition 6.3.2. *Let* k-taut *be the tautology problem restricted to* k-DNF*'s. The* Non-uniform Nondeterministic Strong Exponential Time Hypothesis *(*NUNSETH*) is the statement:* $\forall \varepsilon > 0 \, \exists k \geq 0$, *so that there are no nondeterministic circuit families of size* $\mathsf{O}(2^{n(1-\varepsilon)})$ *recognizing the language* k-taut.

While we do not have any general conservation of non-uniform nondeterministic time by randomized reductions, we do have a limit for the special case of problems that are SETH-hard under randomized reductions.

Lemma 6.3.2. *Assume L is* SETH*-hard at time $T(N)$ via a randomized reduction. If* NUNSETH, *then there is no $\delta > 0$ so that $L \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T^{1-\delta}(n)]$.*

Proof. Let ε be the constant corresponding to δ in the reduction, and let $\mathscr{M}^L$ be the corresponding randomized oracle machine. Let $m < n^k$ be the length in bits of a description of a $\mathsf{k\text{-}sat}$ formula on n inputs. By repeating $\mathscr{M}^L$ $\mathrm{O}(m)$ times and taking the majority answer, we can make the error probability less than 2^{-m}. Therefore, there is one random tape that has no errors, using the standard argument that $\mathsf{BPP} \in \mathsf{P/poly}$. Since $\mathscr{M}$ runs in total time $2^{(1-\varepsilon)n}$, this tape will have length at most $m2^{(1-\varepsilon)n}$, and so will be an exponential improvement over 2^n. Once we have fixed the tape, we can simulate the oracle queries nondeterministically as in the case of deterministic reductions, with total complexity $\mathrm{O}(m)$ times what it is for one run. Thus, we get a nondeterministic circuit with total size $\mathrm{O}(m2^{(1-\varepsilon)n})$. □

Note that the above argument, in addition to needing advice, multiplies the complexity by an amount polynomial in the input size. While this is not an issue for $\mathsf{k\text{-}sat}$, it would render the consequences of randomized reductions for problems within P moot, since we are trying to preserve exact polynomial complexities.

While $\mathsf{NUNSETH}$ seems plausible, we should exercise some caution before adopting it as an axiom. First, there are no known consequences if $\mathsf{NUNSETH}$ fails to be true. Secondly, Williams' result [148] disproving MASETH does remind us that counter-intuitive things can happen when randomness and nondeterminism are combined, so we should be cautious in assuming non-uniformity might not speed up computation in this circumstance. Because there is a polynomial overhead in making such a protocol a nondeterministic algorithm with advice, the efficient Merlin-Arthur protocol for $\mathsf{k\text{-}sat}$ does however not contradict $\mathsf{NUNSETH}$ directly.

6.4 What if NSETH is false?

SETH is an interesting hypothesis because for both the case where SETH is true and the case where it is false, we can prove consequences that seem difficult to prove unconditionally. While the main focus of this dissertation is on consequences of SETH being true, in this section we explore some consequences of it being false. In particular we observe that we can reformulate earlier results by Williams [143] and Jahanjou, Miles and Viola [89] on consequences of SETH being false as consequences that NSETH is false. This is evidence that NSETH will be hard to refute.

A general framework to show that algorithms for $\mathscr{C}$-sat imply circuit lower bounds is due to Williams (see [143] and [147], and Section 2.9 for a brief summary). We observe that the framework is more general than presented by Williams, as co-nondeterministic algorithms for $\mathscr{C}$-sat are sufficient to show the same lower bounds.

In Williams' proof we show a contradiction to the nondeterministic time hierarchy theorem by starting with an arbitrary language in $\mathsf{NTIME}[2^n]$ and show that it is $\mathsf{NTIME}[2^n/\omega]$ for some superpolynomial ω. The proof constructs a circuit using nondeterminism and then verifies that the circuit is unsatisfiable using using the supposed efficient satisfiability algorithm. However, since we have nondeterminism at our disposal, a co-nondeterministic algorithm for satisfiability, i.e. a nondeterministic algorithm to prove unsatisfiability, is sufficient. Williams' framework therefore does not need to assume efficient deterministic satisfiability algorithm, but only efficient co-nondeterministic satisfiability algorithms.

Jahanjou, Miles and Viola [89] give optimized reductions from $\mathscr{C}$-sat for a number of circuit classes to k-sat. Reformulated their result using our nondeterministic extensions, the following result is implicit in their work:

Theorem 6.4.1. *We have the following implications from the failure of* NETH *and*

NSETH*:*

1. *If* NETH *is false; i.e., for every* $\varepsilon > 0$, 3-sat *is in co-nondeterministic time* $2^{\varepsilon n}$*, then there exists a problem* $f \in \mathsf{E}^{\mathsf{NP}}$ *such that* f *does not have linear-size circuits.*

2. *If* NSETH *is false; i.e., there is a* $\delta < 1$ *such that for every* k*,* k-sat *is in co-nondeterministic time* $2^{\delta n}$*, then there exists a problem* $f \in \mathsf{E}^{\mathsf{NP}}$ *such that* f *does not have linear-size series-parallel circuits.*

3. *If there is* $\alpha > 0$ *such that* n^{α}-sat *is in co-nondeterministic time* $2^{n-\omega(n/\log\log n)}$*, then there is a problem* $f \in \mathsf{E}^{\mathsf{NP}}$ *such that* f *does not have linear-size log-depth circuits.*

6.5 The nondeterministic time complexity of problems in P

How could we show that one language is not reducible to another language? There is an ever-growing web of problems, hypotheses, and reductions that reflect the fine-grained complexity approach to explaining hardness. Could this structure collapse into a radically simpler graph, with just a few equivalence classes? If we assume NSETH, probably not as much as one might hope.

We can broadly categorize computational problems into two sets. In the first category, the deterministic time complexity is higher than both the nondeterministic and co-nondeterministic time complexity. In the second category, at least one of nondeterminism or co-nondeterminism does not help in solving the problem more efficiently. Lemma 6.3.1 shows that savings in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ are preserved under deterministic fine-grained reductions. As a result, we can rule out tight reductions from a problem that is hard using nondeterminism or co-nondeterminism to a problem that is easy in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$.

If NSETH holds, then $\mathsf{k\text{-}sat}$ is in the category of problems that do not benefit from co-nondeterminism. So, any problem that is SETH-hard under deterministic reductions also falls into this category.

In this section we explore problems that do benefit from $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$, i.e. we give nondeterministic algorithms that are faster than their presumed deterministic time complexities. This rules out deterministic fine-grained reductions from $\mathsf{CNF\text{-}sat}$ to these problems with their presumed time complexities. As a consequence, it is not possible to show that these problems are SETH-hard using a deterministic reduction.

We begin by formalizing the notion of non-reducibility.

Theorem 6.5.1 (NSETH implies no reduction from $\mathsf{CNF\text{-}sat}$)**.** *If* NSETH *and* $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T]$ *for some problem C and time T, then* $(\mathsf{CNF\text{-}sat}, 2^n) \not\leq_{FGR} (C, T^{1+\gamma})$ *for any* $\gamma > 0$.

Proof. Assume NSETH and $(\mathsf{CNF\text{-}sat}, 2^n) \leq_{FGR} (C, T^{1+\gamma})$, and $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T]$. By Lemma 6.3.1, preservation of $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ savings under fine-grained reductions, there exists $\delta > 0$ such that $\mathsf{CNF\text{-}sat} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[2^{n(1-\delta)}]$. This contradicts NSETH, therefore it cannot be the case (under NSETH) that $(\mathsf{SAT}, 2^n) \leq_{FGR} (C, T)$. $\square$

Corollary 6.5.1 (NSETH implies no reductions from SETH-hard problems)**.** *If* NSETH *holds and* $C \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[T_1]$*, then for any problem B that is* SETH*-hard under deterministic reductions with time T_2, and* $\gamma > 0$*, we have*

$$(B, T_2) \not\leq_{FGR} (C, T_1^{1+\gamma})$$

Proof. Assume NSETH, and that (B, T_2) is SETH-hard. Therefore, we know

$$(\mathsf{CNF\text{-}sat}, 2^n) \leq_{FGR} (B, T_2) \tag{6.2}$$

Now assume $(B,T_2) \leq_{FGR} (C,T_1^{1+\gamma})$. Then by Lemma 2.4.1, composition of fine-grained reductions, we have that $(\mathsf{CNF\text{-}sat}, 2^n) \leq_{FGR} (C,T_1)$. But by Theorem 6.5.1 above, this is impossible under NSETH. □

We now give the main result of this section.

Theorem 6.5.2. *Under* NSETH, *there is no deterministic or zero-error fine-grained reduction from* SAT *or any* SETH-*hard problem to the following problems with the following time complexities for any* $\gamma > 0$.

- MaxFlow, *min-cost* MaxFlow, *and maximum matching with* $T(m) = m^{1+\gamma}$
- HittingSet *with* $T(m) = m^{1+\gamma}$
- 3-sum *with* $T(n) = n^{1.5+\gamma}$
- *All-pairs shortest path with* $T(n) = n^{\frac{3+\omega}{2}+\gamma}$

Note that for graph problems, n refers to the number of vertices, m refers to the number of edges, and ω is the matrix multiplication exponent.

To prove Theorem 6.5.2 we give both nondeterministic and co-nondeterministic algorithms for these problems.

6.5.1 Maximum Flow

The maximum flow problem has been an extensively studied problem for decades and has a large number of theoretical and practical applications. While approximate maximum flow on undirected graphs has a $\tilde{O}(m)$ algorithm [92], where m is the number of edges, no linear time algorithm is known for the exact version of the problem.

A natural question from the point of conditional hardness is if we can prove a superlinear lower bound by proving that the problem is SETH-hard.

In this section we use the max-flow/min-cut theorem to give a $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$ algorithm for the decision version of max-flow with time linear in the number of edges. Assuming NSETH, we can then conclude that there is no deterministic fine-grained reduction from any SETH-hard problem to maximum flow with a superlinear time bound.

Problem 46 (MaxFlow)**.** *Let $G = (V, E)$ be a connected, directed graph with capacity constraints, $s, t \in V$ be vertices and $k \in \mathbb{R}$.*

*The maximum flow problem (*MaxFlow*) is to decide if there exists a flow from s to t of value at least k.*

The nondeterministic algorithm for maximum flow is straight-forward and the co-nondeterministic algorithm follows directly for the max-flow/min-cut theorem.

Lemma 6.5.1. $\mathsf{MaxFlow} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathsf{O}(m)]$

Proof. For the nondeterministic algorithm, nondeterministically guess the flow on each edge. We can verify in linear time that the value of the flow is at least k, that no edge flow exceeds the edge capacity, and that for all nodes the inflow is equal to the outflow.

For the co-nondeterministic algorithm, nondeterministically guess a cut (S, T) such that $s \in S$ and $t \in T$ with value l where $l < k$. By the max-flow/min-cut theorem there is no flow with value strictly greater than l. The value of a cut can be computed in $\mathsf{O}(m)$ time. □

This completes the part of Theorem 6.5.2 concerning maximum flow. In contrast, the SingleSourceMaxFlow problem requires quadratic time under SETH [6]. In the single-source maximum flow problem we are given a source s and need to output the maximum flow from s to all other nodes. As a consequence, there is no deterministic fine-grained reduction from single-source maximum flow to maximum flow under NSETH. Similarly, the AllPairsMaxFlow problem is SETH-hard at time mn [100].

6.5.2 Hitting Set

Problem 47 (HittingSet)**.** *Given two families of non-empty sets $\mathscr{S}$ and $\mathscr{T}$ defined on universe U, a set $S \in \mathscr{S}$ is a* hitting set *if it has nonempty intersections with all members in $\mathscr{T}$. The* HittingSet *problem accepts input $(\mathscr{S}, \mathscr{T}, U)$ iff*

$$\exists S \in \mathscr{S}\ \forall T \in \mathscr{T}\ \exists u \in U\ \ ((u \in S) \wedge (u \in T))$$

Let the size of input be $m = \sum_{S \in \mathscr{S}} |S| + \sum_{T \in \mathscr{T}} |T|$. We assume for any $u \in U$, we can in constant time decide if $u \in S$ or $u \in T$. It is conjectured, that this problem does not admit a subquadratic time algorithm [8]. We show that HittingSet and its negation are both solvable in nondeterministic linear time.

Lemma 6.5.2. $\mathsf{HittingSet} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathsf{O}(m)]$

HittingSet can be solved nondeterministically in linear time, by guessing an S, enumerating all $T \in \mathscr{T}$, and guessing a $u \in T$.

The negation of the HittingSet problem ¬HittingSet, which is defined as

$$\forall S \in \mathscr{S}\ \exists T \in \mathscr{T}\ \forall u \in U\ \ ((u \notin S) \vee (u \notin T))$$

can be solved by the following algorithm.

Algorithm 7: ¬HittingSet

```
for each $S \in \mathscr{S}$ do
    Nondeterministically select $T$ from $\mathscr{T}$
    for each $u \in S$ do
        if $u \in T$ then
            Reject
Accept.
```

The algorithm runs in time $\mathsf{O}(\sum_{S \in \mathscr{S}} |S|) = \mathsf{O}(m)$.

In Section 6.6 we generalize this algorithm for model checking of arbitrary k-quantifier sentences with at least one existential quantifier and ending with a universal quantifier.

6.5.3 Min-Cost Maximum Flow

The min-cost maximum flow problem (MinCostMaxFlow) is an important generalization of the MaxFlow problem that also generalizes problems such as shortest path and bipartite minimum cost perfect matching.

In the MinCostMaxFlow problem on a graph $G = (V,E)$ we consider flow networks where the edges e have additional costs $\psi(e)$. The cost of a flow is defined as

$$\sum_{e \in E} \psi(e)\mathrm{flow}(e)$$

Problem 48 (MinCostMaxFlow)**.** *Let $G = (V,E)$ be a connected directed graph with capacity constraints and edge costs, let $s,t \in V$ be vertices and $k,c \in \mathbb{R}$.*

The MinCostMaxFlow *problem is to decide if there either exists a flow from s to t of value strictly more than k, or if there is a flow from s to t of value exactly k and cost at most c.*

Orlin [115] gives a $\mathrm{O}(m^2)$ algorithm for MinCostMaxFlow. In this section consider the question if it is possible to show SETH-hardness of this problem and show that there is a $\mathrm{O}(m)$ nondeterministic and co-nondeterministic algorithm. Therefore, assuming NSETH, this problem is not SETH-hard under deterministic reductions for any superlinear time.

It is easy to see that this problem is in $\mathsf{NTIME}[\mathrm{O}(m)]$ where m is the number of edges. Simply either guess a maximum flow with minimum cost and verify that it is indeed a flow with the correct value and cost. We therefore concentrate on the co-nondeterministic time complexity.

Lemma 6.5.3. *The* MinCostMaxFlow *problem is in* $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathrm{O}(m)]$.

Proof. Klein [95] showed that a for any flow f, there is a flow of the same value as f but smaller cost if and only if there is a negative cost cycle in the residual graph.

Furthermore, as observed in the analysis of the Bellmann-Ford algorithm [23, 59], there is a nondeterministic algorithm for the nonexistence of a negative weight cycle in a graph. A potential for a weighted graph $G = (V, E, w)$ is a map $p : V \to \mathbb{R}$ such that for all edges $(u, v) \in E$ we have $p(v) \leq p(u) + w(u, v)$. Bellman and Ford show that there is a negative weight cycle in G if and only if there is no potential for G.

The co-nondeterministic algorithm for min-cost maximum flow has two cases. If there is no flow of value k, then we nondeterministically guess a cut of value less than k. Otherwise, nondeterministically guess a flow of value k with minimum cost. We then certify that the flow is a maximum flow by guessing a cut of value k. Furthermore we guess a potential for the residual graph. The cut certifies that there is no flow of value greater than k, and the potential certifies that there is no maximum flow of smaller cost.

Verifying all nondeterministic guesses can be done in time $\mathrm{O}(m)$. □

Since the MaxFlow problem is a special case of the MinCostMaxFlow problem, Lemma 6.5.1 also follows as a corollary of 6.5.3.

6.5.4 Maximum Matching

The MaximumMatching problem in general graphs is one of the most fundamental problems in computer science.

Problem 49 (MaximumMatching)**.** *Given a graph $G = (V, E)$ and a number k, decide if there is a set of edges $M \subseteq E$ with $|M| \geq k$ such that each vertex $v \in V$ is adjacent to at most one edge in M.*

The MaximumMatching problem is in time $\mathrm{O}(m\sqrt{n})$ [108], matching the time complexity of the bipartite case [78].

In this section we show that there is a linear time co-nondeterministic algorithm, and that there is therefore no fine-grained reduction from CNF-sat to MaximumMatching for any superlinear time, assuming NSETH.

We give an $\mathrm{O}(m)$ co-nondeterministic algorithm for this problem. The $\mathrm{O}(m)$ nondeterministic algorithm is trivial.

Lemma 6.5.4. *The* MaximumMatching *problem is in* $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathrm{O}(m)]$.

Proof. Edmonds' Theorem [55] relates maximum matchings of a graph $G = (V, E)$ with odd set covers. An odd set cover is a map $f : V \to \mathbb{N}$, such that each edge is either adjacent to a vertex v with $f(v) = 1$, or is adjacent to two vertices u, v such that $f(u) = f(v) \geq 2$. Furthermore, for $n_i = |\{v \mid v \in V, f(v) = i\}|$ we have n_i is odd for all $i \geq 2$.

For an odd set cover O, let $\mathrm{val}(O) = n_1 + \sum_{i \geq 2} \lfloor \frac{n_i}{2} \rfloor$ be the value of the set cover. Edmonds' Theorem says that for any matching M and any odd set cover O, we have $|M| \leq \mathrm{val}(O)$. Furthermore, for any maximum matching M there is an odd set cover O such that $|M| = \mathrm{val}(O)$. Therefore a matching M is maximum if and only if there is an odd set cover O such that $|M| = \mathrm{val}(O)$.

The co-nondeterministic algorithm then guesses a maximum matching M and an odd set cover O such that $|M| = \mathrm{val}(O)$.

Verifying that M is a matching and O an odd set cover, as well as computing the value of the set cover can easily be done in time $\mathrm{O}(m)$. $\square$

6.5.5 3-SUM

In this section, we consider the 3-sum problem. Recall the definition.

Problem 50 (3-sum)**.** *Given n integers $a_1 \dots a_n$ in the range $[-W, W]$ for some $W = \mathsf{poly}(n)$, the* 3-sum *problem is the problem of determining if there is $1 \le i, j, k \le n$ such that $a_i + a_j + a_k = 0$.*

The conjecture that the 3-sum problem admits no $O(n^{2-\varepsilon})$ algorithm for any $\varepsilon > 0$ has proven immensely useful to show the conditional hardness of a large number of problems (see Section 2.8 for an overview), most of which are not known to be hard under SETH. A fine-grained reduction from SAT to 3-sum would therefore have a large impact, proving the 3-sum conjecture under SETH.

We give a subquadratic algorithm for 3-sum in both nondeterministic and co-nondeterministic time, which rules out a deterministic fine-grained reduction from SAT to 3-sum under NSETH.

Lemma 6.5.5. 3-sum $\in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{1.5})]$

Proof. There is a trivial constant time nondeterministic algorithm of guessing the triplet of indices. The more interesting part is to show that there is an efficient nondeterministic algorithm to show that there is no such triplet.

We nondeterministically guess a proof of the form (p, t, S), such that

- p is a prime number, such that $p \le \text{prime}_{n^{1.5}}$, where prime_i is ith prime number.
- t is a nonnegative integer with $t \le 3cn^{1.5} \log n$ such that

$$t = |\{(i, j, k) \mid a_i + a_j + a_k = 0 \mod p\}| \tag{6.3}$$

  is the number of three-sums modulo p.
- $S = \{(i_1, j_1, k_1), \dots, (i_t, j_t, k_t)\}$ is a set of t triples of indices, such that for all $r :\ 0 < r \le t$ we have $a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p$ and $a_{i_r} + a_{j_r} + a_{k_r} \ne 0$

We first show that such a proof exists. Let us assume that there is no triple of elements that sum up to zero. Let R be the set of all pairs $((i,j,k),p)$, such that p is a prime $\leq \text{prime}_{n^{1.5}}$ and $a_i + a_j + a_k = 0 \mod p$. Then $|R| \leq n^3 \log(3n^c) < 3cn^3 \log n$, as any integer z can have at most $\log(z)$ prime factors. Then, by a simple counting argument, there indeed exists a prime $p_0 \leq \text{prime}_{n^{1.5}}$, such that the number of pairs of the form $((i,j,k),p_0)$ in R is at most $\frac{3cn^3 \log n}{n^{1.5}} = 3cn^{1.5} \log n$.

To verify a proof of that form we first need to check that for all $r \leq t$:

$$a_{i_r} + a_{j_r} + a_{k_r} = 0 \mod p \tag{6.4}$$

$$a_{i_r} + a_{j_r} + a_{k_r} \neq 0 \tag{6.5}$$

Then we compute the number of 3-sums modulo p and compare it with t. In order to do this we expand the following expression using Fast Fourier Transform in time $\tilde{\mathrm{O}}(t)$:

$$\left(\sum_i x^{(a_i \mod p)} \right)^3 \tag{6.6}$$

Let b_j be a coefficient before x^j. The number of triplets that sum to $0 \mod p$ is given by $b_0 + b_p + b_{2p}$. Hence we need to check that

$$b_0 + b_p + b_{2p} = t \tag{6.7}$$

If it is true, then the proof is accepted, otherwise it is rejected.

The time complexity of verification is $\tilde{\mathrm{O}}(n^{1.5})$ for reading and checking the properties of all the triples and $\tilde{\mathrm{O}}(t) = \tilde{\mathrm{O}}(n^{1.5})$ for counting the number of triples that sum to 0 modulo p. Therefore the total time complexity is $\tilde{\mathrm{O}}(n^{1.5})$. $\square$

While we show non-reducibility, a weaker form of SETH-hardness for 3-sum does in fact exists. Williams and Pătraşcu [123] show that there is no algorithm for k-sum with time $n^{o(k)}$.

6.5.6 All-pairs shortest paths and related problems

Recall the definition of the all-pairs shortest path problem (APSP).

Problem 51 (All-Pairs Shortest Path (APSP))**.** *Given an undirected, weighted graph $G = (V, E)$ with weights $w : E \to [-W, W]$ for some $W = \mathsf{poly}(|V|)$, compute for every pair $v_1, v_2 \in V$, compute the length of the shortest path from v_1 to v_2.*

Like the 3-sum conjecture and SETH, the conjecture that APSP does not admit an $\mathrm{O}(n^{3-\varepsilon})$ time algorithm for any $\varepsilon > 0$ has been used successfully to show the conditional hardness of a number of problems, e.g. [150, 139].

We use a similar technique as in the algorithm for 3-sum to show that the Zero Weight Triangle problem (ZeroWeightTriangle), which is hard under APSP, admits an efficient algorithm in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}$.

Problem 52 (ZeroWeightTriangle)**.** *Given a tripartite graph $G(V_1, V_2, V_3, E)$ with $|V_1| = |V_2| = |V_3| = n$ and edge weights in $[-n^a, n^a]$ for some constant a, the* ZeroWeightTriangle *problem is the problem of determining if there is a triangle such that the sum of the edge weights is* 0*.*

We first show that if the range is small enough, then we can count the number of zero weight triangles efficiently.

Lemma 6.5.6. *For a prime p, there is a deterministic algorithm for counting the number of zero weight triangles $\mod p$ in time $\mathrm{O}(n^{\omega} p)$*

Proof. For $i \in \mathrm{GF}(p)$, let $q(i)$ be the polynomial x^i. Let A be the weight matrix of the input graph G ($\mod p$). We define matrix B as $B[i, j] = q(A[i, j])$. For a polynomial

r and integer i, let $b_i(r)$ be the coefficient of x^i in r. Every triangle with weight zero mod p has weight either 0, p and $2p$. We have that $b_j(B^3[i,i])$ is the number of triangles of weight j that involve vertex i. Therefore

$$\sum_{i=1}^{n} \sum_{j\in\{0,p,2p\}} b_j(B^3[i,i]) = 3t \tag{6.8}$$

where t is the number of zero weight triangles modulo p.

The time to compute B^3 is $\mathrm{O}(n^{\omega} p \log p)$ if we multiply the polynomials using Fast Fourier Transform. □

In particular, we will be using Lemma 6.5.6 to verify that our nondeterministic guess of the number of false positives is correct.

Lemma 6.5.7. *The Zero Weight Triangle Problem is in* $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{\mathrm{O}}(n^{\frac{\omega+3}{2}})]$.

Proof. As for 3-sum, the nondeterministic algorithm is trivial and we concentrate on the co-nondeterministic algorithm.

Let $\mu = \frac{3-\omega}{2}$. Further let c be a large constant such that there are at least n^{μ} primes in the range $R = [n^{\mu}, cn^{\mu}\log n]$. We assume that there is no zero weight triangle and consider any fixed triangle. Since all edge weights are in the range $[-n^a, n^a]$, the total weight of the triangle is in the range $[-3n^a, 3n^a]$ and the number of primes $p \in R$ such that the triangle has weight $0 \mod p$ is at most $\log(3n^a)/\log(n^{\mu}) < \frac{2}{\mu}a$. Since R contains at least n^{μ} primes, there is a prime $p \in R$ such that the number of triangles with weight $0 \mod p$ is at most $\frac{2}{\mu}an^{\frac{3+\omega}{2}}$.

The nondeterministic algorithm now proceeds as follows: Nondeterministically pick p as above. By Lemma 6.5.6 we can deterministically count the number t of triangles with weight $0 \mod p$ in time $\mathrm{O}(n^{\omega} p \log p) = \tilde{\mathrm{O}}(n^{\frac{3+\omega}{2}})$. Nondeterministically pick t distinct triangles and check that each of them has weight $w \neq 0$ with $w = 0 \mod p$.

The total time is bounded by $\tilde{O}(n^{\frac{3+\omega}{2}})$ as claimed. □

Corollary 6.5.2. $\mathsf{APSP} \in (\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{\frac{3+\omega}{2}})]$.

Proof. A deterministic fine-grained reduction from the problem of finding a negative weight triangle to ZeroWeightTriangle can be found in [139], such that the negative weight triangle problem is also in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\tilde{O}(n^{\frac{3+\omega}{2}})]$.

Finally, [150] give a deterministic fine-grained reduction from APSP to the negative weight triangle problem with time $\tilde{O}(n^2 T(n^{1/3}))$, where $T(n)$ is the time complexity of the negative weight triangle problem.

Instead of applying this reduction directly, which would still give a subcubic nondeterministic upper bound for APSP, we instead modify their reduction to a nondeterministic reduction that preserves the savings in the exponent. The reduction from [150] loses savings in the exponent when reducing from min-plus product to negative weight triangle. The fine-grained reduction from APSP to min-plus product is folklore and does not change the exponent.

For two matrices A and B the min-plus product C is the matrix such that $C[i,j] = \min_k\{A[i,k]+B[k,j]\}$. Given an instance of min-plus product, nondeterministically guess C as well as a matrix K such that $K[i,j] = \text{argmin}_k\{A[i,k]+B[k,j]\}$. We can easily check that $C[i,j] = A[i,K[i,j]]+B[K[i,j],j]$ for all i and j, which proves that none of the entries in C are too large.

To verify none of the entries in C are too small, we construct a complete $n \times n \times n$ tripartite graph $G = (V_1,V_2,V_3,E)$ such that matrix $-A$ is the weight matrix for the edges between V_1 and V_2, $-B$ corresponds to the weights between V_2 and V_3, and C corresponds to the weights between V_1 and V_3. There are i, j, k such that $C[i,j] < A[i,k]+B[k,j]$ if and only if there is a negative weight triangle in this graph.

We use this reduction along with the co-nondeterministic algorithm for the

ZeroWeightTriangle problem above to get a nondeterministic algorithm for APSP with the claimed time complexity. □

Note that [150] in fact give a sizable list of problems that are equivalent to APSP under subcubic deterministic fine-grained reductions (including negative weight triangle, but not zero weight triangle). Our non-reducibility result therefore applies to all of these problems.

6.6 Characterizing the quantifier structure of SETH-hard graph problems

There are many problems within P that are known to be SETH-hard, but few of them are graph problems. And of the ones that are, they tend to have similar logical forms. For instance, k-DominatingSet [123] is definable by a $\forall^k\exists$ quantified formula; GraphDiameter-2 and BipartiteGraphDominatedVertex [29] are definable by $\forall\forall\exists$ quantified formulas. Here we study the relations between SETH-hardness and the logical structures of model checking problems. A result by Ryan Williams [145] explored the first-order graph properties on dense graphs, while here we look into sparse graphs whose input is a list of edges. For a more expansive treatment of the fine-grained complexity of sparse graph problems based on their quantifier structure, see [67].

We define graph property quite broadly. The input to a graph property is a many-sorted universe that we view as sets of vertices, together with a number of unary relations (node colors), and binary relations, viewed as different categories or colors of edges. The binary relations are not symmetric in general. We specify the problem to be solved by a first order sentence. Let φ be a first order sentence in prenex normal form, with k quantifiers:

$$\varphi = Q_1 x_1 \in X_1, Q_2 x_2 \in X_2, \dots Q_k x_k \in X_k \psi \tag{6.9}$$

or shortened as

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_k x_k \psi \tag{6.10}$$

where φ is a quantifier-free formula whose atoms are unary or binary predicates on $x_1, \dots, x_k$.

An instance of the FirstOrderModelChecking problem of a formula φ with $k \geq 3$ quantifiers specifies sets $X_1, \dots X_k$, where variable x_i is an element of set X_i, as well as all the unary and binary relations that occur in φ. We assume without loss of generality that the sets X_i are disjoint and that the domain of any predicate is restricted to one pair (X_i, X_j). We can always duplicate elements and adjust the corresponding relations accordingly. We also assume equality is one of the relations, so we can tell when $x_i = x_j$. To reformulate the problem as a graph problem, we view he sets $X_1, \dots, X_k$ as the sets of nodes in a k-partite graph, and the binary predicates as (colored) edges, i.e. for some predicate P, if $P(x_i, x_j)$ is true then there is an edge between the nodes x_i and x_j. We refer to the k-partite graph with edges defined by predicate P as G_P, and the colored union of graphs defined on all predicates as G.

We assume that the input is given as follows: For each unary relation, we are given a Boolean vector indexed by the vertices saying whether the relation holds, and for each binary predicate, the list representation of the corresponding directed graph. We want to decide if φ is true for the input model.

Examples of this problem include k-Clique, which is defined by

$$\varphi = \exists x_1 \dots \exists x_k \bigwedge_{i,j \in \{1,\dots,k\}, i \neq j} E(x_i, x_j) \tag{6.11}$$

k-DominatingSet, defined by

$$\varphi = \exists x_1 \dots \exists x_k \forall x_{k+1} \left(E(x_1, x_{k+1}) \vee \dots \vee E(x_k, x_{k+1}) \right) \tag{6.12}$$

and GraphRadius2, defined by

$$\varphi = \exists x_1 \forall x_2 \exists x_3 \, (E(x_1,x_3) \wedge E(x_3,x_2)) \tag{6.13}$$

We let $n = \max_i |X_i|$ be the maximum size of the node parts, and m be the number of edges in the union of the graphs. The size is $n+m$, but for convenience, we will assume $m > n$ and use m as the size.

The maximum deterministic complexity of a k-quantifier formula for $k \geq 2$ is $\mathsf{O}(m^{k-1})$. For $k = 2$, this is just linear in the input size, so matching lower bounds follow. So the interesting case is $k \geq 3$. If SETH is true, some formulas require approximately this time. But if NSETH holds, all such formulas that are SETH hard are of the same logical form. This is made precise as follows:

Theorem 6.6.1. *Let $k \geq 3$. If* NSETH *is true, then there is a k-quantifier formula whose model checking problem is* $\mathsf{O}(m^{k-1})$ SETH*-hard, and all such formulas have the form* $\forall^{k-1}\exists$ *or* $\exists^{k-1}\forall$.

Theorem 6.6.1 comes directly from the following lemmas:

Lemma 6.6.1. *There are graph properties with* $\forall^{k-1}\exists$ *and* $\exists^{k-1}\forall$ *structure that are* SETH*-hard for time* $\mathsf{O}(m^{k-1})$.

Proof. The $(\mathsf{k}-1)$-OrthogonalVectors problem is equivalent to the graph problem

$$\exists x_1 \ldots \exists x_{k-1} \forall x_k \, (P(x_1,x_k) \wedge \cdots \wedge P(x_{k-1},x_k)) \tag{6.14}$$

Section 2.6 gives a proof of the SETH-hardness of (dense) k-OrthogonalVectors for $k \geq 2$. Note that the sparse formulation where the parameter is the number of edges m is more general than the dense version and SETH-hardness therefore translates. The negation of k-OrthogonalVectors is also SETH-hard and has a $\forall^k\exists$ quantifier structure. □

On the other hand if a problem is of any form other than $\forall^{k-1}\exists$ or its negation, we will show it has both smaller nondeterministic and co-nondeterministic complexity. We will assume without loss of generality that the outermost quantifiers is universal. We can handle the other case by simply negating the problem. A problem that does not have an $\forall^{k-1}\exists$ structure either has exactly one existential quantifier, but not in the innermost position, no existential quantifiers, or at least two existential quantifiers.

Lemma 6.6.2. *If φ has more than one existential quantifier, then it can be solved in nondeterministic time $\mathsf{O}(m^{k-2})$. If φ has more than one universal quantifier, then it can be solved in co-nondeterministic time $\mathsf{O}(m^{k-2})$*

Proof. We concentrate on the case with more than one existential quantifier. The other case is symmetric.

These problems can be solved by guessing the existentially quantified variables, and exhaustive search on universally quantified variables. Because there are at most $k-2$ universal quantifiers, the algorithm runs in nondeterministic time $\mathsf{O}(m^{k-2})$. Note that we operate on the sparse representation of the predicates. We assume that we are given the list of pairs that satisfy any predicate in a sorted order that allows us to do this exhaustive search without overhead. $\square$

Symmetrically, if φ has more than one universal quantifier, then it can be solved in co-nondeterministic time $\mathsf{O}(m^{k-2})$. In particular, since $k \geq 3$, the model checking problem is always easy either nondeterministically or co-nondeterministically.

Lemma 6.6.3. *If φ has exactly one existential quantifier, but it is not on the innermost position, then it can be solved in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathsf{O}(m^{k-2})]$. Symmetrically, if φ has exactly one universal quantifier, but it is not on the innermost position, then it can be solved in $(\mathsf{N} \cap \mathsf{coN})\mathsf{TIME}[\mathsf{O}(m^{k-2})]$.*

Proof. We only show the case with one existentially quantified variable. Since $k \geq 3$, the problem is in $\mathsf{coNTIME}[\mathrm{O}(m^{k-2})]$ by Lemma 6.6.2.

Let the existentially quantified variable be x_j.

For a pair of nodes (x_u, x_v), we define its color $\chi(x_u, x_v)$ to be binary strings corresponding to the truth values of all predicates on them. By preprocessing we can set up a table that allows us to check whether $\chi(x_u, x_v) = c$ for given x_u, x_v and color c in constant time. We also define $\chi(x_k | x_1 \ldots x_{k-1})$ to be the concatenation of $\chi(x_1, x_k), \ldots, \chi(x_{k-1}, x_k)$. For colors composed of only 0 (where all the related predicates are false), we call them "background".

Our algorithm can count the number of x_k's with color $\chi(x_k | x_1 \ldots x_{k-1}) = c$ for all $(x_1, \ldots, x_{k-1})$ and c in time $\mathrm{O}(m^{k-2})$. The main idea of the algorithm is to nondeterministically guess x_j and count valid values of x_k, so that it saves the exhaustive search on x_j and x_k.

1. For each combination of $(x_1, \ldots, x_{j-1})$ nodes, we nondeterministically bundle a fixed x_j value to it. This takes time $\mathrm{O}(m^{j-1})$, which is at most $\mathrm{O}(m^{k-2})$ because $j < k$. In the rest of this algorithm, given any $(x_1, \ldots, x_{j-1})$ values we can find their corresponding x_j value in constant time.

2. We do a $(k-2)$-layer nested loop. On each layer we loop through all (x_i, x_k) edges where x_i is a variable other than x_j or x_k. Then inside all the loops, for each x_k in the $(k-2)$ current (x_i, x_k) edges, we record the color $\chi(x_k | x_1 \ldots x_{k-1})$, where the values of x_j come from the current $(x_1, \ldots, x_{j-1})$.

   Then after the loops we can count the number of x_k's of for each $(x_1, \ldots, x_{k-1})$ and each color.

   This step can be done in time $\mathrm{O}(m^{k-2})$.

3. The previous step did not count the x_k's that only appear in (x_j, x_k) edges, i.e. whose $\chi(x_k|x_1 \dots x_{k-1})$ is all-zero on all non-(x_j, x_k) predicate positions, but not all-zero on some (x_k, x_j) predicate positions. We will count these x_k's in this step.

   For each x_j, we can enumerate all the (x_j, x_k) edges to count the number of x_k's where $\chi(x_j, x_k) = c_{jk}$ for any not all-zero c_{jk}. Also, from the previous step, we can count the number of x_k's s where $\chi(x_k|x_1 \dots x_{k-1})$ is not all-zero on some non-(x_j, x_k) predicate positions, and also equals c_{ij} on its (x_j, x_k) predicate positions. By subtraction we can get the number of x_k's, where $\chi(x_k|x_1 \dots x_{k-1})$ is all-zero on non-(x_j, x_k) predicate positions, and equals c_{ij} on its (x_j, x_k) predicate positions. Similarly as the previous step, this process runs in time $O(m^{k-2})$.

4. Now we have counted the x_k's with all non-background colors for all $(x_1, \dots, x_{k-1})$. The number of x_k's where $\chi(x_k|x_1 \dots x_{k-1})$ is background can be computed by $|X_k|$ subtracting the numbers of all non-background x_k's.

5. Finally, for all $(x_1, \dots, x_{k-1})$, we sum the number of x_k's of all colors that satisfy φ. If it always equals $|X_k|$, then the algorithm accepts.

□

The last case, where either all quantifiers are existential or all quantifiers are universal is easy deterministically and therefore also not a candidate for SETH-hardness, independent of NSETH.

Lemma 6.6.4. *If all quantifiers are existential, or all quantifiers are universal, then the problem can be solved in deterministic time* $O(m^{k-1.5})$.

Proof. This case is computationally easy even deterministically. We concentrate on the case where all quantifiers are existential. For simplicity, we will restrict all predicates to be binary. The algorithms can easily be modified to work for unary predicates.

Assume there is at most one predicate on each pair of variables. Otherwise for a pair of variable we can take the disjunction of all the predicates to be the value of the only predicate. Let the only predicate be E.

First, we write quantifier-free part ψ in DNF with t terms, i.e. $\psi = \bigvee_{i=1}^{t} \psi_i$, and then split its terms, so that $\varphi = \bigvee_{i=1}^{t}(\exists x_1 \dots \exists x_k \psi_i)$. Thus the problem becomes a constant number of model checking sub-problems with form $\exists x_1 \dots \exists x_k \psi_i$ where ψ_i is a conjunction of either positive or negative predicates.

Let the number of predicates be p. We assume some canonical order on the predicates, so that all truth values for a tuple of vertices $(x_1, \dots, x_k)$ correspond to strings in $\{0,1\}^p$. We call the strings *colors*, denoted by $\chi(x_1, \dots, x_k)$. Specifically, the color 0^p (where all predicates are false) is called the *background color*.

For each ψ_i, we first find all colors satisfying it, and then for each satisfying color c, we use the following algorithm to count the number of $(x_1, \dots, x_k)$ tuples such that $\chi(x_1, \dots, x_k) = c$.

Case 1: The color c contains at least two positive predicates.

Case 1-1: There exists four distinct variables v, w, x, y such that $E_1(v, w)$ and $E_2(x, y)$ are true in c. Then we exhaustively search all $k-4$ variables other than v, w, x, y in time $O(n^{k-4}) = O(m^{k-4})$, and enumerate all $E_1(v, w)$ and $E_2(x, y)$ edges, which takes time $O(m^2)$. The overall running time is $O(m^{k-2})$. For each k-tuple enumerated, if $\chi(x_1, \dots, x_k) = c$, we increment the counter for color c.

Case 1-2: There are three distinct variables x, y, z such that $E_1(x, y)$ and $E_2(y, z)$ are true in c. We take time $O(m^{k-3})$ to exhaustively search all $k-3$ other variables. Then we consider the x nodes of large and small degree separately.

- For x's whose degree is at most $\sqrt{m}$, we enumerate all $E_1(x,y)$ edges. Then for each y, we enumerate all (y,z) edges. The running time is $O(\sum_x \deg(x)) \leq O(m\sqrt{m}) = O(m^{1.5})$.
- For x's whose degree is greater than $\sqrt{m}$, we enumerate all such x and all (y,z) edges. Because there are at most $O(m/\sqrt{m}) = \sqrt{m}$ such x's, the running time is $O(\sqrt{m}m) = O(m^{1.5})$.

For each k-tuple enumerated, check if $\chi(x_1,\ldots,x_k) = c$. The overall running time is $O(m^{k-3}m^{1.5}) = O(m^{k-1.5})$.

Case 2: The color c contains only one positive predicate. Let the positive predicate be $E(x,y)$.

We can count the number of tuples with $\chi(x_1,\ldots,x_k) = c$ indirectly by subtracting the number of tuples also satisfying another predicate from the total number of predicates satisfying $E(x,y)$. As Case 1 shows, for colors where at least two predicates are true, we can count the number of tuples with that color. Also, we can easily compute the number of k-tuples satisfying $E(x,y)$ regardless of other predicates, by simply multiplying the number of $E(x,y)$ edges with the sizes of all other sets. Because there are only a constant number of colors, the running time is $O(m^{k-1.5})$.

Case 3: The color c is the background color. Similar to Case 2, we compute the number of $(x_1,\ldots,x_k)$ of all other colors, and subtract it from the total number of tuples. The running time is $O(m^{k-1.5})$.

□

Thus, only $\forall^{k-1}\exists$ formulas require $O(m^{k-1})$ nondeterministic time, and by looking at the complements, only $\exists^{k-1}\forall$ formulas require $O(m^{k-1})$ co-nondeterministic time. Thus, assuming NSETH, only these two types of first-order properties might be SETH-hard for the maximum difficulty of a k-quantifier formula.

6.7 Consequences for verification of solutions

Besides implying that some problems are not SETH-hard, NSETH also implies some new lower bounds on problems in P. Namely, if NSETH is true, then any SETH-hard problem with a deterministic reduction such as Fréchet distance, edit distance, and LongestCommonSubsequence also require quadratic co-nondeterministic time, i.e. the problem of showing that there is no solution better than a given value is as hard as solving the problem deterministically. This immediately implies that, even given a solution, testing optimality requires quadratic time. We can formalize this as follows:

Theorem 6.7.1. *Let $F(x,y)$ be some function that can be computed in time $T_F(|x|+|y|) \geq |x|+|y|$. We assume $|y| = l(|x|)$ for some l and define $n = |x|+l(|x|)$. Let Opt_F denote the optimization problem for F: Given x, find y such that $F(x,y)$ is maximized. The verification problem* Ver_F *is: Given x and y, is y an optimal solution for Opt_F, i.e. is there no y' with $F(x,y') > F(x,y)$? Assume that Opt_F is SETH-hard at some time bound $T(n)$ which is greater than $T_F^{1+\gamma}(n)$ for some $\gamma > 0$. Then if NSETH is true,* Ver_F *cannot be solved in any time T' so that $T'(n) < T^{1-\varepsilon}(n)$ for any $\varepsilon > 0$.*

Proof. If we assume for the sake of contradiction that Ver_F can be solved in such a time T', then we can solve Opt_F by nondeterministically guessing the witness y and verifying that it is optimal. This takes time $T^{1-\varepsilon}(n)$, hence using the fine-grained reduction from CNF-sat to Opt_F we get both a nondeterministic and co-nondeterministic algorithm for CNF-sat that contradicts SETH. □

So NSETH gives us a way to argue that not only finding but also verifying optimal solutions is computationally intensive.

Verification problems have the property that they are nondeterministically easy. Similarly, many natural problems the property that they are easy either nondeterministically or co-nondeterministically. A natural problem that is an exception is the StablePair problem discussed in Section 4.5.4.

6.8 Conclusions and open problems

A theme running through computational complexity is that looking at general relationships between models of computing and complexity classes can frequently shed light on the difficulty of specific problems. In this chapter, we introduce this general technique to the study of fine-grained complexity by comparing nondeterministic complexities of problems. This raises the more general question of what other notions and models of complexity might be useful in distinguishing the fine-grained complexity of problems. For example, we show that neither 3-sum nor APSP can be SETH-hard if NSETH holds. This still leaves open the possibility that the two conjectures are equivalent to each other (if not to SETH). One might be able to prove such an equivalence, or give evidence against it by showing a different notion of complexity that distinguishes the two and is preserved by fine-grained reductions.

Chapter 6 is based on material as it appears in the following publications: 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, pp. 261-270. ACM, 2016. [36] The author of this dissertation was a principal author of this publication. Material from Chapter 6 is currently in preparation for submission for publication, by Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. The author of this dissertation was a principal author of this publication. We would like to thank Amir Abboud, Karl Bringmann, Bart Jansen, Sebastian Krinninger, Virginia Vassilevska Williams, Ryan Williams and the anonymous reviewers for many helpful comments on the material in this Chapter 6.

Appendix A

Dramatis Personae

We take a problem-centric approach. In the resource-centric approach to complexity theory it is sometimes difficult to keep the overview over all the complexity classes, giving rise to websites such as the complexity zoo [1]. Similarly, in our problem-centric approach the space of problems is large. This chapter aims to collect all the definitions and key results of problems mentioned anywhere in this dissertation. Note that in the PDF version of this document, all problem names are links to the respective definition in this chapter.

A.1 Satisfiability Problems

Problem 53 ($\mathscr{C}$-sat)**.** *The satisfiability problem for Boolean circuits on n variables in the circuit class* $\mathscr{C} \subseteq \mathsf{P/Poly}$

- **Upper Bounds:** Varies on the circuit class $\mathscr{C}$. In general $\tilde{\mathrm{O}}(2^n)$
- **Lower Bounds:** SETH-hard at time 2^n for any $\mathscr{C}$ that contains all k-CNF formulas

Problem 54 (k-sat)**.** $\mathscr{C}$-sat *where* $\mathscr{C}$ *is the set of all* k-CNF *formulas, Boolean formulas in conjunctive normal form on n variables and with clause width at most k.*

- **Upper Bounds:** $2^{(1-1/\mathrm{O}(k))n}$ [117, 130]

- **Lower Bounds:** SETH-hard at time 2^n for $k = \omega(1)$ by definition of SETH

Problem 55 (Sparse-k-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all k-CNF formulas on n variables, with clause width at most k and at most $f(k)n$ clauses for some function f.*

- **Upper Bounds:** $2^{(1-1/\mathrm{O}(k))n}$ [117, 130]
- **Lower Bounds:** SETH-hard at time 2^n for $k = \omega(1)$ by by the sparsification lemma [85]

Problem 56 (CNF-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all CNF formulas, Boolean formulas in conjunctive normal form on n variables and cn clauses.*

- **Upper Bounds:** $2^{(1-1/\mathrm{O}(\log c))n}$ [131]
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes Sparse-k-sat

Problem 57 (AC^0-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all AC^0 formulas, constant-depth Boolean circuits over AND and OR on n variables. The problem is parametrized by the number of gates cn and the depth d.*

- **Upper Bounds:** $2^{(1-1/\mathrm{O}(\log c + d \log d)^{d-1})n}$ [81]
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes CNF-sat

Problem 58 (ACC^0-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all ACC^0 formulas, constant-depth Boolean circuits over AND, OR and MOD$_6$ on n variables. The problem is parametrized by the number of gates $m = 2^{n^\delta}$ and the depth d.*

- **Upper Bounds:** $\mathrm{O}(2^{n-n^{\varepsilon_{\delta,d}}})$ [147]
- **Lower Bounds:** SETH-hard at time 2^n for $m = \omega(n)$ as it generalizes AC^0-sat

Problem 59 (DeMorgan-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all deMorgan formulas, Boolean circuits over* AND *and* OR *of fanin* 2 *on n variables. The problem is parametrized by the number of leaves $m = cn$.*

- **Upper Bounds:** $2^{(1-1/\mathsf{poly}(c))n}$ [128]
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes Sparse-k-sat

Problem 60 (Succinct-k-sat)**.** *Given a circuit $C \in \mathsf{P/Poly}$ on n variables, decide if the truth table of C is an encoding of a satisfiable k-*CNF *formula.*

- **Upper Bounds:** $\tilde{\mathsf{O}}(2^{2^n})$
- **Lower Bounds:** NEXP-complete

We consider a number of circuit classes with weighted threshold gates (THR).

Problem 61 (0-1-ILP)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all* $\mathsf{AND} \circ \mathsf{THR}$ *circuits on n variables and with cn bottom-level gates.*

- **Upper Bounds:** $2^{(1-1/\mathsf{O}(c\log^2 c))n}$ [80, 38], see Chapter 3
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes CNF-sat

Problem 62 (max-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all* $\mathsf{THR} \circ \mathsf{AND}$ *circuits on n variables and with cn bottom-level gates.*

- **Upper Bounds:** $2^{(1-1/\mathsf{O}(\sqrt{c}))n}$ [42]
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes CNF-sat

Problem 63 (max-k-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all* $\mathsf{THR} \circ \mathsf{AND}$ *circuits on n variables and bottom-level fan-in k.*

- **Upper Bounds:** If $k = 2$, $\mathrm{O}(2^{\omega n/3})$, where $\omega < 3$ is the matrix multiplication exponent [142]
- **Lower Bounds:** SETH-hard at time 2^n for $k = \omega(1)$ as it generalizes k-sat

Problem 64 (DepthTwoThr-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all $\mathsf{THR} \circ \mathsf{THR}$ circuits on n variables and with m bottom-level gates.*

- **Upper Bounds:** $2^{n-1/\mathsf{poly}(m)}$ [136]
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes CNF-sat

Problem 65 (SparseDepthTwoThr-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all $\mathsf{THR} \circ \mathsf{THR}$ circuits on n variables and with cn wires.*

- **Upper Bounds:** $2^{(1-1/c^{\mathrm{O}(c)})n}$ [42], see Chapter 3
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes Sparse-k-sat

Problem 66 (SymFormula-sat)**.** *$\mathscr{C}$-sat where $\mathscr{C}$ is the set of all formulas on n variables and with cn wires where the gates are arbitrary (unweighted) symmetric gates. A gate is symmetric if the output only depends on the number of inputs that are* 1.

- **Upper Bounds:** $2^{(1-1/c^{\mathrm{O}(c^2)})n}$ [84], see Chapter 3
- **Lower Bounds:** SETH-hard at time 2^n for $c = \omega(1)$ as it generalizes Sparse-k-sat

A.2 Vector Problems

Problem 67 (OrthogonalVectors)**.** *Given vectors $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0,1\}^d$, determine if there is $i, j \in [n]$ satisfying $\langle a_i, b_j \rangle = 0$.*

- **Upper Bounds:** $n^{2-1/\mathrm{O}(\log c)}$ where $c = d/\log n$ [7]
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ [142]

Problem 68 (k-OrthogonalVectors)**.** *Given n vectors $x_1, \ldots, x_n \in \{0,1\}^d$, determine if there is a set of k indices $i_1, \ldots, i_k$, such that for each dimension $j \in [d]$ there is a l such that $x_{i_l}[j] = 0$.*

- **Upper Bounds:** $n^{k-1/\mathrm{O}(\log c)}$ where $c = d/\log n$ [7]
- **Lower Bounds:** SETH-hard at time n^k for $d = \omega(\log n)$ [142], see Section 2.6

Problem 69 (MinInnProd)**.** *Given vectors $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0,1\}^d$ and an integer $r \in \mathbb{N}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \leq r$.*

- **Upper Bounds:** $n^{2-1/\mathrm{O}(c \log^2 c)}$ [15]
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ as it directly generalizes OrthogonalVectors

Problem 70 (MaxInnProd)**.** *Given vectors $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0,1\}^d$ and an integer $r \in \mathbb{N}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \geq r$.*

- **Upper Bounds:** $n^{2-1/\mathrm{O}(c \log^2 c)}$ [15]
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ [15], see Section 2.7

All problems above on Boolean vectors have integer and real variants. We allow both positive and negative values and for the integer variants we typically assume that the absolute values are bounded by some $W = \mathrm{poly}(n)$. For the real variants, we assume a real RAM model, and for the integer variants a word RAM model with such that any integer fits into a constant number of words. In this section we only list the variants that we actually use in at some point in this dissertation.

Problem 71 (IntegerOrthogonalVectors)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in [-W,W]^d$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle = 0$.*

- **Upper Bounds:** $\tilde{O}(n^2)$
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log\log n)$ [141]

Problem 72 (IntegerMinInnProd)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in [-W,,W]^d$ and a natural number $r \in \mathbb{N}$, determine if there are i, j satisfying $\langle a_i, b_j \rangle \leq r$.*

- **Upper Bounds:** $O(n^{2-1/\lceil \frac{d}{2} \rceil})$ [106]
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ as it directly generalizes OrthogonalVectors

Since we allow for negative values, the maximization version is triavially equivalent.

Problem 73 (IntegerMaxInnProd)**.** *Given vectors $a_1, \dots, a_n, b_1, \dots, b_n \in [-W,W]^d$ and an integer $r \in \mathbb{N}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \geq r$.*

- Equivalent to IntegerMinInnProd

Problem 74 (RealMaxInnProd)**.** *Given vectors $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ and an integer $r \in \mathbb{N}$, determine if there is a pair i, j satisfying $\langle a_i, b_j \rangle \geq r$.*

- **Upper Bounds:** $O(n^{2-1/\lceil \frac{d}{2} \rceil})$ [106]
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ as it is generalizes IntegerMaxInnProd

Problem 75 (AllInnProd)**.** *Given $a_1, \dots, a_n \in [-W,W]^d$ and $b_1, \dots, b_n \in [-W,W]^d$, determine* for all $j \in [n]$, *the value* $\min_{i\in[n]} \langle a_i, b_j \rangle$.

- **Upper Bounds:** $O(n^{2-1/\lceil \frac{d}{2} \rceil})$, see Chapter 5
- **Lower Bounds:** SETH-hard at time n^2

Problem 76 (VectorDomination)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ determine if there is i, j such that $a_i \leq b_j$ component-wise.*

- **Upper Bounds:** $n^{2-1/(c\log^2 c)}$ where $c = d/\log n$ [80, 38], see Chapter 3
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ as it is generalizes OrthogonalVectors

Problem 77 (MostDominantVectors)**.** *Given $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{R}^d$ and an integer k, determine if there is a pair i, j such that $a_i \leq b_j$ in at least k components.*

- **Upper Bounds:** $2^d n^{2-1/O(c\log^2 c)}$ where $c = \frac{d}{\log n}$ by a trivial reduction to the VectorDomination problem
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$ as it is generalizes OrthogonalVectors

Problem 78 (SetContainment)**.** *Given sets $a_1, \dots, a_n, b_1, \dots, b_n \subseteq [d]$ given as vectors in $\{0,1\}^d$ determine if there are i, j such that $a_i \subseteq b_j$.*

- Equivalent to OrthogonalVectors

A.3 Least Weight Subsequence Problems

The least weight subsequence problem and its succinct instantiations are discussed in Chapter 5.

Problem 79 (LWS)**.** *We are given weights $w_{i,j} \in [-W,W] \cup \{\infty\}$ for every $0 \le i < j \le n$ and an arbitrary function $g : \mathbb{N} \to \mathbb{N}$. The* LWS *problem is to determine $F[n]$ which is defined by the following DP formulation.*

$$\begin{aligned} F[0] &= 0, \\ F[j] &= \min_{0 \le i < j} g(F[i]) + w_{i,j} \qquad \textit{for } j = 1, \ldots, n \end{aligned} \tag{A.1}$$

- **Upper Bounds:** $O(n^2)$ by definition
- **Lower Bounds:** Input size $\Omega(n^2)$ in general

Problem 80 (LowRankLWS)**.** LowRankLWS *is the* LWS *problem where the weight matrix* $\mathbf{W}$ *is of rank $d \ll n$. The input is given succinctly as two matrices A and B, which are $(n \times d)$- and $(d \times n)$-matrices respectively, and $\mathbf{W} = A \cdot B$.*

- Equivalent to IntegerMinInnProd

Problem 81 (CoinChange)**.** *Given a weight sequence $x_1, \ldots, x_n$ with $x_i \in [-W,W] \cup \{\infty\}$, that is the coin with value i has weight x_i. Find the weight of the multiset of denominations I such that $\sum_{i \in I} i = n$ and the sum of the weights $\sum_{i \in I} x_i$ is minimized.*

- Equivalent to $(\min,+)$-Convolution

Problem 82 (oiCoinChange)**.** *The output-intensive version of* CoinChange *is to determine, given an input to* CoinChange*, the weight of the optimal multiset such that the denominations sum up to j* for all $1 \leq j \leq n$.

- Equivalent to $(\min,+)$-Convolution

Problem 83 (UnboundedKnapsack)**.** *We are given a sequence of profits $p = (p_1, \ldots, p_n)$ with $p_i \in [0,W]$, that is the item of size i has profit p_i. Find the total profit of the multiset of indices I such that $\sum_{i \in I} i \leq n$ and the total profit $\sum_{i \in I} p_i$ is maximized.*

- Equivalent to $(\min,+)$-Convolution

Problem 84 (UnboundedSubsetSum)**.** *Given a subset $S \subseteq [n]$, determine whether there is a multiset of elements of S that sums up to exactly n.*

- **Upper Bounds:** $\tilde{O}(n)$

Problem 85 (ChainLWS)**.** *Fix a set X of objects and a relation $R \subseteq X \times X$. The Weighted Chain Least-Weight Subsequence Problem for R, denoted* ChainLWS(R)*, is the following problem: Given data items $x_0, \ldots, x_n \in X$, weights $y_1, \ldots, y_{n-1} \in [-W,W]$, find the weight of the increasing sequence $i_0 = 0 < i_1 < i_2 < \ldots < i_k = n$ such that for all j with $1 \leq j \leq k$ the pair $(x_{i_{j-1}}, x_{i_j})$ is in the relation R and the weight $\sum_{j=1}^{k-1} y_{i_j}$ is minimized.*

- **Upper Bounds:** $O(n^2)$
- **Lower Bounds:** P/Poly-SETH-hard in general

Problem 86 (NestedBoxes)**.** *Given n boxes in d dimensions, given as non-negative, d-dimensional vectors $(b_1, \ldots, b_n)$, find the longest chain such that each box fits into the next (without rotation). We say box that box a fits into box b if for all dimensions $1 \leq i \leq d$, $a_i \leq b_i$.*

- Equivelent to VectorDomination

Problem 87 (SubsetChain). *Given n sets from a universe U of size d, given as Boolean, d-dimensional vectors $(b_1, \ldots, b_n)$, find the longest chain such that each set is a subset of the next.*

- Equivalent to OrthogonalVectors

Problem 88 (LIS). *Given a sequence of n integers $x_1, \ldots, x_n$, compute the length of the longest subsequence that is strictly increasing.*

- **Upper Bounds:** $\tilde{O}(n)$

Problem 89 (ConcaveLWS). *Given an LWS instance in which the weights satisfy the quadrangle inequality*

$$w_{i,j} + w_{i',j'} \leq w_{i',j} + w_{i,j'} \qquad \text{for } i \leq i' \leq j \leq j',$$

solve it. The weights are not explicitly given, but each $w_{i,j}$ can be queried in constant time.

- **Upper Bounds:** $O(n)$ [140]

Problem 90 (AirplaneRefueling). *Given airport locations on a line, and a preferred distance per hop k (in miles), we define the penalty for flying k' miles as $(k-k')^2$. The goal is then to find a sequence of airports starting at the first airport and terminating at the last airport that minimizes the sum of the penalties.*

- **Upper Bounds:** $O(n)$, special case of ConcaveLWS

Problem 91 (PrettyPrinting)**.** *Given a string consisting of n space-separated words, and a line length M, split the string into lines $l_1, \dots, l_m$ at the spaces, such that the penalty $\sum_{i=1}^{m}(l_i - M)^2$ is minimized.*

- **Upper Bounds:** $\mathsf{O}(n)$, special case of ConcaveLWS

Problem 92 (1DKMeansClustering)**.** *Given n points on a line $x_1, \dots, x_n \in \mathbb{R}$ and a parameter k, find k points $y_1, \dots, y_k$ that minimize*

$$\sum_{i=1}^{n} \min_{j} (x_i - y_j)^2 \tag{A.2}$$

- **Upper Bounds:** $\mathsf{O}(n \log n + kn)$ [69]

Problem 93 (Static-LWS($\mathbf{W}$))**.** *Fix an instance of* LWS($\mathbf{W}$). *Given intervals of indices $I := \{a+1, \dots, a+N\}$ and $J := \{a+N+1, \dots, a+2N\}$ with a, N such that $I, J \subseteq [n]$, together with the values $F[a+1], \dots, F[a+N]$, the Static Least-Weight Subsequence Problem (*Static-LWS*) asks to determine*

$$F'[j] := \min_{i \in I} F[i] + w_{i,j} \qquad \qquad \textit{for all } j \in J.$$

- At least as hard as the corresponding LWS instantiation

Problem 94 (SMAWK)**.** *A matrix M is called totally monotone if for all i, j, a, b such that $i < j$ and $a < b$ we have $M_{ja} < M_{ia} \implies M_{jb} < M_{ib}$.*

Given a totally monotone $n \times n$ matrix, find for every column the minimum element.

- **Upper Bounds:** $\mathsf{O}(n)$ [12]

A.4 Graph Problems

Problem 95 (MaxFlow)**.** *Let $G = (V,E)$ be a connected, directed graph with capacity constraints, $s,t \in V$ be vertices and $k \in \mathbb{R}$.*

*The maximum flow problem (*MaxFlow*) is to decide if there exists a flow from s to t of value at least k.*

- **Upper Bounds:** $\mathsf{O}(|V||E|)$ [116]

Problem 96 (SingleSourceMaxFlow)**.** *Given a sparse, connected, directed graph $G = (V,E)$ with $n = |V|$, $|E| = \tilde{\mathsf{O}}(n)$ and with capacity contraints in $[n]$, and $s \in V$ a vertex, find for each $t \in V$ the value of the maximum flow from s to t.*

- **Lower Bounds:** SETH-hard at time n^2 [6]

Problem 97 (AllPairsMaxFlow)**.** *Given a connected, directed graph $G = (V,E)$ with $n = |V|$, $|E| = m$ and with (unbounded) capacity contraints, find for each $s,t \in V$ the value of the maximum flow from s to t.*

- **Upper Bounds:** $\mathsf{O}(m^{\omega} n)$ [44]
- **Lower Bounds:** SETH-hard at time mn [100]

Problem 98 (MinCostMaxFlow)**.** *Let $G = (V,E)$ be a connected directed graph with capacity constraints and edge costs, let $s,t \in V$ be vertices and $k,c \in \mathbb{R}$.*

The MinCostMaxFlow *problem is to decide if there either exists a flow from s to t of value strictly more than k, or if there is a flow from s to t of value exactly k and cost at most c.*

- **Upper Bounds:** $\mathsf{O}(|E|^2)$ [115]

Problem 99 (MaximumMatching). *Given a graph $G = (V,E)$ and a number k, decide if there is a set of edges $M \subseteq E$ with $|M| \geq k$ such that each vertex $v \in V$ is adjacent to at most one edge in M.*

- **Upper Bounds:** $\mathsf{O}(|E|\sqrt{|V|})$ [108]

Problem 100 (All-Pairs Shortest Path (APSP)). *Given an undirected, weighted graph $G = (V,E)$ with weights $w : E \to [-W,W]$ for some $W = \mathsf{poly}(|V|)$, compute for every pair $v_1, v_2 \in V$, compute the length of the shortest path from v_1 to v_2.*

- **Upper Bounds:** $n^3/2^{\sqrt{\Omega(\log n)}}$ [144]
- **Lower Bounds:** No algorithm with time $\mathsf{O}(n^{3-\varepsilon})$ for any $\varepsilon > 0$ according to the APSP conjecture [61]

Problem 101 (ZeroWeightTriangle). *Given a tripartite graph $G(V_1,V_2,V_3,E)$ with $|V_1| = |V_2| = |V_3| = n$ and edge weights in $[-n^a, n^a]$ for some constant a, the* ZeroWeightTriangle *problem is the problem of determining if there is a triangle such that the sum of the edge weights is* 0.

- **Lower Bounds:** 3-sum-hard (under randomized reductions) and APSP-hard [139, 150] at time n^3

Problem 102 (NegativeWeightTriangle). *Given a tripartite graph $G(V_1,V_2,V_3,E)$ with $|V_1| = |V_2| = |V_3| = n$ and edge weights in $[-n^a, n^a]$ for some constant a, determine if there is a triangle such that the sum of the edge weights is negative.*

- **Lower Bounds:** Subcubic equivalent to APSP [150]

Problem 103 (FirstOrderModelChecking)**.** *Given a quantified Boolean formula with $k+1$ quantifiers, a universe of size n and a list representation for all relations with total size m, determine if the formula is true.*

- **Upper Bounds:** $\mathsf{O}(m^k/2^{\Omega(\sqrt{\log n})})$ [67]
- **Lower Bounds:** SETH-hard at time m^k [142]

Problem 104 (k-DominatingSet)**.** *Given a graph $G=(V,E)$ with $|V|=n$ and $|E|=m$, determine if there is a set $S \subseteq V$ with $|S|=k$ such that each node v in V is either in S or adjacent to a node in S.*

- **Upper Bounds:** $\mathsf{O}(m^k/2^{\Omega(\sqrt{\log n})})$, special case of FirstOrderModelChecking
- **Lower Bounds:** SETH-hard at time m^k [123]

Problem 105 (GraphDiameter)**.** *Given a graph $G=(V,E)$, determine the diameter, i.e.*

$$\max_{u,v\in V} \operatorname{dist}(u,v) \tag{A.3}$$

- Subcubic equivalent to APSP [3].

Problem 106 (GraphDiameter-2)**.** *Given a graph $G=(V,E)$ with $|V|=n$ and $|E|=m$, determine if if the diameter of the graph is at most* 2.

- **Upper Bounds:** $\mathsf{O}(m^k/2^{\Omega(\sqrt{\log n})})$, special case of FirstOrderModelChecking

Problem 107 (GraphRadius)**.** *Given a graph $G=(V,E)$, determine the radius, i.e.*

$$\min_{c\in V}\max_{v\in V} \operatorname{dist}(c,v) \tag{A.4}$$

- Subcubic equivalent to APSP [3].

Problem 108 (GraphRadius2)**.** *Given a graph $G = (V,E)$ with $|V| = n$ and $|E| = m$, determine if if the radius of the graph is at most 2.*

- **Upper Bounds:** $O(m^k/2^{\Omega(\sqrt{\log n})})$, special case of FirstOrderModelChecking

Problem 109 (k-Clique)**.** *Given a graph $G = (V,E)$ with $|V| = n$, determine if there is a set $S \subseteq V$ with $|S| = k$ such that there is an edge between any two nodes in S.*

- **Upper Bounds:** $O(n^{\omega k/3})$ [113]

A.5 Stable Matching Problems

In Chapter 4, we consider a number of variants of the StableMatching problem.

Problem 110 (StableMatching)**.** *Given a matching market consisting of of a set of men M and a set of women W with $|M| = |W| = n$ with a preference list over M for each node in W and a preference list over W for each node in M, find a bipartite matching, such that there is no blocking pair, i.e. a pair of unmatched nodes (m,w), such that m prefers w over the partner in the matching, and w prefers m over the partner in the matching.*

- **Upper Bounds:** $O(n^2)$ [62]
- **Lower Bounds:** $\Omega(n^2)$ even in a communication complexity setting [68]

Problem 111 (AttributeMatching)**.** *The* StableMatching *problem where each participant p is defined by attributes $A_i(p) \in \mathbb{R}$ for $1 \le i \le d$ and weights $\alpha_i(p) \in \mathbb{R}$ for $1 \le i \le d$. The preference list for $m \in M$ is defined by the value $\mathrm{val}_m(w) = \sum_{i=1}^{d} \alpha_i(m)A_i(w)$ in decreasing order. The preference lists for women is defined analogously.*

- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$, as it is a generalization of BooleanAttributeMatching

Problem 112 (BoundedAttributeMatching)**.** *The* AttributeMatching *problem, where the attributes and weights are restricted to be from a set of of size C.*

- **Upper Bounds:** $\mathrm{O}(C^{2d} n(d+\log n))$
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$, as it is a generalization of BooleanAttributeMatching

Problem 113 (BooleanAttributeMatching)**.** *The* AttributeMatching *problem, where the attributes and weights are restricted to* 0 *or* 1*.*

- **Upper Bounds:** $\mathrm{O}(2^{2^d} n(d+\log n))$, special case of BooleanAttributeMatching
- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$

Problem 114 (OneSidedAttributeMatching)**.** *The* AttributeMatching *problem, where the preference lists for one side are defined by d attributes, and the preferences for the other side are defined by a single attribute.*

- **Upper Bounds:** $\tilde{\mathrm{O}}(n^{2-1/\lfloor d/2 \rfloor})$

Problem 115 (ListMatching)**.** *The* StableMatching *problem with at most d distinct preference lists per side.*

Problem 116 (SinglePeakedMatching)**.** *The* StableMatching *problem where the preference lists are single-peaked. We say the men's preferences over the women in a matching market are* single-peaked *if the women can be ordered as points along a line* $(p(w_1) < p(w_2) < \cdots < p(w_n))$ *and for each man m there is a point* $q(m)$ *and a binary preference relation* $\succ_m$ *such that if* $p(w_i) \leq q(m)$ *then* $p(w_i) \succ_m p(w_j)$ *for* $j < i$ *and if* $p(w_i) \geq q(m)$ *then* $p(w_i) \succ_m p(w_j)$ *for* $j > i$*.*

- **Upper Bounds:** $\mathrm{O}(n^2)$

Problem 117 (GeometricMatching)**.** *We say the men's preferences over the women in a matching market are* geometric *in d dimensions if each women w is defined by a* location $p(w)$ *and for each man m there is an* ideal $q(m)$ *such that m prefers woman w_1 to w_2 if and only if $\|p(m)-q(w_1)\|_2^2 < \|p(m)-q(w_2)\|_2^2$, i.e. $p(w_1)$ has smaller euclidean distance from the man's ideal than $p(w_2)$. The* GeometricMatching *problem is the* StableMatching *problem where both all preferences are geometric.*

- Equivalent to AttributeMatching, see Section 4.6.2 for results on variants of this problem

Problem 118 (VerifyStableMatching)**.** *Given a matching market as well as a bipartite matching, decide if the matching is a stable matching.*

- **Upper Bounds:** $\mathrm{O}(n^2)$
- **Lower Bounds:** $\Omega(n^2)$ even in a communication complexity setting [68]

Problem 119 (VerifyAttributeMatching)**.** *Verification problem of* AttributeMatching

- **Upper Bounds:** $\tilde{\mathrm{O}}(n^{2-1/2d})$
- **Lower Bounds:** SETH-hard at time n^2 for $d=\omega(\log n)$, as it is a generalization of VerifyBooleanAttributeMatching

Problem 120 (VerifyBooleanAttributeMatching)**.** *Verification for a matching market in the Boolean attribute model.*

- **Upper Bounds:** $\tilde{\mathrm{O}}(n^{2-1/\mathrm{O}(c\log^2(c))})$ for $d=c\log n$
- **Lower Bounds:** SETH-hard at time n^2 for $d=\omega(\log n)$

Problem 121 (VerifyListMatching)**.** *Verification problem of* ListMatching

- **Upper Bounds:** $\mathsf{O}(dn)$

Problem 122 (VerifySinglePeakedMatching)**.** *Verification for a single-peaked matching market*

- **Upper Bounds:** $\mathsf{O}(n \log n)$

Problem 123 (StablePair)**.** *Given a matching market and a pair (m, w), decide if the pair is matched in any stable matching.*

- **Upper Bounds:** $\mathsf{O}(n^2)$ [87, 70]
- **Lower Bounds:** $\Omega(n^2)$ even in a communication complexity setting [68]

Problem 124 (BooleanAttributeStablePair)**.** *The* StablePair *problem with Boolean attributes and weights.*

- **Lower Bounds:** SETH-hard at time n^2 for $d = \omega(\log n)$

A.6 Other Problems

Problem 125 ($(\min,+)$-Convolution)**.** *Given n-dimensional vectors $a = (a_0, \dots, a_{n-1})$, $b = (b_0, \dots, b_{n-1}) \in [-W, W]^n$ for some $W = \mathsf{poly}(n)$, the $(\min,+)$-Convolution $a * b$ is defined by*

$$(a * b)_k = \min_{0 \le i, j < n : i + j = k} a_i + b_j \qquad \textit{for all } 0 \le k \le 2n - 2.$$

- **Upper Bounds:** $\mathsf{O}(n^2)$

Problem 126 (Convolution)**.** *Given two n-dimensional vectors* $a = (a_0, \ldots, a_{n-1})$*,* $b = (b_0, \ldots, b_{n-1}) \in [-W, W]^n$ *for some* $W = \mathsf{poly}(n)$*, the convolution* $a \circledast b$ *is defined by*

$$(a \circledast b)_k = \sum_{0 \le i,j < n: i+j=k} a_i \cdot b_j \qquad \textit{for all } 0 \le k \le 2n-2.$$

- **Upper Bounds:** $\mathrm{O}(n \log n)$

Problem 127 (Selection)**.** *Let D be a set of objects, and let* $D_1, D_2 \subseteq D^n$*. Given two sequences of inputs* $(a_1, \ldots, a_n) \in D_1$ *and* $(b_1, \ldots, b_n) \in D_2$ *and a relation* $R \subseteq D \times D$*, determine if there is* i, j *satisfying* $R(a_i, b_j)$*. We denote this selection problem with respect to a relation R and sets* D_1, D_2 *by* $\mathsf{Selection}(R^{D_1, D_2})$*. If* $D_1 = D_2 = D^n$*, we denote the problem by* $\mathsf{Selection}(R)$.

- **Upper Bounds:** $\mathrm{O}(n^2)$

Problem 128 (Sorting)**.** *Given n real numbers, find the permutation* $x_1, \ldots, x_n$ *such that* $x_1 \le \cdots \le x_n$*.*

- **Upper Bounds:** $\mathrm{O}(n \log n)$

Problem 129 (ChainSet)**.** *Let* $\{x_0, \ldots, x_n\}$ *be a set of data items, weights* $w_1, \ldots, w_{n-1} \in [-W, W]$ *and a relation* $R(x_i, x_j)$ *be given. The chain set problem for R, denoted* $\mathsf{ChainSet}(R)$ *asks to find the sequence* $i_0, i_1, i_2, \ldots, i_k$ *such that for all j with* $1 \le j \le k$ *the pair* $(x_{i_{j-1}}, x_{i_j})$ *is in the relation R and the weight* $\sum_{j=1}^{k-1} w_{i_j}$ *is minimized.*

- **Lower Bounds:** NP-complete

Problem 130 (HittingSet)**.** *Given two families of sets* $\mathscr{A}$ *and* $\mathscr{B}$ *with* $|\mathscr{A}| = |\mathscr{B}| = n$*, decide if there exists* $A \in \mathscr{A}$ *such that* $A \cup B \neq \emptyset$ *for all* $B \in \mathscr{B}$*.*

- **Upper Bounds:** $\mathsf{O}(n^2)$

Problem 131 (SparseHittingSet)**.** *Given two families of sets $\mathscr{A}$ and $\mathscr{B}$ with*

$$\sum_{A\in\mathscr{A}} |A| + \sum_{B\in\mathscr{B}} |B| = m$$

, decide if there exists $A \in \mathscr{A}$ such that $A \cup B \neq \emptyset$ for all $B \in \mathscr{B}$.

- **Upper Bounds:** $\mathsf{O}(m^2)$

Problem 132 (k-sum)**.** *Given a set of integers $A \subseteq [-W,W]^n$ for some $W = \mathsf{poly}(n)$, determine if there is $a_i \in A$ for $1 \leq i \leq k$ such that $\sum_{i=1}^{k} a_i = 0$.*

- **Upper Bounds:** $\tilde{\mathsf{O}}(n^{\lceil \frac{k}{2} \rceil})$
- **Lower Bounds:** No algorithm with time $\mathsf{O}(n^{\lceil \frac{k}{2} \rceil - \varepsilon})$ for any $\varepsilon > 0$ according to the k-sum conjecture [61]

Problem 133 (Table-k-sum)**.** *Given a k sets of integers $A_i \subseteq [-W,W]^n$ for for $1 \leq i \leq k$ and some $W = \mathsf{poly}(n)$, determine if there is $a_i \in A_i$ for $1 \leq i \leq k$ such that $\sum_{i=1}^{k} a_i = 0$.*

- Equivalent to k-sum

Problem 134 (SubsetSum)**.** *Given a set of integers $A \subseteq [-W,W]^n$ for some $W = \mathsf{poly}(n)$, determine if there is $A' \subseteq A$ for such that $\sum_{a\in A'} a = 0$.*

- **Upper Bounds:** $\tilde{\mathsf{O}}(2^{n/2})$

Problem 135 (SparseSubsetSum)**.** *Given a target value $t \in \mathbb{N}$ and a set of integers $A \subseteq [t]^n$, determine if there is $A' \subseteq A$ for such that $\sum_{a\in A'} a = t$.*

- **Upper Bounds:** $\tilde{\mathsf{O}}(n+t)$ [32]

Problem 136 (FréchetDistance)**.** *For a given n let a walk for n be a sequence $1 = i_1, \ldots, i_m = n$ such that $i_k - i_{k-1} \in \{0,1\}$ for all $k > 1$.*

Given two sequences of points in two dimensions $a_1, \ldots, a_n \in \mathbb{R}^2$ and $b_1, \ldots, b_n \in \mathbb{R}^2$, find the minimum over all walks for n $i_1, \ldots, i_m$ and $j_1, \ldots, j_m$ of

$$\max_{k=1}^{m} ||a_{i_k} - b_{j_k}|| \tag{A.5}$$

- ***Upper Bounds:*** $O(n^2/\mathsf{polylog}(n))$ *[10]*
- ***Lower Bounds:*** SETH-*hard at time* n^2 *[31]*

Problem 137 (EditDistance)**.** *Given two strings a and b with $|a| = |b| = n$, compute the minimum number of deletions, insertions and substitutions to transform a into b.*

- **Upper Bounds:** $O(n^2/\mathsf{polylog}(n))$ [105]
- **Lower Bounds:** SETH-hard at time n^2 [17]

Problem 138 (LongestCommonSubsequence)**.** *Given two strings x and y of length at most n, compute the length of the longest string z that is a subsequence of both x and y.*

- **Upper Bounds:** $O(n^2/\mathsf{polylog}(n))$ [105]
- **Lower Bounds:** SETH-hard at time n^2 [2, 33]

Bibliography

[1] Scott Aaronson, Greg Kuperberg, and Christopher Granade. Complexity zoo. https://complexityzoo.uwaterloo.ca/Complexity_Zoo.

[2] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, 2015.

[3] 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.

[4] 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 Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 375–388, New York, NY, USA, 2016. ACM.

[5] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, pages 39–51, 2014.

[6] 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 '15, pages 41–50, New York, NY, USA, 2015. ACM.

[7] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 218–230, 2015.

[8] Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 377–391. Society for Industrial and Applied Mathematics, 2016.

[9] Pankaj K Agarwal, Lars Arge, Jeff Erickson, Paolo G Franciosa, and Jeffry Scott Vitter. Efficient searching with linear constraints. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 169–178. ACM, 1998.

[10] Pankaj K Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014.

[11] Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.

[12] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter W. Shor, and Robert E. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.

[13] Alfred V. Aho, Daniel S. Hirschberg, and Jeffrey D. Ullman. Bounds on the complexity of the longest common subsequence problem. *Journal of the ACM*, 23(1):1–12, 1976.

[14] 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, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476, 2016.

[15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, 2015.

[16] Esther M. Arkin, Sang Won Bae, Alon Efrat, Kazuya Okamoto, Joseph S.B. Mitchell, and Valentin Polishchuk. Geometric stable roommates. *Information Processing Letters*, 109(4):219–224, 2009.

[17] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 51–58, 2015.

[18] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2215–2229. SIAM, 2017.

[19] Ilya Baran, Erik D Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, 2008.

[20] Gill Barequet and Sariel Har-Peled. Polygon containment and translational in-hausdorff-distance between segment sets are 3sum-hard. *International Journal of Computational Geometry & Applications*, 11(04):465–474, 2001.

[21] John Bartholdi and Michael A. Trick. Stable matching with preferences derived from a psychological model. *Operations Research Letters*, 5(4):165–169, 1986.

[22] Christopher Beck and Russell Impagliazzo. Strong ETH holds for regular resolution. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 487–494, 2013.

[23] Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[24] Nayantara Bhatnagar, Sam Greenberg, and Dana Randall. Sampling stable marriages: why spouse-swapping won't work. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1223–1232. Society for Industrial and Applied Mathematics, 2008.

[25] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[26] Andreas Björklund and Thore Husfeldt. Exact graph coloring using inclusion–exclusion. In *Encyclopedia of Algorithms*, pages 289–289. Springer, 2008.

[27] Anna Bogomolnaia and Jean-François Laslier. Euclidean preferences. *Journal of Mathematical Economics*, 43(2):87–98, 2007.

[28] Ravi B. Boppana and Michael Sipser. Handbook of theoretical computer science (vol. a). chapter The complexity of finite functions, pages 757–804. MIT Press, Cambridge, MA, USA, 1990.

[29] Michele Borassi, Pierluigi Crescenzi, and Michel Habib. Into the square: On the complexity of some quadratic-time solvable problems. *Electronic Notes in Theoretical Computer Science*, 322:51–67, 2016.

[30] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pǎtraşcu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014.

[31] Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless seth fails. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 661–670. IEEE, 2014.

[32] Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 1073–1084, 2017.

[33] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 79–97, 2015.

[34] Chris Calabro. *The exponential complexity of satisfiability problems*. PhD thesis, University of California San Diego, 2009.

[35] 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, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers, Lecture Notes in Computer Science 5917*, pages 75–85, Berlin, Heidelberg, 2009. Springer-Verlag.

[36] 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*, pages 261–270. ACM, 2016.

[37] Timothy M Chan. All-pairs shortest paths with real weights in o (n 3/log n) time. *Algorithmica*, 50(2):236–243, 2008.

[38] Timothy M Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 212–217. Society for Industrial and Applied Mathematics, 2015.

[39] Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In *Proc. 47th Annual ACM Symposium on Theory of Computing, (STOC'15)*, pages 31–40, 2015.

[40] Ashok K Chandra, Larry Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

[41] Prasad Chebolu, Leslie Ann Goldberg, and Russell Martin. The complexity of approximately counting stable matchings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 81–94. Springer, 2010.

[42] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and max-k-csp. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

[43] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. In *LIPIcs-Leibniz*

International Proceedings in Informatics, volume 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[44] Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. *SIAM Journal on Computing*, 42(3):733–751, 2013.

[45] Kim-Sau Chung. On the existence of stable roommate matchings. *Games and economic behavior*, 33(2):206–230, 2000.

[46] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science: Proceedings of the 1964 International Congress (Studies in Logic and the Foundations of Mathematics)*, pages 24–30. North-Holland Publishing, 1965.

[47] S.A. Cook. The complexity of theorem-proving procedures. pages 151–158, 1971.

[48] Stephen A Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[49] M. Cygan, M. Mucha, K. Węgrzycki, and M. Włodarczyk. On problems equivalent to (min,+)-convolution. *ArXiv e-prints*, February 2017.

[50] John Dabney and Brian C. Dean. Adaptive stable marriage algorithms. In *Proceedings of the 48th Annual Southeast Regional Conference*, page 35. ACM, 2010.

[51] Evgeny Dantsin and Alexander Wolpert. Max-sat for formulas with constant clause density can be solved faster than in $\mathcal{O}(2^n)$ time. In Armin Biere and CarlaP. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *Lecture Notes in Computer Science*, pages 266–276. Springer Berlin Heidelberg, 2006.

[52] Sashka Davis and Russell Impagliazzo. Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009.

[53] Mark de Berg, Kevin Buchin, Bart M. P. Jansen, and Gerhard J. Woeginger. Fine-grained complexity analysis of two classic TSP variants. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, pages 5:1–5:14, 2016.

[54] David P. Dobkin and David G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *Journal of Algorithms*, 6(3):381–392, 1985.

[55] Jack Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B*, 69(1965):125–130, 1965.

[56] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[57] David Eppstein. Sequence comparison with mixed convex and concave costs. *J. Algorithms*, 11(1):85–101, 1990.

[58] David A. Eppstein. *Efficient algorithms for sequence analysis with concave and convex gap costs*. PhD thesis, Columbia University, 1989.

[59] Lester R Ford Jr. Network flow theory. Technical report, DTIC Document, 1956.

[60] Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29 – 35, 1975.

[61] Anka Gajentaan and Mark H Overmars. On a class of o (n 2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

[62] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[63] David Gale and Marilda Sotomayor. Ms. machiavelli and the stable matching problem. *The American Mathematical Monthly*, 92(4):261–268, 1985.

[64] Zvi Galil and Raffaele Giancarlo. Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64(1):107–118, 1989.

[65] Zvi Galil and Kunsoo Park. A linear-time algorithm for concave one-dimensional dynamic programming. *Inf. Process. Lett.*, 33(6):309–311, 1990.

[66] Zvi Galil and Kunsoo Park. Parallel algorithms for dynamic programming recurrences with more than O(1) dependency. *J. Parallel Distrib. Comput.*, 21(2):213–222, 1994.

[67] 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, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2162–2181, 2017.

[68] Yannai A. Gonczarowski, Noam Nisan, Rafail Ostrovsky, and Will Rosenbaum. A stable marriage requires communication. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1003–1017. SIAM, 2015.

[69] A. Grønlund, K. Green Larsen, A. Mathiasen, J. Sindahl Nielsen, S. Schneider, and M. Song. Fast Exact k-Means, k-Medians and Bregman Divergence Clustering in 1D. *ArXiv e-prints*, January 2017.

[70] Dan Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987.

[71] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. Foundations of Computing Series. Mit Press, 1989.

[72] Andras Hajnal, Wolfgang Maass, Pavel Pudlak, Mario Szegedy, and Gyorgy Turan. Threshold circuits of bounded depth. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 99–110, Washington, DC, USA, 1987. IEEE Computer Society.

[73] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[74] Johan Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM, 1986.

[75] John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, 1995.

[76] Timon Hertli. 3-sat faster and simpler—unique-sat bounds for ppsz hold in general. *SIAM Journal on Computing*, 43(2):718–729, 2014.

[77] Daniel S. Hirschberg and Lawrence L. Larmore. The least weight subsequence problem. *SIAM Journal on Computing*, 16(4):628–638, 1987.

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

[79] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[80] Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. 0-1 integer linear programming with a linear number of constraints. *arXiv preprint arXiv:1401.5512*, 2014.

[81] Russell Impagliazzo, Williams Matthews, and Ramamohan Paturi. A Satisfiability Algorithm for AC^0. pages 961–972, 2012.

[82] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367 – 375, 2001.

[83] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size–depth trade-offs for threshold circuits. *SIAM J. Comput.*, 26(3):693–707, 1997. preliminary version published in STOC 1993.

[84] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 479–488. IEEE, 2013.

[85] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[86] Robert W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, 1985.

[87] Robert W. Irving and Paul Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986.

[88] Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[89] Hamid Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 749–760, 2015.

[90] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[91] Henry A Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, volume 92, pages 359–363. Citeseer, 1992.

[92] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 217–226. SIAM, 2014.

[93] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

[94] Maria M. Klawe and Daniel J. Kleitman. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discrete Math.*, 3(1):81–97, 1990.

[95] Morton Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, 1967.

[96] Donald E. Knuth. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. Amer Mathematical Society, 1997.

[97] Donald E. Knuth and Michael F. Plass. Breaking paragraphs into lines. *Softw., Pract. Exper.*, 11(11):1119–1184, 1981.

[98] Hirotatsu Kobayashi and Tomomi Matsui. Cheating strategies for the gale-shapley algorithm with complete preference lists. *Algorithmica*, 58(1):151–169, 2010.

[99] Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for subset sum. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'17)*, pages 1062–1072, 2017.

[100] Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. *CoRR*, abs/1702.05805, 2017.

[101] Marvin Künnemann, Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. *CoRR*, abs/1510.06452v5, 2016.

[102] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

[103] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[104] Inês Lynce and João Marques-Silva. Efficient haplotype inference with boolean satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 104. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[105] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

[106] Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(1):315–334, 1992.

[107] Jiří Matoušek and Otfried Schwarzkopf. Linear optimization queries. In *Proceedings of the eighth annual symposium on Computational geometry*, pages 16–25. ACM, 1992.

[108] Silvio Micali and Vijay V Vazirani. An o (v— v— c— e—) algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.

[109] Webb Miller and Eugene W. Myers. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology*, 50(2):97–120, 1988.

[110] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *International Computer Science Symposium in Russia*, pages 294–308. Springer, 2016.

[111] Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.

[112] Saburo Muroga, Iwao Toda, and Satoru Takasu. Theory of majority decision elements. *Journal of the Franklin Institute*, 271(5):376–418, 1961.

[113] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[114] Cheng Ng and Daniel S. Hirschberg. Lower bounds for the stable marriage problem and its variants. *SIAM Journal on Computing*, 19(1):71–77, 1990.

[115] James B Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.

[116] James B Orlin. Max flows in o(nm) time, or better. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 765–774. ACM, 2013.

[117] Ramamohan Paturi, Pavel Pudlák, Michael E Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM (JACM)*, 52(3):337–364, 2005.

[118] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 566–574. IEEE, 1997.

[119] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for steiner tree on planar graphs. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 20. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[120] David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003.

[121] Pavel Pudlak and Russell Impagliazzo. A lower bound for dll algorithms for k-sat. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, San Francisco, CA, USA, January 9-11, 2000*, pages 128–136, 2000.

[122] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610. ACM, 2010.

[123] Mihai Pătraşcu 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, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075, 2010.

[124] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

[125] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.

[126] Alvin E. Roth. The economics of matching: Stability and incentives. *Mathematics of operations research*, 7(4):617–628, 1982.

[127] Alvin E. Roth and Marilda A. Oliveira Sotomayor. *Two-sided Matching: A Study in Game - Theoretic Modeling and Analysis*. Econometric Society Monographs. Cambridge University, 1990.

[128] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and qbf satisfiability. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 183–192, Washington, DC, USA, 2010. IEEE Computer Society.

[129] Stefan Schneider. Satisfiability algorithms for restricted circuit classes. *CoRR*, abs/1306.4029, 2013.

[130] Uwe Schöning. A probabilistic algorithm for k-sat based on limited local search and restart. *Algorithmica*, 32(4):615–623, 2002.

[131] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. *Journal of Algorithms*, 54(1):40–44, 2005.

[132] Ilya Segal. The communication requirements of social choice rules and supporting budget sets. *Journal of Economic Theory*, 136(1):341–378, 2007.

[133] Joel I Seiferas, Michael J Fischer, and Albert R Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM (JACM)*, 25(1):146–167, 1978.

[134] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[135] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[136] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a subquadratic number of symmetric and threshold gates. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, page 4, 2016.

[137] Chung-Piaw Teo, Jay Sethuraman, and Wee-Peng Tan. Gale-shapley stable marriage problem revisited: Strategic issues and applications. *Management Science*, 47(9):1252–1267, 2001.

[138] Kenya Ueno. Exact algorithms for 0-1 integer programs with linear equality constraints. *arXiv preprint arXiv:1405.6851*, 2014.

[139] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.

[140] Robert E. Wilber. The concave least-weight subsequence problem revisited. *J. Algorithms*, 9(3):418–425, 1988.

[141] Ryan Williams. Personal communication.

[142] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[143] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[144] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 664–673. ACM, 2014.

[145] Ryan Williams. Faster decision of first-order graph properties. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 80:1–80:6, 2014.

[146] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 194–202. ACM, 2014.

[147] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[148] Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 2:1–2:17, 2016.

[149] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics.*, volume 43, 2015.

[150] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.

[151] F. Frances Yao. Efficient dynamic programming using quadrangle inequalities. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC'80)*, pages 429–435, 1980.