On some fine-grained questions in algorithms and complexity

Virginia Vassilevska Williams
Massachusetts Institute of Technology, EECS and CSAIL

Abstract

In recent years, a new "fine-grained" theory of computational hardness has been developed, based on "fine-grained reductions" that focus on exact running times for problems. Mimicking NP-hardness, the approach is to (1) select a key problem X that for some function t, is conjectured to not be solvable by any $O(t(n)^{1-\varepsilon})$ time algorithm for $\varepsilon > 0$, and (2) reduce X in a fine-grained way to many important problems, thus giving tight conditional time lower bounds for them. This approach has led to the discovery of many meaningful relationships between problems, and to equivalence classes.

The main key problems used to base hardness on have been: the 3-SUM problem, the CNF-SAT problem (based on the Strong Exponential Time Hypothesis (SETH)) and the All Pairs Shortest Paths Problem. Research on SETH-based lower bounds has flourished in particular in recent years showing that the classical algorithms are optimal for problems such as Approximate Diameter, Edit Distance, Frechet Distance and Longest Common Subsequence.

This paper surveys the current progress in this area, and highlights some exciting new developments.

1 Introduction

Arguably the main goal of the theory of algorithms is to study the worst case time complexity of fundamental computational problems. When considering a problem P, we fix a computational model, such as a Random Access Machine (RAM) or a Turing machine (TM). Then we strive to develop an efficient algorithm that solves P and to prove that for a (hopefully slow growing) function $t(n)$, the algorithm solves P on instances of size n in $O(t(n))$ time in that computational model. The gold standard for the running time $t(n)$ is linear time, $O(n)$; to solve most problems, one needs to at least read the input, and so linear time is necessary.

The theory of algorithms has developed a wide variety of techniques. These have yielded near-linear time algorithms for many diverse problems. For instance, it is known since the 1960s and 70s (e.g. [143, 144, 145, 99]) that Depth-First Search (DFS) and Breadth-First Search (BFS) run in linear time in graphs, and that using these techniques one can obtain linear time algorithms (on a RAM) for many interesting graph problems: Single-Source Shortest paths, Topological Sort of a Directed Acyclic Graph, Strongly Connected Components, Testing Graph Planarity etc. More recent work has shown that even more complex problems such as Approximate Max Flow, Maximum Bipartite Matching, Linear Systems on Structured Matrices, and many others, admit close to linear time algorithms, by combining combinatorial and linear algebraic techniques (see e.g. [140, 64, 141, 116, 117, 67, 68, 65, 66, 113]).

Nevertheless, for most problems of interest, the fastest known algorithms run much slower than linear time. This is perhaps not too surprising. Time hierarchy theorems show that for most computational models, for any computable function $t(n) \geq n$, there exist problems that are solvable in $O(t(n))$ time but are NOT solvable in $O(t(n)^{1-\varepsilon})$ time for $\varepsilon > 0$ (this was first proven for TMs [95], see [124] for more).

Time hierarchy theorems are proven by the diagonalization method pioneered by Cantor in the 19th century. Unfortunately, however, these theorems say almost nothing about *particular* problems of interest.

Consider for instance the ubiquitous Boolean Satisfiability (SAT) problem: given a Boolean expression F over n variables and Boolean operators AND, OR and NOT, is there a Boolean assignment to the variables that makes F evaluate to true?

A simple algorithm to solve SAT is to try all possible 2^n assignments and evaluate F on each of them. The runtime depends on how F is represented. In Circuit-SAT, F is given as a (directed acyclic) circuit with AND, OR and NOT gates, n input gates representing the variables and a designated output gate. The evaluation of a circuit can be performed in $O(m+n)$ time, where m is the number of gates and wires, by evaluating its gates in topological order. A much more structured version of SAT is CNF-SAT. Here, F is given as a Boolean expression in Conjunctive Normal Form (CNF): an AND of m *clauses* that are ORs of *literals* (variables and their negations), i.e. one needs to satisfy every clause by setting at least one literal to TRUE. A CNF-Formula can be evaluated in $O(m+n)$ time. Regardless, of the representation, Circuit or CNF, the enumeration of all 2^n assignments dominates if m is, say, subexponential.

When the maximum clause length is a constant k, CNF-SAT can be solved in $O^*(2^{n-cn/k})$ for constant c independent of n and k (see e.g., [98, 122, 126, 125, 137, 138]). Nevertheless, as k grows, this runtime approaches 2^n, and the exhaustive search algorithm is essentially the best known for general CNF-SAT. For general Circuit-SAT, there is *no better* algorithm known than exhaustive search.

A natural question then is, can one prove, for a robust model of computation, that this 2^n runtime dependence is inherent to the problem? Unfortunately, such a result is very far from current techniques in computational complexity. In fact, it is not even known whether SAT can be solved in *linear time*!

The only known superlinear runtime lower bounds for SAT are obtained by restricting the algorithms, for instance, to use only a small amount of space. The best of these is by R. Williams [155] who showed that if an algorithm running on a RAM uses $n^{o(1)}$ space, then it requires at least $n^{2\cos(\pi/7)-o(1)} \geq \Omega(n^{1.8})$ time to solve SAT on n variables. This runtime lower bound is very far from the 2^n upper bound, and in fact, it was shown [51] that this is the best result one can obtain with known techniques.

Since unconditional lower bounds seem so challenging to derive, the computer science community has long resorted to lower bounds that are conditioned on plausible, but so far unproven hypotheses. One of the most commonly used hardness hypotheses is P $\neq$ NP. The hypothesis is formally about *decision* problems — problems whose outputs are binary — YES or NO. E.g. CNF-SAT is the decision problem that asks whether the given CNF formula is satisfiable. P is the set of decision problems that can be decided by a polynomial time algorithm[1] on a TM. NP is the set of decision problems that have polynomial time algorithms (on a TM) that can verify a polynomial sized solution to the instance[2]: e.g. CNF-SAT is in NP because we can check in polynomial time if any given Boolean assignment satisfies the formula.

P vs NP asks whether all decision problems that can be verified in polynomial time (in the sense of the above paragraph), can also be decided in polynomial time. P vs NP is one of the most famous open problems in computer science. It is one of the Clay Millennium problems. While current techniques seem very far from resolving this problem, most researchers believe that P $\neq$ NP.

The most fascinating implication of the P $\neq$ NP hypothesis is that many problems such as SAT *cannot* be solved in polynomial time. A problem A is NP-hard if every instance of every problem in NP can be encoded in polynomial time as an instance of A. A problem in NP which is NP-hard is called NP-Complete.

Clearly, if an NP-hard problem has a polynomial time algorithm, then P = NP. Thus, if we assume that P $\neq$ NP, no NP-hard problem can have a polynomial time algorithm. Starting with the work of Cook and Levin (who showed that SAT is NP-complete) and Karp (who added 21 more NP-complete problems),

[1]When we say polynomial, we mean $O(n^c)$ for constant $c > 0$, where n is the size of the input instance.

[2]More formally, $\pi \in$NP if there is a polynomial time algorithm V such that if x is a YES instance of π, then there is a string y of size $O(|x|^c)$ for some constant c, such that $V(x, y)$ returns YES, and if x is a NO instance, $V(x, y)$ returns NO for all y.

NP-hardness took off. Now there are many thousands of problems known to be NP-hard.

NP-hardness is arguably the biggest export of theoretical computer science (TCS) to other disciplines. It is routinely used to explain why it is so hard to find efficient algorithms for problems occurring in practice, and why one should probably use specialized algorithms and heuristics to solve them.

P and NP are defined for TMs. However, due to polynomial time reductions between computational models (see [146] for a thorough treatment), whether we consider a TM or RAM in the definition of P and NP does not actually matter. P vs NP is essentially model-independent. This model-independence was one of the reasons to focus on *polynomial* time as a model of efficiency. (Another simple reason is that polynomials compose into polynomials.) Nevertheless, no-one would argue that all polynomial runtimes are actually efficient. In fact, for today's large inputs, even *quadratic* time is inefficient.

There are many fundamental problems for which the fastest known algorithms run in quadratic time or slower. A simple example is the *Edit Distance* problem, with many diverse applications from computational biology to linguistics: given two strings α and β, over some finite alphabet, what is the smallest sequence of symbol insertions, deletions and substitutions that can be performed on α to obtain β?

The problem has a long history: a classical dynamic programming algorithm by Wagner and Fisher [152] runs in $O(n^2)$ time, and despite many efforts, the best known algorithm [118] only shaves a $\log^2 n$ factor. On inputs where n is in the billions (such as the human genome), quadratic runtime is prohibitive.

Another simple problem, from Computational Geometry, asks for a given set of points in the plane, are any three colinear; conversely, are the points are in general position. Before running a computational geometry algorithm, one typically needs to check this important primitive. Unfortunately, the best known algorithms for this Colinearity question for n points run in $n^{2-o(1)}$, i.e. quadratic time.

There are many such examples within P, from a vast variety of research areas. Why has it been so hard to find faster algorithms for such problems? Addressing this is impossible using P $\neq$ NP as an assumption: no problem that is already in P can be NP-Complete, unless P = NP. We need a different approach.

The Fine-Grained Question. Let us delve further into the issue described above. From now on let us fix the computational model to a word-RAM with $O(\log n)$ bit words. Informally, this is a RAM machine that can read from memory, write to memory and perform operations on $O(\log n)$ bit chunks of data in constant time. We can fix any computational model; we pick the word-RAM because it is simple to work with.

When faced with a computational problem P, we can usually apply well-known algorithmic techniques (such as dynamic programming, greedy, divide and conquer etc.) and come up with a simple algorithm that runs in $O(t(n))$ time on inputs of size n.

Often this algorithm is obtained by the brute-force approach — enumerate all candidate solutions in a search space. This is the case for SAT, but also for a large variety of other problems.

Sometimes the simple algorithm is not brute-force but uses textbook techniques in the natural way. Consider for instance the Longest Common Subsequence problem (LCS), a simplified version of Edit Distance: given two sequences A and B of length n over an alphabet Σ, determine the maximum length sequence S that appears in both A and B, with symbols in the same order, but possibly not consecutively. For instance, an LCS of (b, b, c, a, d, e) and (b, d, c, b, e) is (b, c, e). The textbook approach here is to apply dynamic programming, concluding that $\text{LCS}((b, b, c, a, d, e), (b, d, c, b, e))$ is the longest of $\text{LCS}((b, b, c, a, d), (b, d, c, b)) \odot e$, $\text{LCS}((b, b, c, a, d, e), (b, d, c, b))$, and $\text{LCS}((b, b, c, a, d), (b, d, c, b, e))$. The runtime is $O(n^2)$ since throughout the computation, one at most needs to memoize the computed longest common subsequences of n^2 pairs of prefixes. (The textbook algorithm for Edit Distance is similar.)

More often than not, the obtained "textbook" running time seems difficult to improve upon: improvements have been sought after for decades, and the simple algorithm has stood almost unchallenged. We

mentioned earlier that this is the case for CNF-SAT, Colinearity and Edit Distance. The situation is similar for LCS and Edit Distance (fastest runtime $O(n^2/\log^2 n)$ [118] for constant size alphabet and otherwise $O(n^2 \log\log n/\log^2 n)$ [39, 94]), and for a large variety of other problems from all over computer science and beyond. The central question that needs to be addressed is:

For each of the problems of interest with textbook running time $O(t(n))$ and nothing much better known, is there a barrier to obtaining an $O(t(n)^{1-\varepsilon})$ time algorithm for $\varepsilon > 0$?

Relatedly, is the reason for this difficulty the same for all problems of interest?

2 Fine-Grained Complexity (and algorithms).

We would like to mimic NP-Completeness. The approach will be as follows.

1. We will identify some believable *fine-grained* hardness hypotheses. These will be about specific conjectured running times for very well-studied computational problems.
2. Using *fine-grained* reductions we will show that for a problem with textbook running time $t(n)$, obtaining an $O(t(n)^{1-\varepsilon})$ time algorithm for $\varepsilon > 0$ would violate one or more of the hypotheses. The reductions we employ cannot be mere polynomial time reductions - they would have to be tailored to the specific textbook runtime $t(n)$. As we will see, they will differ in other ways as well from most reductions used in traditional complexity.

We would also like to give equivalences, i.e. to show that problem A with textbook running time $a(n)$ and problem B with textbook running time $b(n)$ are equivalent in the sense that if A admits an $a(n)^{1-\varepsilon}$ time algorithm for $\varepsilon > 0$, then B admits an $b(n)^{1-\varepsilon'}$ time algorithm for some $\varepsilon' > 0$. This would mean that the reason why it has been hard to improve on A and on B is the same.

In the following we will discuss some of the most prominent hardness hypotheses in fine-grained complexity, and the reductions we employ to achieve fine-grained hardness.

2.1 Key Hypotheses

Much of fine-grained complexity is based on hypotheses of the time complexity of three problems: CNF-SAT, All-Pairs Shortest Paths (APSP) and 3-SUM. Below we will introduce these, and a few more related hypotheses. There are no known reductions between CNF-SAT, APSP and 3-SUM: they are potentially unrelated. All hypotheses are about the word-RAM model of computation with $O(\log n)$ bit words, where n is the size of the input.

SETH. Impagliazzo, Paturi and Zane [101] introduced the Strong Exponential Time Hypothesis (SETH) to address the complexity of CNF-SAT. At the time they only considered deterministic algorithms, but nowadays it is common to extend SETH to allow randomization.

Hypothesis 1 (Strong Exponential Time Hypothesis (SETH))**.** *For every $\varepsilon > 0$ there exists an integer $k \geq 3$ such that CNF-SAT on formulas with clause size at most k (the so called k-SAT problem) and n variables cannot be solved in $O(2^{(1-\varepsilon)n})$ time even by a randomized algorithm.*

As the clause size k grows, the lower bound given by SETH converges to 2^n. SETH also implies that general CNF-SAT on formulas with n variables and m clauses requires $2^{n-o(n)}\text{poly}(m)$ time.

SETH is motivated by the lack of fast algorithms for k-SAT as k grows. It is a much stronger assumption than P $\neq$ NP which assumes that SAT requires superpolynomial time. A weaker version, the Exponential Time Hypothesis (ETH) asserts that there is some constant $\delta > 0$ such that CNF-SAT requires $\Omega(2^{\delta n})$.

Both ETH and SETH are used within Fixed Parameter and Exponential Time algorithms as hardness hypotheses, and they imply meaningful hardness results for a variety of problems (see e.g. [72]). Because we are concerned with tight, fine-grained, runtime bounds, we focus on SETH as opposed to ETH.

3-SUM Hypothesis The 3-SUM problem is as follows: given a set S of n integers from $\{-n^c, \ldots n^c\}$ for some constant c, determine whether there are $x, y, z \in S$ such that $x + y + z = 0$. A standard hashing trick allows us to assume that $c \leq 3 + \delta$ for any $\delta > 0$.[3]

Hypothesis 2 (3-SUM Hypothesis)**.** *3-SUM on n integers in $\{-n^4, \ldots, n^4\}$ cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$ by a randomized algorithm.*

The hypothesis was introduced by Gajentaan and Overmars [87, 88] who used it to show that many problems in computational geometry require quadratic time, assuming that 3-SUM does. Quadratic lower bounds for 3-SUM are known in restricted models of computation such as the linear decision tree model in which each decision is based on the sign of an affine combination of at most 3 inputs (see e.g. [82, 79]). However, in the more general linear decision tree model, Kane et al. [105] show that $O(n \log^2 n)$ queries suffice to solve 3-SUM, so that such lower bounds should be taken with a grain of salt.

The 3-SUM problem is very simple and has been studied extensively. The textbook algorithm is a simple $O(n^2 \log n)$ time enumeration algorithm: sort S and then for every $x, y \in S$, check if $-z \in S$ using binary search. An $O(n^2)$ runtime can be obtained by traversing the sorted order of S in both directions. Baran, Demaine and Pătraşcu [37] improved this running time to $O(n^2(\log\log n)^2/\log^2 n)$ time. If the input numbers are real numbers instead of integers (now in the Real-RAM model of computation), Grønlund and Pettie [104] gave an $O(n^2(\log\log n)^{2/3}/\log^{2/3} n)$ time algorithm. This runtime was recently improved by Chan [56] to $n^2(\log\log n)^{O(1)}/\log^2 n$, almost matching the known running time for integer inputs.

All-Pairs Shortest Paths (APSP). The APSP problem is as follows: given an n node graph $G = (V, E)$, and integer edge weights $w : E \to \{-M, \ldots, M\}$ for some $M = \text{poly}(n)$, compute for every $u, v \in V$, the (shortest path) distance $d(u, v)$ in G from u to v, i.e. the minimum over all paths from u to v of the total weight sum of the edges of the path. G is assumed to contain no negative weight cycles.

The textbook algorithm for APSP is the $O(n^3)$ time Floyd-Warshall algorithm from the 1960s based on dynamic programming. Many other algorithms run in the same time. For instance, one can run Dijkstra's algorithm from every vertex, after computing new nonnegative edge weights using Johnson's trick [103]. Following many polylogarithmic improvements (e.g. [86, 54]), the current best APSP running time is a breakthrough $n^3/\exp(\sqrt{\log n})$ runtime by R. Williams [159]. Despite the long history, the cubic runtime of the textbook algorithm has remained unchallenged. This motivates the APSP Hypothesis below, implicitly used in many papers (e.g. [136]). Its first explicit use as a hardness hypothesis is in [148].

[3] One can pick a random prime p that is between n and $n^{3+\delta}$. The number of distinct primes in this range that can divide any particular sum of three input integers is $O(1)$, and hence the total number of distinct primes that can divide the sum of some three input integers is $O(n^3)$. However, there are $\Omega(n^{3+\delta'})$ primes in the interval between n and $n^{3+\delta}$, for any $0 < \delta' < \delta$, and the probability that p divides one of the sums from S is $\leq O(1/n^{\delta'})$. We can then reduce 3-SUM mod p to three instances of the original 3-SUM problem with integers in the range $\{-2p, \ldots, p-1\}$ — checking if x, y, z sum to 0,p or $2p$.

Hypothesis 3 (APSP Hypothesis)**.** *No randomized algorithm can solve APSP in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$ on n node graphs with edge weights in $\{-n^c, \ldots, n^c\}$ and no negative cycles for large enough c.*

2.2 Fine-grained reductions

Our goal is as follows. Consider problem A with textbook runtime $a(n)$ and problem B with textbook runtime $b(n)$. Given a supposed $O(b(n)^{1-\varepsilon})$ time algorithm for B for $\varepsilon > 0$, we would like to compose it with another algorithm (the reduction) that transforms instances of A into instances of B, to obtain an algorithm for A running in time $O(a(n)^{1-\varepsilon'})$ time for $\varepsilon' > 0$ (a function of ε).

The most common reductions used in complexity are polynomial time (or sometimes logspace) reductions. For our purposes such reductions are not sufficient since we truly care about the runtimes $a(n)$ and $b(n)$ that we are trying to relate, and our reductions need to run faster than $a(n)$ time for sure; merely polynomial time does not suffice.

Beyond the time restriction, reductions differ in whether they are Karp or Turing reductions. Karp (also called many-one) reductions transform an instance of A into a single instance of B. Turing reductions are allowed to produce multiple instances, i.e. oracle calls to B. If we restrict ourselves to Karp-style reductions, then we wouldn't be able to reduce a search problem to any decision problem: decision problems return a single bit and if we only make one oracle call to a decision problem, in general we would not get enough information to solve the original search problem. We hence use Turing-style reductions.

The most general definition is:

Definition 2.1 (Fine-grained reduction)**.** *Assume that A and B are computational problems and $a(n)$ and $b(n)$ are their conjectured running time lower bounds, respectively. Then we say A (a, b)-reduces to B, $A \leq_{a,b} B$, if for every $\varepsilon > 0$, there exists $\delta > 0$, and an algorithm R for A that runs in time $a(n)^{1-\delta}$ on inputs of length n, making q calls to an oracle for B with query lengths $n_1, \ldots, n_q$, where*

$$\sum_{i=1}^{q} (b(n_i))^{1-\varepsilon} \leq (a(n))^{1-\delta}.$$

If $A \leq_{a,b} B$ and $B \leq_{b,a} A$, we say that A and B are fine-grained equivalent, $A \equiv_{a,b} B$.

The definition implies that if $A \leq_{a,b} B$ and B has an algorithm with running time $O(b(n)^{1-\varepsilon})$, then, A can be solved by replacing the oracle calls by the corresponding runs of the algorithm, obtaining a runtime of $O(a(n)^{1-\delta})$ for A for some $\delta > 0$. If $A \equiv_{a,b} B$, then arguably the reason why we have not been able to improve upon the runtimes $a(n)$ and $b(n)$ for A and B, respectively, is the same.

Notice that the oracle calls in the definition need not be independent — the ith oracle call might be adaptively chosen, according to the outcomes of the first $i - 1$ oracle calls.

3 Hardness results from SETH.

SETH was first used to give conditional hardness for other NP-hard problems. For instance, Cygan et al. [71] show that several other problems (such as k-Hitting Set and k-NAE-SAT) are equivalent to k-SAT, in that an $O(2^{(1-\varepsilon)n})$ time algorithm for $\varepsilon > 0$ for one of them (for all k) would imply such an algorithm for all of them, and would refute SETH.

The introduction of SETH as a hardness hypothesis for polynomial time problems was initiated by R. Williams [157]. Among other things, [157] show that the so called *Orthogonal Vectors* (OV) problem, a problem in quadratic time, requires quadratic time under SETH. We will describe the reduction shortly.

Orthogonal Vectors. The OV problem, and its generalization k-OV, form the basis of many fine-grained hardness results for problems in P.

The *OV problem* is defined as follows: Let $d = \omega(\log n)$; given two sets $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$, determine whether there exist $a \in A, b \in B$ so that $a \cdot b = 0$ where $a \cdot b = \sum_{i=1}^{d} a[i] \cdot b[i]$.

The *k-OV problem* for constant $k \geq 2$ is the generalization of OV to k sets: Let $d = \omega(\log n)$; given k sets $A_1, \ldots, A_k \subseteq \{0,1\}^d$ with $|A_i| = n$ for all i, determine whether there exist $a_1 \in A_1, \ldots, a_k \in A_k$ so that $a_1 \cdot \ldots \cdot a_k = 0$ where $a_1 \cdot \ldots \cdot a_k := \sum_{i=1}^{d} \prod_{j=1}^{k} a_j[i]$.

OV is a special case of Hopcroft's problem: given two sets R and B of n vectors each in $\mathbb{R}^d$, detect $r \in R$, $b \in B$ such that $\langle r, b \rangle = 0$ (an equivalent version that Hopcroft posed is when we are given points and hyperplanes through the origin, and we want to detect a point lying on one of the hyperplanes). The fastest algorithms for Hopcroft's problem for general d run in $2^{O(d)} n^{2-\Theta(d)}$ time [119, 61].

OV is equivalent to the Batch Subset Query problem from databases [133, 91, 20, 120]: given two sets S and T of sets over $[d]$, check if there is some $s \in S, t \in T$ such that $s \subseteq t$. It is also known to be equivalent to the classical Partial Match problem.

It is not hard to solve k-OV in $O(n^k d)$ time by exhaustive search, for any $k \geq 2$. The fastest known algorithms for the problem run in time $n^{k-1/\Theta(\log(d/\log n))}$ [17, 57]. It seems that $n^{k-o(1)}$ is necessary. This motivates the now widely used *k-OV Hypothesis*.

Hypothesis 4 (k-OV Hypothesis)**.** *No randomized algorithm can solve k-OV on instances of size n in $n^{k-\varepsilon}\text{poly}(d)$ time for constant $\varepsilon > 0$.*

Interestingly, Williams and Yu [161] show that the 2-OV Hypothesis is false when operations are over the ring $\mathbb{Z}_m$, or over the field $\mathbb{F}_m$ for any prime power $m = p^k$. In the first case, OV can be solved in $O(nd^{m-1})$ time, and in the second case, in $O(nd^{p(k-1)}$ time. Although the problem is easier in these cases, [161] actually also show that these runtimes cannot be improved very much, unless SETH fails, so there is still some hidden hardness. Over $\mathbb{Z}_6$, it turns out that OV does still require quadratic time under SETH: no $n^{2-\varepsilon} d^{o_d(\log d/\log\log d)}$ time algorithm $\varepsilon > 0$ can exist.

Gao et al. [90] show that OV is complete for a large class of problems: the class of all first order properties. They consider properties expressible by a first-order formula with $k+1$ quantifiers on a given structure with m records; checking if any such property holds can easily be done in $O(m^k)$ time, and [90] give an improved $m^k/2^{\Theta(\sqrt{\log m})}$ time algorithm. The completeness of OV is as follows. The First-Order Property Conjecture (FOPC) [90] asserts that there is some $k \geq 2$ s.t. for all $\varepsilon > 0$ there is a first order property on $k+1$ quantifiers that cannot be decided in $O(m^{k-\varepsilon})$ time. [90] show that FOPC is equivalent to the 2-OV hypothesis.

Here we present Williams' [157] result that k-OV requires essentially n^k time, under SETH. Afterwards we will see some applications of this result.

Theorem 3.1 ([157])**.** *If k-OV on sets with N vectors from $\{0,1\}^m$ can be solved in $N^{k-\varepsilon}\text{poly}(m)$ time for any $\varepsilon > 0$, then CNF-SAT on n variables and m clauses can be solved in $2^{(1-\varepsilon')n}\text{poly}(m)$ time for some $\varepsilon' > 0$ and SETH is false.*

Proof. We present a fine-grained reduction from CNF-SAT to k-OV. Let the given formula F have n variables and m clauses. Split variables into k parts $V_1, \ldots, V_k$ on n/k variables each. For every $j = 1, \ldots k$ create a set A_j containing a length m binary vector $a^j(\phi)$ for every one of the $N = 2^{n/k}$ Boolean assignments ϕ to the variables in V_j, where

$$a^j(\phi)[c] = 0 \text{ if the } c\text{th clause of } F \text{ is satisfied by } \phi, \text{ and 1 otherwise.}$$

The instance of k-OV formed by $A_1, \ldots, A_k$ has all $|A_j| = N = 2^{n/k}$.

Suppose that for some $a_1(\phi_1) \in A_1, \ldots, a_k(\phi_k) \in A_k$, we have $\sum_c \prod_j a_j(\phi_j)[c] = 0$, then for every clause c, there is some vector $a_j(\phi_j)$ that is 0 in clause c, and hence the Boolean assignment ϕ_j to the variables in V_j satisfies clause c. Thus, the concatenation $\phi_1 \odot \ldots \odot \phi_k$ is a Boolean assignment to all variables V of F that satisfies all clauses. Conversely, if ϕ satisfies all clauses, then we define ϕ_j to be the restriction of ϕ to V_j, and we see that $\sum_c \prod_j a_j(\phi_j)[c] = 0$, as every clause must be satisfied by some ϕ_j.

If k-OV on k sets of N vectors each in $\{0,1\}^m$ can be solved in $N^{k-\varepsilon}\text{poly}(m)$ time, then CNF-SAT on n variables and m clauses can be solved in time $(2^{n/k})^{k-\varepsilon}\text{poly}(m) = 2^{n-\varepsilon'}\text{poly}(m)$ time for $\varepsilon' = \varepsilon/k > 0$. This contradicts SETH. □

We note that due to the Sparsification Lemma [101], one can assume that the n-variable ℓ-CNF instance that one reduces to k-OV has $O(n)$ clauses. Thus, to refute SETH, one only needs to obtain an $N^{k-\varepsilon}\text{poly}(d)$ time algorithm for $\varepsilon > 0$ for k-OV where the dimension d of the vectors is any slowly growing function of N that is $\omega(\log N)$, for instance $d = \log^2 N$.

Besides k-OV, Williams also considers the k-Dominating set problem: for a fixed constant k, given an n node graph $G = (V, E)$, determine whether there is a subset $S \subseteq V$ of size k so that for every $v \in V$ there is some $s \in S$ so that $(s, v) \in E$. Williams ([158], later in [130]) shows via a reduction from CNF-SAT, k-Dominating set requires $n^{k-o(1)}$ time. The reduction from CNF-SAT to k-OV can be routed through k-Dominating Set, showing that that problem is in a sense between CNF-SAT and k-OV.

The k-OV problem is the basis for most reductions from CNF-SAT to problems within Polynomial Time. We will give two examples, and will then give a short summary of most known results.

It is simple to reduce k-OV to $(k-1)$-OV: go over all vectors in the first set, and solve a $(k-1)$-OV instance for each. Hence 2-OV is the hardest out of all k-OV problems. Also, k-OV is potentially strictly harder than SETH. Thus, even if SETH turns out to be false, the k-OV Hypothesis might still hold.

Orthogonal Vectors and Graph Diameter. Arguably the first reduction from SETH to a graph problem in P is from a paper by Roditty and Vassilevska Williams [135] that considers the Diameter problem: given an n node, m edge graph $G = (V, E)$, determine its diameter, i.e. $\max_{u,v \in V} d(u, v)$.

For directed or undirected graphs with arbitrary (real) edge weights, the fastest known algorithm for the Diameter problem computes all the pairwise distances in G, solving APSP. As mentioned earlier, the fastest known algorithm for APSP in dense graphs runs in $n^3/\exp(\sqrt{\log n})$ time. For sparser graphs, the fastest known algorithms run in $\tilde{O}(mn)$ time[4] [129, 127, 128].

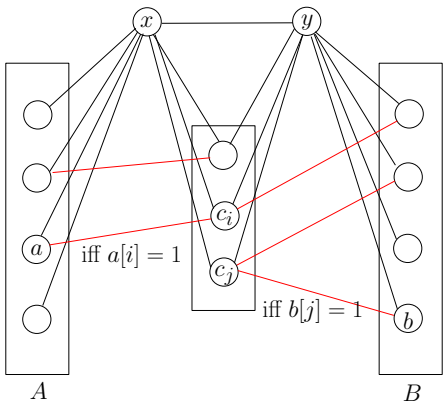
If the graph is unweighted, one can solve Diameter in $\tilde{O}(n^\omega)$ time, where $\omega < 2.373$ [150, 111] is the exponent of square matrix multiplication. If the graph has small integer edge weights in $\{0, \ldots, M\}$, Diameter is in $\tilde{O}(Mn^\omega)$ time (see e.g. [70]). However, since $\omega \geq 2$, all known algorithms for Diameter run in $\Omega(n^2)$ time, even when the graph is unweighted, and undirected[5], and has $m \leq O(n)$ edges.

With a simple reduction, [135] show that under SETH, the Diameter problem in undirected unweighted graphs with n nodes and $O(n)$ edges requires $n^{2-o(1)}$ time. Moreover, their reduction shows that under SETH, even distinguishing between graphs with Diameter 2 and 3 requires $n^{2-o(1)}$ time, and hence no $3/2 - \varepsilon$ approximation algorithm can run in $O(n^{2-\delta})$ time for $\varepsilon, \delta > 0$ even on sparse graphs.

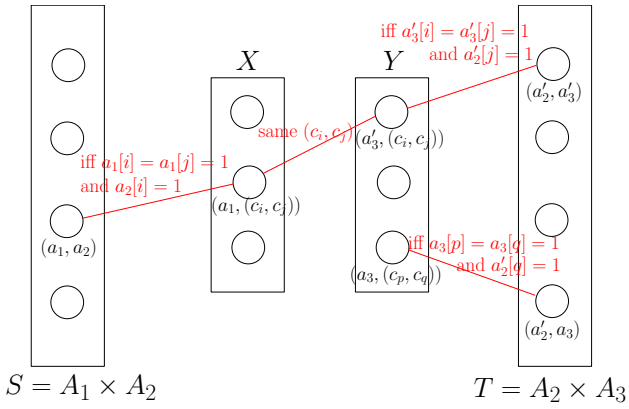
Chechik et al. [62] obtained a $3/2$-approximation algorithm for Diameter that runs in $\tilde{O}(m^{3/2})$ time in m-edge graphs; their algorithm was based on a previous $\tilde{O}(m\sqrt{n})$ time algorithm from [135] that is a $3/2$

[4] $\tilde{O}(f(n))$ denotes $f(n)\text{polylog}(n)$.

[5] All shortest paths problems, including Diameter, are at least as hard in directed graphs as they are in undirected graphs; similarly, they are at least as hard in weighted graphs as they are in unweighted graphs, and at least as hard in denser graphs than they are in sparser graphs.



(a) Reduction from OV to Diameter 2 vs 3.



(b) Reduction from 3-OV to S-T-Diameter 3 vs 7.

Figure 1: Two reductions to Diameter Problems.

approximation when the diameter is divisible by 3 (and slightly worse otherwise). This algorithm is thus in a sense optimal, under SETH: it runs in truly subquadratic time in sparse graphs, but if one wants to improve upon the approximation factor even slightly, all of a sudden $n^{2-o(1)}$ time is needed. An $\tilde{O}(n^2)$ runtime in sparse graphs is very easy to achieve: just solve APSP by running BFS from every node! Thus, under SETH, there is essentially nothing more to do besides the easy algorithm, for approximation below $3/2$.

Theorem 3.2. *If one can distinguish between Diameter* 2 *and* 3 *in an undirected unweighted graph with* $O(N)$ *nodes and edges in* $O(N^{2-\varepsilon})$ *time for some* $\varepsilon > 0$*, then* 2*-OV on two sets of* n *vectors in* d *dimensions can be solved in* $n^{2-\varepsilon}\text{poly}(d)$ *time and SETH is false.*

Proof. Suppose we are given an instance of 2-OV, A, B of n vectors each in $\{0,1\}^d$, where $|A| = |B| = n$. Let's create a graph G as follows. See Figure 1a.

For every vector $a \in A$, create a node a of G. For every vector $b \in B$, create a node b of G. For every $i \in [d]$, create a node c_i. We add two additional nodes x and y.

The edges are as follows. For every $a \in A$ and $i \in [d]$, if $a[i] = 1$, add an edge between a and c_i in G. Similarly, for every $b \in B$ and $i \in [d]$, if $b[i] = 1$, add an edge between b and c_i in G.

The edges incident to x are as follows: (x, a) for every $a \in A$, (x, c_i) for every $i \in [d]$ and (x, y). The edges incident to y are as follows: (y, b) for every $b \in B$ and (y, c_i) for every $i \in [d]$ (and (x, y)).

Now, if $a \in A$ and $b \in B$ are not orthogonal, then there is some i such that $a[i] = b[i] = 1$, and so $d(a, b) = 2$ via the path through c_i. Otherwise, if a and b are orthogonal, then there is no such c_i and the shortest $a - b$ path goes through (x, y), and $d(a, b) = 3$. All nodes in the graph are at distance at most 2 to x, y, and each c_i, and hence the Diameter is 3 if there is an orthogonal pair, and 2 otherwise.

Let $N = nd$. The number of nodes and edges is at most $O(N)$. If Diameter 2 vs 3 can be solved in $O(N^{2-\varepsilon})$ time for some $\varepsilon > 0$, then 2-OV is in $O((nd)^{2-\varepsilon}) \leq n^{2-\varepsilon}\text{poly}(d)$ time for $\varepsilon > 0$. □

By the above result, we get that under the 2-OV Hypothesis, improving upon the approximation factor of the known Diameter algorithms [135, 62] is impossible without blowing up the running time to $n^{2-o(1)}$. However, all known $3/2$-approximation algorithms run in $\tilde{O}(n^{1.5})$ time in sparse graphs. Can this runtime be improved? Can it be made linear?

Cairo et al. [52] presented faster approximation algorithms for Diameter. Generalizing [135, 62], they presented for every integer $k \geq 1$, an $\tilde{O}(mn^{1/(k+1)})$ time algorithm that is a $2 - 1/2^k$-approximation to the

Diameter of undirected graphs (if it is divisible by $2^{k+1}-1$, and slightly worse otherwise). Unfortunately, the approximation quality degrades as the runtime decreases. Thus their results do not answer the question of whether there are faster $3/2$-approximation algorithms.

In recent work, Backurs et al. [33] show that unless the 3-OV Hypothesis (and hence SETH) is false, any $3/2$-approximation algorithm to the Diameter in sparse graphs needs $n^{1.5-o(1)}$ time, thus resolving the question. They also obtain a variety of other tight conditional lower bounds based on k-OV for different k for graph Eccentricities, and variants of Diameter.

The hardness result for $3/2$-approximate Diameter is based on a hardness construction for a slightly more difficult problem called S-T Diameter. In it, one is given a graph $G = (V, E)$ and two subsets $S, T \subseteq V$ and is asked to compute $D_{S,T} := \max_{s \in S, t \in T} d(s, t)$, the so called S-T Diameter which is the largest distance between a node of S and a node of T.

When it comes to exact computation in sparse weighted graphs, S-T Diameter is (n^2, n^2)-equivalent to Diameter (see [33]). When it comes to approximation, the problems differ a bit. In linear time, Diameter admits a 2-approximation, while S-T Diameter admits a 3-approximation. In $\tilde{O}(m^{3/2})$ time, Diameter admits a $3/2$-approximation, whereas S-T Diameter admits a 2-approximation. Thus, the starting point of the hardness for $3/2$-approximate Diameter is a hardness construction for 2-approximate S-T Diameter.

Theorem 3.3 ([33])**.** *Under the 3-OV Hypothesis, no $O(N^{1.5-\varepsilon})$ time algorithm for $\varepsilon > 0$, can distinguish between S-T Diameter 3 and 7 in graphs with at most N nodes and edges.*

Since any 2-approximation algorithm can distinguish between S-T Diameter 3 and 7, the Theorem above implies that $n^{1.5-o(1)}$ time is needed to 2-approximate the S-T Diameter of a sparse graph. We will present the proof of Theorem 3.3. To complete the reduction to Diameter, some extra gadgets are needed; these create a graph in which the Diameter is either 5 or 9 and thus give hardness for $3/2$-Diameter approximation. We refer the reader to the presentation in [33]. Theorem 3.2 and the extension to Theorem 3.3 to Diameter can be generalized to a reduction from k-OV for arbitrary k to Diameter, thus showing a time/approximation tradeoff lower bound [33].

Proof Sketch of Theorem 3.3. Let $A_1, A_2, A_3 \subseteq \{0,1\}^d$ be the n-sets forming the 3-OV instance.

For every pair of vectors $a_1 \in A_1, a_2 \in A_2$, we create a node (a_1, a_2) in a set S. For every pair of vectors $a_2 \in A_2, a_3 \in A_3$, we create a node (a_2, a_3) in a set T.

For every node $a_1 \in A_1$ and every pair of coordinates $i, j \in [d]$, create a node (a_1, x_i, x_j) in a set X. For every node $a_3 \in A_3$ and every pair of coordinates $i, j \in [d]$, create a node (a_3, x_i, x_j) in a set Y.

See Figure 1b. The edges are as follows.

For every $i, j \in [d]$, and every $a_1 \in A_1, a_3 \in A_3$, add an edge between (a_1, x_i, x_j) and (a_3, x_i, x_j); we get bicliques in $X \times Y$ corresponding to each pair of coordinates i, j.

For each $(a_1, a_2) \in S$, we add an edge to $(a_1, x_i, x_j) \in X$ if and only if $a_1[i] = a_1[j] = 1$ and $a_2[i] = 1$. For each $(a_2, a_3) \in T$, we add an edge to $(a_3, x_i, x_j) \in Y$ if and only if $a_3[i] = a_3[j] = 1$ and $a_2[j] = 1$.

Suppose that there is no 3-OV solution. Then, for every $a_1 \in A_1, a_2 \in A_2, a_3 \in A_3$, there exists a coordinate k such that $a_1[k] = a_2[k] = a_3[k] = 1$. Consider an arbitrary $(a_1, a_2) \in S$ and $(a_2', a_3) \in T$. There is a coordinate i for which $a_1[i] = a_2[i] = a_3[i] = 1$ and a coordinate j for which $a_1[j] = a_2'[j] = a_3[j] = 1$. By construction, there is an edge between $(a_1, a_2) \in S$ and $(a_1, x_i, x_j) \in X$ and between $(a_2', a_3) \in T$ and $(a_3, x_i, x_j) \in Y$. Together with the edge between (a_1, x_i, x_j) and (a_3, x_i, x_j), we get that the distance between $(a_1, a_2) \in S$ and $(a_2', a_3) \in T$ is 3. Thus the S-T-Diameter is 3.

Suppose now that there is a 3-OV solution, $a_1 \in A_1, a_2 \in A_2, a_3 \in A_3$. Then one can show that if $d((a_1, a_2), (a_2, a_3)) \leq 5$, then there is a coordinate i such that $a_1[i] = a_2[i] = a_3[i] = 1$, giving a contradiction. Because the graph is bipartite, the distance must be ≥ 7, and we can conclude.

Thus, the S-T Diameter is 3 if there is no 3-OV solution or ≥ 7 if there is one. The number of vertices is $O(n^2 + nd^2)$ and the number of edges is $O(n^2d^2)$. Let $N = n^2d^2$. If there is an $O(N^{3/2-\varepsilon})$ time algorithm distinguishing 3 and 7 for $\varepsilon > 0$, then 3-OV can be solved in $n^{3-2\varepsilon}\text{poly}(d)$ time. □

Other known hardness results under SETH and k-OV. In recent years, there has been an explosion of conditional hardness results based on OV and hence SETH:

1. Tight lower bounds for approximating the Graph Diameter and Graph Eccentricities [135, 62, 14, 33].
2. Tight quadratic lower bounds for the Local Alignment problem [15].
3. Tight lower bounds for dynamic problems. The first comprehensive paper to consider multiple hardness hypotheses to explain the difficulty of dynamic problems was by Abboud and Vassilevska Williams [13]. Under SETH, the main hardness results concern the following dynamic problems: maintaining under edge insertions and deletions, the strongly connected components of a graph, the number of nodes reachable from a fixed source, a 1.3 approximation of the graph diameter, or given fixed node sets S and T, whether there is a pair of nodes $s \in S, t \in T$ so that s can reach t.
4. Strong hardness for the All Pairs Max Flow problem [107]: in n node, m edge graphs $mn^{2-o(1)}$ time is needed. Lower bounds from OV and from Max-CNF-SAT. These results are based on previous hardness for variants of the Max Flow problem under SETH by Abboud et al. [16].
5. Lower bounds for incremental and decremental Max-Flow [74] following [16] and [13]. This is among the handful of lower bounds that address *amortized* runtimes for partially dynamic algorithms. The prior techniques could only provide worst case lower bounds here.
6. Lower bounds for sensitivity problems. Sensitivity problems are similar to dynamic problems in that they need to preprocess the input and prepare a data structure that answers queries after some sequence of updates. The difference is that once the queries are answered, the updates must be rolled back to the original state of the input. That is, the sensitivity problem is to prepare for any set of small changes and be able to answer queries on them. [97] give lower bounds under SETH for sensitivity data structures for graph problems such as answering for any small (constant) size set of edge insertions, approximate Graph Diameter queries or queries about the number of nodes reachable from a fixed source node.
7. Closest Pair in d-dimensional Hamming Space cannot be solved in $n^{2-\varepsilon}2^{o(d)}$ time for $\varepsilon > 0$ [22]. The best algorithm for this problem and several others (e.g. offline bichromatic furthest neighbors) is by [21] and runs in $n^{2-1/O(c\log^2(c))}$ time for $d = c\log n$.
8. Quadratic lower bounds for LCS [4, 48], Edit Distance [30], Frechet Distance [43]. [4] also give an $n^{k-o(1)}$ lower bound for computing the LCS of k strings for any $k \geq 2$.
9. Tight lower bounds for problems like LCS and RNA-Folding where the input strings are represented as a context free grammar whose only output is the input string [1]. Some of the lower bounds are also based on Hypotheses about the complexity of k-Clique and k-SUM.
10. Subset Sum on n integers and target T, cannot be solved in $T^{1-\varepsilon}2^{o(n)}$ time for any $\varepsilon > 0$ [6]. Similar results apply to the Bicriteria Path problem.
11. Tight lower bounds for the Subtree Isomorphism problem: given rooted trees on n total nodes T and T', is T a subtree of T'? [2] show that truly subquadratic algorithms for the following refute the OV Hypothesis: for binary, rooted trees, or for rooted trees of depth $O(\log\log n)$. Conversely, for every constant d, there is a constant $\varepsilon_d > 0$ and a randomized, truly subquadratic algorithm for degree-d rooted trees of depth at most $(1+\varepsilon_d)\log_d n$.

12. Frechet distance on n-length strings requires $n^{2-o(1)}$ time [43], and is hard to approximate [47, 49].
13. Tight results for regular expression matching ([31, 45]): here one is given a pattern of length m, text of length n, and the pattern involves concatenation, OR, Kleene star and Kleene plus. Under SETH, there is a dichotomy of problems (proven for depth 2 by [31] and for > 2 by [45]): either they are solvable in near-linear time, or they require $mn^{1-o(1)}$ time. There is a single exception: the Word Break problem solvable in $\tilde{O}(m + nm^{1/3})$ time [45].
14. Tight lower bounds for problems in model checking: for Büchi objectives [59] and others [60].
15. Tight lower bounds for succinct stable matching [121].
16. Quadratic hardness results for problems in Machine Learning [32].
17. Tight hardness for some one dimensional Dynamic Programming problems [108].
18. Furthest pair in R^d (ℓ_2) on n vectors, when $d = \omega(\log \log n)$ requires $n^{2-o(1)}$ time [160]. This is to be contrasted with Closest Pair in the same dimensions which can be solved in $n^{1+o(1)}$ time.
19. Very strong inapproximability several problems via the introduction of *Distributed PCPs* for Fine-Grained Hardness of Approximation [12]: Bichromatic Max-Inner Product on N vectors in $\{0, 1\}^d$ cannot be approximated better than a factor of $2^{(\log N)^{1-o(1)}}$ if you do not spend $N^{2-o(1)}$ time. Similar inapproximability for approximation versions of Subset Query, Bichromatic LCS Closest Pair, Regular Expression Matching and Diameter in Product Metrics.

4 Hardness results from 3-SUM.

A seminal paper by Gajentaan and Overmars [87, 88] from the 1990s introduces the 3-SUM Hypothesis and proves that a large set of problems in computational geometry require quadratic time, under this hypothesis:

1. Given a set of points in the plane, decide whether any three are colinear (Colinearity/3 Points on Line).
2. Given a set of lines in the plane, decide whether any three of them pass through the same point (Point on 3 Lines).
3. Given a set of non-intersecting, axis-parallel line segments, decide whether some line separates them into two non-empty subsets (Separator).
4. Given a set of (infinite) strips in the plane and a rectangle, decide whether they fully cover the rectangle (Strips Cover Box).
5. Given a set of triangles in the plane, compute their measure (Triangle Measure).
6. Given a set of horizontal opaque triangles in three dimensional space, a view point p and another triangle T, decide whether there is a point on T that can be seen from p (Visible Triangle).
7. Given a set of non-intersecting, axis-parallel line segment obstacles in the plane, a rod and a source and a destination point, decide whether the rod can be moved by translations and rotations from the source to the destination without colliding with the obstacles (Planar Motion Planning).
8. Given a set of horizontal triangle obstacles in three dimensional space, a vertical rod, and a source and destination, decide whether the rod can be translated (without rotation) from the source to the destination without colliding with the obstacles (3D Motion Planning).

The notion of 3-SUM hardness reduction used in [87, 88] is more restrictive than the fine-grained reduction defined later on. It only allows the creation of $O(1)$ number of instances, each of no more than linear size. Even though the reduction notion is limited, it is still possible to obtain all of the above hardness results using more or less simple algebraic transformations. The paper inspired many other 3-SUM hardness results in computational geometry. Some of these include polygon containment [38], testing whether a dihedral rotation will cause a chain to self-intersect [139] and many others [76, 80, 28, 63, 42, 81, 26, 25, 18].

A transformative paper in 3-SUM research by Pătraşcu [131] shows that 3-SUM is equivalent (under subquadratic reductions) to a slightly simpler looking problem, 3-*SUM Convolution*: Given three length n arrays A, B and C of integers, decide whether there exist i, k such that $C[k] = A[i] + B[k - i]$.

Unlike for 3-SUM, $O(n^2)$ is the brute-force algorithm runtime for 3-SUM Convolution (for 3-SUM the trivial runtime is $O(n^3)$). This makes it easier to reduce 3-SUM Convolution to other problems whose best known algorithm is the brute-force one. Also, because now the search is reduced to finding two indices i, k, as opposed to searching for a sum of two integers, one can use 3-SUM Convolution in reductions to problems that are more combinatorial in nature. Pătraşcu reduces 3-SUM Convolution to problems such as Listing Triangles in a graph. He shows that listing up to m triangles in an m-edge graph requires $m^{4/3-o(1)}$ time under the 3-SUM Hypothesis. This is the first hardness result for a truly combinatorial problem (no numbers in the instance).

Prior to Pătraşcu's results, there is one other 3-SUM hardness result for a problem outside computational geometry, by Vassilevska and Williams [147]. They show that under the 3-SUM Hypothesis, the following *Exact Triangle* problem requires $n^{2.5-o(1)}$ time on an n node edge-weighted graph G: determine whether there is a triangle G whose edge weights sum to 0. Pătraşcu's equivalence between 3-SUM and 3-SUM Convolution allows this hardness to be improved to $n^{3-o(1)}$, thus showing that the brute-force cubic algorithm for the problem might be optimal [151].

After [131] and [151], many other combinatorial problems were proven to be 3-SUM hard: Abboud and Vassilevska Williams [13] continue Pătraşcu's work, giving lower bounds for many dynamic problems under the 3-SUM hypothesis. Example graph problems of consideration are to maintain under edge deletions and insertions: $s - t$ Reach (whether a given fixed source can reach a given fixed destination in a directed graph), SCC (the strongly connected components of a graph, or even just their number), BPMatch (whether a bipartite graph as a perfect matching), and many others. Kopelowitz et al. [106] improve Pătraşcu's reduction to triangle listing and show that the known algorithms for listing triangles in graphs [40] are optimal if $\omega = 2$ and under the 3-SUM Hypothesis. They also give amortized conditional lower bound for maintaining a maximum matching in a graph under edge insertions. [15] show that the Local Alignment requires quadratic time under 3-SUM. The following other problems are also known to be hard under 3-SUM: jumbled indexing [23], online pattern matching with gaps [24], partial matrix multiplication, and witness reporting versions of convolution problems [92], and others.

5 Hardness results from APSP.

APSP is now known to be equivalent to many other problems on n node graphs and $n \times n$ matrices so that either all these problems admit $O(n^{3-\varepsilon})$ time algorithms for $\varepsilon > 0$, or none of them do. A partial list of these equivalent problems is below. The main references are the original paper by Vassilevska Williams and Williams [148, 149] (bullets 1-9), and also [29] (bullet 9), [9] (bullets 10-12), and [115] (bullet 13).

1. The all-pairs shortest paths problem on weighted digraphs (APSP).
2. Detecting if an edge-weighted graph has a triangle of negative total edge weight (Negative Triangle).

3. Listing up to $n^{2.99}$ negative triangles in an edge-weighted graph (Triangle listing).
4. Finding a minimum weight cycle in a graph of non-negative edge weights (Shortest Cycle).
5. The replacement paths problem on weighted digraphs (RP).
6. Finding the second shortest simple path between two nodes in a weighted digraph (2nd Shortest Path).
7. Checking whether a given matrix defines a metric (Metricity).
8. Verifying a matrix product over the $(\min, +)$-semiring (Distance Product Verification).
9. Finding a maximum subarray in a given matrix (Max Subarray).
10. Finding the Median node of a weighted graph (Median).
11. Finding the Radius of a weighted graph (Radius).
12. Computing the Betweenness Centrality of a given node in a weighted graph (BC).
13. Computing the Wiener Index of a weighted graph (Wiener Index).

Some of the equivalences above have been strengthened to preserve sparsity [19, 115] and even the range of weights [134]. Beyond the above equivalences, there have been multiple APSP-*hardness* results. Computing the edit distance between two rooted ordered trees with nodes labeled from a fixed alphabet (Tree Edit Distance) [44] is known to require cubic time if APSP does. An equivalence with APSP is an open problem. [13] provided tight hardness for dynamic problems under the APSP Hypothesis. The main results are for Bipartite Maximum Weight Matching and s-t Shortest Path, showing that the trivial dynamic algorithms are optimal, unless APSP can be solved faster. For instance, any algorithm that can maintain the distance in a weighted graph between a fixed source node s and a fixed target t, while supporting edge deletions, must either perform $n^{3-o(1)}$ time preprocessing, or either the update or the query time must be $n^{2-o(1)}$ ([13] following [136]). Henzinger et al. [97] give tight lower bounds under the APSP Hypothesis for *sensitivity* problems such as answering Graph Diameter or $s-t$ Shortest Path queries for any single edge failure. Abboud and Dahlgaard [8] gave the first fine-grained lower bound for a problem in *planar* graphs: no algorithm for dynamic shortest paths or maximum weight bipartite matching in planar graphs can support both updates and queries in amortized $O(n^{1/2-\varepsilon})$ time, for any $\varepsilon > 0$, unless the APSP Hypothesis fails.

The main technical hurdle in showing the equivalences and most hardness results above, overcome by [148], is in reducing APSP to the Negative Triangle Problem. Negative Triangle is a simple decision problem, and reducing it to the other problems above is doable, with sufficient gadgetry.

Below we will outline the reduction *from APSP to Negative Triangle*. It is a true fine-grained reduction — it produces many instances, reducing a function problem to a decision problem.

The first step is to formulate APSP as a problem involving triangles, the *All-Pairs Negative Triangles (APNT)* problem defined as follows: given a tripartite graph G with node partitions R, T, C, with arbitrary edges in $R \times T, T \times C, C \times R$, with integer edge weights $w(\cdot)$, for every $i \in R, j \in C$, determine whether there exists a $t \in T$ so that $w(i,t) + w(t,j) + w(j,i) < 0$.

Reducing APSP to APNT (see [148]) is done by using a known equivalence [83] between APSP and the Distance Product problem of computing a product of two matrices over the $(\min, +)$ semiring. Then Distance Product is solved by using calls to APNT to binary search for the entries in the output matrix.

Now, it suffices to reduce the All-Pairs Negative Triangles problem to just detecting a Negative Triangle. The reduction picks a parameter $t = n^{2/3}$. Then, it arbitrarily partitions R, T, C into t pieces each of size roughly n/t: $(R_1, \ldots, R_t)$, $(T_1, \ldots, T_t)$, $(C_1, \ldots, C_t)$. Every negative triangle is in some triple (R_i, T_j, C_k). We will create Negative Triangle instances for these triples as follows. See Figure 2.

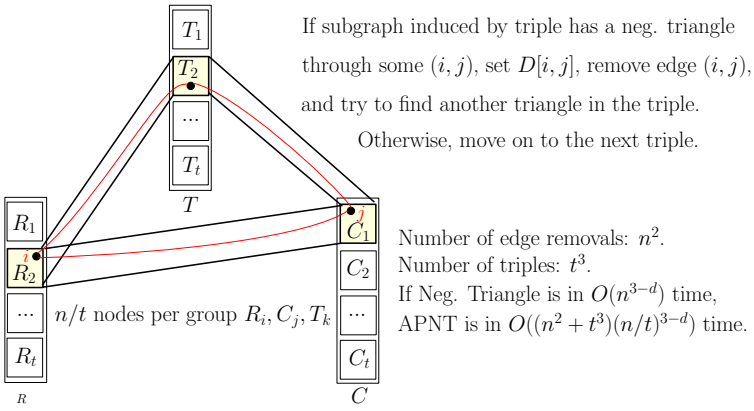


Figure 2

Create an $n \times n$ all 0 matrix D that at the end will have $D[i,j] = 1$ if and only if there is some $\ell \in T$ so that i, ℓ, j is a negative triangle.

Now, for every triple of parts (R_i, T_j, C_k) in turn, starting with (R_1, T_1, C_1), while the subgraph induced by the triple (R_i, T_j, C_k) contains a negative triangle (this is a call to Negative Triangle), find one such negative triangle $i \in R, \ell \in T, j \in C$ via self-reduction (see [148]). Set $D[i,j] = 1$ and remove (i,j) from the entire graph; we do this so that there will be no more negative triangles containing (i,j) in any of the subsequent Negative Triangle calls. Thus, the number of calls that do find a negative triangle is $\leq n^2$.

The algorithm keeps calling Negative Triangle on a triple until the triple has no more negative triangles, and then moves on to the next triple. At the end it just returns D. The number of calls to Negative Triangle that do not return a negative triangle are bounded by the number of triples which is t^3.

The number of calls to Negative Triangle is at most $t^3 + n^2$, and each call is on a graph with $O(n/t)$ nodes. As $t = n^{2/3}$, we get $O(n^2)$ calls to instances of size $O(n^{1/3})$, i.e. a subcubic fine-grained reduction.

6 Other hypotheses.

Beyond the three main hardness hypotheses, there are several other ones that have come to the forefront of fine-grained complexity. Again, the model of computation is the word-RAM with $O(\log n)$ bit words.

Hitting set. The *Hitting Set (HS)* problem is as follows: given two sets of vectors $S, T \subseteq \{0,1\}^d$ for $d = \omega(\log n)$, determine whether there is some $s \in S$ such that $s \cdot t \neq 0$ for all $t \in T$, in other words a vector in S that hits all vectors in T.

Hypothesis 5 (HS Hypothesis [14])**.** *No randomized algorithm can solve HS on n vectors in $\{0,1\}^d$ in $n^{2-\varepsilon}\text{poly}(d)$ time for $\varepsilon > 0$.*

To see why this new hypothesis is useful, consider the converse HS: decide whether $\forall s \in S\ \exists t \in T$ such that $s \cdot t = 0$. This is OV with the first $\exists$ quantifier replaced with $\forall$. This quantifier flip allows for different hardness results to be proven. For instance the Radius Problem asks, given a graph, whether there exists a vertex c such that for all other vertices v, $d(v,c)$ is most R. The converse asks whether $\forall c\ \exists v$ such that $d(v,c) > R$ which is the $\forall\exists$ variant of Diameter, which is $\exists\exists$. While it was not hard to reduce the $\exists\exists$ problem OV to Diameter, reducing it to the $\exists\forall$ Radius seemed problematic. On the other hand, since HS

has the $\forall\exists$ structure, [14] are able to reduce it Radius, so that Radius on sparse graphs requires $n^{2-o(1)}$ time under the HS Hypothesis. HS and its hypothesis are also studied in [90].

The HS Hypothesis implies the OV Hypothesis [14, 17], but the reverse is not known to be true.

Hypotheses on the complexity of k-Clique. For a constant $k \geq 3$, the *k-Clique* problem is as follows: given a graph $G = (V, E)$ on n vertices, does G contain k distinct vertices $a_1, \ldots, a_k$ so that for every i, j, $i \neq j$, $(a_i, a_j) \in E$? Such a k node graph is called a k-clique.

The k-Clique problem can easily be solved in $O(n^k)$ time by enumerating all k-tuples of vertices. A faster algorithm [102, 123] reduces the problem to multiplying square matrices, giving an $O(n^{\omega k/3}) \leq O(n^{0.8k})$ time algorithm when k is divisible by 3. Recall, $\omega < 2.373$ is the exponent of square matrix multiplication [150, 142, 111]. If k is not divisible by 3, the fastest known algorithm for k-Clique runs asymptotically in the time to multiply an $n^{\lfloor k/3 \rfloor} \times n^{\lceil k/3 \rceil}$ matrix by an $n^{\lceil k/3 \rceil} \times n^{k - \lfloor k/3 \rfloor - \lceil k/3 \rceil}$ matrix, which is no more than $O(n^{2+\omega k/3})$; tighter bounds are known ([110, 69, 89, 100]).

Hypothesis 6 (k-Clique Hypothesis)**.** *No randomized algorithm can detect a k-Clique in an n node graph in $O(n^{\frac{\omega k}{3} - \varepsilon})$ time for $\varepsilon > 0$.*

The Hypothesis is usually used for k divisible by 3. Also, since $\omega \geq 2$ (one needs to output a matrix with n^2 entries), the hypothesis asserts in particular that k-Clique requires $n^{2k/3 - o(1)}$ time.

Two harder problems are the *Min-Weight k-Clique* and *Exact k-Clique* problems. In both problems, one is given a graph on n vertices and edge weights in $\{-n^{100k}, \ldots, n^{100k}\}$. In the first, one seeks a k-Clique that minimizes the total sum of its edge weights. In the second, one seeks a k-Clique with weight sum of exactly 0. Neither of these problems are known to be solvable in $O(n^{k-\varepsilon})$ time for any constant $\varepsilon > 0$.

Hypothesis 7 (Min-Weight k-Clique Hypothesis)**.** *The Min-Weight k-Clique problem on n node graphs with edge weights in $\{-n^{100k}, \ldots, n^{100k}\}$ requires (randomized) $n^{k-o(1)}$ time.*

Hypothesis 8 (Exact k-Clique Hypothesis)**.** *The Exact k-Clique problem on n node graphs with edge weights in $\{-n^{100k}, \ldots, n^{100k}\}$ requires (randomized) $n^{k-o(1)}$ time.*

It is known that the Min-Weight k-Clique Hypothesis implies the Exact k-Clique Hypothesis [147]. The version of Min-Weight k-Clique in which the weights are on the nodes, rather than on the edges, can be solved in the same time as the (unweighted) k-Clique problem [73, 147, 11], so that the Min-Weight k-Clique Hypothesis does not hold for node-weighted graphs.

Using results from [11] and [157] and the known reduction from k-Clique to k-Dominating Set, one can show that Exact-Weight and Min-Weight k-Clique are (n^k, n^2)-reducible to 2-OV, and hence their hypotheses imply the OV Hypothesis [5].

Notably, the Min-Weight 3-Clique problem is equivalent to the Negative Triangle problem and hence also to APSP, under subcubic fine-grained reductions [148]. Exact 3-Clique is just the Exact Triangle problem studied by [151]. Exact 3-Clique seems genuinely more difficult than Min-Weight 3-Clique. First, the latter problem can be solved in $n^3/\exp(\sqrt{\log n})$ time [159], whereas the fastest algorithm for Exact 3-Clique runs in $n^3(\log\log n)^2/\log n$ time [104]. Second, as we mentioned in the section on 3-SUM, Exact 3-Clique requires $n^{3-o(1)}$ under both the 3-SUM and the APSP Hypotheses, whereas Min-Weight 3-Clique is equivalent to APSP which is not known to be related to 3-SUM and could be potentially easier.

The following tight lower bounds under the k-Clique Hypothesis are known: Context Free Grammar Recognition for $O(1)$ size grammars, RNA-Folding, and Language Edit Distance require $n^{\omega - o(1)}$ time [3, 58], and Tree-adjoining grammar parsing [50] requires the unusual running time of $n^{2\omega}$, tight due to [132].

The following problems have tight conditional lower bounds under the Min-Weight k-Clique Hypothesis: all problems hard under the APSP Hypothesis, the Local Alignment Problem [15], the Viterbi problem of finding the most likely path in a Hidden Markov Model (HMM) that results in a given sequence of observations [34], the Maximum Weight Box problem that given weighted points (positive or negative) in d dimensions, asks to find the axis-aligned box which maximizes the total weight of the points it contains [29].

Recently, the k-Clique and Min-Weight k-Clique Hypotheses have been used to show hardness for graph problems for almost all sparsities. Recall that under SETH one could show that many problems in very sparse graphs (with a near-linear number of edges) are hard. On the other hand, the APSP Hypothesis implied hardness for problems in dense graphs, i.e. when the runtime is measured solely in terms of the number of vertices. However, neither of these hypotheses seem to address questions such as "Can APSP be solved in $O(n^2 + m^{3/2})$ time?". Such a runtime would be consistent with the APSP Hypothesis and with the fact that in sparse graphs one needs $\Omega(n^2)$ time to write down the output.

For APSP and many other graph problems on m edges and n vertices, the best known running times are of the form $\tilde{O}(mn)$: APSP, Shortest Cycle, Replacement Paths, Radius, Wiener Index etc. There is no faster algorithm for any sparsity m. Lincoln et al. [115] address this by showing that for any constant $k \geq 1$, if one assumes the Min-Weight $2k + 1$-Clique Hypothesis, then APSP, Shortest Cycle, Replacement Paths, Radius, Wiener Index etc. require $mn^{1-o(1)}$ time in weighted graphs with $m = \Theta(n^{1+1/k})$ edges. In other words, for an infinite number of sparsities, mn is the right answer. Under the k-Clique Hypothesis, [115] provide weaker lower bounds for the same problems in unweighted graphs.

Boolean Matrix Multiplication (BMM). The BMM problem is, given two $n \times n$ matrices A and B, to compute the $n \times n$ matrix C with $C[i, j] = \vee_{i=1}^{n}(A[i, k] \wedge B[k, j])$ for all i, j. BMM can be solved using the known matrix multiplication algorithms over a field by embedding the Boolean semiring into the Rationals. Thus BMM on $n \times n$ matrices is in $O(n^{2.373})$ time [150, 111]. However, the theoretically fast algorithms for matrix multiplication are considered inefficient. The desire for more practical algorithms motivates the notion of "combinatorial" algorithms. This notion is not well-defined, however it roughly means that the runtime should have a small constant in the big-O, and that the algorithm is feasibly implementable.

There is a *"BMM hypothesis"* (in quotes as this is not well-defined) asserting that any combinatorial BMM algorithm requires $n^{3-o(1)}$ time. This is supported by the lack of truly subcubic combinatorial BMM algorithms: the fastest is by Yu [162] and runs in $n^3(\log \log n)^{O(1)}/\log^4 n$ time. The first combinatorial BMM algorithm is the so called Four-Russians algorithm [27], which was later improved by [55, 36, 162].

The BMM Hypothesis has been used to explain the lack of fast combinatorial algorithms for many problems: many dynamic problems [13, 136], Context Free Grammar Parsing [112], $2k$-Cycle in undirected graphs [75], etc. Also many fine-grained combinatorial equivalences to BMM are known (e.g. [148]).

Online Matrix Vector Multiplication (OMV). The BMM Hypothesis is unsatisfactory due to the undefined combinatorial notion, and there has been some work to replace it with something else. Henzinger et al. [96] define the Online Matrix Vector (OMV) hypothesis which makes the BMM hypothesis about an online version of the problem for which even non-combinatorial subcubic algorithms seem out of reach. The *OMV* problem is well-studied [154, 41, 109]: given an $n \times n$ Boolean matrix, preprocess it so that future products with arbitrary query $n \times 1$ vectors are efficient.

Hypothesis 9 (OMV Hypothesis)**.** *Every (randomized) algorithm that can process a given $n \times n$ Boolean matrix A, and then in an online way can compute the products Av_i for any n vectors $v_1, \ldots, v_n$, must take total time $n^{3-o(1)}$.*

The best algorithm for OMV is by Larsen and Williams [109] who show that the OMV problem (for n queries) can be solved in total time $n^3/\exp(\sqrt{\log n})$ via a reduction to the OV problem. Moreover, [109] give a cell probe algorithm that can solve the problem using $O(n^{11/4}/\sqrt{\log n})$ probes, thus ruling out an unconditional lower bound for OMV using purely information theoretic techniques.

The OMV Hypothesis is particular suited to proving conditional lower bounds for dynamic problems. Such lower bounds are known for practically all dynamic problems for which there is a known BMM-based combinatorial lower bound [96], and many other problems (e.g. [74]).

Nondeterministic Strong Exponential Time Hypothesis (NSETH). Surprisingly, the fastest algorithm for CNF-SAT on formulas on n variables, even using *nondeterminism*, still runs in roughly 2^n time. This motivated Carmosino et al. [53] to define the following.

Hypothesis 10 (Nondeterministic Strong Exponential Time Hypothesis (NSETH))**.** *Refuting unsatisfiable k-CNF formulas on n variables requires nondeterministic $2^{n-o(n)}$ time for unbounded k.*

It is worth noting that NSETH does not allow randomization. In early work, Carmosino et al. also proposed a Merlin-Arthur and Arthur-Merlin SETH that assert that no constant round probabilistic proof system can refute unsatisfiable k-CNF formulas in $2^{n-\Omega(n)}$ time. Williams [156] shows that these hypotheses are false in a very strong way, exhibiting proof systems that prove that the number of satisfying assignments of any given $o(n)$-depth, bounded-fan-in circuit is a given value, using a proof of length $2^{n/2}\text{poly}(n)$ that can be verified in $2^{n/2}\text{poly}(n)$ time with high probability, using only $O(n)$ random bits.

The fact that the AM and MA versions of NSETH are false casts doubt on the veracity of NSETH. Nevertheless, disproving NSETH seems challenging. Assuming NSETH, [53] prove that there can be no deterministic reduction from OV to 3-SUM or APSP. This is done by exhibiting fast nondeterministic algorithms for the latter two problems, whereas OV cannot have a nontrivial nondeterministic refutation algorithm, under NSETH, via Williams' [157] reduction from CNF-SAT to OV that we presented earlier.

SETH for other Circuit Satisfiability Problems. As we mentioned in the introduction, CNF places a restriction on the input of the more general SAT problem. When represented as a circuit, a k-CNF formula has depth two — it is an AND of ORs. Moreover, due to the Sparsification Lemma of [101], SETH really concerns the satisfiability problem for depth two circuits of $O(n)$ size. To make SETH more believable, we can instead consider the satisfiability of less restricted classes of inputs to Circuit SAT.

Consider a Boolean function f on n bit inputs for which we want to prove satisfiability. It is not hard to see that any algorithm whose only access to f is by querying the value of f on various inputs, must spend $\Omega(2^n)$ time to check if there is an n-bit x for which $f(x) = 1$. A clever algorithm would do more than query the function. It would attempt to analyze f to decrease the runtime of SAT. How much power algorithms have to analyze f depends crucially on the representation of f. It is impossible for a black box representation, and it is quite trivial if f is given as a DNF formula (ORs of ANDs of literals).

For each class C of representations, we can define the corresponding *C-SETH* that states that SAT with a representation from C cannot be solved in $O(2^{(1-\varepsilon)n})$ time for $\varepsilon > 0$.

C-SETH for k-CNF Formulas as k grows is just SETH. NC-SETH on the other hand asserts that SAT of polynomial size, polylogarithmic depth circuits requires $2^{n-o(n)}$ time. NC circuits are much more powerful than CNF Formulas. They can perform most linear algebraic operations, and they can implement cryptographic primitives like One Way Functions and Pseudorandom Generators, for which the ability to hide satisfiability is crucial.

C-SETH was defined by Abboud et al. [10] who gave fine-grained lower bounds for sequence alignment problems such as Edit Distance, Frechet Distance, LCS. For instance, these problems on n length sequences require $n^{2-o(1)}$ time unless NC-SETH fails, thus replacing the prior SETH hardness results with NC-SETH hardness. The results of [10] also imply that a truly subquadratic algorithm for any of these problems would imply novel circuit lower bounds for classes such as E^{NP}. More surprisingly, if these problems can be solved in $O(n^2/\log^c n)$ time for all constants c, then NTIME[$2^{O(n)}$] does not have non-uniform polynomial-size log-depth circuits. Hence, shaving all polylogs over the textbook quadratic runtime would result in a major advance in complexity theory.

Two Problems Harder than CNF-SAT, 3-SUM and APSP. The search for more believable hypotheses than SETH, and the APSP and 3-SUM Hypotheses motivates the following more believable conjecture: "At least one of SETH, the APSP Hypothesis and the 3-SUM Hypothesis is true."

To prove hardness under this conjecture, one would have to perform three reductions (from k-SAT, APSP and from 3-SUM) instead of just one; this can be cumbersome. It is also not apriori clear that any natural problems are hard under all three conjectures. Abboud et al. [16] define two simple combinatorial problems and reduce k-SAT, APSP and 3-SUM to them. The goal is then to use these as the basis of hardness.

The first problem is *Triangle Collection*: Given a graph $G = (V, E)$ with node colors $c : V \to \{1, \ldots, n\}$, decide whether there exist colors $c_1, c_2, c_3 \in \{1, \ldots, n\}$ such that there are NO triangles u, v, w in G (i.e. $(u, v), (v, w), (w, u) \in E$), such that $c(u) = c_1, c(v) = c_2, c(w) = c_3$.

The second problem is *Matching Triangles*: Given a graph $G = (V, E)$ with node colors $c : V \to \{1, \ldots, n\}$ and an integer Δ, decide whether there exist colors $c_1, c_2, c_3 \in \{1, \ldots, n\}$ such that there are at least Δ triangles u, v, w in G (i.e. $(u, v), (v, w), (w, u) \in E$), such that $c(u) = c_1, c(v) = c_2, c(w) = c_3$.

Abboud et al. [16] show that if either Triangle Collection or Matching Triangles on n node graphs admit an $O(n^{3-\varepsilon})$ time algorithm for any $\varepsilon > 0$, then all three of SETH, the APSP Hypothesis and the 3-SUM Hypothesis are false. In fact, this is true for a restricted version Triangle Collection* of Triangle Collection that is easier to work with, and [16] give conditional hardness under it for several dynamic graph problems under edge insertions and deletions such dynamic Max Flow, or maintaining the number of nodes reachable from a fixed source node. Dahlgaard [74] gave conditional lower bounds for approximating the graph Diameter both statically and dynamically, under the Triangle Collection* hypothesis. Hence many problems are known to be difficult under all three main hypotheses.

7 Further Applications of Fine-grained Complexity.

The fine-grained approach has found applications in many other areas of TCS:

- **FPT in P.** Parameterized complexity strives to classify problems according to their time complexity as a function of multiple parameters of the input or output. This is a different way to classify problems on a finer scale. It is particularly interesting for NP-hard problems.

  A problem is FPT with respect to a set of parameters if it can be solved in time $f(k_1, \ldots, k_t)\mathrm{poly}(n)$ on inputs of size n and parameters set to $k_1, \ldots, k_t$; here f can be any computable function. FPT problems can be solved in polynomial time when the parameters are constant; this can often make NP-hard FPT problems tractable. Parameterized complexity has identified many problems that are FPT and has developed a theory to explain which problems are likely not to be FPT (see e.g. [84, 72]).

  Abboud et al. [14] consider a notion of FPT for polynomial time problems: Fixed Parameter Subquadratic (FPS)– parameterized problems that admit algorithms running in time $f(k)n^{2-\varepsilon}$ for $\varepsilon > 0$

on inputs of size n and parameter(s) set to k, for some computable function f. [14] show that the Diameter and Radius problems in graphs with parameter treewidth are FPS. They also give conditional lower bounds on the function f for these problems. Their work was continued by Fomin et al. [85] who added fixed parameter results for other polynomial time problems. Notice that parameterized algorithms have long been used within Algorithms: e.g. for graph problems, runtimes are often measured in terms of both the number of edges and the number of vertices. Abboud et al. [14] are the first to give fine-grained conditional lower bounds for parameterized polynomial time solvable problems.

- **Unconditional CONGEST Lower Bounds.** Abboud et al. [7] consider the CONGEST model in distributed computing in which processors are n nodes in a graph, and computation proceeds in rounds in which every processor can send $O(\log n)$ bits of information to all adjacent processors. [7] (see also [46]) show how to convert some conditional lower bounds based on the OV Hypothesis to *unconditional* lower bounds in the CONGEST model. For instance, they show that in the CONGEST model, any algorithm that can compute a $3/2-\varepsilon$ approximation to the diameter of the graph of a $5/3-\varepsilon$ approximation to the eccentricities for any $\varepsilon > 0$ needs $\Omega(n)$ rounds of communication.

  The basic idea in [7] is that OV is equivalent to Set Disjointness which has an unconditional $\Theta(n)$ lower bound in communication complexity. The proofs show that any Diameter or Eccentricities protocol that takes too few rounds is solving Set Disjointness with too little communication.

- **Fine-Grained Cryptography.** Two papers begin the study of creating cryptographic primitives from fine-grained assumptions. Degwekar et al. [77] develop cryptographic protocols secure against adversaries that are at most as powerful as low circuit classes within P such as NC_1 — this is more fine-grained but does not address runtime. More recently, [35], provide several problems that are provably hard on average, under SETH or the 3-SUM or APSP Hypotheses. Then they use these problems to construct a Proof of Work scheme. They leave as an open problem to develop more cryptographic primitives, such as One Way Functions, from fine-grained assumptions.

- **Fine-Grained Time/Space Tradeoffs for Algorithms.** Besides considering the runtime as the main measure of complexity, one can also consider the space usage. Lincoln et al. [114] study the time/space tradeoffs of 3-SUM, building on prior work by Wang [153]. Besides developing new algorithms, [114] show that the 3-SUM hypothesis is equivalent to the following hypothesis: *There is some $\delta > 0$, such that every algorithm that uses $O(n^{0.5+\delta})$ space, needs $n^{2-o(1)}$ time to solve* 3*-SUM.*

  This makes the 3-SUM Hypothesis look even more plausible as it only applies to space bounded algorithms. Also, one might conceivably be able to prove it unconditionally: restricting the space usage has been sufficient to prove unconditional lower bounds for SAT, among other problems [155].

- **Fine-Grained Time/Space Tradeoffs for Data Structures.** Goldstein et al. [93] define various data structure variants of 3-SUM, BMM and Directed Reachability, formulate novel conjectures and show consequences for the time/space tradeoffs for various data structure problems.

- **Fine-Grained Complexity in the I/O Model.** Demaine et al. [78] initiate the study of the I/O model from the perspective of fine-grained complexity. The paper proposes plausible I/O hardness hypotheses, and uses these, together with fine-grained I/O reductions, to show that many known I/O upper bounds are tight. For instance, the best known upper bound on the I/O complexity of LCS is tight under one of the assumptions. Finally, they prove an analogue of the Time Hierarchy Theorem in the I/O model.

Fine-grained complexity is a growing field and we hope that its ideas will spread to many other parts of TCS and beyond.

Acknowledgments. The author would like to thank Ryan Williams for useful comments and Erik Demaine and Timothy M. Chan for pointers for references on 3-SUM hard problems.

References

[1] Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 192–203, 2017.

[2] Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1256–1271, 2016.

[3] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 98–117, 2015.

[4] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.

[5] Amir Abboud, Karl Bringmann, Holger Dell, and Jesper Nederlof. Personal communication, 2017.

[6] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. *CoRR*, abs/1704.04546, 2017.

[7] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. Near-linear lower bounds for distributed distance computations, even in sparse networks. In *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, pages 29–42, 2016.

[8] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 477–486, 2016.

[9] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1681–1697, 2015.

[10] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016.

[11] Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 1–12, 2014.

[12] Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36, 2017.

[13] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

[14] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. 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, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391, 2016.

[15] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 39–51, 2014.

[16] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 41–50, 2015.

[17] Amir Abboud, Richard Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 218–230, 2015.

[18] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. Smallest color-spanning objects. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 278–289, 2001.

[19] Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity and conditional hardness for sparse graphs. *CoRR*, abs/1611.07008, 2016.

[20] Parag Agrawal, Arvind Arasu, and Raghav Kaushik. On indexing error-tolerant set containment. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 927–938, 2010.

[21] 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.

[22] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150, 2015.

[23] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 114–125, 2014.

[24] Amihood Amir, Tsvi Kopelowitz, Avivit Levy, Seth Pettie, Ely Porat, and B. Riva Shalom. Mind the gap: Essentially optimal algorithms for online dictionary matching with one gap. In *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, pages 12:1–12:12, 2016.

[25] Daniel Archambault, William S. Evans, and David G. Kirkpatrick. Computing the set of all the distant horizons of a terrain. *Int. J. Comput. Geometry Appl.*, 15(6):547–564, 2005.

[26] Esther M. Arkin, Yi-Jen Chiang, Martin Held, Joseph S. B. Mitchell, Vera Sacristán, Steven Skiena, and Tae-Heng Yang. On minimum-area hulls. *Algorithmica*, 21(1):119–136, 1998.

[27] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradzev. On economical construction of the transitive closure of an oriented graph. *Soviet Math. Dokl.*, 11:1209–1210, 1970.

[28] Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.

[29] Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 81:1–81:13, 2016.

[30] 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.

[31] Arturs Backurs and Piotr Indyk. Which regular expression patterns are hard to match? In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 457–466, 2016.

[32] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4311–4321, 2017.

[33] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Unpublished Manuscript, submitted*, 2018.

[34] Arturs Backurs and Christos Tzamos. Improving viterbi is hard: Better runtimes imply faster clique algorithms. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 311–321, 2017.

[35] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 483–496, 2017.

[36] N. Bansal and R. Williams. Regularity lemmas and combinatorial algorithms. In *Proc. FOCS*, pages 745–754, 2009.

[37] I. Baran, E.D. Demaine, and M. Pătraşcu. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, 2008.

[38] Gill Barequet and Sariel Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3SUM-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

[39] P. Bille and M. Farach-Colton. Fast and compact regular expression matching. *Theoretical Computer Science,*, 409(3):486–496, 2008.

[40] Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 223–234, 2014.

[41] G. Blelloch, V. Vassilevska, and R. Williams. A new combinatorial approach to sparse graph problems. In *Proc. ICALP*, pages 108–120, 2008.

[42] Prosenjit Bose, Marc J. van Kreveld, and Godfried T. Toussaint. Filling polyhedral molds. *Computer-Aided Design*, 30(4):245–254, 1998.

[43] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014.

[44] Karl Bringmann, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Tree edit distance cannot be computed in strongly subcubic time (unless APSP can). *CoRR*, abs/1703.08940, 2017.

[45] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. A dichotomy for regular expression membership testing. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 307–318, 2017.

[46] Karl Bringmann and Sebastian Krinninger. Brief announcement: A note on hardness of diameter approximation. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 44:1–44:3, 2017.

[47] Karl Bringmann and Marvin Künnemann. Improved approximation for fréchet distance on c-packed curves matching conditional lower bounds. In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 517–528, 2015.

[48] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 79–97, 2015.

[49] Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete fréchet distance. *JoCG*, 7(2):46–76, 2016.

[50] Karl Bringmann and Philip Wellnitz. Clique-based lower bounds for parsing tree-adjoining grammars. In *28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017, July 4-6, 2017, Warsaw, Poland*, pages 12:1–12:14, 2017.

[51] Samuel R. Buss and Ryan Williams. Limits on alternation-trading proofs for time-space lower bounds. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 181–191, 2012.

[52] Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New bounds for approximating extremal distances in undirected graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376, 2016.

[53] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 261–270, 2016.

[54] Timothy M. Chan. All-pairs shortest paths for unweighted undirected graphs in *o(mn)* time. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 514–523, 2006.

[55] Timothy M. Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 212–217, 2015.

[56] Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median,+)-convolution, and some geometric 3sum-hard problems. In *Proceedings of SODA 2018*, 2018. to appear.

[57] Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016.

[58] Yi-Jun Chang. Hardness of RNA folding problem with four symbols. In *27th Annual Symposium on Combinatorial Pattern Matching, CPM 2016, June 27-29, 2016, Tel Aviv, Israel*, pages 13:1–13:12, 2016.

[59] Krishnendu Chatterjee, Wolfgang Dvorák, Monika Henzinger, and Veronika Loitzenbauer. Conditionally optimal algorithms for generalized büchi games. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, pages 25:1–25:15, 2016.

[60] Krishnendu Chatterjee, Wolfgang Dvorák, Monika Henzinger, and Veronika Loitzenbauer. Model and objective separation with conditional lower bounds: Disjunction is harder than conjunction. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 197–206, 2016.

[61] Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993.

[62] Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1041–1052, 2014.

[63] Otfried Cheong, Alon Efrat, and Sariel Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete & Computational Geometry*, 37(4):545–563, 2007.

[64] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 273–282, 2011.

[65] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 410–419, 2017.

[66] Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 9–21, 2016.

[67] Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{o}$ ($m^{10/7}$ log W) time (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 752–771, 2017.

[68] Michael B. Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 902–913, 2017.

[69] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Complexity*, 13:42–49, 1997.

[70] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of baur-strassen's theorem: Shortest cycles, diameter and matchings. In *Proc. FOCS*, 2012.

[71] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as cnf-sat. *ACM Trans. Algorithms*, 12(3):41:1–41:24, May 2016.

[72] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[73] A. Czumaj and A. Lingas. Finding a heaviest triangle is not harder than matrix multiplication. In *Proc. SODA*, pages 986–994, 2007.

[74] Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 48:1–48:14, 2016.

[75] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Morten Stöckel. Finding even cycles faster via capped k-walks. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 112–120, 2017.

[76] Mark de Berg, Marko de Groot, and Mark H. Overmars. Perfect binary space partitions. *Comput. Geom.*, 7:81–91, 1997.

[77] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 533–562, 2016.

[78] Erik D. Demaine, Andrea Lincoln, Quanquan C. Liu, Jayson Lynch, and Virginia Vassilevska Williams. Fine-grained i/o complexity via reductions: New lower bounds, faster algorithms, and a time hierarchy. In *Proc. ITCS*, page to appear, 2018.

[79] Jeff Erickson. Lower bounds for linear satisfiability problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1995. San Francisco, California.*, pages 388–395, 1995.

[80] Jeff Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28(4):1198–1214, 1999.

[81] Jeff Erickson, Sariel Har-Peled, and David M. Mount. On the least median square problem. *Discrete & Computational Geometry*, 36(4):593–607, 2006.

[82] Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete & Computational Geometry*, 13:41–57, 1995.

[83] M. J. Fischer and A. R. Meyer. Boolean matrix multiplication and transitive closure. In *Proc. FOCS*, pages 129–131, 1971.

[84] J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[85] Fedor V. Fomin, Daniel Lokshtanov, Michal Pilipczuk, Saket Saurabh, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1419–1432, 2017.

[86] M.L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5:49–60, 1976.

[87] A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

[88] Anka Gajentaan and Mark H. Overmars. On a class of o(n^2) problems in computational geometry. *Comput. Geom.*, 45(4):140–152, 2012.

[89] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proc. SODA*, page to appear, 2018.

[90] 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.

[91] Ashish Goel and Pankaj Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. In *SIGMETRICS 2010, Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 14-18 June 2010*, pages 143–154, 2010.

[92] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 45:1–45:16, 2016.

[93] Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. Conditional lower bounds for space/time tradeoffs. In *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, pages 421–436, 2017.

[94] Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. In *Stringology*, pages 202–211, 2014.

[95] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285—-306, 1965.

[96] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 21–30, 2015.

[97] Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional hardness for sensitivity problems. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 26:1–26:31, 2017.

[98] E. A. Hirsch. Two new upper bounds for SAT. In *Proc. SODA*, pages 521–530, 1998.

[99] John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[100] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *J. of Complexity*, 14(2):257–299, 1998.

[101] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[102] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM J. Computing*, 7(4):413–423, 1978.

[103] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, 1977.

[104] Allan Grønlund Jørgensen and Seth Pettie. Threesomes, degenerates, and love triangles. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 621–630, 2014.

[105] Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. *CoRR*, abs/1705.01720, 2017.

[106] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287, 2016.

[107] Robert Krauthgamer and Ohad Trabelsi. Conditional lower bounds for all-pairs max-flow. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 20:1–20:13, 2017.

[108] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 21:1–21:15, 2017.

[109] Kasper Green Larsen and R. Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2182–2189, 2017.

[110] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 514–523, 2012.

[111] François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

[112] Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, 2002.

[113] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014.

[114] Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-SUM. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 58:1–58:14, 2016.

[115] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. SODA*, page to appear, 2018.

[116] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 253–262, 2013.

[117] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 593–602, 2016.

[118] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18 – 31, 1980.

[119] Jiří Matoušek. Range searching with efficient hiearchical cutting. *Discrete & Computational Geometry*, 10:157–182, 1993.

[120] Sergey Melnik and Hector Garcia-Molina. Adaptive algorithms for set containment joins. *ACM Trans. Database Syst.*, 28:56–99, 2003.

[121] Daniel Moeller, Ramamohan Paturi, and Stefan Schneider. Subquadratic algorithms for succinct stable matching. In *Computer Science - Theory and Applications - 11th International Computer Science Symposium in Russia, CSR 2016, St. Petersburg, Russia, June 9-13, 2016, Proceedings*, pages 294–308, 2016.

[122] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287 – 295, 1985.

[123] J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Math. Universitatis Carolinae*, 26(2):415–419, 1985.

[124] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[125] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005.

[126] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[127] S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004.

[128] Seth Pettie. All pairs shortest paths in sparse graphs. In *Encyclopedia of Algorithms*. 2008.

[129] Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

[130] M. Pătraşcu and R. Williams. On the possibility of faster SAT algorithms. In *Proc. SODA*, pages 1065–1075, 2010.

[131] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610, 2010.

[132] Sanguthevar Rajasekaran and Shibu Yooseph. TAL recognition in O(M(n^2)) time. *J. Comput. Syst. Sci.*, 56(1):83–89, 1998.

[133] Karthikeyan Ramasamy, Jignesh M. Patel, Jeffrey F. Naughton, and Raghav Kaushik. Set containment joins: The good, the bad and the ugly. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 351–362, 2000.

[134] Liam Roditty and Virginia Vassilevska Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 180–189, 2011.

[135] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, STOC '13, pages 515–524, 2013.

[136] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *Algorithms - ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings*, pages 580–591, 2004.

[137] I. Schiermeyer. Solving 3-satisfiability in less then 1.579^n steps. In *CSL*, pages 379–394, 1992.

[138] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proc. FOCS*, pages 410–414, 1999.

[139] Michael A. Soss, Jeff Erickson, and Mark H. Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Comput. Geom.*, 26(3):235–246, 2003.

[140] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90, 2004.

[141] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Analysis Applications*, 35(3):835–885, 2014.

[142] A. Stothers. On the complexity of matrix multiplication. *Ph.D. Thesis, U. Edinburgh*, 2010.

[143] Robert Endre Tarjan. Depth-first search and linear graph algorithms (working paper). In *12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971*, pages 114–121, 1971.

[144] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[145] Robert Endre Tarjan. Testing graph connectivity. In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA*, pages 185–193, 1974.

[146] P. van Emde Boas. Machine models and simulations. In J. van Leeuwen, editor, *The Handbook of Theoretical Computer Science, vol. I: Algorithms and Complexity*, chapter 1, pages 1–61. MIT Press, Cambridge, Massachusetts, 1990.

[147] V. Vassilevska and R. Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. STOC*, pages 455–464, 2009.

[148] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. FOCS*, pages 645–654, 2010.

[149] V. Vassilevska Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. *Journal of the ACM*, 2018. to appear.

[150] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898, 2012.

[151] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

[152] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.

[153] Joshua R. Wang. Space-efficient randomized algorithms for K-SUM. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 810–829, 2014.

[154] R. Williams. Matrix-vector multiplication in sub-quadratic time (some preprocessing required). In *Proc. SODA*, pages 995–1001, 2007.

[155] R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity*, 17(2):179–219, 2008.

[156] Richard 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.

[157] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.

[158] Ryan Williams. *Algorithms and Resource Requirements for Fundamental Problems*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2007. CMU-CS-07-147.

[159] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673, 2014.

[160] Ryan Williams. On the complexity of furthest, closest, and orthogonality problems in low dimensions. In *Proc. SODA 2018*, 2018. to appear.

[161] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1867–1877, 2014.

[162] Huacheng Yu. An improved combinatorial algorithm for boolean matrix multiplication. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 1094–1105, 2015.