Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis

Virginia V. Williams[*]

Stanford University, Computer Science Department, Stanford, CA, USA

Abstract

Algorithmic research strives to develop fast algorithms for fundamental problems. Despite its many successes, however, many problems still do not have very efficient algorithms. For years researchers have explained the hardness for key problems by proving NP-hardness, utilizing polynomial time reductions to base the hardness of key problems on the famous conjecture $\mathrm{P} \neq \mathrm{NP}$. For problems that already have polynomial time algorithms, however, it does not seem that one can show any sort of hardness based on $\mathrm{P} \neq \mathrm{NP}$. Nevertheless, we would like to provide evidence that a problem A with a running time $O(n^k)$ that has not been improved in decades, also *requires* $n^{k-o(1)}$ time, thus explaining the lack of progress on the problem. Such unconditional time lower bounds seem very difficult to obtain, unfortunately. Recent work has concentrated on an approach mimicking NP-hardness: (1) select a few key problems that are conjectured to require $T(n)$ time to solve, (2) use special, fine-grained reductions to prove time lower bounds for many diverse problems in P based on the conjectured hardness of the key problems. In this abstract we outline the approach, give some examples of hardness results based on the Strong Exponential Time Hypothesis, and present an overview of some of the recent work on the topic.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.1.3 Complexity Measures and Classes

Keywords and phrases reductions, satisfiability, strong exponential time hypothesis, shortest paths, 3SUM, equivalences, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.IPEC.2015.17

Category Invited Paper

1 Introduction

The core goal of algorithmic research is to determine how fast computational problems can be solved in the worst case. Important time hierarchy theorems from complexity theory imply that (for natural models of computation) for all time-constructible functions $T(n)$, there are problems that can be solved in $T(n)$ time but not in $O(T(n)^{1-\varepsilon})$ time for $\varepsilon > 0$. The main challenge is to determine where in this hierarchy various natural and fundamental problems lie. Throughout the years, many ingenious algorithmic techniques have been developed and applied to obtain blazingly fast algorithms for many important problems. For instance, many useful graph problems can be solved in near-linear time (*e.g.*, finding connected components or computing single source shortest paths). Recent breakthroughs have shown that problems like graph matching, linear programming and max flow have surprisingly fast algorithms as

[*] This work was partially supported by NSF Grants CCF-1417238 and CCF-1514339, and BSF Grant BSF:2012338.

well ([46, 42, 43]). Nevertheless, for many other central problems the best known running times are essentially those of the classical algorithms devised for them in the 1950s and 1960s.

A prominent example is the CNF-SAT problem: given a boolean formula on n variables in conjunctive normal form, determine whether it has a satisfying assignment. The naïve algorithm for the problem is to try all possible 2^n assignments to the variables and check whether they satisfy the clause. This algorithm runs in $O^*(2^n)$ time, where O^* hides polynomial factors in the number of variables and clauses. When the maximum clause length is a constant k, CNF-SAT is called k-SAT. This problem can be solved in $O^*(2^{n-cn/k})$ for various constants c independent of n and k (*e.g.*, [36, 48, 51, 50, 56, 57]), thus improving over the 2^n running time exponentially. Nevertheless, as k grows, this running time approaches 2^n. Thus the naïve algorithm is essentially the best known algorithm for CNF-SAT even when the clause length is (an arbitrary) constant.

The theory of NP-hardness has been the main tool for explaining why problems such as CNF-SAT are hard. The seminal result that k-SAT is NP-complete for all $k \geq 3$ [41] implies that if we believe that $P \neq NP$, k-SAT cannot have a polynomial time algorithm. NP-hardness however, does not say anything about running times that are not polynomial, and it seems difficult, if not impossible, to show that the 2^n time algorithm for CNF-SAT is optimal, assuming only $P \neq NP$. The optimality of the 2^n time algorithm for CNF-SAT is currently only a conjecture, known as the Strong Exponential Time Hypothesis (SETH).

The type of hardness that SETH asserts about CNF-SAT is not unique to NP-hard problems. Although for the purposes of complexity theory, problems in the class P are considered "easy", this is not because anyone believes that running times such as $O(n^{100})$ or even $O(n^3)$ are efficient. P was initially studied mainly because polynomials have useful properties, *e.g.*, they are closed under composition and also polynomial running times allow us model independence. It became interesting to distinguish the problems in P from those that require superpolynomial time, and questions such as P vs NP emerged.

There are many important problems within P that have (often brute-force) classical algorithms running in $\tilde{O}(n^k)$ time for some constant k,[1] but whose running time has not been improved upon except (possibly) by $n^{o(1)}$ factors. Some prominent examples of such problems from different areas of computer science include: (1) from graph algorithms: the center of a graph G, *i.e.*, $\arg\min_{v\in G}\max_{x\in G}\operatorname{dist}(v,x)$, can be computed in $O(n^3)$ time using Floyd–Warshall's classical algorithm for All-Pairs Shortest Paths, and no faster algorithm is known; (2) from computational biology: given two length n DNA sequences, their longest common subsequence (LCS) can be computed in $O(n^2)$ time using a classical dynamic programming algorithm, and the fastest known algorithm runs faster but only by logarithmic factors [47, 16, 35]; (3) from computational geometry, given n points in the plane, one can determine whether they are in general position with a simple classical algorithm in $\tilde{O}(n^2)$ time, and this is the best known, up to $n^{o(1)}$ factors.

Unconditional lower bounds seem very difficult to obtain – it is not even known whether SAT can be solved in linear time. Mimicking NP-hardness, we would like to give evidence that for problems like the above, the classical algorithms are probably optimal, and that the polynomial time solvable problem is "hard". NP-hardness itself seems to have little use in showing that problems that already have polynomial time algorithms are hard. Instead, a more *fine-grained* approach has been used in recent years. In this approach, we pick a widely believed hypothesis about the time complexity of a *key problem*, and then use *fine-grained* reductions to reduce this key problem to other important problems, giving conditional lower bounds on how fast these problems can be solved.

[1] Here, $\tilde{O}$ hides $n^{o(1)}$ factors.

2 The key problems

The majority of conditional hardness results for problems within P are based on the conjectured hardness of three problems: 3SUM, All-Pairs Shortest Paths, and CNF-SAT. Since we are focusing on exact running times, we need to fix the model of computation. Here we assume that we are working with a Word RAM model with $O(\log n)$ bit words.

2.1 3SUM

The *3SUM* problem asks to determine whether a given set of n integers contains three integers a, b, c so that $a+b=c$. The problem has a simple $\tilde{O}(n^2)$ time algorithm: sort the integers, and for every pair a, b, check whether their sum is in the list. Baran, Demaine and Pătraşcu [14] showed that in the Word RAM model with $O(\log n)$ bit words, 3SUM can be solved in $O(n^3(\log\log n)^2/\log^2 n)$ time, thus obtaining a speed-up[2]. Chan and Lewenstein [20] showed that some interesting structured instances of 3SUM *can* be solved in $O(n^{1.9})$ time. However, there are no known $O(n^{2-\varepsilon})$ time (so-called "truly subquadratic") algorithms for the general problem for any $\varepsilon > 0$, even when randomization can be used. The lack of progress on the problem has led to the following conjecture [53, 32].

▶ **Conjecture 1** (No truly subquadratic 3SUM). In the Word RAM model with $O(\log n)$ bit words, any algorithm requires $n^{2-o(1)}$ time in expectation to determine whether a set $S \subset \{-n^3, \dots, n^3\}$ of size n contains three distinct elements $a, b, c \in S$ with $a + b = c$.

(By standard hashing arguments, one can assume that the size of the integers in the 3SUM instance is $O(n^3)$, and so the conjecture is not for a restricted version of the problem.)

The 3SUM conjecture is widely believed, especially in computational geometry. In 1995, Gajentaan and Overmars [32] formed a theory of "3SUM-hard problems" by showing that one can reduce 3SUM to many problems in computational geometry and that the conjecture above implies that none of these problems have truly subquadratic algorithms. One example of a 3SUM-hard problem is the planar points in general position problem that we mentioned earlier. Following [32] many other papers proved the 3SUM hardness of geometric problems, *e.g.*, [28, 45, 29, 10, 30, 12, 22, 15]. Vassilevska Williams and Williams [58, 59] showed that a certain weighted graph triangle problem cannot be found efficiently unless Conjecture 1 is false, relating 3SUM to problems in weighted graphs. Their work was extended [4] for other weighted subgraph problems. The 3SUM conjecture has also recently been shown to imply hardness for various problems on strings such as jumbled indexing [11] and versions of sequence local alignment [7]. Pătraşcu [53] initiated the research of proving lower bounds on dynamic problems based on Conjecture 1. He showed that for several dynamic problems Conjecture 1 implies update time lower bounds that are *polynomial* in the input size, whereas the best (unconditional) cell probe lower bounds known are polylogarithmic at best. Follow-up work by [5] extended and tightened Pătraşcu's results.

2.2 All-Pairs Shortest Paths

The *All-Pairs Shortest Paths* problem (APSP) is as follows: given a directed or undirected graph with integer edge weights, determine the distances between every pair of vertices in the graph. Classical algorithms such as Floyd–Warshall's provide $O(n^3)$ running times for

[2] When the inputs are real numbers, the problem can also be sped up by a logarithmic factor [40].

APSP in n-node graphs. For years, researchers obtained larger and larger *polylogarithmic* improvements over the cubic running time, until in a breakthrough result Williams [64] designed an algorithm that runs faster than $O(n^3/(\log n)^c)$ time for all constants c; the exact running time is $n^3/\exp(\Theta(\sqrt{\log n}))$. Nevertheless, no truly subcubic time ($O(n^{3-\varepsilon})$ for $\varepsilon > 0$) algorithm for APSP is known. This led to the following conjecture assumed in many papers, *e.g.* [55, 61].

▶ **Conjecture 2** (No truly subcubic APSP). There is a constant c, such that in the Word RAM model with $O(\log n)$ bit words, any algorithm requires $n^{3-o(1)}$ time in expectation to compute the distances between every pair of vertices in an n node graph with edge weights in $\{1, \dots, n^c\}$.

Vassilevska Williams and Williams [61] and Abboud, Grandoni and Vassilevska Williams [3] showed that many other graph problems are equivalent to APSP under subcubic reductions, and as a consequence any truly subcubic algorithm for them would violate Conjecture 2. Some examples of these problems include detecting a negative weight triangle in a graph, computing replacement paths and finding the radius of a graph. The APSP conjecture implies strong lower bounds for dynamic problems [55, 5], *e.g.*, for single source shortest paths even if the algorithm is required to support only insertions and deletions.

2.3 Satisfiability

Here we formally describe the Strong Exponential Time Hypothesis (SETH) that we discussed in the introduction. Impagliazzo, Paturi, and Zane [37, 38] introduced SETH to address the question of how fast one can solve k-SAT as k grows. They define:

$$s_k = \inf\{\, \delta \mid \text{there is a } O^*(2^{\delta n}) \text{ time algorithm for } k\text{-SAT with } n \text{ variables} \,\}.$$

The sequence s_k is clearly nondecreasing. As the best known algorithms for k-SAT have running times that converge to $O^*(2^n)$ as k grows, it is natural to conjecture that $\lim_{k\to\infty} s_k = 1$, which is exactly what SETH hypothesizes.

▶ **Conjecture 3** (SETH). For every $\varepsilon > 0$, there exists an integer k, such that Satisfiability on k-CNF formulas on n variables cannot be solved in $O(2^{(1-\varepsilon)n} \operatorname{poly} n)$ time in expectation.

SETH is an extremely popular conjecture in the exact exponential-time algorithms community. For instance, Cygan et al. [24] showed that the SETH is also equivalent to the assumption that several other NP-hard problems cannot be solved faster than by exhaustive search, and the best algorithms for these problems are the exhaustive search ones. Some other work that proves conditional lower bounds based on SETH for NP-hard problems includes [24, 18, 27, 44, 26, 65, 52, 23, 25, 31]. SETH has been the basis for many conditional hardness results for problems in P, including edit-distance [13], LCS [2, 17], graph diameter [54, 21] and many others.

2.4 Orthogonal Vectors

Many hardness results based on SETH go through an intermediate problem, *Orthogonal Vectors* (OV). In this problem, we are given two sets U and V of n vectors each over $\{0,1\}^d$ where $d = \omega(\log n)$, and want to determine whether there are $u \in U$ and $v \in V$ such that $\sum_{i=1}^d u_i \cdot v_i = 0$. An equivalent version of the problem has $U = V$.

The naïve algorithm for the problem runs in $O(n^2 d) \leq \tilde{O}(n^2)$ time, and the best known algorithm [9] runs slightly faster, in $O(n^{2-1/O(\log(d/\log n))})$ time. Obtaining a truly subquadratic algorithm for OV has been elusive, and Williams [63] showed that SETH implies the nonexistence of such an algorithm. (We will see the proof of this later.) No reduction is known from OV to CNF-SAT, and the OV problem may require essentially quadratic time even if SETH is false. Thus it is often better to base hardness on the following OV conjecture.

▶ **Conjecture 4** (OVC). In the Word RAM model with $O(\log n)$ bit words, any algorithm requires $n^{2-o(1)}$ time in expectation to determine whether a set of n vectors over $\{0,1\}^d$ for $d = \omega(\log n)$ contains an orthogonal pair.

2.5 The Small Universe Hitting Set Problem

A version of the Orthogonal Vectors problem is the *Small Universe Hitting Set* (HS) problem: given two sets U and V of n sets each over the universe $\{1,\ldots,d\}$ where $d = \omega(\log n)$, determine whether there is a $u \in U$ that hits every $v \in V$ in at least one element. The HS problem can be described analogously as, given two sets U and V of vectors over $\{0,1\}^d$ for $d = \omega(\log n)$, is there a $u \in U$ such that for all $v \in V$, we have $\sum_{i=1}^d u_i \cdot v_i > 0$.

Just as with OV, the fastest known algorithm for HS runs in $n^{2-o(1)}$ time. Similar to OVC, there is a conjecture concerning the HS problem:

▶ **Conjecture 5** (HSC). In the Word RAM model with $O(\log n)$ bit words, any algorithm, given two sets U and V of n vectors each over $\{0,1\}^d$ for $d = \omega(\log n)$, requires $n^{2-o(1)}$ time in expectation to determine if U contains a vector that is not orthogonal to any vector in V.

We will show that HSC implies OVC and is thus potentially stronger. An example hardness result based on HSC is that computing the radius of a sparse graph requires $n^{2-o(1)}$ time. Such a result does not seem to be possible based on OVC.

2.6 Relationships between the conjectures

Besides the reduction from CNF-SAT to OV by Williams [63], there are no known reductions between the main key problems. Potentially, any subset of the first three key conjectures could be true while the others are false. Recent work [19] gives evidence that it might be difficult to reduce these conjectures to one another, showing that if for instance one could reduce OV to 3SUM or APSP in a tight way, then a nondeterministic version of SETH would fail. It seems to be difficult to solve SAT quickly even by an algorithm that can exploit nondeterminism since showing that the formula is unsatisfiable seems to require a lot of nondeterministic time. Nevertheless, in recent work, Williams [62] has shown that if the nondeterministic algorithm can use randomness, k-SAT can be solved (for all k), in $O((2-\varepsilon)^n)$ time for a constant $\varepsilon > 0$. Derandomizing Williams' algorithm seems challenging, however, so that the nondeterministic version of SETH might still hold.

Even though there are no known (tight) reductions between CNF-SAT, 3SUM and APSP, there are two known problems that one can reduce all three problems to.

The first problem, *Matching Triangles*, takes as an input an integer Δ, a graph $G = (V,E)$ on n nodes, and a coloring of the nodes $c : V \to \{1,\ldots,n\}$. The problem asks, is there a triple of colors, $a,b,c \in \{1,\ldots,n\}$, such that the number of triangles in G that have node colors a,b,c, is at least Δ.

The second problem, *Triangle Collection*, takes as an input a graph $G = (V,E)$ on n nodes, and a coloring of the nodes $c : V \to \{1,\ldots,n\}$, and asks whether there there is a triple of colors, $a, b, c \in \{1, \ldots, n\}$, such that there are no triangles in G that have node colors a, b, c. (An alternative name for this problem might be Triangle-Free Color Triple.)

Both Triangle Collection and Matching Triangles can be solved in $O(n^3)$ time by enumerating all triangles in the input graph. A surprising result [8] is that if either of these problems has an $O(n^{3-\varepsilon})$ time algorithm for $\varepsilon > 0$, then the 3SUM, APSP and SETH conjectures are all false. The result holds for Matching Triangles even when Δ is polylogarithmic, and it also holds for a restricted version of Triangle Collection called *Triangle Collection∗*. The latter problem has been used as a basis for hardness for several problems in dynamic algorithms, such as maintaining the strongly connected components of a graph, and for a few static problems related to Max Flow.

3 The reductions

To prove tight reductions between the running times for solving problems in P, one cannot merely use polynomial time reductions since these do not distinguish between different polynomial running times. Instead, we define *fine-grained reductions* that for problems A and B and running times $a(n), b(n)$ imply that an $O(b(n)^{1-\varepsilon})$ time algorithm for B for constant $\varepsilon > 0$ would imply an $O(a(n)^{1-\delta})$ algorithm for A for constant δ. Notice that we allow δ to be different from ε. This is fine since we merely want to know when any improvement for B carries over to some improvement for A. Having δ and ε differ gives us much more freedom in designing reductions.

An initial attempt would be to say that given an instance of A, the reduction should transform it into a single instance of B, *i.e.*, giving a many-one reduction. This, however, limits us quite a bit. For instance, we wouldn't be able to show that a multi-output problem such as APSP can be reduced to a decision problem such as whether the radius of the graph is less than R. Instead, we define the reductions as special Turing reductions, *i.e.*, each instance of A can be reduced to multiple instances of B.

▶ **Definition 6** (Fine-grained reduction)**.** Let $a(n)$ and $b(n)$ be nondecreasing functions of n. Problem A is (a, b)-reducible to problem B, denoted as $A \leq_{a,b} B$, if for every $\varepsilon > 0$, there is a $\delta > 0$, an algorithm F with access to an oracle to B, a constant d, and for every $n \geq 1$ an integer $k(n)$, such that for every n, the algorithm F takes any instance of A of size n and

- F runs in at most $d \cdot (a(n))^{1-\delta}$ time,
- F produces at most $k(n)$ instances of B adaptively, that is, the jth instance B_j is a function of $\{(B_i, a_i)\}_{1 \leq i < j}$ where B_i is the ith instance produced and a_i is the answer of the oracle for B on instance B_i, and
- the sizes n_i of the instances B_i for any choice of oracle answers a_i obey the inequality $\sum_{i=1}^{k(n)} (b(n_i))^{1-\varepsilon} \leq d \cdot (a(n))^{1-\delta}$.

An immediate consequence of the reductions is that improvements over $b(n)$ for B imply improvements over $a(n)$ for A. More formally, if there is an algorithm for B of running time $c \cdot N^{1-\varepsilon}$ on instances of size N, then composing the algorithm with a fine-grained (a, b)-reduction from A to B produces an algorithm for A running in time

$$d \cdot (a(n))^{1-\delta} + \sum_{i=1}^{k(n)} c \cdot (b(n_i))^{1-\varepsilon} \leq d \cdot (c+1) \cdot (a(n))^{1-\delta},$$

where the first summand is for running the reduction algorithm F, and the second summand is for running the algorithm for B on the instances $\{B_i\}$. As c and d are constants, the running time for A is $O(a(n)^{1-\delta})$.

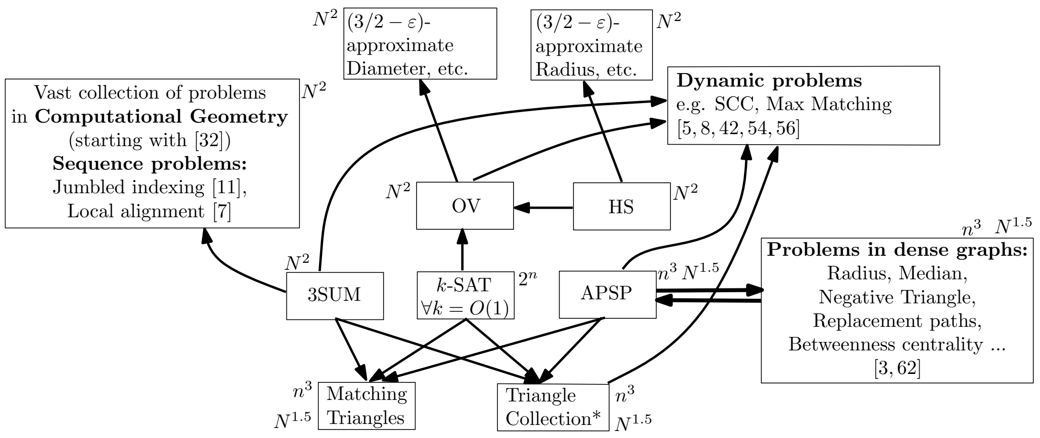
N^2 $(3/2-\varepsilon)$-approximate Diameter, etc.
$(3/2-\varepsilon)$-approximate Radius, etc. N^2
Vast collection of problems in **Computational Geometry** (starting with [32]) **Sequence problems:** Jumbled indexing [11], Local alignment [7] N^2
Dynamic problems e.g. SCC, Max Matching [5, 8, 42, 54, 56]
N^2 OV
HS N^2
N^2 3SUM
k-SAT $\forall k = O(1)$ 2^n
APSP n^3 $N^{1.5}$
n^3 $N^{1.5}$ **Problems in dense graphs:** Radius, Median, Negative Triangle, Replacement paths, Betweenness centrality ... [3, 62]
n^3 $N^{1.5}$ Matching Triangles
Triangle Collection* n^3 $N^{1.5}$

Figure 1 Partial summary of the implications of the main conjectures. An arrow from problem A to problem B, where A has $a(n)$ next to it, B has $b(n)$ next to it, implies that $A \leq_{a,b} B$. When the inputs are graphs, n stands for the number of nodes. N always stands for the total input size. When both n and N are present for a problem, we assume that $N = n^2$; the meaning is that the reductions are only for dense graphs in which case the input size is quadratic in n. For k-SAT, n denotes the number of variables. For the dynamic problems, different key problems can be reduced to different key problems, and the update/query time tradeoffs vary. References are not comprehensive.

Notice that by the above definition, the instance sizes may depend on n and ε, and d can depend on ε, but not on n.

4 Example results

A partial summary of the implications of the main conjectures (3SUM, OV, SETH and APSP) can be found in Figure 1. Many of the reductions in the known results are quite intricate. Here we will present a few simple proofs to illustrate the approach.

We begin with the starting point of many of the reductions from SETH to problems in P. This is Williams' reduction from CNF-SAT to OV.

▶ **Theorem 7** (k-SAT $\leq_{2^n,n^2}$ OV [63]). *For each positive integer k, the k-SAT problem on n variables is $(2^n, n^2)$-reducible to Orthogonal Vectors on n vectors.*

Proof. Let F be a k-SAT formula on n variables. Using the sparsification lemma [38], we can assume that F has $m = O(n)$ clauses, as this only gives a $2^{\alpha n}$ overhead for arbitrarily small $\alpha > 0$. Create two sets U_1, U_2 of vectors over $\{0,1\}^m$. Split the variables into two sets V_1 and V_2 of size $n/2$ each. For each $i \in \{1,2\}$ and each partial assignment ϕ to the variables of V_i, add a vector $v_{i,\phi}$ to U_i where $v_{i,\phi}[c] = 1$ if ϕ does not satisfy clause c. Now, $v_{0,\phi}$ and $v_{1,\psi}$ are orthogonal if and only if for every clause c at least one of ϕ and ψ satisfies clause c. Thus, there is an orthogonal pair if and only if ϕ and ψ together form a satisfying assignment. The number of vectors in the instances created is $N = O(2^{n/2})$ and the number of coordinates is $O(n) = O(\log N)$. ◀

Due to the use of the sparsification lemma, the above reduction produces an exponential number of instances. However, the instances do not depend on the answers of the OV oracle. Below we give a different reduction that is actually adaptive and each new instance depends on the answers to the previous ones.

▶ **Theorem 8** (HS $\le_{n^2,n^2}$ OV [9, 6]). *Small-Universe Hitting Set on n vectors is (n^2, n^2)-reducible to Orthogonal Vectors on n vectors.*

Proof. Let U, V be an instance of HS where $|U| = |V| = n$. Let s be a parameter to be set later. Partition U into s sets $U_1, \ldots, U_s$ of size at most $\lceil n/s \rceil \le 2n/s$ each. Similarly, partition V into $V_1, \ldots, V_s$ of size $\le 2n/s$.

Now, for every choice of $i, j \in \{1, \ldots, s\}$ in turn: while (U_i, V_j) contains an orthogonal pair (u, v), remove u from U (and hence from all U_k) and ask about (U_i, V_j) again; if no orthogonal pair is found, continue to the next choice of (i, j).

If at the end of this procedure U contains some u, then u must be nonorthogonal to all vectors in V, and hence the HS instance is a "yes" instance. Otherwise, if U is empty, then every $u \in U$ was orthogonal to some $v \in V$ and the HS instance is a "no" instance.

The running time is as follows: every call to the OV problem either returns an orthogonal pair (u, v) or determines that no such pair exists in $U_i \times V_j$. The number of times an orthogonal pair can be returned is at most n since when (u, v) is discovered, u is removed from U. On the other hand, each (U_i, V_j) instance can be a "no"-instance of OV at most once, so that the number of calls that return a "no" is at most s^2. Thus, if we set $s = \sqrt{n}$, the number of instances created of OV is at most $2n$ and their sizes are all at most $2\sqrt{n}$. Hence, for every $\varepsilon > 0$, there is a fine-grained reduction that creates at most $2n$ instances whose sizes n_i obey $\sum_{i=1}^{2n} n_i^{2-\varepsilon} \le 4 \sum_{i=1}^{2n} n^{1-\varepsilon/2} = 8n^{2-\varepsilon/2}$. The reduction only does simple partitioning of the instance and removals from U, and hence runs in $O(n^{2-\varepsilon/2})$ time. ◀

Finally, we give a reduction from OV to the problem of approximating the diameter of a sparse graph, *i.e.*, the largest distance. There is a $\frac{3}{2}$-approximation algorithm for diameter that runs in $\tilde{O}(n^{3/2})$ time in n-node, $\tilde{O}(n)$-edge graphs [54, 21]. In such graphs, the diameter can computed exactly in $\tilde{O}(n^2)$ time by computing APSP. The theorem below shows that, assuming OVC (or SETH), if you want a $(\frac{3}{2} - \varepsilon)$-approximation, you might as well compute all pairwise distances in the graph.

▶ **Theorem 9** (OV $\le_{n^2,n^2}$ $(3/2 - \varepsilon)$−Diameter [54]). *Orthogonal Vectors on n vectors is (n^2, n^2)-reducible to distinguishing between diameter 2 and 3 in a graph with n nodes and $O(n)$ edges.*

Proof. Let U, V be an instance of OV. For each $u \in U$ create a node u (abusing notation) and for each $v \in V$ create a node v. For each coordinate c, create a node c. For any vector node x (in U or V), add an edge from x to any coordinate node c if and only if $x[c] = 1$. Add two extra nodes a and b with an edge between them. Add an edge from all $u \in U$ to a and from b to all $v \in V$. Now, all pairs of nodes except those in $U \times V$ are at distance at most 2. Two nodes $u \in U$ and $v \in V$ are at distance 2 if there is some c for which $u[c] = v[c] = 1$, and are at distance 3 otherwise, via (a, b). Thus the diameter is 3 if there is an orthogonal pair and is 2 otherwise. ◀

5 Some open problems

5.1 Connections to exact exponential-time algorithms

Many problems in P have fine-grained reductions from CNF-SAT. There are other NP-hard problems, however, that are not known to be equivalent (in the fine-grained sense) to CNF-SAT.

What problems in P do other NP-hard problems reduce to?

Recent work by Abboud et al. [1] gives fine-grained reductions from k-Clique (for any constant k) to two problems in cubic time, CFG recognition and RNA folding. When k is divisible by 3, the fastest known algorithm for k-Clique runs in $n^{\omega k/3+o(k)}$ time [49] where $\omega < 2.373$ is the matrix multiplication exponent [60, 33]. Via the reduction from [1], the conjecture that this algorithm is optimal implies $n^{\omega-o(1)}$ conditional lower bounds for both RNA folding and CFG recognition. Valiant showed in the 1970s that the latter problem can be solved in $O(n^{\omega})$ time, and hence this running time is tight assuming the k-clique conjecture.

Williams [63] gave a fine-grained (c^n, n^k)-reduction from any 2-CSP, *i.e.*, any constraint satisfaction problem with at most two variables per constraint (such as MAX-CUT or MAX-2-SAT) to k-Clique for any constant $k \geq 3$. Here c^n represents the brute-force running time for the 2-CSP problem, and for MAX-CUT and MAX-2-SAT, $c = 2$. Since one can find a triangle (*i.e.*, a 3-clique) in a graph in $O(n^{2.373})$ time [39], this immediately gave faster than brute-force algorithms for these problems. Due to Williams' reduction, one can view the result of Abboud et al. [1] as reducing any 2-CSP to CFG recognition and RNA folding. The fine-grained reduction implies that any improvement over the current algorithms for the latter two problems would improve upon the best known algorithms for solving 2-CSPs. What other problems can one reduce 2-CSPs to? What can we reduce Set Cover, TSP, etc. to?

5.2 Connections to fixed parameter tractability

The fixed parameter tractability community studies which NP-hard problems, when parameterized by some parameter k of the input, can be solved in $f(k)\,\mathrm{poly}\,n$ time, where n is the size of the input. The idea is that while we believe that NP-hard problems cannot be solved in polynomial time, the hardness can be pushed into some parameter k, so that for all constant values of k, the running time is the same polynomial, independent of k. In a sense, $f(k)$ can be thought of as the constant factor overhead in the running time.

This idea does not have to be restricted to NP-hard problems. Consider a problem that can be solved in $O(n^c)$ time for some constant c and is believed to also require $n^{c-o(1)}$ time. Then, we can ask, for what parameters k is this problem in $f(k)n^{c-\varepsilon}$ time for $\varepsilon > 0$? This question was asked independently by Giannopoulou et al. [34] and Abboud et al. [6]. Giannopoulou et al. consider the Longest Path on interval graphs that is known to be solvable in $O(n^4)$ time. They show that when parameterized by the vertex deletion number k to proper interval graphs, the problem has an $O(k^9 n)$ time algorithm. Abboud et al. explored the diameter problem in sparse graphs which is solvable in $O(n^2)$ time. They developed a fixed parameter subquadratic $2^{O(t \log t)}n$ time algorithm, where t is the treewidth of the input graph, also showing an almost matching lower bound, under SETH.

Under what parameters are APSP, 3SUM, OV in fixed-parameter linear time?

The above question is of course interesting for all hard problems in P.

References

1 Amir Abboud, Arturs Backurs, and Virginia V. Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, October 17-20, 2015*, page to appear, 2015.

2 Amir Abboud, Arturs Backurs, and Virginia V. Williams. Quadratic-time hardness of LCS and other sequence similarity measures. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, October 17-20, 2015*, page to appear, 2015.

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

4 Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 1–12, 2013.

5 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.

6 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, page to appear, 2016.

7 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.

8 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.

9 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.

10 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.

11 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.

12 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 886–894, 2005.

13 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.

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

15 Gill Barequet and Sariel Har-Peled. Polygon-containment and translational min-Hausdorff-distance between segment sets are 3SUM-hard. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.*, pages 862–863, 1999.

16 Philip Bille and Martin Farach-Colton. Fast and compact regular expression matching. *Theoretical Computer Science*, 409(3):486–496, 2008.

17 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, October 17-20, 2015*, page to appear, 2015.

18 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *Parameterized and Exact Computation*, pages 75–85. Springer, 2009.

19 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mikhailin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. Technical Report TR15-148, Electronic Colloquium on Computational Complexity (ECCC), 2015.

20 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40, 2015.

21 Shiri Chechik, Daniel 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.

22 Otfried Cheong, Alon Efrat, and Sariel Har-Peled. On finding a guard that sees most and a shop that sells most. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 1098–1107, 2004.

23 Marek Cygan. Deterministic parameterized connected vertex cover. In *Algorithm Theory – SWAT 2012 – 13th Scandinavian Symposium and Workshops, Helsinki, Finland, July 4-6, 2012. Proceedings*, pages 95–106, 2012.

24 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. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 74–84, 2012.

25 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 301–310, 2013.

26 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011.

27 Evgeny Dantsin and Alexander Wolpert. On moderately exponential time for SAT. In *Proc. 13th International Conference on Theory and Applications of Satisfiability Testing*, pages 313–325, 2010.

28 Mark de Berg, Marko de Groot, and Mark H. Overmars. Perfect binary space partitions. *Computational Geometry: Theory and Applications*, 7(81):81–91, 1997.

29 J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM Journal on Computing*, 28(4):1198–1214, 1999.

30 William S. Evans, Daniel Archambault, and David G. Kirkpatrick. Computing the set of all distant horizons of a terrain. In *Proceedings of the 16th Canadian Conference on Computational Geometry, CCCG'04, Concordia University, Montréal, Québec, Canada, August 9-11, 2004*, pages 76–79, 2004.

31 Henning Fernau, Pinar Heggernes, and Yngve Villanger. A multivariate analysis of some DFA problems. In *Language and Automata Theory and Applications – 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 275–286, 2013.

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

33 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.

34 Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. Polynomial fixed-parameter algorithms: A case study for longest path on interval graphs. *CoRR*, abs/1506.01652, 2015.

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

36 Edward A. Hirsch. Two new upper bounds for SAT. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California.*, pages 521–530, 1998.

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

38 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

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

40 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.

41 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972.

42 Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 147–156, 2013.

43 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\text{vrank})$ 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.

44 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 777–789, 2011.

45 J. Erickson M. Soss and M. H. Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Computational Geometry: Theory and Applications*, 26(3):235–246, 2002.

46 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.

47 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.

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

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

50 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.

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

52 Marcin Pilipczuk and Michał Pilipczuk. Finding a maximum induced degenerate subgraph faster than 2^n. In *IPEC'12: Parameterized and Exact Computation*, pages 3–12, 2012.

53 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.

54 L. Roditty and V. 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, New York, NY, USA, 2013. ACM.

55 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.

56 Ingo Schiermeyer. Solving 3-satisfiability in less than 1.579^n steps. In *Computer Science Logic, 6th Workshop, CSL'92, San Miniato, Italy, September 28 – October 2, 1992, Selected Papers*, pages 379–394, 1992.

57 Uwe Schöning. A probabilistic algorithm for k-sat and constraint satisfaction problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS'99, 17-18 October, 1999, New York, NY, USA*, pages 410–414, 1999.

58 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 455–464, 2009.

59 V. Vassilevska Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

60 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.

61 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654, 2010.

62 R. Williams. Personal communication, 2015.

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

64 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.

65 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.