Open Problems Column
Edited by William Gasarch

1 This Issues Column!

This issue's Open Problem Column is by Virginia Vassilevska Williams. It is titled: *Some Open Problems in Fine-Grained Complexity*.

2 Request for Columns!

I invite any reader who has knowledge of some area to contact me and arrange to write a column about open problems in that area. That area can be (1) broad or narrow or anywhere inbetween, and (2) really important or really unimportant or anywhere inbetween.

Some Open Problems in Fine-Grained Complexity

Virginia Vassilevska Williams
MIT EECS and CSAIL

3 Introduction

Fine-grained complexity studies problems that are "hard" in the following sense. Consider a computational problem for which existing techniques easily give an algorithm running in $a(n)$ time for inputs of size n, for some a. The algorithm is often brute-force, and despite decades of research, no $O(a(n)^{1-\varepsilon})$ time algorithm for constant $\varepsilon > 0$ has been developed.

There are many diverse examples of such problems. Here are two: CNF-SAT on n variables and m clauses can be solved via exhaustive search in $O(2^n mn)$ time, and no $2^{(1-\varepsilon)n}\text{poly}(m,n)$ time algorithm for constant $\varepsilon > 0$ is known. The Longest Common Subsequence (LCS) problem on strings of length n has a classical $O(n^2)$ time algorithm, and no $O(n^{2-\varepsilon})$ time algorithm for $\varepsilon > 0$ is known. Let's call these running times the "textbook running times". (Note that this is not well-defined but for many fundamental problems such as SAT or LCS, it is natural. The textbook runtime is the runtime of the algorithm a bright student in an algorithms class would come up with.)

Fine-grained complexity aims to answer questions such as:

(1) Why is a particular problem A hard in the sense described above?

(2) Suppose we have two hard problems A and B, with textbook runtimes $a(n)$ and $b(n)$. Are they hard for the same reason? Are we lacking some magical technique that will both solve A in $O(a(n)^{1-\varepsilon})$ time and B in $O(b(n)^{1-\varepsilon})$ time for some $\varepsilon > 0$?

The approach of fine-grained complexity is similar to that of NP-hardness: we address these problems via *reductions*. A *fine-grained reduction* from problem A with textbook runtime $a(n)$ to a problem B with textbook runtime $b(n)$, aims to show that if we can get a much faster than $b(n)$ time algorithm for B, then we can also get a much faster than $a(n)$ algorithm for A. The formal definition (see [22, 23]) is as follows:

Definition 3.1 (Fine-grained reduction)**.** *Assume that A and B are computational problems and $a(n)$ and $b(n)$ are time-constructible. 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 solving A that runs in time $O(a(n)^{1-\delta})$ on inputs of size 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, $A \equiv_{a,b} B$.

Unlike in NP-hardness, where P≠NP is the single hardness assumption, fine-grained complexity has (so far) used hardness hypotheses for *several* key problems. The main ones are: CNF-SAT, the Orthogonal Vectors Problem (OV), 3-SUM and All-Pairs Shortest Paths (APSP). Together, the hardness hypotheses for these problems explain the "natural" running times for a large variety of problems. (See [23].)

Computational model. Since we care about the actual running times of algorithms (as opposed to merely whether they are polynomial), we need to fix the computational model we are working with. Let us assume the Word-RAM model with $O(\log n)$ bit words, for inputs of size n.

SETH and k-OV. The first hypothesis, formulated by Impagliazzo, Paturi and Zane [12, 13] concerns the CNF-SAT problem:

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.*

An equivalent formulation is that there is no $O(2^{(1-\varepsilon)n})$ time algorithm for CNF-SAT on n variables and $O(n)$ clauses, for any $\varepsilon > 0$.

Notice that SETH is a **much stronger assumption** than P$\neq$NP: it not only assumes that CNF-SAT requires superpolynomial time, but that it actually requires essentially the time to brute-force over the solution space, and that this is also true of formulas with a very small number of clauses. Nevertheless, this assumption is completely consistent with our current knowledge of the world.

A very useful problem to start reductions from is k-OV, a variant of set-disjointness defined below.. Williams [25] reduced CNF-SAT to k-OV and showed that SETH implies a potentially more believable hardness hypothesis for k-OV.

The *k-Orthogonal Vectors (k-OV) problem* for constant $k \geq 2$ is as follows: 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 $a_1 \cdot \ldots \cdot a_k := \sum_{i=1}^{d} \prod_{j=1}^{k} a_j[i]$. The 2-OV problem is also known as just OV.

Hypothesis 2 (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 $\varepsilon > 0$.*

Williams' result shows that if SETH is true, then the k-OV hypothesis is also true for all integers $k \geq 2$. There is no reduction in the other direction, so that the k-OV hypothesis might hold even if SETH fails.

3-SUM. Starting with the work of Gajentaan and Overmars [9], a hardness hypothesis concerning the 3-SUM problem defined below became very popular in computational geometry, as it implied tight hardness results for many geometry problems.

The 3-*SUM Problem* is: given a set S of n integers, determine whether there are $a, b, c \in S$ with $a + b + c = 0$.

Hypothesis 3 (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.*

APSP. A classical graph problem is APSP: given a graph with weights on its edges, determine for every pair of vertices, the shortest path distance between them. A popular hypothesis asserts that the classical cubic time algorithms for APSP are essentially optimal.

Hypothesis 4 (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.*

4 Open Problems

The most tantalizing open problems in fine-grained complexity concern the main hypotheses: how believable are they, what are the relationships between them, can we replace them with better hypotheses?

Problem 1. How fast can we actually solve k-SAT? The fastest known algorithms have running times of the form $2^{n-cn/k}\text{poly}(n)$, for various constants c (see e.g., [11, 14, 16, 15, 20, 21]). Can we improve these algorithms? For instance, can we solve k-SAT in $O(2^{n-cn(\log k)/k})$ time for some c? Notably, the known hard instances [17] for the PPSZ [15] algorithm only say that PPSZ does not run faster than $O(2^{n-cn(\log k)/k})$ time, so that PPSZ itself might give such a runtime. Alternatively, does anything interesting follow from assuming that the best runtime for k-SAT is $2^{n-cn/k-o(n)}$?

Problem 2. How fast can we solve 3-SUM? For APSP and k-OV, significant (though still $n^{o(1)}$) improvements over the textbook runtime are possible. APSP has an $n^3/\exp(\sqrt{\log n})$ time algorithm [26], and k-OV on vectors of dimension d can be solved in $n^{k-1/\Theta(\log(d/\log n))}$ time [2, 7].

Thus both APSP and k-OV of dimension up to $\exp(\sqrt{\log(n)})$ have an improvement of $\exp(\sqrt{\log(n)})$ over their textbook runtimes; this subsumes all polylogarithmic improvements, for example. Meanwhile, the fastest known algorithms for 3-SUM, up to poly $\log\log n$ factors, run in $O(n^2/\log^2 n)$ time [3, 6].

Can one achieve an improvement for 3-SUM, similar to the ones for APSP and k-OV? Or, can we show that some bizarre consequence follows from such an improved algorithm? More modestly, is 3-SUM in, say, $O(n^2/\log^3 n)$ time?

It seems that the current techniques that worked for APSP and OV, are not sufficient to give improvements for 3-SUM. Nevertheless, my guess is that $O(n^2/\log^2 n)$ cannot be the best running time, and that further improvements should be possible.

Problem 3. Can we refute NSETH? Carmosino et al. [5] formulated a nondeterministic version of SETH as follows:

Hypothesis 5. *(NSETH) For every $\varepsilon > 0$ there is a k, so that k-TAUT is not in NTIME$[2^{(1-\varepsilon)n}]$, where k-TAUT is the language of k-DNFs that are tautologies.*

In other words, NSETH asserts that there is no substantially sub-2^n time nondeterministic algorithm for CNF-UNSAT. NSETH sounds to me ridiculously unbelievable – somehow with all the power of nondeterminism we are still unable to improve upon the brute-force algorithm.

Partial progress towards refuting NSETH was achieved by Williams [28]: One can define versions of SETH for AM and MA-protocols, and Williams refuted these, providing very efficient AM and MA-protocols for CNF-UNSAT. Nevertheless, the approach crucially uses randomness, and with the current state of SAT algorithms, NSETH might still be true. Can we refute it?

Problem 4. Relate the key hard problems to each other. Carmosino et al [5] showed that under NSETH, there can be no deterministic fine-grained reduction from OV to 3-SUM or APSP. These results do not rule out randomized reductions (AMSETH and MASETH are false, and NSETH might also be false).

Is there a (randomized) fine-grained reduction from OV to 3-SUM or APSP?

There is a problem, *Exact Triangle*, that 3-SUM and APSP both reduce to. In Exact Triangle, one is given a complete n-node graph with integer edge weights in $\{-n^C, \ldots, n^C\}$ for some constant C, and one is asked, do there exist three nodes a, b, c so that $w(a,b) + w(b,c) + w(c,a) = 0$?

Vassilevska W. and Williams [24] showed that if Exact Triangle is in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$, then APSP is in $O(n^{3-\varepsilon'})$ time for $\varepsilon' > 0$ and 3-SUM is in $O(n^{2-\varepsilon'})$ time for $\varepsilon' > 0$. Thus a potentially simpler question is: *Is there an (n^2, n^3) (randomized) fine-grained reduction from OV to Exact Triangle?*

Finally, NSETH does not rule out reductions from APSP and 3-SUM to OV (or between APSP and 3-SUM). *Is there an (n^3, n^2) fine-grained reduction from APSP to OV, or an (n^2, n^2) reduction from 3-SUM to OV?* My guess: most likely yes.

There is another problem that seems hard, and can also be reduced in a fine-grained way to both APSP and 3-SUM. This is the $(\min, +)$-*Convolution* problem: Given two arrays A and B of length n and integers in $\{-n^c, \ldots, n^c\}$ for some c, compute an array C so that $C[k] = \min_i A[i] + B[k-i]$.

There is no known $O(n^{2-\varepsilon})$ time algorithm for $\varepsilon > 0$ for $(\min, +)$-Convolution, and the brute-force running time is $O(n^2)$. Techniques from [4] and [24] show that if $(\min, +)$-Convolution requires $n^{2-o(1)}$ time, then APSP requires $n^{3-o(1)}$ time, and 3-SUM requires $n^{2-o(1)}$ time (see e.g. [8]). Thus, one can base the hardness of both APSP and 3-SUM on $(\min, +)$-Convolution.

Can one (n^2, n^2) reduce $(\min, +)$-Convolution to OV as well?

We want to point out that the following simpler looking *Negative Convolution Pair* problem is (n^2, n^2)-equivalent to $(\min, +)$-Convolution, using techniques from [22]: Given three n-length arrays A, B, C of integers in $\{-n^c, \ldots, n^c\}$ for large enough constant c, are there some i, k so that $A[i] + B[k-i] < C[k]$?

Can Negative Convolution Pair be (n^2, n^2) reduced to OV?

Problem 5. Does k-OV get easier as k grows? We know that CNF-SAT can be reduced to k-OV for every k, and that SETH implies that k-OV should require $n^{k-o(1)}$ time for each k. It is also not hard to see that k-OV can be (n^k, n^{k-1}) reduced to $(k-1)$-OV for every $k \geq 3$, so that 2-OV is in a sense the hardest k-OV problem. Two questions emerge:

Can one (n^k, n^{k+1})-reduce k-OV to $(k+1)$-OV?

If this can be done for all k, then all k-OV problems would be fine-grained equivalent. However, we don't have such a result for any k. Perhaps, for instance, 2-OV and 3-OV are not equivalent? My feeling is that 2-OV actually does require $n^{2-o(1)}$ time, but I wouldn't be too astonished if say 100-OV has an $O(n^{99})$ time algorithm (and hence SETH is false). We are quite far from proving such a statement of course.

On the other hand, suppose that we assume that for every constant $k \geq 2$, k-OV requires $n^{k-o(1)}$ time. Can we prove that SETH holds from this? In other words, *is SETH equivalent to the AND of all k-OV Hypotheses?*

Problem 6. Hitting Set vs 3-SUM. Perhaps it is difficult (or even impossible) to reduce OV to 3-SUM or APSP. As a first step then, one can consider the potentially easier Hitting Set problem.

In *Hitting Set* one is given two n-sets of vectors $A, B \subseteq \{0,1\}^d$ and is asked, does there exist an $a \in A$ such that for all $b \in B$, $a \cdot b \neq 0$?

The negation of Hitting Set asks whether $\forall a \in A, \exists b \in B$ such that $a \cdot b = 0$. In other words, (the negation of) Hitting Set is obtained from OV via a quantifier swap - from $\exists\exists$ to $\forall\exists$. It is known (e.g. [1, 2])

that Hitting Set can be (n^2, n^2)-reduced to OV, but a reduction in the reverse direction is not known. It is thus possible that Hitting Set is truly easier than OV.

Can we reduce Hitting Set to 3-SUM or APSP?

Problem 7. Algorithms for Balanced QBF? When studying the Graph Diameter and Radius problems, Abboud et al. [1] noted that Diameter and Radius only differ in their quantifiers $\exists\exists$ vs $\forall\exists$. A fine-grained lower bound based on OV was known for Diameter ([18]). To obtain a similar lower bound for Radius, one needs the Hitting Set problem which, as mentioned earlier, is obtained from the OV problem by the same quantifier swap as going from Diameter to Radius.

Similar quantifier swaps define various variants of k-OV, many of which, similarly to k-OV, are not known to be solvable faster than $n^{k-o(1)}$ time. A typical quantified k-OV problem is of the form $Q_1 a_1 \in A_1 Q_2 a_2 \in A_2 \ldots Q_k a_k \in A_k : \ a_1 \cdot a_2 \cdot \ldots \cdot a_k = 0$, where the input is $A_1, \ldots, A_k \subseteq \{0,1\}^d$ with $|A_j| = n, \forall j$, and $Q_j \in \{\exists, \forall\}, \forall j$.

Such quantified problems are studied by Gao et al. [10] (see also Williams [27]) in the context of first-order properties on sparse structures. While k-OV is known to be the hardest problem among the variants with k quantifiers (follows from [2]), it is unclear whether the rest of the quantified variants are equivalent to k-OV or potentially easier. Each quantified OV problem can explain the hardness of a variety of problems with a similar quantifier structure, and thus these variants are all of interest.

The hardness of k-OV is often justified via SETH. A natural question is whether a similar SAT-based hypothesis is believable for the rest of the quantified variants. To this end, let us defined a Balanced Quantified Boolean Formula as follows. Let $x_1, \ldots, x_n$ be given Boolean variables, and let $F(x_1, \ldots, x_n)$ be a CNF formula. Then

$$Q_1 x_1, \ldots, x_{n/k} Q_2 x_{1+n/k}, \ldots x_{2n/k} \ldots Q_k (x_{1+(k-1)n/k}, \ldots, x_n) F(x_1, \ldots, x_n)$$

is a balanced QBF where $Q_1, \ldots, Q_k \in \{\exists, \forall\}$, $Q_k = \exists$ and the Q_i alternate. In other words, this is just a quantified CNF in which every one of the k quantifier blocks have n/k variables.

Are there constants $k \geq 1$ and $\varepsilon > 0$ for which there is an $2^{(1-\varepsilon)n}\text{poly}(m)$ time algorithm to determine whether a balanced QBF with k quantifiers, n variables and m clauses is satisfiable?

Clearly, if the balance condition is removed, one can easily reduce CNF-SAT to QBF (for CNFs), and SETH implies that QBF with any constant number of quantifiers requires $2^{n-o(n)}$ time. However, when the quantifier blocks are balanced, no such fine-grained reduction from CNF-SAT is known. Perhaps the problem gets easier as the number of quantifiers increase? Santhanam and Williams [19] gave several counterintuitive algorithms for the case when the blocks are not balanced: for instance, QBF on circuits of size s and k quantifier blocks can be solved in time $2^{n-\Omega(k)}\text{poly}(s)$ by a zero-error randomized algorithm, and quantified CNFs on n variables, k quantifier blocks and size $\text{poly}(n)$ can be solved in time $2^{n-n^{1/(k+1)}}\text{poly}(n)$ also by a zero-error randomized algorithm. Could similar techniques be used to show that when the blocks are balanced, much better improvements are possible? My guess is that balanced QBF does get easier as the number of quantifiers k grows, and there might actually be an $O(1.99^n)$ time algorithm for some k, though who knows.

References

[1] 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.

[2] 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.

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

[4] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014.

[5] 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.

[6] 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.

[7] 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.

[8] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. On problems equivalent to (min, +)-convolution. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 22:1–22:15, 2017.

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

[10] 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.

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

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

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

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

[15] 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.

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

[17] Pavel Pudlák, Dominik Scheder, and Navid Talebanfard. Tighter hard instances for PPSZ. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 85:1–85:13, 2017.

[18] 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.

[19] Rahul Santhanam and Richard Ryan Williams. Beating exhaustive search for quantified boolean formulas and connections to circuit complexity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 231–241, 2015.

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

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

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

[23] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians*, page to appear, 2018.

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

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

[26] 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.

[27] Ryan Williams. Faster decision of first-order graph properties. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 80:1–80:6, 2014.

[28] 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.