Hardness for Triangle Problems under Even More Believable Hypotheses: Reductions from Real APSP, Real 3SUM, and OV*

Timothy M. Chan†
tmc@illinois.edu
UIUC
Champaign, IL, USA

Virginia Vassilevska Williams‡
virgi@mit.edu
MIT
Cambridge, MA, USA

Yinzhan Xu§
xyzhan@mit.edu
MIT
Cambridge, MA, USA

ABSTRACT

The 3SUM hypothesis, the All-Pairs Shortest Paths (APSP) hypothesis and the Strong Exponential Time Hypothesis are the three main hypotheses in the area of fine-grained complexity. So far, within the area, the first two hypotheses have mainly been about integer inputs in the Word RAM model of computation. The "Real APSP" and "Real 3SUM" hypotheses, which assert that the APSP and 3SUM hypotheses hold for real-valued inputs in a reasonable version of the Real RAM model, are even more believable than their integer counterparts.

Under the very believable hypothesis that at least one of the Integer 3SUM hypothesis, Integer APSP hypothesis or SETH is true, Abboud, Vassilevska W. and Yu [STOC 2015] showed that a problem called Triangle Collection requires $n^{3-o(1)}$ time on an n-node graph.

The main result of this paper is a nontrivial lower bound for a slight generalization of Triangle Collection, called All-Color-Pairs Triangle Collection, under the even more believable hypothesis that at least one of the Real 3SUM, the Real APSP, and the Orthogonal Vector (OV) hypotheses is true. Combined with slight modifications of prior reductions from Triangle Collection, we obtain polynomial conditional lower bounds for problems such as the (static) ST-Max Flow problem and dynamic versions of Max Flow, Single-Source Reachability Count, and Counting Strongly Connected Components, now under the new weaker hypothesis.

Our main result is built on the following two lines of reductions. In the first line of reductions, we show Real APSP and Real 3SUM hardness for the All-Edges Sparse Triangle problem. Prior reductions only worked from the integer variants of these problems. In the second line of reductions, we show Real APSP and OV hardness for a variant of the Boolean Matrix Multiplication problem.

Along the way we show that Triangle Collection is equivalent to a simpler restricted version of the problem, simplifying prior work. Our techniques also have other interesting implications, such as a super-linear lower bound of Integer All-Numbers 3SUM based on the Real 3SUM hypothesis, and a tight lower bound for a string matching problem based on the OV hypothesis.

CCS CONCEPTS

• **Theory of computation → Data structures design and analysis**.

KEYWORDS

fine-grained complexity

ACM Reference Format:
Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. 2022. Hardness for Triangle Problems under Even More Believable Hypotheses: Reductions from Real APSP, Real 3SUM, and OV. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3519935.3520032

1 INTRODUCTION

Fine-grained complexity is an active area of study that gives a problem-centric approach to complexity. Its major goal is to prove relationships and equivalences between problems whose best known running times have not been improved in decades. Through a variety of techniques and sophisticated reductions many problems from a huge variety of domains and with potentially vastly different running time complexities are now known to be related via *fine-grained reductions* (see e.g. the survey [54]).

As a consequence of the known reductions, the hardness of most of the studied problems in fine-grained complexity can be based on the presumed hardness of three key problems: the 3SUM problem, the All-Pairs Shortest Paths (APSP) problem and CNF-SAT. Their associated hardness hypotheses below are all defined for the Word RAM model of computation with $O(\log n)$-bit words:

- **The (Integer) 3SUM hypothesis.** There is no algorithm that can check whether a list of n integers from $\pm[n^c]$ for some constant c contains three integers that sum up to zero in $O(n^{2-\varepsilon})$ time for $\varepsilon > 0$.[1]
- **The (Integer) APSP hypothesis.** There is no algorithm that can solve the APSP problem in an n-node graph whose

*The full version of this paper is available at https://arxiv.org/abs/2203.08356.
†Supported by NSF Grant CCF-1814026.
‡Supported by NSF Grants CCF-2129139 and CCF-1909429, a BSF Grant BSF:2020356, a Google Research Fellowship and a Sloan Research Fellowship.
§Partially supported by NSF Grant CCF-2129139.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520032

[1]Throughout this paper, $[N]$ denotes $\{0, 1, \ldots, N-1\}$ and $\pm[N]$ denotes $\{-(N-1), \ldots, N-1\}$.

edge weights are from $\pm[n^c]$ for some constant c in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$.

- **The Strong Exponential Time Hypothesis (SETH).** For every $\varepsilon > 0$, there exists an integer k such that k-SAT on n variables cannot be solved in $O(2^{(1-\varepsilon)n})$ time. An equivalent formulation states that there is no $O(2^{(1-\varepsilon)n})$ time algorithm for CNF-SAT with n variables and $O(n)$ clauses for any $\varepsilon > 0$.

All three hardness hypotheses were studied before Fine-Grained Complexity even got its name. The complexity of 3SUM was first used as a basis for hardness in the computational geometry community by Gajentaan and Overmars [39]. The complexity of APSP has been used as a basis of hardness in the graph algorithms community at least since the early 2000s (e.g. [50]). SETH was first studied in 1999 by Impagliazzo and Paturi [44], though the name "SETH" was first given later in [19]. Together, the three hypotheses have been very influential, giving very strong lower bounds for a wide range of problems.

The 3SUM hypothesis is now known to imply tight hardness results for many geometric problems (e.g. [6, 12, 13, 16, 23, 30, 32, 33, 52]), and also for some non-algebraic problems (e.g. [2, 3, 46, 47, 49]); some convolution problems [47, 49] are known to be equivalent to 3SUM. The APSP hypothesis is known to imply tight hardness results for many problems (e.g. [2, 17, 49]), and many problems are also known to be fine-grained equivalent to APSP ([1, 56]). SETH is now known to imply an enormous number of lower bounds both for problems in exponential time (e.g. [18, 26, 28, 48]), and in polynomial time (e.g. [9, 14, 58] and many more); the hardness of a small number of problems is also known to be equivalent to SETH [25]. See [54] for more known implications.

There are no known direct relationships between the three hypotheses, and there is some evidence (e.g. [20]) that reducing between them might be difficult. As we do not really know which, if any, of these hypotheses actually hold, it is important to consider weaker hypotheses that still give meaningful hardness results.

A natural hardness hypothesis considered by Abboud, Vassilevska W. and Yu [4] is the following:

Hypothesis 1. *At least one of SETH, the (Integer) 3SUM or the (Integer) APSP hypothesis is true.*

Under Hypothesis 1, Abboud, Vassilevska W. and Yu proved polynomial lower bounds for a variant of maximum flow and several problems in dynamic graph algorithms: dynamically maintaining the maximum flow in a graph, the number of nodes reachable from a fixed source in a graph (#SSR), the number of strongly connected components in a directed graph (#SCC), and more.

Dahlgaard [27] showed that computing the diameter of an unweighted graph with n nodes and m edges requires $n^{1-o(1)}\sqrt{m}$ time under Hypothesis 1. The reductions of both [4] and [27] utilized a problem called Triangle-Collection, which we will abbreviate as Tri-Co.

In the Tri-Co problem, given a node-colored graph with n nodes, one is asked whether it is true that for all triples of distinct colors (a, b, c), there exists a triangle whose nodes have these colors. A key step in the above reductions proved in [4] is the following tight hardness result for Tri-Co.

Theorem 1.1 ([4]). *Assuming Hypothesis 1,* Tri-Co *requires* $n^{3-o(1)}$ *time.*

The main question that inspires this work is the following:

Is there a natural hypothesis that is weaker than Hypothesis 1 and implies similar hardness results?

As mentioned, recent conditional lower bound results based on APSP and 3SUM (especially since Pătraşcu's seminal paper [49]) typically assumed the *integer* variants of these hypotheses. However, algorithms for APSP and 3SUM are more often designed for the *real*-valued versions of the problems (this includes not only the traditional cubic or quadratic time algorithms, but the celebrated, slightly subcubic or subquadratic algorithms of Williams [59] or Grønlund and Pettie [43], as we will review shortly). This discrepancy between the literature on lower bounds and upper bounds raises another intriguing question:

Do known conditional lower bounds derived from the integer versions of the APSP and 3SUM hypotheses hold for the real versions of these hypotheses?

Our work will give a positive answer to this second question for a plethora of known conditional lower bound results, and will hence provide an answer to the first question as well, since the real versions of the hypotheses are weaker/more believable.

Remarks on models of computation. Within fine-grained complexity it is standard to work in the Word RAM model of computation with $O(\log n)$-bit words. As we will consider variants of APSP and 3SUM with real-valued inputs, we will need to work in the Real RAM, a standard model in computational geometry.

The Real RAM (see e.g. Section 6 in the full version of [34]) supports unit cost comparisons and arithmetic operations (addition, subtraction, multiplication, division) on real numbers, unit cost casting integers into reals, in addition to the standard unit cost operations supported by an $O(\log n)$-bit Word RAM. No conversions from real numbers to integers are allowed, and randomization only happens by taking random $O(\log n)$-bit integers, not random reals.

Without further restrictions, the Real RAM can be unrealistically powerful. However, it is not difficult to define a "reasonable" restricted Real RAM model for which our reductions still work, such that any algorithm for real-valued inputs in such a model can be converted into an algorithm in the word RAM for integer-valued inputs, running in roughly the same time. See the full paper for a detailed discussion, and several natural ways to define such a reasonable Real RAM model. Thus, the real versions of the hardness hypotheses under such a model are indeed even more believable than the integer versions.

Real 3SUM hypothesis. Historically, the early papers on the 3SUM hypothesis from computational geometry were concerned with the real instead of the integer case.[2]

Let Real-3SUM refer to the version of 3SUM for real numbers (in contrast to Int-3SUM, which refers to the integer version). In its original form, the Real 3SUM hypothesis stated that there is no $o(n^2)$ time algorithm that solves Real-3SUM in the Real RAM model, and some evidence was provided by Erickson [32], who

[2]Technically, the original paper by Gajentaan and Overmars [39] stated the 3SUM hypothesis for integer inputs, but they assumed a Real RAM model of computation, as in most work in computational geometry.

proved quadratic lower bounds for algorithms that are allowed to use only restricted forms of real comparisons (testing the signs of linear functions involving just 3 input reals). Grønlund and Pettie refuted this hypothesis by giving an $O(n^2(\log\log n)^{2/3}/(\log n)^{2/3})$ time deterministic algorithm and an $O(n^2(\log\log n)^2/\log n)$ time randomized algorithm for Real-3SUM [43]. This was subsequently improved by Freund [38], Gold and Sharir [42], and Chan [21], reaching an $n^2(\log\log n)^{O(1)}/\log^2 n$ deterministic running time. Grønlund and Pettie obtained their initial breakthrough by first proving a truly subquadratic upper bound, near $n^{3/2}$, on the linear-decision-tree complexity of Real-3SUM (using comparisons of two sums of pairs of input reals). More recently, Kane, Lovett, and Moran [45] in another breakthrough improved the linear-decision-tree complexity upper bound to $\tilde{O}(n)$,[3] although their approach has not yet led to improved algorithms. A truly subquadratic time algorithm remains open.

The modern Real 3SUM hypothesis states that no $O(n^{2-\varepsilon})$ time algorithm exists for Real-3SUM with real-valued inputs, in a reasonable Real RAM model, for any $\varepsilon > 0$.

The integer version of 3SUM has gained more attention after a groundbreaking result by Pătraşcu [49] which showed that the Integer 3SUM hypothesis implies lower bounds for many dynamic problems that may not even have numbers in their inputs. Notably, his reduction uses a hashing technique which does not quite apply to Real-3SUM. Thus, these hardness results and many subsequent results [2, 46] following [49] were not known to be true under the Real 3SUM hypothesis. Thus, an intriguing question is whether these problems are also hard under the Real 3SUM hypothesis.

Real-3SUM could conceivably be harder than Int-3SUM. Baran, Demaine and Pătraşcu gave an $n^2(\log\log n)^{O(1)}/\log^2 n$ time algorithm for Int-3SUM as early as 2005 [15]. However, Real-3SUM did not have an algorithm with the same asymptotic running time (up to $(\log\log n)^{O(1)}$ factors) until more than ten years later [21]. Also, many techniques that are useful for Int-3SUM stop working for Real-3SUM, for example, the hashing technique used by Baran, Demaine and Pătraşcu. Therefore, the Real 3SUM hypothesis is arguably weaker than the Integer 3SUM hypothesis.

Real APSP hypothesis. Similarly, one could naturally consider the APSP problem where the edge weights of the input graph are real numbers. In fact, historically, the APSP problem was first studied when the edge weights are real numbers [36]. Fredman [37] gave the first slightly subcubic time algorithm for Real-APSP, running in $n^3(\log\log n)^{O(1)}/(\log n)^{1/3}$ time. Fredman obtained his result by first proving a truly subcubic upper bound, near $n^{5/2}$, on the linear-decision-tree complexity of Real-APSP (using what is now known as "Fredman's trick" that later inspired Grønlund and Pettie's work on Real-3SUM). After a series of further improved poly-logarithmic speedups, Williams [60] developed the current fastest algorithm for Real-APSP, with running time $n^3/2^{\Omega(\sqrt{\log n})}$ (which was subsequently derandomized by Chan and Williams [22]). A truly subcubic algorithm remains open.

The Real APSP hypothesis states that no $O(n^{3-\varepsilon})$ time algorithm exists for Real-APSP in graphs with real-valued weights for any $\varepsilon > 0$ in a reasonable Real RAM model.

[3] In this work, we use $\tilde{O}$ to suppress poly-logarithmic factors.

Although the current upper bounds for Real-APSP and Int-APSP are the same, Int-APSP could conceivably be easier than Real-APSP. For example, in Williams' APSP paper [60], he described a "relatively short argument" for an $n^3/2^{\Omega(\log^\delta n)}$ algorithm in the integer case as a warm-up, which did not immediately generalize to the real case (which required further ideas).

It is known [35] that Int-APSP (resp. Real-APSP) is equivalent to Int-(min, +)-Product (resp. Real-(min, +)-Product), where one is given two $n\times n$ integer (resp. real) matrices A, B, and is asked to compute an $n \times n$ matrix C where $C[i, j] = \min_{k\in[n]}(A[i, k] + B[k, j])$. Thus, (min, +)-Product has been the main focus of algorithms and reductions for APSP.

The Orthogonal Vectors (OV) hypothesis. In the OV problem, given a set of n Boolean vectors in d-dimension for some $d = \omega(\log n)$, one needs to determine if there are two vectors u, v such that $\bigvee_{i=1}^{d}(u[i] \wedge v[i])$ is false. The OV hypothesis (OVH) states that no $O(n^{2-\varepsilon})$ time algorithm exists for OV for any $\varepsilon > 0$. Williams [58] showed that SETH implies OVH. In fact, a large fraction of the conditional lower bounds based on SETH actually use the OV problem as an intermediate problem (see the survey [54] for an overview of problems that are hard under OVH and SETH).

In the k-OV problem we are given k sets of Boolean vectors of small dimension and are asked if there exists a k-tuple of vectors, one from each set, that is orthogonal, i.e. there is no coordinate in which all k vectors have 1s. Williams' argument [58] in fact implies a fine-grained reduction from CNF-SAT with n variables and $O(n)$ clauses to k-OV for any integer $k \geq 2$, so that under SETH, k-OV requires $n^{k-o(1)}$ time.

It is easy to reduce k-OV for any $k > 2$ to OV, however a reduction in the reverse direction has been elusive. It is quite possible that, say 1000-OV has an $O(n^{999.99})$ time algorithm, yet OV still requires $n^{2-o(1)}$ time. Thus basing hardness on OV is arguably better than basing hardness on k-OV for $k > 2$, and basing hardness on any fixed k-OV is better than basing hardness on SETH, as all one needs to do to refute SETH is to refute the presumed hardness of k-OV for some k.

1.1 Our Results

We first define our new hypothesis and then outline our reductions.

Combining OVH, the Real 3SUM hypothesis and the Real APSP hypothesis, we consider the following very weak hypothesis.

Hypothesis 2. *At least one of OVH, the Real 3SUM hypothesis or the Real APSP hypothesis is true.*

1.1.1 Main Result: Hardness for Triangle Collection. It is unclear whether any of the Real 3SUM hypothesis, the Real APSP hypothesis, and the OV hypothesis (let alone Hypothesis 2) implies nontrivial lower bounds for the Tri-Co problem. Abboud, Vassilevska W. and Yu's prior reductions [4] from Int-3SUM and Int-APSP to Tri-Co used hashing tricks that are not applicable to real inputs; furthermore, their reduction from CNF-SAT to Tri-Co does not go through OV, but was from the stronger 3-OV problem.

The proof of Theorem 1.1 in [4] uses an intermediate problem, Int-Exact-Tri, between Int-3SUM / Int-APSP and Tri-Co. In Int-Exact-Tri, one is given an n-node graph with edge weights in $\pm[n^c]$ for some constant c and one needs to determine whether the

graph contains a triangle whose edge weights sum up to zero. One can define Real-Exact-Tri analogously by replacing edge weights with real numbers. The Integer (resp. Real) Exact-Triangle hypothesis states that no $O(n^{3-\varepsilon})$ time algorithm for $\varepsilon > 0$ exists for Int-Exact-Tri (resp. Real-Exact-Tri) in the Word RAM model (resp. a reasonable Real RAM model), and it is well-known that the Integer 3SUM hypothesis and the Integer APSP hypothesis imply the Integer Exact-Triangle hypothesis [55].

If one tries to mimic the approach of going through Int-Exact-Tri in the real case, one would first reduce Real-APSP and Real-3SUM to Real-Exact-Tri, and then reduce Real-Exact-Tri to Tri-Co. However, it was unclear whether Real-APSP reduces to Real-Exact-Tri, since the previous reduction for the integer case relies on fixing the results bit by bit [55]. The tight reduction from Int-3SUM to Int-Exact-Tri also doesn't seem to apply since it relies on Pătraşcu's hashing technique [49]. (However, a previous non-tight reduction from Int-3SUM to Int-Exact-Tri by Vassilevska W. and Williams [55] does generalize to a reduction from Real-3SUM to Real-Exact-Tri; see the full paper for more details.)

Secondly, even if one is able to successfully reduce Real-APSP to Real-Exact-Tri, one still faces an obstacle in reducing further to Tri-Co: the proof in [4] relies on transforming the integer edge weights into vectors of tiny integers, which isn't possible if the edge weights are real numbers.

The reduction from SETH to Tri-Co in [4] implicitly goes through the 3-OV problem. It is very natural to relate 3-OV to Tri-Co, since 3-OV asks whether all triples of vectors are not orthogonal, and Tri-Co asks whether all triples of colors have a triangle. It is then sufficient to embed 3-OV into Tri-Co so that a triple of vectors are not orthogonal if and only if their corresponding colors have a triangle. However, OV and Tri-Co are seemingly conceptually different (OV is about pairs, whereas Tri-Co is about triples) and thus it is not quite clear whether one can reduce OV to Tri-Co.

We consider the ACP-Tri-Co problem, a natural generalization of the Tri-Co problem ("ACP" stands for "All-Color-Pairs"). The input of ACP-Tri-Co is the same as the input of Tri-Co, while ACP-Tri-Co requires the algorithm to output for every pair of distinct colors (a, b), whether there exists a triangle with colors (a, b, c) for every c different from a and b.

As our main result, we give a nontrivial fine-grained lower bound for ACP-Tri-Co based on the very weak Hypothesis 2.

Theorem 1.2. *Assuming Hypothesis 2,* ACP-Tri-Co *requires* $n^{2+\delta-o(1)}$ *time for some* $\delta > 0$*, in a reasonable Real RAM model.*

Our reductions prove the above theorem for $\delta = 0.25$. Combining Theorem 1.2 with the reductions by Abboud, Vassilevska W. and Yu [4], we obtain the following corollary. In the following, the #SS-Sub-Conn problem asks to maintain the number of nodes reachable from a fixed source in an undirected graph under node updates, and the ST-Max-Flow problem asks to compute the maximum flow in a graph for every pair of vertices $(s, t) \in S \times T$ for two given subsets of nodes S, T.

Corollary 1.3. *Assuming Hypothesis 2, there exists a constant* $\delta > 0$ *such that any fully dynamic algorithm for* #SSR, #SCC, #SS-Sub-Conn, *and* Max-Flow *requires either amortized* $n^{\delta-o(1)}$ *update or query times, or* $n^{2+\delta-o(1)}$ *preprocessing time.*

Also, assuming Hypothesis 2, ST-Max-Flow *on a network with* n *nodes and* $O(n)$ *edges requires* $n^{1+\delta-o(1)}$ *time for some* $\delta > 0$*, even when* $|S| = |T| = \sqrt{n}$*.*

We obtain Theorem 1.2 via two conceptually different webs of reductions, together with equivalence results between variants of Triangle Collection.

1.1.2 Real APSP and Real 3SUM Hardness via All-Edges Sparse Triangle. In the All-Edges Sparse Triangle problem, which we will abbreviate as AE-Sparse-Tri, one is given a graph with m edges, and is asked whether each edge in the graph is in a triangle. This problem has an $O(m^{2\omega/(\omega+1)})$ time algorithm [11] where ω is the square matrix multiplication exponent. This running time is $O(m^{1.41})$ with the current bound $\omega < 2.373$ [8, 40, 53], and $\tilde{O}(m^{4/3})$ if $\omega = 2$.

Pătraşcu [49] showed that this problem requires $m^{4/3-o(1)}$ time assuming the integer 3SUM hypothesis. Kopelowitz, Pettie, and Porat [46] extended Pătraşcu's result by showing conditional lower bounds for Set-Disjointness and Set-Intersection, which generalize AE-Sparse-Tri. Recently, Vassilevska W. and Xu [57] showed a reduction from Int-Exact-Tri to AE-Sparse-Tri. Combined with the reductions from Int-APSP and Int-3SUM to Int-Exact-Tri [55, 56], we obtain that AE-Sparse-Tri requires $m^{4/3-o(1)}$ time assuming either the Integer 3SUM hypothesis or the Integer APSP hypothesis.

These reductions fail under the Real APSP hypothesis or the Real 3SUM hypothesis. Pătraşcu's and Kopelowitz, Pettie, and Porat's reductions rely heavily on hashing, which does not seem to apply for Real-3SUM. Vassilevska W. and Xu's reduction uses Int-Exact-Tri as an intermediate problem. As discussed earlier, it is unclear how to reduce Real-APSP to Real-Exact-Tri. Even if such a reduction is possible, one still needs to replace a hashing trick used by Vassilevska W. and Xu with some other technique that works for Real-Exact-Tri.

We overcome these difficulties by designing conceptually very different reductions from before. On a very high level, instead of hashing, we use techniques inspired by "Fredman's trick" [37]. Using our new techniques, we obtain the following theorem. The resulting reduction also appears simpler than previous reductions, partly because we don't need to design and analyze any hash functions. For example, the entire proof of our reduction from Real-APSP to AE-Sparse-Tri fits in under two pages, and is included at the end of the introduction in Section 1.2. The reader is invited to browse through Vassilevska W. and Xu's longer, more complicated proof [57] for a comparison.

Theorem 1.4. AE-Sparse-Tri *on a graph with* m *edges requires*

- $m^{4/3-o(1)}$ *time assuming the Real APSP hypothesis;*
- $m^{5/4-o(1)}$ *time assuming the Real Exact-Triangle hypothesis;*
- $m^{6/5-o(1)}$ *time assuming the Real* 3*SUM hypothesis.*

Theorem 1.4 immediately implies Real APSP and Real 3SUM hardness for a large list of problems that were shown to be Integer APSP and Integer 3SUM hard via AE-Sparse-Tri, such as dynamic reachability, dynamic shortest paths and Pagh's problem [2, 49].

We obtain higher conditional lower bounds if we consider the counting version of AE-Sparse-Tri, #AE-Sparse-Tri, where one is given a graph with m edges, and is asked to output the number of triangles each edge is in. Note that the $O(m^{2\omega/(\omega+1)})$ time algorithm by Alon, Yuster, and Zwick [11] still works for #AE-Sparse-Tri. Therefore, the following theorem is tight if $\omega = 2$.

Theorem 1.5. #AE-Sparse-Tri *on a graph with m edges requires $m^{4/3-o(1)}$ time if at least one of the Real APSP hypothesis, the Real Exact-Triangle hypothesis or the Real 3SUM hypothesis is true.*

We actually obtain slightly stronger results than Theorem 1.4: we can reduce each of Real-APSP, Real-Exact-Tri and Real-3SUM to some number of instances of AE-Sparse-Tri. Consequently, we obtain Real APSP and Real 3SUM hardness for a problem called the AE-Mono-Tri (i.e. All-Edges Monochromatic Triangles, defined in Section 2), by combining a known reduction by Lincoln, Polak, and Vassilevska W. [47] from multiple instances of AE-Sparse-Tri to AE-Mono-Tri.

Finally, we will reduce AE-Mono-Tri to ACP-Tri-Co, thus proving the Real APSP and Real 3SUM hardness in Theorem 1.2.

1.1.3 Real APSP and OV Hardness via Colorful Boolean Matrix Multiplication. As a key problem in our second line of reductions, we define a natural generalization of the Boolean Matrix Multiplication problem, Colorful-BMM. In the Colorful-BMM problem, we are given an $n \times n$ Boolean matrix A and an $n \times n$ Boolean matrix B and a mapping $color : [n] \to \Gamma$. For each $i \in [n]$ and $j \in [n]$, we want to decide whether $\{color(k) : A[i,k] \wedge B[k,j],\ k \in [n]\} = \Gamma$. In other words, for every pair of i, j, we want to determine whether the witnesses cover all the colors.

We show that Real-APSP can be reduced to the Colorful-BMM problem. Our reduction uses an idea from Williams' algorithm for Real-APSP [60]: to compute the Min-Plus product of two real matrices, it suffices to compute several logical ANDs of ORs. We then show that these logical operations can be naturally reduced to Colorful-BMM. Since OV also has a similar formulation of logical ANDs of ORs [5], we similarly reduce OV to Colorful-BMM. We obtain the following theorem using this idea.

Theorem 1.6. Colorful-BMM *between two $n \times n$ matrices requires*

- $n^{2.25-o(1)}$ *time assuming the Real APSP hypothesis;*
- $n^{3-o(1)}$ *time assuming OVH.*

We also obtain Real-APSP and OV hardness for several natural matrix product problems, such as (distinct, =)-Product and (distinct, +)-Product (see the full paper for their definitions).

We will then reduce Colorful-BMM to ACP-Tri-Co, as detailed in the following.

1.1.4 Equivalence between Variants of Triangle Collection. We also show some equivalence results between variants of Tri-Co. These equivalences will be useful when we further reduce the previous two lines of reductions to ACP-Tri-Co to finish the proof of Theorem 1.2.

Triangle-Collection* (Tri-Co* for short) as defined by [4] is a "restricted" version of Tri-Co whose definition has two parameters t and p along with other details about the structure of the input graphs. All previous reductions from Tri-Co [4, 27] are actually from the Tri-Co* problem with small parameters $t, p \le n^{o(1)}$ (or $t, p \le n^{\varepsilon}$ for every $\varepsilon > 0$).

We consider a conceptually much simpler variant of Tri-Co, which we call Tri-Co$_{\text{light}}$. The input and output of Tri-Co$_{\text{light}}$ are the same as those of Tri-Co; however, we use a parameter which denotes an upper bound for the number of nodes that can share the same color ("light" means that the colors are light, i.e. have few vertices each).

We then show the following equivalence between Tri-Co* and Tri-Co$_{\text{light}}$.

Theorem 1.7. *(*ACP-*)* Tri-Co$_{\text{light}}$ *with parameter $n^{o(1)}$ and (*ACP-*)* Tri-Co* *with $t, p \le n^{o(1)}$ are equivalent up to $n^{o(1)}$ factors.*

Therefore, in order to reduce problems to Tri-Co* and thus to other problems known to be reducible from Tri-Co*, it suffices to reduce them to the much cleaner problem Tri-Co$_{\text{light}}$.

We obtain our main result, Theorem 1.2, by combining our previous reductions with Theorem 1.7.

First, by reducing AE-Mono-Tri to ACP-Tri-Co$_{\text{light}}$, we obtain Real-APSP and Real-3SUM hardness of ACP-Tri-Co*.

Second, Colorful-BMM easily reduces to ACP-Tri-Co, establishing the OV hardness of ACP-Tri-Co and yielding another route of reduction from the Real-APSP problem. However, Colorful-BMM doesn't seem to reduce to the more restricted problem ACP-Tri-Co*. By unrolling our reduction from OV to ACP-Tri-Co, we show that OV actually reduces to the original Tri-Co* problem.

Theorem 1.8. Tri-Co$_{\text{light}}$ *with parameter $n^{o(1)}$ (thus* Tri-Co* *with parameters $n^{o(1)}$ and* Tri-Co*) requires $n^{3-o(1)}$ time assuming OVH.*

Using Theorem 1.7 we are also able to prove the following surprising result:

Theorem 1.9. *If* Tri-Co* *with parameters $t, p \le n^{\varepsilon}$ for some $\varepsilon > 0$ has a truly subcubic time algorithm, then so does* Tri-Co.

Theorem 1.9 establishes a subcubic equivalence between Tri-Co and Tri-Co* with parameters n^{ε}, and in fact implies that all the known hardness results so far that were proven from Tri-Co* also hold from Tri-Co itself.

1.1.5 Other Reductions. Using our reductions and techniques, we also obtain the following list of interesting applications.

A hard colorful version of AE-Sparse-Triangle. We give a *tight* conditional lower bound under Hypothesis 2 for the parameterized time complexity of a natural variant of AE-Sparse-Tri. Specifically, in the AE-Colorful-Sparse-Tri problem, we are given a graph $G = (V, E)$ with m edges and a mapping $color : V \to \Gamma$. For each edge uv, we want to decide whether $\{color(w) : uwv \text{ is a triangle}\} = \Gamma$. When the degeneracy of the graph is m^{α}, we can clearly solve the problem in $\tilde{O}(m^{1+\alpha})$ time by enumerating all triangles in the graph [24]. We show that, under Hypothesis 2, for any constant $0 < \alpha \le 1/5$, no algorithm can solve AE-Colorful-Sparse-Tri in a graph with m edges and degeneracy $O(m^{\alpha})$ in $\tilde{O}(m^{1+\alpha-\varepsilon})$ time for $\varepsilon > 0$.

Real-to-integer reductions. It is an intriguing question whether we can base the hardness of a problem with integer inputs on the hardness of the same problem but with real inputs. For instance, it would be extremely interesting if one could show that the Real 3SUM hypothesis implies the Integer 3SUM hypothesis. We partially answer this question by showing two conditional lower bounds of this nature.

First, if Int-All-Nums-3SUM (a variant of Integer 3SUM where one needs to output whether each input number is in a 3SUM

solution) can be solved in $\tilde{O}(n^{6/5-\varepsilon})$ time for some $\varepsilon > 0$, then Real-All-Nums-3SUM can be solved in truly subquadratic time, falsifying the Real 3SUM hypothesis.

Second, if Int-AE-Exact-Tri, which is the "All-Edges" variant of Int-Exact-Tri, can be solved in $\tilde{O}(n^{7/3-\varepsilon})$ time for $\varepsilon > 0$, then Real-AE-Exact-Tri can be solved in truly subcubic time. We also show variants of this result such as an analogous result for the counting versions of these problems.

An application to string matching. In the pattern-to-text Hamming distance problem, we are given a text string $T = t_1 \cdots t_N$ and a pattern string $P = p_1 \cdots p_M$ in Σ^* with $M \le N$, and we want to compute for every $i = 0, \ldots, N - M$, the Hamming distance between P and $t_{i+1} \cdots t_{i+M}$, which is defined as $M - |\{j : p_j = t_{i+j}\}|$. The current best algorithm in terms of N runs in $\tilde{O}(N^{3/2})$ time [7] while unfortunately there isn't a matching conditional lower bound under standard hypotheses (though there is an unpublished non-matching $N^{\omega/2-o(1)}$ time conditional lower bound based on the presumed hardness of Boolean Matrix Multiplication that has been attributed to Indyk, see e.g. [41]).

We consider a similar string matching problem which we call pattern-to-text distinct Hamming similarity. The input to pattern-to-text distinct Hamming similarity is the same as the input to pattern-to-text Hamming distance, but for each $i = 0, \ldots, N - M$, we need to output $|\{p_j : p_j = t_{i+j}\}|$ instead. The $\tilde{O}(N^{3/2})$ time algorithm for pattern-to-text Hamming distance can be easily adapted to an $\tilde{O}(N^{3/2})$ time algorithm for pattern-to-text distinct Hamming similarity. Using our reduction from OV to Colorful-BMM, we show a matching $N^{3/2-o(1)}$ lower bound for pattern-to-text distinct Hamming similarity based on OVH.

Real APSP hardness of Set-Disjointness and Set-Intersection. The Set-Disjointness problem and the Set-Intersection problem are two generalized versions of AE-Sparse-Tri (see the full paper for their formal definitions). Kopelowitz, Pettie, and Porat [46] showed Integer 3SUM hardness of these two problems, and used them as intermediate steps for showing 3SUM hardness of many graph problems, such as Triangle Enumeration and Maximum Cardinality Matching. Later, Vassilevska W. and Xu [57] showed reductions from Int-Exact-Tri to these two problems, and thus obtained Integer APSP hardness for Set-Disjointness and Set-Intersection. By generalizing the techniques used in our reduction from Real-APSP to AE-Sparse-Tri, we obtain Real APSP hardness for these two problems. Therefore, all the hardness results shown by Kopelowitz, Pettie, and Porat [46] now also have Real APSP hardness.

1.2 An Illustration of Our Techniques: Reduction from Real-APSP to AE-Sparse-Tri

In this subsection, we will include our complete reduction from Real-APSP to AE-Sparse-Tri to exemplify how our techniques are different from, and simpler than, the previous hashing techniques [57].

Our main new insight is simple: we observe that Fredman's beautiful method for Real-APSP [37], which yielded an $\tilde{O}(n^{5/2})$-depth decision tree but not a truly subcubic time algorithm, can actually be converted to an efficient algorithm when given an oracle to AE-Sparse-Tri, if we combine the method with a standard randomized search trick and an interesting use of "dyadic intervals" (essentially corresponding to a one-dimensional "range tree" [29]).

We first recall the well-known fact [35] that Real-APSP is equivalent to computing the $(\min,+)$-product of two $n \times n$ real-valued matrices A and B, defined as the matrix C with $C[i,j] = \min_k(A[i,k] + B[k,j])$. This problem in turn reduces to $O(n/d)$ instances of computing the $(\min,+)$-product of an $n \times d$ matrix and $d \times n$ matrix.

We define the following intermediate problem, which we show is equivalent to the original problem by a random sampling trick:

Problem 1.10 (Real-$(\min,+)$-Product-Variant)**.** *We are given an $n \times d$ real matrix A and a $d \times n$ real matrix B, where $d \le n$. For each $i, j \in [n]$, we are also given an index $k_{ij} \in [d]$.*

For each $i, j \in [n]$, we want to find an index $k'_{ij} \in [d]$ (if it exists) satisfying

$$A[i,k'_{ij}] + B[k'_{ij},j] < A[i,k_{ij}] + B[k_{ij},j]. \tag{1}$$

Lemma 1.11. Real-$(\min,+)$-Product *of an $n \times d$ real matrix A and a $d \times n$ real matrix B reduces to $\tilde{O}(1)$ calls of an oracle for* Real-$(\min,+)$-Product-Variant *using Las Vegas randomization.*

Proof. We compute the $(\min,+)$-product of A and B as follows: take a random subset $R \subseteq [d]$ of size $d/2$. For each $i, j \in [n]$, first compute

$$k_{ij}^{(R)} = \arg\min_{k \in R}(A[i,k] + B[k,j]).$$

This can be done recursively by $(\min,+)$-multiplying an $n \times (d/2)$ and a $(d/2) \times n$ matrix.

Next, we initialize $k_{ij} = k_{ij}^{(R)}$. Then we invoke the oracle for Real-$(\min,+)$-Product-Variant to find some k'_{ij} satisfying (1) for each $i, j \in [n]$. If k'_{ij} exists, reset $k_{ij} = k'_{ij}$. Now, repeat. When k'_{ij} is nonexistent for all $i, j \in [n]$, we can stop, since we would have $k_{ij} = \arg\min_{k \in [d]}(A[i,k] + B[k,j])$ for all $i, j \in [n]$.

To bound the number of iterations, observe that for each $i, j \in [n]$, the number of indices $k' \in [d]$ satisfying $A[i,k'] + B[k',j] < A[i,k_{ij}^{(R)}] + B[k_{ij}^{(R)},j]$ is $O(\log n)$ w.h.p.[4] (since in a set of d values, at most $O(\log n)$ values are smaller than the minimum of a random subset of size $d/2$ w.h.p.). Thus, $O(\log n)$ iterations suffice w.h.p.

As the recursion has $O(\log d)$ depth, the total number of oracle calls is $O(\log d \log n)$ w.h.p. □

With Lemma 1.11, we are ready to present our main reduction from Real-$(\min,+)$-Product-Variant to AE-Sparse-Tri, which is inspired by "Fredman's trick" [37]—namely, the obvious but crucial observation that $a' + b' < a + b$ is equivalent to $a' - a < b - b'$.

Lemma 1.12. Real-$(\min,+)$-Product-Variant *reduces to $O(d)$ instances of* AE-Sparse-Tri *on graphs with $\tilde{O}(n^2/d + dn)$ edges.*

Proof. To solve Real-$(\min,+)$-Product-Variant, we first sort the following list of $O(d^2 n)$ elements in $O(d^2 n \log n)$ time:

$$L = \{A[i,k'] - A[i,k] : i \in [n],\ k,k' \in [d]\} \cup \{B[k,j] - B[k',j] : j \in [n],\ k,k' \in [d]\}.$$

[4] "w.h.p." is short for "with high probability", i.e., with probability $1 - O(1/n^c)$ for an arbitrarily large constant c.

Fix $k \in [d]$. Let $P_k = \{(i,j) \in [n]^2 : k_{ij} = k\}$. Divide P_k into $\left\lceil \frac{|P_k|}{n^2/d} \right\rceil$ subsets of size $O(n^2/d)$. Fix one such subset $P \subseteq P_k$. We solve AE-Sparse-Tri on the following tripartite graph $G_{k,P}$:

(0) The left nodes are $\{x[i] : i \in [n]\}$, the middle nodes are $\{y[k', I] : k' \in [d],\ I$ is a dyadic interval$\}$, and the right nodes are $\{z[j] : j \in [n]\}$. Here, a *dyadic* interval refers to an interval of the form $[2^s t, 2^s(t+1)) \subset [0, 4d^2 n)$ for nonnegative integers s and t (here, $4d^2n - 1$ is an upper bound for the smallest power of 2 that is larger than or equal to $2d^2n$). We don't explicitly enumerate all these nodes, but generate a node when we need to add an edge to it.
(1) For each $(i,j) \in P$, create an edge $x[i]\, z[j]$.
(2) For each $i \in [n]$, $k' \in [d]$, and each dyadic interval I, create an edge $x[i]\, y[k', I]$ if the rank of $A[i,k'] - A[i,k]$ in L is in the left half of I.
(3) For each $j \in [n]$, $k' \in [d]$, and each dyadic interval I, create an edge $y[k', I]\, z[j]$ if the rank of $B[k,j] - B[k',j]$ in L is in the right half of I.

Step 1 creates $O(n^2/d)$ edges. Steps 2–3 create $O(dn \log n)$ edges, since any fixed value lies in $O(\log n)$ dyadic intervals. Thus, the graph $G_{k,P}$ has $\tilde{O}(n^2/d + dn)$ edges. The total number of graphs is $\sum_{k \in [d]} \left\lceil \frac{|P_k|}{n^2/d} \right\rceil = O(d)$.

For any given $(i,j) \in [n]^2$, take $k = k_{ij}$ and P to be the subset of P_k containing (i,j). Finding a k' with $A[i,k'] + B[k',j] < A[i,k_{ij}] + B[k_{ij},j]$ is equivalent to finding a k' with $A[i,k'] - A[i,k] < B[k,j] - B[k',j]$, which is equivalent to finding a triangle $x[i]\, y[k', I]\, z[j]$ in the graph $G_{k,P}$. Here, we are using the following fact: for any two numbers $a, b \in [2d^2n]$, we have $a < b$ iff there is a (unique) dyadic interval I such that a lies on the left half of I and b lies on the right half of I. Thus, the answers to Real-(min, +)-Product-Variant can be deduced from the answers to the AE-Sparse-Tri instances. Note that we have assumed an equivalent version of the AE-Sparse-Tri problem with "witnesses", i.e., that outputs a triangle through each edge whenever such a triangle exists [31]. □

A near $m^{4/3}$ lower bound for AE-Sparse-Tri immediately follows, under the Real-APSP hypothesis; this matches known upper bounds for AE-Sparse-Tri if $\omega = 2$. The new proof not only strengthens the previous proof by Vassilevska W. and Xu [57], which reduces from integer APSP, but it is also simpler (not requiring more complicated hashing arguments).

Theorem 1.13. *If* AE-Sparse-Tri *could be solved in* $\tilde{O}(m^{4/3-\varepsilon})$ *time, then* Real-APSP *could be solved in* $\tilde{O}(n^{3-3\varepsilon/2})$ *time using Las Vegas randomization.*

Proof. By combining the above two lemmas, if AE-Sparse-Tri could be solved in $T(m)$ time, then the (min, +)-product of an $n \times d$ and a $d \times n$ real matrix could be computed in $\tilde{O}(d \cdot T(n^2/d + dn))$ time. The (min, +)-product of two $n \times n$ real matrices, and thus Real-APSP, reduce to n/d instances of such rectangular products and could then be computed in $\tilde{O}(n \cdot T(n^2/d + dn))$ time. By choosing $d = \sqrt{n}$, the time bound becomes $\tilde{O}(n \cdot T(n^{3/2})) = \tilde{O}(n^{3-3\varepsilon/2})$ if $T(m) = \tilde{O}(m^{4/3-\varepsilon})$. □

Our reduction from Real-3SUM to AE-Sparse-Tri (see Section 3.2) is based on a similar insight: we observe that Grønlund and Pettie's elegant method (based on Fredman's work), which yielded an $\tilde{O}(n^{3/2})$-depth decision tree for Real-3SUM, can also be converted to an efficient algorithm when given an oracle to AE-Sparse-Tri. Since Grønlund and Pettie's method is a bit cleverer, this reduction is technically more challenging (though it is still comparable in simplicity with Pătraşcu's original reduction from Int-3SUM [49]), and because of these extra complications, the resulting conditional bounds are not tight. Nevertheless, it is remarkable that nontrivial conditional lower bounds can be obtained at all, and that we do get tight bounds for a certain range of degeneracy values, and tight bounds for reductions to the counting variant of AE-Sparse-Tri, when $\omega = 2$. (For applications to some of the dynamic graph problems, the earlier polynomial lower bounds weren't tight anyways.)

Our reduction from Real-APSP to Colorful-BMM (see Section 4) is even simpler (under one page long), and is inspired by ideas from Williams' Real-APSP algorithm [59], also based on Fredman's trick. Our reduction from OV to Colorful-BMM is more straightforward, but we view the main innovation here to be the introduction of the Colorful-BMM problem itself, which we hope will find further applications. Sometimes, the key to conditional lower bound proofs lies in formulating the right intermediate subproblems. Our combined reduction from OV to ACP-Tri-Co via Colorful-BMM is essentially as simple as Abboud, Vassilevska W. and Yu's original reduction from 3-OV to Tri-Co [4], but adds more understanding of the Tri-Co and related problems as it reveals that a weaker hypothesis is sufficient to yield conditional lower bounds for all the dynamic graph problems in Corollary 1.3.

1.3 Paper Organization

In Section 2, we define necessary notations. In Section 3, we show hardness of AE-Sparse-Tri under the Real 3SUM and the Real APSP hypotheses. In Section 4, we show hardness of Colorful-BMM based on the Real APSP and the OV hypotheses. We prove our main theorem in Section 5, by reducing problems in Section 3 and Section 4 to variants of Tri-Co. We defer the following proofs to the full paper: hardness of AE-Sparse-Tri under the Real Exact-Triangle hypothesis, hardness of #AE-Sparse-Tri under the Real 3SUM and the Real Exact-Triangle hypotheses, hardness of AE-Mono-Tri (and its counting version) under the Real 3SUM, the Real APSP and the Real Exact-Triangle hypotheses. We also defer further applications of our techniques and results, including Real-to-integer reductions and conditional hardness for pattern-to-text distinct Hamming similarity, Set-Disjointness and Set-Intersection, to the full paper.

Table 1 summarizes our main lower bound results.

2 PRELIMINARIES

In this section, we give definitions and abbreviations of terminologies and problems we will consider throughout the paper. We group related definitions together for easier navigation.

Some problems require to output whether each edge in a given graph is in some triangle with certain property. We use the prefix AE- (short for "All-Edges-") to denote that we need to output a Boolean value for each edge indicating whether each edge is in such a triangle. For these problems, we can instead use the prefix

Table 1: Summary of some of our results and previous results. Ignoring $n^{o(1)}$ factors, entries marked † match known upper bounds assuming $\omega = 2$; entries marked ‡ match known upper bounds without the assumption.

Problems	Lower Bounds		Hypotheses	References
AE-Sparse-Tri	$m^{4/3-o(1)}$	†	Int-3SUM	[49]
	$m^{4/3-o(1)}$	†	Int-APSP, Int-Exact-Tri	[57]
	$m^{6/5-o(1)}$		Real-3SUM	Thm. 3.6
	$m^{5/4-o(1)}$		Real-Exact-Tri	Full paper
	$m^{4/3-o(1)}$	†	Real-APSP	Thm. 1.13
#AE-Sparse-Tri	$m^{4/3-o(1)}$	†	Real-3SUM	Full paper
	$m^{4/3-o(1)}$	†	Real-Exact-Tri	Full paper
AE-Mono-Tri	$n^{5/2-o(1)}$	†	Int-3SUM	[47]
	$n^{5/2-o(1)}$	†	Int-APSP	[57]
	$n^{9/4-o(1)}$		Real-3SUM	Full paper
	$n^{7/3-o(1)}$		Real-Exact-Tri	Full paper
	$n^{5/2-o(1)}$	†	Real-APSP	Full paper
#AE-Mono-Tri	$n^{5/2-o(1)}$	†	Real-Exact-Tri	Full paper
	$n^{5/2-o(1)}$	†	Real-3SUM	Full paper
Colorful-BMM	$n^{9/4-o(1)}$		Real-APSP	Thm. 4.2
	$n^{3-o(1)}$	‡	OV	Thm. 4.4
Tri-Co	$n^{3-o(1)}$	‡	Int-APSP, Int-3SUM, SETH	[4]
ACP-Tri-Co	$n^{9/4-o(1)}$		Real-3SUM	Full paper
	$n^{5/2-o(1)}$		Real-APSP	Full paper
	$n^{3-o(1)}$	‡	OV	Cor. 5.3
Int-All-Nums-3SUM	$n^{6/5-o(1)}$		Real-3SUM	Full paper
Int-AE-Exact-Tri	$n^{7/3-o(1)}$		Real-AE-Exact-Tri	Full paper
pattern-to-text distinct Hamming similarity	$n^{3/2-o(1)}$	‡	OV	Full paper

#AE- to indicate that we need to count the number of such triangles involving each edge.

2.1 Fine-Grained Reductions

We use the notion of fine-grained reduction [54, 56] which is as follows. Let A and B be problems, and let $a(n)$ and $b(n)$ be running time functions where n is the input size or a suitable measure related to the input size such as the number of nodes in a graph.

We say that A is (a, b)-fine-grained reducible to B if for every $\varepsilon > 0$ there is a $\delta > 0$ and an $O(a(n)^{1-\delta})$ time algorithm that solves size (or size measure) n instances of A making calls to an oracle for problem B, so that the sizes (or size measures) of the instances of B in the oracle calls are $n_1, n_2, \ldots, n_k$ and $\sum_{i=1}^{k}(b(n_i))^{1-\varepsilon} \leq a(n)^{1-\delta}$.

If A is (a, b)-fine-grained reducible to B, and there is an $O(b(n)^{1-\varepsilon})$ time algorithm for B for some $\varepsilon > 0$, then there is also an $O(a(n)^{1-\delta})$ time algorithm for some $\delta > 0$.

All the reductions in the paper will be fine-grained, and hence we will often omit "fine-grained" when we say reduction.

2.2 Hard Problems

In this section, we define the problems that we will consider as our hardness sources. In case a problem P has numbers in the input, we will define the generic version of problem P and use Int-P to denote the version where the input numbers are integers from $\pm[n^c]$ for some sufficiently large constant c, and use Real-P to denote the version where the input numbers are reals.

Problem 2.1 (3SUM). *Given three sets A, B, C of numbers of size n, determine whether there exist 3 numbers $a \in A, b \in B, c \in C$ such that $a + b + c = 0$.*

Problem 2.2 (All-Nums-3SUM). *Given three sets A, B, C of numbers of size n, for each $c \in C$, determine whether there exist $a \in A, b \in B$ such that $a + b + c = 0$.*

Int-All-Nums-3SUM (resp. Real-All-Nums-3SUM) and Int-3SUM (resp. Real-3SUM) are subquadratically equivalent [56]. Note that a "one-set" version of 3SUM (instead of the above "three-sets" version) has also been used in the literature, but they are known to be equivalent. Often, it is more convenient to consider the version of the problem where the third set is negated, in which case we are seeking a triple (a, b, c) with $a + b = c$.

Problem 2.3 (APSP). *Given an n-node directed graph whose edge weights are given as numbers and which has no negative cycles, compute the shortest path distances from u to v for every pair of nodes u and v in the graph.*

Problem 2.4 ($(\min, +)$-Product). *Given an $n \times d$ matrix A and a $d \times n$ matrix B whose entries are given as numbers, compute an $n \times n$ matrix C such that $C[i, j] = \min_{k \in [d]}(A[i, k] + B[k, j])$.*

It is known [35, 56] that Real-APSPand Real-$(\min, +)$-Product between $n \times n$ real-valued matrices are subcubically equivalent; similarly, Int-APSPand Int-$(\min, +)$-Product between $n \times n$ integer-valued matrices are subcubically equivalent as well.

Problem 2.5 (AE-Exact-Triangle (AE-Exact-Tri)). *Given an n-node graph whose edge weights are given as numbers, for each edge determine whether there is a triangle containing that edge whose edge weights sum up to zero.*

It is known that the Integer 3SUM and the Integer APSP hypotheses imply that Int-AE-Exact-Tri requires $n^{3-o(1)}$ time [55]. However, such tight reductions aren't known for their real variants. However, we can adapt one previous non-tight reduction from Int-3SUM to Int-AE-Exact-Tri [55] to the real case, which implies Real-AE-Exact-Tri requires $n^{2.5-o(1)}$ time assuming the Real 3SUM hypothesis (see more details in the full paper).

Problem 2.6 (OV). *Given a set of n Boolean vectors in f dimensions, determine whether the set contains two vectors that are orthogonal.*

2.3 Triangle Collection and Variants

We define a tripartite version of Triangle-Collection, which is slightly different from the original definition of Abbound, Vassilevska W. and Yu [4]. In the full version of the paper, we will show these two definitions are equivalent.

For brevity, we will give short names for Triangle-Collection and Triangle-Collection*, Tri-Co and Tri-Co* respectively. Similar abbreviations will also be given for the All-Color-Pairs versions.

Problem 2.7 (Triangle-Collection (Tri-Co)). *Given a tripartite graph $G = (V, E)$ on partitions A, B, C such that the colors of the nodes in A are from a set K_A, the colors of the nodes in B are from a set K_B and the colors of the nodes in C are from a set K_C, where $K_A \cap K_B \cap K_C = \emptyset$, and one needs to determine whether for all triples of colors $a \in K_A, b \in K_B, c \in K_C$ there exists some triangle $x, y, z \in V$ such that $color(x) = a, color(y) = b, color(z) = c$.*

Problem 2.8 (ACP-Triangle-Collection (ACP-Tri-Co)). *Given the input to a* Tri-Co *instance, one needs to determine for every $a \in K_A, b \in K_B$, whether for all $c \in K_C$ there exists some triangle $x, y, z \in V$ such that $color(x) = a, color(y) = b, color(z) = c$.*

All other variants of Tri-Co will have the same output as Tri-Co, so they also naturally have an All-Color-Pairs (ACP) variant. We will only define their normal versions for conciseness.

Problem 2.9 (Triangle-Collection* (Tri-Co*)). *An* Tri-Co* *instance is a restricted* Tri-Co*instance. For parameters p and t it is a node-colored graph G which is a disjoint union of graphs $G_1, \ldots, G_t$. G (and hence all the G_is) is tripartite on partitions A, B, C. The aforementioned value p is an upper bound on the number of nodes of any particular color in any G_i.*

The node colors are from $[3] \times [n]$. For every t, the nodes of G_t are:

- *Nodes in A of the form (a, t) of color $(1, a)$. Note that this means that each G_t has nodes of distinct colors in A.*
- *Nodes in B of the form (b, t, j) of color $(2, b)$, where $j \leq p$. For each $(a, t) \in A$ and every color $(2, b)$, there is at most one node (b, t, j) in B that (a, t) has an edge to.*
- *Nodes in C of the form (c, t, j) of color $(3, c)$ for $j \leq p$. For each $(a, t) \in A$ and every color $(3, c)$, there is at most one node (c, t, j) in C that (a, t) has an edge to.*

The last two bullets mean that in each G_t all the neighbors of a node in A have distinct colors. There is no restriction on the edges between nodes in B and C (beyond that the graphs G_i are disjoint).

An algorithm for Tri-Co* *needs to output whether for all triples (a, b, c), there is a triangle with node colors $(1, a), (2, b), (3, c)$.*

We now define a another version of Tri-Co , Tri-Co$_{\text{light}}$, that has Tri-Co* as a special case. The "light" part of the name stands for "Light Colors", meaning that each colors has few nodes.

Problem 2.10 (Tri-Co$_{\text{light}}$). *An instance of* Tri-Co$_{\text{light}}$ *is a restricted instance of* Tri-Co. *For an integer parameter p, it is a graph that has at most p nodes of any fixed color.*

2.4 Other Problems

Problem 2.11 (AE-Sparse-Tri. (AE-Sparse-Tri)). *Given a graph with m edges, for each edge determine whether it is in a triangle.*

Problem 2.12 (AE-Monochromatic-Tri. (AE-Mono-Tri)). *Given a graph on n nodes where each edge has a color, for each edge determine whether it is in a triangle whose three edges share the same color.*

In some of our reductions, it is convenient to assume that in AE-Sparse-Tri, for each edge where the answer is yes, a witness (a triangle through the edge) must also be provided. This version is equivalent (up to poly-logarithmic factors) by standard random sampling techniques for witness finding [10, 51], which requires only Las Vegas randomization. The same is true for AE-Mono-Tri as well.

Problem 2.13 (Colorful Boolean Matrix Mult. (Colorful-BMM)). *Given an $n \times n$ Boolean matrix A and an $n \times n$ Boolean matrix B and a mapping $color : [n] \to \Gamma$, for each $i \in [n]$ and $j \in [n]$, decide whether $\{color(k) : A[i, k] \wedge B[k, j],\ k \in [n]\} = \Gamma$.*

3 HARDNESS OF ALL-EDGES SPARSE TRIANGLE

In this section, we show lower bounds of AE-Sparse-Tri based on the conjectured hardness of Real-APSP or Real-3SUM.

3.1 Real-APSP → AE-Sparse-Tri

Recall that we proved Theorem 1.13 in Section 1.2:

Theorem 1.13. *If* AE-Sparse-Tri *could be solved in $\tilde{O}(m^{4/3-\varepsilon})$ time, then* Real-APSP *could be solved in $\tilde{O}(n^{3-3\varepsilon/2})$ time using Las Vegas randomization.*

More generally, we can obtain a near mD conditional lower bound in terms of the degeneracy D, if $D \ll m^{1/3}$, which again matches known upper bounds (regardless of the value of ω).

Theorem 3.1. *If* AE-Sparse-Tri *for graphs with m edges and degeneracy $\tilde{O}(m^{\alpha})$ could be solved in $\tilde{O}(m^{1+\alpha-\varepsilon})$ time for some constant $\alpha \leq 1/3$, then* Real-APSP *could be solved in $\tilde{O}(n^{3-2\varepsilon/(1+\alpha)})$ time using Las Vegas randomization.*

Proof. First, observe that the graph $G_{k,P}$ in Lemma 1.12's proof can be modified to have degeneracy $\tilde{O}(d)$: Whenever a left node x has $\Delta_x > d$ neighbors among the right nodes, we split x into $\left\lceil \frac{\Delta_x}{d} \right\rceil$ copies, where each copy is linked to up to d neighbors among the right nodes. Each copy is also linked to all of the original $O(d \log n)$ neighbors among the middle nodes. The number of left nodes increases to $\sum_x \left\lceil \frac{\Delta_x}{d} \right\rceil$, and so the number of edges increases by

$\tilde{O}(d\sum_x \lceil \frac{\Delta_x}{d} \rceil) = \tilde{O}(\sum_x \Delta_x + dn) = \tilde{O}(n^2/d + dn)$. Now, each left node has $\tilde{O}(d)$ neighbors among the middle and right nodes. Each right node has $\tilde{O}(d)$ neighbors among the middle nodes. It follows that the degeneracy of the modified graph is $\tilde{O}(d)$.

To prove the theorem, choose d so that $d = (n^2/d)^\alpha$, i.e., $d = n^{2\alpha/(1+\alpha)}$. Since $\alpha \le 1/3$, we have $d \le \sqrt{n}$ and so $dn \le n^2/d$. If AE-Sparse-Tri with m edges and degeneracy D could be solved in $T(m, D)$ time, then Real-APSP could be solved in time $\tilde{O}(n \cdot T(n^2/d, d)) = \tilde{O}(n^{3-2\varepsilon/(1+\alpha)})$ if $T(m) = \tilde{O}(m^{1+\alpha-\varepsilon})$. □

3.2 Real-3SUM → AE-Sparse-Tri

We next adapt our proof to reduce from Real-3SUM. We will more generally consider an asymmetric version of Real-3SUM with three sets A, B, and C of sizes n, n, and $\hat{n}$ respectively with $\hat{n} \le n$ (the standard version has $\hat{n} = n$). Sort A and B and divide the sorted lists of A and B into sublists $A_1, \ldots, A_{n/d}$ and $B_1, \ldots, B_{n/d}$ of size d, for a given parameter $d \le n$. Let $A[i, k]$ denote the k-th element of A_i for each $i \in [n/d]$ and $k \in [d]$, and $B[j, \ell]$ denote the ℓ-th element of B_j for each $j \in [n/d]$ and $\ell \in [d]$. We will consider a slightly stronger problem: for each $c \in C$, find the predecessor and successor of c among $A + B$. (If c is in this set, we will deviate from convention and define the predecessor of c to be itself.) By a known observation (which was used in Grønlund and Pettie's Real-3SUM algorithm [43] and in subsequent algorithms [21]), for each $c \in C$, it suffices to search for c in $A_i + B_j$ for $O(n/d)$ pairs (i, j) (since these (i, j) pairs form a "staircase" in the $[n/d] \times [n/d]$ grid). Thus, Real-3SUM (or Real-All-Nums-3SUM) reduces to the following problem (which was explicitly formulated, for example, in Chan's paper on Real-3SUM [21]):

Problem 3.2 (Real-3SUM-Variant$_1$)**.** *We are given an $(n/d) \times d_A$ real matrix A and an $(n/d) \times d_B$ real matrix B with $d_A, d_B \le d$. For each $i, j \in [n/d]$, we are also given a set C_{ij} of real numbers with $\sum_{i,j} |C_{ij}| = O(\hat{n}n/d)$.*

For each $i, j \in [n/d]$ and each $c \in C_{ij}$, we want to find the predecessor and successor of the value c among the elements in $\{A[i, k] + B[j, \ell] : k \in [d_A], \ell \in [d_B]\}$.

We again introduce an intermediate problem, which the original problem reduces to via random sampling:

Problem 3.3 (Real-3SUM-Variant$_2$)**.** *We are given an $(n/d) \times d$ real matrix A and an $(n/d) \times d$ real matrix B. For each $i, j \in [n/d]$, we are also given a set $Q_{ij} \subseteq [d]^4$ of quadruples with $\sum_{i,j} |Q_{ij}| = O(\hat{n}n/d)$.*

For each $i, j \in [n/d]$ and each $q = (k^-, \ell^-, k^+, \ell^+) \in Q_{ij}$, we want to find indices $k'_{ijq}, \ell'_{ijq} \in [d]$ (if they exist) satisfying

$$A[i, k^-] + B[j, \ell^-] < A[i, k'_{ijq}] + B[j, \ell'_{ijq}] < A[i, k^+] + B[j, \ell^+]. \quad (2)$$

Lemma 3.4. Real-3SUM-Variant$_1$ *reduces to $\tilde{O}(1)$ calls to an oracle for* Real-3SUM-Variant$_2$ *using Las Vegas randomization.*

Proof. We solve Real-3SUM-Variant$_1$ as follows: Take a random subset $R \subseteq [d_A]$ of size $d_A/2$. For each $i, j \in [n/d]$ and each $c \in C_{ij}$, first compute indices $k^{-(R)}_{ijc}, \ell^{-(R)}_{ijc}, k^{+(R)}_{ijc}, \ell^{+(R)}_{ijc} \in [d]$ such that $A[i, k^{-(R)}_{ijc}] + B[j, \ell^{-(R)}_{ijc}]$ and $A[i, k^{+(R)}_{ijc}] + B[j, \ell^{+(R)}_{ijc}]$ are the predecessor and successor (respectively) of the value c among the elements in $\{A[i, k] + B[j, \ell] : k \in R, \ell \in [d_B]\}$. This computation can be done recursively.

Next, initialize $(k^-_{ijc}, \ell^-_{ijc}) = (k^{-(R)}_{ijc}, \ell^{-(R)}_{ijc})$ and $(k^+_{ijc}, \ell^+_{ijc}) = (k^{+(R)}_{ijc}, \ell^{+(R)}_{ijc})$. Invoke the oracle for Real-3SUM-Variant$_2$ to find some $(k'_{ijc}, \ell'_{ijc}) \in [d_A] \times [d_B]$ satisfying (2) for each $i, j \in [n]$ and $c \in C_{ij}$. If (k'_{ijc}, ℓ'_{ijc}) exists and $A[i, k'_{ijc}] + B[j, \ell'_{ijc}] \le c$, find the index $\ell \in [d_B]$ such that $A[i, k'_{ijc}] + B[j, \ell]$ is the predecessor of the value c among the elements in $\{A[i, k'_{ijc}] + B[j, \ell] : \ell \in [d_B]\}$. This index can be found in $O(\log d)$ time by binary search, assuming that each row of B has been sorted (which requires only $O(n \log n)$ preprocessing time). Reset $(k^-_{ijc}, \ell^-_{ijc}) = (k'_{ijc}, \ell)$. If (k'_{ijc}, ℓ'_{ijc}) exists and $A[i, k'_{ijc}] + B[j, \ell'_{ijc}] > c$, we proceed similarly, replacing "predecessor" with "successor" and $-$ superscripts with $+$. Now, repeat. When (k'_{ijc}, ℓ'_{ijc}) is nonexistent for all $i, j \in [n]$, we can stop.

To bound the number of iterations, observe that for each $i, j \in [n]$ and each $c \in C_{ij}$, the number of indices $k' \in [d_A]$ such that the predecessor of c among $\{A[i, k'] + B[j, \ell] : \ell \in [d_B]\}$ is greater than $A[i, k^{-(R)}_{ijc}] + B[\ell, k^{-(R)}_{ijc}]$ is $O(\log n)$ w.h.p (since in a set of d_A values, there are at most $O(\log n)$ values greater than the maximum of a random subset of size $d_A/2$). A similar statement, replacing "predecessor" with "successor", $-$ superscripts with $+$, and "greater" with "less", holds. Thus, $O(\log n)$ iterations suffice w.h.p.

As the recursion has $O(\log d_A)$ depth, the total number of oracle calls is $O(\log d \log n)$ w.h.p., and the extra cost of the binary searches is $O((\hat{n}n/d) \log^2 d \log n)$, which is negligible. □

We now present our reduction from Real-3SUM-Variant$_2$ to AE-Sparse-Tri, which is inspired by Grønlund and Pettie's work [43] and is also based on Fredman's trick. Although we now need to work with even more indices, the basic idea is similar to our earlier reductions. The extra complications cause further loss of efficiency (but we still obtain a tight conditional lower bound for AE-Sparse-Tri, albeit for a more restricted range of degeneracy $\ll m^{1/5}$). The whole proof is still simple (comparable to the original reductions from Int-3SUM by Pătraşcu [49] or Kopelowitz, Pettie and Porat [46], but completely bypassing hashing arguments).

Lemma 3.5. Real-3SUM-Variant$_2$ *reduces to a single instance of* AE-Sparse-Tri *on a graph with $\tilde{O}(\hat{n}n/d + d^2 n)$ edges and degeneracy $\tilde{O}(d)$.*

Proof. To solve Real-3SUM-Variant$_2$, first sort the following list of $O(dn)$ elements in $O(dn \log n)$ time:

$$L = \{A[i, k'] - A[i, k] : i \in [n/d],\ k, k' \in [d]\} \cup \{B[j, \ell] - B[j, \ell'] : j \in [n/d],\ \ell, \ell' \in [d]\}.$$

We then solve AE-Sparse-Tri on a tripartite graph G which is defined as follows:

(0) The left nodes are $\{x[i, k^-, k^+] : i \in [n/d],\ k^-, k^+ \in [d]\}$, the middle nodes are $\{y[I^-, I^+] : I^-, I^+ \text{ are dyadic intervals}\}$, and the right nodes are $\{z[j, \ell^-, \ell^+] : j \in [n/d], \ell^-, \ell^+ \in [d]\}$.
(1) For each $i, j \in [n/d]$ and $(k^-, \ell^-, k^+, \ell^+) \in Q_{ij}$, create an edge $x[i, k^-, k^+]\, z[j, \ell^-, \ell^+]$.
(2) For each $i \in [n/d]$, $k^-, k^+, k' \in [d]$, and dyadic intervals I^- and I^+, create an edge $x[i, k^-, k^+]\, y[I^-, I^+]$ if the rank of

$A[i,k^-] - A[i,k']$ in L is in the left half of I^- and the rank of $A[i,k'] - A[i,k^+]$ in L is in the left half of I^+.

(3) For each $j \in [n/d]$, $\ell^-, \ell^+, \ell' \in [d]$, and dyadic intervals I^- and I^+, create an edge $y[I^-, I^+]\, z[j, \ell^-, \ell^+]$ if the rank of $B[j,\ell'] - B[j,\ell^-]$ in L is in the right half of I^- and the rank of $B[j,\ell^+] - B[j,\ell']$ in L is in the right half of I^+.

Step 1 creates $\sum_{i,j} |Q_{ij}| = O(\hat{n}n/d)$ edges. Steps 2–3 create $O((n/d)d^3 \log^2 n)$ edges, since any fixed value lies in $O(\log n)$ dyadic intervals. Thus, the graph G has $\tilde{O}(\hat{n}n/d + d^2 n)$ edges.

For any given $(i,j) \in [n]^2$ and $(k^-,\ell^-,k^+,\ell^+) \in Q_{ij}$, finding a (k',ℓ') with $A[i,k^-] + B[j,\ell^-] < A[i,k'] + B[j,\ell'] < A[i,k^+] + B[j,\ell^+]$ is equivalent to finding a (k',ℓ') with $A[i,k^-] - A[i,k'] < B[j,\ell'] - B[j,\ell^-]$ and $A[i,k'] - A[i,k^+] < B[j,\ell^+] - B[j,\ell']$, which is equivalent to finding a triangle $x[i,k^-,k^+]\, y[I^-,I^+]\, z[j,\ell^-,\ell^+]$ in the graph G. Thus, the answers to Real-3SUM-Variant$_2$ can be deduced from the answers to the AE-Sparse-Tri instances.

Finally, to ensure that the graph G has degeneracy $\tilde{O}(d)$, we modify the graph by splitting nodes in the same way as in the last paragraph of Theorem 3.1's proof. The number of edges increases by $\tilde{O}(d \sum_x \left\lceil \frac{\Delta_x}{d} \right\rceil) = \tilde{O}(\sum_x \Delta_x + (n/d)d^2 \cdot d) = \tilde{O}(\hat{n}n/d + d^2 n)$. □

Theorem 3.6. *If* AE-Sparse-Tri *with m edges could be solved in $\tilde{O}(m^{6/5-\varepsilon})$ time, then* Real-3SUM *(and* Real-All-Nums-3SUM*) could be solved in $\tilde{O}(n^{2-5\varepsilon/3})$ time using Las Vegas randomization.*

More generally, if AE-Sparse-Tri *with m edges and degeneracy $\tilde{O}(m^\alpha)$ could be solved in $\tilde{O}(m^{1+\alpha-\varepsilon})$ time for some constant $\alpha \le \beta/(3+2\beta)$, then* Real-3SUM *for three sets of sizes n, n, and $\hat{n} = n^\beta$ ($\beta \le 1$) could be solved in $\tilde{O}(n^{1+\beta-\varepsilon(1+\beta)/(1+\alpha)})$ time using Las Vegas randomization.*

Proof. By combining the above two lemmas, if AE-Sparse-Tri could be solved in $T(m,D)$ time, then Real-3SUM-Variant$_1$ could be solved in $\tilde{O}(T(\hat{n}n/d + d^2 n,\, d))$ time. Also, Real-3SUM (and more generally, Real-All-Nums-3SUM) reduces to a single instance of Real-3SUM-Variant$_1$, with $\hat{n} = n^\beta$. (For the original symmetric version of Real-3SUM, $\beta = 1$.) Choose d so that $d = (n^{1+\beta}/d)^\alpha$, i.e., $d = n^{(1+\beta)\alpha/(1+\alpha)}$. Since $\alpha \le \beta/(2+3\beta)$, we have $d \le n^{\beta/3}$ and so $d^2 n \le n^{1+\beta}/d$. The time bound becomes $\tilde{O}(T(n^{1+\beta}/d, d)) = \tilde{O}(n^{1+\beta-\varepsilon(1+\beta)/(1+\alpha)})$ if $T(m, m^\alpha) = \tilde{O}(m^{1+\alpha-\varepsilon})$. □

4 HARDNESS OF COLORFUL BMM

In this section, we show that Real-APSP and OV can be reduced to Colorful-BMM.

4.1 Real-APSP → Colorful-BMM

We first present our reduction from Real-APSP to Colorful-BMM, which is simple and is inspired by Williams's Real-APSP algorithm [60] (and its derandomization by Chan and Williams [22]) using ANDs of ORs.

Lemma 4.1. Real-(min, +)-Product *of an $n \times d$ real matrix A and a $d \times n$ real matrix B reduces to one instance of* Colorful-BMM *for an $n \times \tilde{O}(d^4 n^\varepsilon)$ and an $\tilde{O}(d^4 n^\varepsilon) \times n$ Boolean matrix with $O(d)$ colors (and $\tilde{O}(d^2 n)$ nonzero input entries), after spending $\tilde{O}(dn^{2-\varepsilon})$ time.*

Proof. For simplicity, we assume that $A[i,k] + B[k,j] \ne A[i,k'] + B[k',j]$ for all $i,j \in [n]$ and $k,k' \in [d]$. This can be ensured, for example, by adding $k\delta$ to $A[i,k]$ for an infinitesimally small $\delta > 0$. Note that we don't need to explicitly store $A[i,k] + k\delta$. Instead, we can store a number $a + b\delta$ as a pair of numbers (a,b). Every time we need to compare two numbers (a_1,b_1) and (a_2,b_2), we first compare a_1 and a_2 and only compare b_1 and b_2 if $a_1 = a_2$. This implementation only incurs a constant factor overhead.

Fix $t \in [\log d]$. We will describe how to compute the t-th bit of $k_{ij} = \arg\min_{k \in [d]} (A[i,k] + B[k,j])$. Let $K_t = \{k \in [d] : \text{the } t\text{-th bit of } k \text{ is } 1\}$. Note that the t-th bit of k_{ij} is 1 iff

$$\bigwedge_{k \in [d]-K_t} \bigvee_{k' \in K_t} \big[A[i,k'] + B[k',j] < A[i,k] + B[k,j]\big]$$
$$= \bigwedge_{k \in [d]-K_t} \bigvee_{k' \in K_t} \big[A[i,k'] - A[i,k] < B[k,j] - B[k',j]\big].$$

Thus, the answers can be determined by solving Colorful-BMM on the following matrices $\mathcal{A}$ and $\mathcal{B}$ and color mapping to $[d] - K_t$:

(1) For each $i \in [n]$, $k \in [d] - K_t$, $k' \in K_t$, and each dyadic interval I, let $\mathcal{A}[i,(k,k',I)] = 1$ iff the rank of $A[i,k'] - A[i,k]$ lies in the left half of the dyadic interval I.

(2) For each $j \in [n]$, $k \in [d] - K_t$, $k' \in K_t$, and each dyadic interval I, let $\mathcal{B}[(k,k',I),j] = 1$ iff the rank of $B[k,j] - B[k',j]$ lies in the right half of the dyadic interval I.

(3) Define $color((k,k',I)) = k$.

However, the inner dimension (i.e., the number of columns of $\mathcal{A}$ or rows of $\mathcal{B}$) is large. We lower the inner dimension by using the "high vs. low degree" trick: Label a triple (k,k',I) "high" if the number of indices i for which $\mathcal{A}[i,(k,k',I)] = 1$ exceeds $n^{1-\varepsilon}/d^2$; otherwise, label it "low". For each low triple (k,k',I), we enumerate all (i,j) for which $\mathcal{A}[i,(k,k',I)] = 1$ and $\mathcal{B}[(k,k',I),j] = 1$, and mark these pairs (i,j) as "bad". There are $O((d^2 n \log n) \cdot n^{1-\varepsilon}/d^2) = \tilde{O}(n^{2-\varepsilon})$ bad pairs, and for each bad pair (i,j), we can compute its corresponding entry of the (min, +)-product in $O(d)$ time by brute force. This takes $\tilde{O}(dn^{2-\varepsilon})$ time. For the remaining good pairs (i,j), it suffices to keep only the high triples, and the number of high triples is $O(\frac{d^2 n \log n}{n^{1-\varepsilon}/d^2}) = \tilde{O}(d^4 n^\varepsilon)$. So, in the remaining Colorful-BMM instance, $\mathcal{A}$ and $\mathcal{B}$ have dimensions $\tilde{O}(n) \times \tilde{O}(d^4 n^\varepsilon)$ and $\tilde{O}(d^4 n^\varepsilon) \times \tilde{O}(n)$. □

Theorem 4.2. *If* Colorful-BMM *for two $n \times n$ Boolean matrices could be solved in $\tilde{O}(n^{9/4-5\varepsilon/4})$ time, then* Real-APSP *could be solved in $\tilde{O}(n^{3-\varepsilon})$ time.*

More generally, if Colorful-BMM *for two $n \times n$ Boolean matrices with $O(n^\alpha)$ colors could be solved in $\tilde{O}(n^{2+\alpha-\varepsilon})$ time for some constant $\alpha \le (1-\varepsilon)/4$, then* Real-APSP *could be solved in $\tilde{O}(n^{3-\varepsilon})$ time.*

Proof. The (min, +)-product of two $n \times n$ real matrices, and thus Real-APSP, reduces to n/d rectangular (min, +)-products between $n \times d$ matrices and $d \times n$ matrices. Choose $d = n^\alpha$. Since $\alpha \le (1-\varepsilon)/4$, we have $d^4 n^\varepsilon \le n$. Then the theorem is immediately implied by Lemma 4.1. □

4.2 OV → Colorful-BMM

We can similarly reduce OV to Colorful-BMM, since the former also reduces to computing expressions involving ANDs of ORs (as exploited in Abboud, Williams, and Yu's OV algorithm [5]). In fact,

the reduction from OV is even simpler, and more efficient, than the reduction from Real-APSP.

Lemma 4.3. *One instance of* OV *for n Boolean vectors in f dimensions reduces in $O(ndf)$ time to one instance of* Colorful-BMM *for an $(n/d) \times (d^2 f)$ and a $(d^2 f) \times (n/d)$ Boolean matrix with d^2 colors for any given $d \le n$.*

Proof. Divide the input set A and B into groups $A_1, \ldots, A_{n/d}$ and $B_1, \ldots, B_{n/d}$ of d Boolean vectors each. Let $A[i,k,s]$ be the s-th bit of the k-th vector in A_i and let $B[j,\ell,s]$ be the s-th bit of the ℓ-th vector in B_j.

For each $i, j \in [n/d]$, there are no orthogonal pairs of vectors in $A_i \times B_j$ iff

$$\bigwedge_{k,\ell \in [d]} \bigvee_{s \in [f]} (A[i,k,s] \wedge B[j,\ell,s]).$$

Thus, the answer can be determined by solving Colorful-BMM on the following matrices $\mathcal{A}$ and $\mathcal{B}$:

(1) For each $i \in [n/d], k, \ell \in [d]$, and $s \in [f]$, let $\mathcal{A}[i,(k,\ell,s))] = A[i,k,s]$.
(2) For each $j \in [n/d], k, \ell \in [d]$, and $s \in [f]$, let $\mathcal{B}[(k,\ell,s),j] = B[j,\ell,s]$.
(3) Define $color((k,\ell,s)) = (k,\ell)$.

The inner dimension (i.e., the number of triples (k,ℓ,s)) is $d^2 f$. □

Theorem 4.4. *If* Colorful-BMM *for two $N \times N$ Boolean matrices could be solved in $\tilde{O}(N^{3-\varepsilon})$ time, then* OV *for n Boolean vectors in f dimensions could be solved in $\tilde{O}(f^{O(1)} n^{2-2\varepsilon/3})$ time.*

More generally, if Colorful-BMM *for two $N \times N$ Boolean matrices with N^α colors could be solved in $\tilde{O}(N^{2+\alpha-\varepsilon})$ time for some constant $\alpha \le 1$, then* OV *for n Boolean vectors in f dimensions could be solved in $\tilde{O}(f^{O(1)} n^{2-2\varepsilon/(2+\alpha)})$ time.*

Proof. Choose d so that $d^2 = (n/d)^\alpha$, i.e., $d = n^{\alpha/(2+\alpha)}$. Since $\alpha \le 1$, we have $d \le n^{1/3}$ and so $d^2 \le n/d$. Thus, if there is an $\tilde{O}(N^{2+\alpha-\varepsilon})$ time algorithm for Colorful-BMM with $O(N^\alpha)$ colors, we can solve OV in $\tilde{O}((f^{O(1)} n/d)^{2+\alpha-\varepsilon}) = \tilde{O}(f^{O(1)} n^{2-2\varepsilon/(2+\alpha)})$ time. □

5 TRIANGLE COLLECTION AND TRIANGLE-COLLECTION*

In the full version of the paper, we will prove Theorem 1.7, which we recall as follows:

Theorem 1.7. *(*ACP-*)* Tri-Co$_{\text{light}}$ *with parameter $n^{o(1)}$ and (*ACP-*)* Tri-Co* *with $t, p \le n^{o(1)}$ are equivalent up to $n^{o(1)}$ factors.*

Recall our main theorem is the following.

Theorem 1.2. *Assuming Hypothesis 2,* ACP-Tri-Co *requires $n^{2+\delta-o(1)}$ time for some $\delta > 0$, in a reasonable Real RAM model.*

In Section 5.1, we will show a reduction from AE-Mono-Tri to ACP-Tri-Co*. Combined with the Real-APSP and Real-3SUM hardness of AE-Mono-Tri (which can be found in the full paper), we obtain the Real-APSP and Real-3SUM hardness in the main theorem. In Section 5.2, we show a reduction from Colorful-BMM to ACP-Tri-Co*. By combining with the reduction from OV to Colorful-BMM and unrolling the whole reduction, we in fact obtain a reduction from OV to Tri-Co*. This shows the OV hardness of the main theorem, and thus concludes the proof of the main theorem.

5.1 From All-Edges Monochromatic Triangle to All-Color-Pairs Triangle Collection

Here we show that the AE-Mono-Tri problem can be reduced to the ACP-Tri-Co* problem. By our results above, it suffices to reduce AE-Mono-Tri to ACP-Tri-Co$_{\text{light}}$ with parameter $n^{o(1)}$.

Recall that in AE-Mono-Tri we are given an n node graph $G = (V,E)$ with colors $color : E \mapsto [n^2]$ on the edges and we want to know for every edge $(u,v) \in E$ if there exists a $w \in V$ so that $(u,v),(v,w),(w,u) \in E$ (i.e. they form a triangle) and $color(u,v) = color(v,w) = color(w,u)$.

Theorem 5.1. *An n-node instance of* AE-Mono-Tri *can be reduced to an $O(n \log n)$-node instance of* ACP-Tri-Co$_{\text{light}}$ *with parameter $O(\log n)$ in $O(n^2 \log n)$ time.*

Proof. Let $G = (V,E)$ be the instance of AE-Mono-Tri with edge colors $color : E \mapsto [n^2]$. Without loss of generality, G is tripartite with node parts A, B, C.

We will build an instance of ACP-Tri-Co$_{\text{light}}$ H. For every $t \in [2 \log n]$ we will create a graph G_t and H will be the disjoint union of these $O(\log n)$ graphs.

For a fixed $t \in [2 \log n]$, the nodes of G_t are as follows. G_t will be a tripartite graph on parts A', B', C'. For every node $z \in C$ of G, we add a node z to G_t; z has color z (we associate the nodes of G with the integers $[n]$). These nodes will be in part C'. For every node $v \in A \cup B$ of G we create two copies of v in G_t: v_0 and v_1, both having color v. If v was in A, v_0, v_1 are in the partition A', and if v was in B, then v_0, v_1 are in partition B'.

For every pair of nodes of G, $v \in A \cup B$ and $z \in C$, if the tth bit of the color $color(v,z)$ is b, then add an edge between v_b and z. For every pair of nodes of G, $v \in A$ and $v' \in B$, add an edge between v_p and $v'_{p'}$, for all choices of p and p' s.t. $p \ne p'$. In addition, add an edge between between v_b and v'_b if the tth bit of the color $color(v,v')$ is not b.

Notice that for every $z \in C, v \in A, v' \in B$ (which are now a triple of colors), G_t does not have a triangle of these colors iff the tth bit of the colors of $(z,v),(z,v'),(v,v')$ in G match. Thus, there are no triangles colored z, v, v' in $H = \cup_t G_t$ if and only if the colors of $(z,v),(z,v'),(v,v')$ in G match for all choices of t, and hence if and only if the colors are exactly the same.

By construction, the number of nodes of each color is at most $O(\log n)$. Thus we have reduced AE-Mono-Tri to an $O(n \log n)$-node instance of ACP-Tri-Co$_{\text{light}}$.

□

5.2 Colorful-BMM → ACP-Tri-Co and OV → Tri-Co*

We give a simple reduction from Colorful-BMM to ACP-Tri-Co.

Lemma 5.2. *Any instance of* Colorful-BMM *on $n \times n$ matrices can be reduced in $O(n^2)$ time to an instance of* ACP-Tri-Co *on n nodes. If the maximum number of k of any given color in the* Colorful-BMM *instance is $n^{o(1)}$, then* Colorful-BMM *also reduces to* ACP-Tri-Co* *with parameter $n^{o(1)}$.*

Proof. Let A and B be two $n \times n$ matrices that constitute an instance of Colorful-BMM. Let $color : [n] \to [K]$ be the color function of the columns of A and rows of B. Note that without loss of generality $K \leq O(n)$.

We will create an instance G of ACP-Tri-Co with $O(n)$ colors.

For every color $k \in [K]$ and every $t \in [n]$ such that $color(t) = k$, create a node $k_{2,t}$ of color k.

For every $i \in [n]$, create a node i_1 of color i_1 and a node i_3 of color i_3, and add an edge between i_1 and j_3 for all $i, j \in [n]$.

For every pair $i \in [n], t \in [n]$ with $color(t) = k$, add an edge $(i_1, k_{2,t})$ if $A[i,t] = 1$ and add an edge $(i_3, k_{2,t})$ if $B[t,i] = 1$.

Notice that for a fixed triple $i \in [n], j \in [n], k \in [K]$, there is a triangle with colors i_1, k, j_3 if and only if there is some $t \in [n]$, $A[i,t] \cdot B[t,j] = 1$.

ACP-Tri-Co asks for every pair of colors i_1, j_3 to compute whether there is some k with no triangles of color triple i_1, j_3, k, conversely, whether for all colors k there is a triangle with color triple i_1, j_3, k, i.e. whether for all i, j, k there is some t of color k such that $A[i,t] \cdot B[t,j] = 1$.

We get an instance of ACP-Tri-Co with $O(n)$ nodes and colors.

Now suppose that the number of k in any given color of the Colorful-BMM instance is at most $T \leq n^{o(1)}$. In this case, the number of nodes of each color in the above reduction is at most $n^{o(1)}$, so we actually get an instance of ACP-Tri-Co$_{\text{light}}$ with parameter $n^{o(1)}$. By our reduction from ACP-Tri-Co$_{\text{light}}$ with parameter $n^{o(1)}$ to ACP-Tri-Co*, we get an instance of ACP-Tri-Co* with $n^{1+o(1)}$ nodes and $O(n)$ colors. □

We get as a corollary a reduction from OV to Tri-Co*.

Corollary 5.3. *For any $1 \leq d \leq n$, OV for n Boolean vectors in f dimensions reduces in $O(n^2/d^2 + ndf)$ time to an instance of* Tri-Co$_{\text{light}}$ *on $O(nf/d + d^2)$ nodes with parameter f. Thus if OV requires $n^{2-o(1)}$ time for $f = n^{o(1)}$, then* Tri-Co$_{\text{light}}$ *on N nodes with parameter $N^{o(1)}$ requires $N^{3-o(1)}$ time, then so does* Tri-Co* *with parameter $n^{o(1)}$.*

Proof. Consider the reduction from Lemma 4.3 from OV to Colorful-BMM. For any $d \leq n$, it partitioned the input sets of vectors A and B into groups $\{A_i\}_{i \in [n/d]}$ and $\{B_i\}_{i \in [n/d]}$ of d vectors each, letting $A[i,j,s]$ be the sth bit of the kth vector of A_i and $B[j,\ell,s]$ be the sth bit of the ℓth vector of B_j.

It then created an $n \times d^2 f$ matrix $\mathcal{A}$ with $\mathcal{A}[i,(k,\ell,s))] = A[i,k,s]$ and an $d^2 f \times n$ matrix $\mathcal{B}$ with $\mathcal{B}[(k,\ell,s),j] = B[j,\ell,s]$. The color of (k,ℓ,s) was (k,ℓ). The number of columns of $\mathcal{A}$ of each color is thus at most f.

With the above reduction, we then apply our reduction from Colorful-BMM to ACP-Tri-Co$_{\text{light}}$ in the proof of Lemma 5.2 to show that OV reduces to ACP-Tri-Co$_{\text{light}}$.

Notice that in the above reduction from OV to Colorful-BMM, there are no pairs of orthogonal vectors iff for all $i, j \in [n/d]$ and $k, \ell \in [d]$, there exists some $s \in [f]$ such that $\mathcal{A}[i,(k,\ell,s))] = \mathcal{B}[(k,\ell,s),j] = 1$. When we create the ACP-Tri-Co$_{\text{light}}$ instance from the Colorful-BMM instance in our reduction, it suffices to figure out whether for all triples of colors there is some triangle, which is exactly the Tri-Co$_{\text{light}}$ problem. □

REFERENCES

[1] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. 2015. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1681–1697. https://doi.org/10.1137/1.9781611973730.112

[2] Amir Abboud and Virginia Vassilevska Williams. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*. 434–443. https://doi.org/10.1109/FOCS.2014.53

[3] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. 2014. Consequences of Faster Alignment of Sequences. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP), Part I (Lecture Notes in Computer Science, Vol. 8572)*. Springer, 39–51. https://doi.org/10.1007/978-3-662-43948-7_4

[4] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. 2018. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM J. Comput.* 47, 3 (2018), 1098–1122. https://doi.org/10.1137/15M1050987 Preliminary version in STOC 2015..

[5] Amir Abboud, Ryan Williams, and Huacheng Yu. 2014. More applications of the polynomial method to algorithm design. In *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 218–230. https://doi.org/10.1137/1.9781611973730.17

[6] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. 2001. Smallest Color-Spanning Objects. In *Proc. 9th Annual European Symposium on Algorithms (ESA)*. 278–289. https://doi.org/10.1007/3-540-44676-1_23

[7] Karl Abrahamson. 1987. Generalized String Matching. *SIAM J. Comput.* 16, 6 (Dec. 1987), 1039–1051. https://doi.org/10.1137/0216067

[8] Josh Alman and Virginia Vassilevska Williams. 2021. A refined laser method and faster matrix multiplication. In *Proc. 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 522–539. https://doi.org/10.1137/1.9781611976465.32

[9] Josh Alman and Ryan Williams. 2015. Probabilistic Polynomials and Hamming Nearest Neighbors. In *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS)*. 136–150. https://doi.org/10.1109/FOCS.2015.18

[10] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. 1992. Witnesses for Boolean Matrix Multiplication and for Shortest Paths. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 417–426. https://doi.org/10.1109/SFCS.1992.267748

[11] Noga Alon, Raphael Yuster, and Uri Zwick. 1997. Finding and Counting Given Length Cycles. *Algorithmica* 17, 3 (1997), 209–223. https://doi.org/10.1007/BF02523189

[12] Daniel Archambault, William Evans, and David Kirkpatrick. 2005. Computing the Set of all the Distant Horizons of a Terrain. *Int. J. Comput. Geometry Appl.* 15 (12 2005), 547–564. https://doi.org/10.1142/S0218195905001841

[13] Boris Aronov and Sariel Har-Peled. 2008. On approximating the depth and related problems. *SIAM J. Comput.* 38, 3 (2008), 899–921. https://doi.org/10.1137/060669474

[14] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. 2018. Towards tight approximation bounds for graph diameter and eccentricities. In *Proc. the 50th ACM Symposium on Theory of Computing (STOC)*. 267–280. https://doi.org/10.1145/3188745.3188950

[15] Ilya Baran, Erik D. Demaine, and Mihai Patrascu. 2008. Subquadratic Algorithms for 3SUM. *Algorithmica* 50, 4 (2008), 584–596. https://doi.org/10.1007/s00453-007-9036-3 Preliminary version in WADS 2005.

[16] Gill Barequet and Sariel Har-Peled. 2001. Polygon Containment and Translational Min-Hausdorff-Distance Between Segment Sets are 3SUM-Hard. *Int. J. Comput. Geom. Appl.* 11, 4 (2001), 465–474. https://doi.org/10.1142/S0218195901000596 Preliminary version in SODA 1999.

[17] Karl Bringmann, Paweł Gawrychowski, Shay Mozes, and Oren Weimann. 2020. Tree Edit Distance Cannot be Computed in Strongly Subcubic Time (Unless APSP Can). *ACM Trans. Algorithms* 16, 4 (2020), 48:1–48:22. https://doi.org/10.1145/3381878

[18] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2009. The complexity of satisfiability of small depth circuits. In *Proc. 4th International Workshop on Parameterized and Exact Computation (IWPEC)*. Springer, 75–85. https://doi.org/10.1007/978-3-642-11269-0_6

[19] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. 2013. On the Exact Complexity of Evaluating Quantified k-CNF. *Algorithmica* 65, 4 (2013), 817–827. https://doi.org/10.1007/s00453-012-9648-0 Preliminary version in IPEC 2010.

[20] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. 2016. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. ACM Conference on Innovations in Theoretical Computer Science (ITCS)*. 261–270. https://doi.org/10.1145/2840728.2840746

[21] Timothy M. Chan. 2020. More Logarithmic-Factor Speedups for 3SUM, (Median,+)-Convolution, and Some Geometric 3SUM-Hard Problems. *ACM Trans. Algorithms* 16, 1, Article 7 (2020), 23 pages. https://doi.org/10.1145/3363541 Preliminary version in SODA 2018.

[22] Timothy M. Chan and R. Ryan Williams. 2021. Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky. *ACM Trans.*

Algorithms 17, 1 (2021), 2:1–2:14. https://doi.org/10.1145/3402926 Preliminary version in SODA 2016.

[23] Otfried Cheong, Alon Efrat, and Sariel Har-Peled. 2007. Finding a Guard that Sees Most and a Shop that Sells Most. *Discret. Comput. Geom.* 37, 4 (2007), 545–563. https://doi.org/10.1007/s00454-007-1328-5 Preliminary version in SODA 2004.

[24] Norishige Chiba and Takao Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14, 1 (1985), 210–223. https://doi.org/10.1137/0214017

[25] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. 2016. On Problems as Hard as CNF-SAT. *ACM Trans. Algorithms* 12, 3 (2016), 41:1–41:24. https://doi.org/10.1145/2925416

[26] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. 2018. Fast Hamiltonicity Checking Via Bases of Perfect Matchings. *J. ACM* 65, 3, Article 12 (mar 2018), 46 pages. https://doi.org/10.1145/3148227

[27] Søren Dahlgaard. 2016. On the Hardness of Partially Dynamic Graph Problems and Connections to Diameter. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs, Vol. 55)*. 48:1–48:14. https://doi.org/10.4230/LIPIcs.ICALP.2016.48

[28] Evgeny Dantsin and Alexander Wolpert. 2010. On moderately exponential time for SAT. In *Proc. 13th International Conference on Theory and Applications of Satisfiability Testing (SAT)*. Springer, 313–325. https://doi.org/10.1007/978-3-642-14186-7_27

[29] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer. https://www.worldcat.org/oclc/227584184

[30] Mark de Berg, Marko M. de Groot, and Mark H. Overmars. 1997. Perfect binary space partitions. *Comput. Geom.* 7, 1 (1997), 81–91. https://doi.org/10.1016/0925-7721(95)00045-3

[31] Lech Duraj, Krzysztof Kleiner, Adam Polak, and Virginia Vassilevska Williams. 2020. Equivalences between triangle and range query problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 30–47. https://doi.org/10.1137/1.9781611975994.3

[32] Jeff Erickson. 1999. Lower Bounds for Linear Satisfiability Problems. *Chic. J. Theor. Comput. Sci.* 1999(8) (1999). https://doi.org/10.4086/cjtcs.1999.008 Preliminary version in SODA 1995.

[33] Jeff Erickson. 1999. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.* 28, 4 (1999), 1198–1214. https://doi.org/10.1137/S0097539797315410

[34] Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. 2020. Smoothing the gap between NP and ER. In *Proc. 61st IEEE Symposium on Foundations of Computer Science*. IEEE, 1022–1033. https://doi.org/10.1109/FOCS46700.2020.00099 Full version available at http://arxiv.org/abs/1912.02278.

[35] Michael J. Fischer and Albert R. Meyer. 1971. Boolean matrix multiplication and transitive closure. In *Proc. 12th Annual Symposium on Switching and Automata Theory (SWAT)*. IEEE, 129–131. https://doi.org/10.1109/SWAT.1971.4

[36] Robert W. Floyd. 1962. Algorithm 97: shortest path. *Commun. ACM* 5, 6 (1962), 345. https://doi.org/10.1145/367766.368168

[37] Michael L. Fredman. 1976. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.* 5, 1 (1976), 83–89. https://doi.org/10.1137/0205006

[38] Ari Freund. 2017. Improved subquadratic 3SUM. *Algorithmica* 77, 2 (2017), 440–458. https://doi.org/10.1007/s00453-015-0079-6

[39] Anka Gajentaan and Mark H. Overmars. 1995. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom.* 5, 3 (1995), 165–185. https://doi.org/10.1016/0925-7721(95)00022-2

[40] François Le Gall. 2014. Powers of tensors and fast matrix multiplication. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC)*. 296–303. https://doi.org/10.1145/2608628.2608664

[41] Paweł Gawrychowski and Przemysław Uznański. 2018. Towards Unified Approximate Pattern Matching for Hamming and L_1 Distance. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs, Vol. 107)*. 62:1–62:13. https://doi.org/10.4230/LIPIcs.ICALP.2018.62

[42] Omer Gold and Micha Sharir. 2017. Improved Bounds for 3SUM, k-SUM, and Linear Degeneracy. In *Proc. 25th European Symposium on Algorithms (ESA) (LIPIcs, Vol. 87)*. 42:1–42:13. https://doi.org/10.4230/LIPIcs.ESA.2017.42

[43] Allan Grønlund and Seth Pettie. 2018. Threesomes, Degenerates, and Love Triangles. *J. ACM* 65, 4 (2018), 22:1–22:25. https://doi.org/10.1145/3185378 Preliminary version in FOCS 2014.

[44] Russell Impagliazzo and Ramamohan Paturi. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62, 2 (2001), 367–375. https://doi.org/10.1006/jcss.2000.1727 Preliminary version in CoCo 1999..

[45] Daniel M. Kane, Shachar Lovett, and Shay Moran. 2019. Near-optimal Linear Decision Trees for k-SUM and Related Problems. *J. ACM* 66, 3 (2019), 16:1–16:18. https://doi.org/10.1145/3285953 Preliminary version in STOC 2018.

[46] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. 2016. Higher lower bounds from the 3SUM conjecture. In *Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1272–1287. https://doi.org/10.1137/1.9781611974331.ch89

[47] Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. 2020. Monochromatic Triangles, Intermediate Matrix Products, and Convolutions. In *Proc. 11th Innovations in Theoretical Computer Science Conference (ITCS) (LIPIcs, Vol. 151)*. 53:1–53:18. https://doi.org/10.4230/LIPIcs.ITCS.2020.53

[48] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2018. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. *ACM Trans. Algorithms* 14, 2, Article 13 (apr 2018), 30 pages. https://doi.org/10.1145/3170442

[49] Mihai Pătraşcu. 2010. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC)*. 603–610. https://doi.org/10.1145/1806689.1806772

[50] Liam Roditty and Uri Zwick. 2011. On Dynamic Shortest Paths Problems. *Algorithmica* 61, 2 (2011), 389–401. https://doi.org/10.1007/s00453-010-9401-5 Preliminary version in ESA 2004.

[51] R. Seidel. 1995. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *J. Comput. Syst. Sci.* 51, 3 (1995), 400–403. https://doi.org/10.1006/jcss.1995.1078

[52] Michael Soss, Jeff Erickson, and Mark Overmars. 2003. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Comput. Geom.* 26, 3 (2003), 235–246. https://doi.org/10.1016/S0925-7721(02)00156-6

[53] Virginia Vassilevska Williams. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. 44th ACM Symposium on Theory of Computing (STOC)*. 887–898. https://doi.org/10.1145/2213977.2214056

[54] Virginia Vassilevska Williams. 2018. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, Vol. 3. World Scientific, 3431–3472. https://doi.org/10.1142/9789813272880_0188

[55] Virginia Vassilevska Williams and Ryan Williams. 2013. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.* 42, 3 (2013), 831–854. https://doi.org/10.1137/09076619X Preliminary version in STOC 2009.

[56] Virginia Vassilevska Williams and R. Ryan Williams. 2018. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM* 65, 5 (2018), 27:1–27:38. https://doi.org/10.1145/3186893 Preliminary version in FOCS 2010.

[57] Virginia Vassilevska Williams and Yinzhan Xu. 2020. Monochromatic Triangles, Triangle Listing and APSP. In *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS)*. 786–797. https://doi.org/10.1109/FOCS46700.2020.00078

[58] Ryan Williams. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348, 2-3 (2005), 357–365. https://doi.org/10.1016/j.tcs.2005.09.023 Preliminary version in ICALP 2004.

[59] Ryan Williams. 2014. 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*, David B. Shmoys (Ed.). ACM, 664–673. https://doi.org/10.1145/2591796.2591811

[60] R. Ryan Williams. 2018. Faster All-Pairs Shortest Paths via Circuit Complexity. *SIAM J. Comput.* 47, 5 (2018), 1965–1985. https://doi.org/10.1137/15M1024524 Preliminary version in STOC 2014.