Fredman's Trick Meets Dominance Product: Fine-Grained Complexity of Unweighted APSP, 3SUM Counting, and More

Timothy M. Chan*
UIUC
tmc@illinois.edu

Virginia Vassilevska Williams†
MIT
virgi@mit.edu

Yinzhan Xu‡
MIT
xyzhan@mit.edu

Abstract

In this paper we carefully combine Fredman's trick [SICOMP'76] and Matoušek's approach for dominance product [IPL'91] to obtain powerful results in fine-grained complexity:

- Under the hypothesis that APSP for undirected graphs with edge weights in $\{1, 2, \ldots, n\}$ requires $n^{3-o(1)}$ time (when $\omega = 2$), we show a variety of conditional lower bounds, including an $n^{7/3-o(1)}$ lower bound for unweighted directed APSP and an $n^{2.2-o(1)}$ lower bound for computing the Minimum Witness Product between two $n \times n$ Boolean matrices, even if $\omega = 2$, improving upon their trivial n^2 lower bounds. Our techniques can also be used to reduce the unweighted directed APSP problem to other problems. In particular, we show that (when $\omega = 2$), if unweighted directed APSP requires $n^{2.5-o(1)}$ time, then Minimum Witness Product requires $n^{7/3-o(1)}$ time.
- We show that, surprisingly, many central problems in fine-grained complexity are equivalent to their natural counting versions. In particular, we show that Min-Plus Product and Exact Triangle are subcubically equivalent to their counting versions, and 3SUM is subquadratically equivalent to its counting version.
- We obtain new algorithms using new variants of the Balog-Szemerédi-Gowers theorem from additive combinatorics. For example, we get an $O(n^{3.83})$ time deterministic algorithm for exactly counting the number of shortest paths in an arbitrary weighted graph, improving the textbook $\widetilde{O}(n^4)$ time algorithm. We also get faster algorithms for 3SUM in preprocessed universes, and deterministic algorithms for 3SUM on monotone sets in $\{1, 2, \ldots, n\}^d$.

*Supported by NSF Grant CCF-2224271.
†Supported by an NSF CAREER Award, NSF Grant CCF-2129139 and BSF Grant BSF:2012338, a Google Research Fellowship and a Sloan Research Fellowship.
‡Partially supported by NSF Grant CCF-2129139.

Contents

1 Introduction

There are many examples of computational problems for which a slight variant can become much harder than the original: 2-SAT (in P) vs. 3-SAT (NP-complete), perfect matching (in P) vs. its counting version (#P-complete), and so on. Fine-grained complexity (FGC) has provided explanations for such differences for many natural problems – e.g. finding a minimum weight triangle in a node-weighted graph is "easy" (in $O(n^\omega)$ time [CL09], where $\omega < 2.373$ [AV21] is the matrix multiplication exponent) while finding a minimum weight triangle in an edge-weighted graph is "hard" (subcubically equivalent to All-Pairs Shortest Paths (APSP) [VW10, VW18]). Nevertheless, many fundamental questions remain unanswered:

- Seidel [Sei95] showed that APSP in unweighted *undirected* graphs can be solved in $\tilde{O}(n^\omega)$ time[1]. The fastest algorithm for APSP in unweighted *directed* graphs by Zwick [Zwi02] runs in $O(n^{2.53})$ for the current bound of rectangular matrix multiplication [LU18]. If $\omega = 2$, then undirected APSP would have an essentially optimal $\tilde{O}(n^2)$ time algorithm, whereas Zwick's algorithm for directed APSP would run in, slower, $\tilde{O}(n^{2.5})$ time.

  Assuming that the runtime of Zwick's algorithm is the best possible for unweighted directed APSP (abbreviated u-dir-APSP) has been used as a hardness hypothesis (see e.g. [LPV20, VX20a]). However, so far there has been *no explanation* for why u-dir-APSP should be harder than its undirected counterpart.

  Q1: Does APSP *in directed unweighted graphs require superquadratic time, even if* $\omega = 2$*?*

- Boolean matrix multiplication (BMM) asks to compute, for two $n \times n$ Boolean matrices A and B, for every $i, j \in \{1, \ldots, n\}$ whether there is some k so that $A[i, k] = B[k, j] = 1$. The Min-Witness product (Min-Witness-Prod) is a well-studied [KL05, VWY10, SYZ11, CY14, KL21] generalization of BMM in which for every i, j, one needs to instead compute the *minimum* k for which $A[i, k] = B[k, j] = 1$. Similarly to u-dir-APSP, the fastest algorithm for Min-Witness-Prod runs in $O(n^{2.53})$ time [CKL07], and would run in $\tilde{O}(n^{2.5})$ time if $\omega = 2$, whereas BMM would have an essentially optimal algorithm in that case.

  The assumption that Min-Witness-Prod requires $n^{2.5-o(1)}$ time has been used as a hardness hypothesis (e.g. [LPV20]), but similar to u-dir-APSP, so far there has been no explanation as to why Min-Witness-Prod should be harder than BMM.

  Q2: Does Min-Witness-Prod *require superquadratic time, even if* $\omega = 2$*?*

- For a large number of problems of interest within FGC, the known algorithms for finding a solution can also compute the *number* of solutions in the same time. This is true for triangle detection in graphs, BMM, Min-Plus product, 3SUM, Exact Triangle, Negative Triangle, and many more. Is this merely a coincidence, or are the decision variants of these problems equivalent to the counting variants?

  A natural and important question is:

  Q3: Are the core FGC problems like 3SUM*, Min-Plus product and Exact Triangle easier than their exact counting variants?*

[1] The notation $\tilde{O}(f(n))$ denotes $f(n) \operatorname{poly} \log(n)$.

A recent line of work [DL21, DLM20] gives fine-grained reductions from *approximate counting* to decision for several core problems of FGC, showing that if the decision versions have improved algorithms, then one can also obtain fast approximation schemes. As an approximate count can solve the decision problem, we get that for many key problems *approximate counting* and decision are equivalent.

This does not imply that *exact counting* would be equivalent to decision however. In fact, quite often the decision and counting versions of computational problems can have vastly different complexities. There are many examples of polynomial time decision problems (e.g. perfect matching [Val79]) whose counting variants become #P-complete. For many of these problems, polynomial time approximation schemes are possible (see e.g. [JSV04] for perfect matching), however obtaining a fast *exact* counting algorithm is considered infeasible.

Within FGC, there are more examples. Consider for example the case of induced subgraph isomorphism for pattern graphs of constant size k. It is known (see e.g. [CDM17]) that for *every* k-node pattern graph H, *counting* the induced copies of H in an n-node graph is fine-grained equivalent to *counting* the k-cliques in an n-node graph. Meanwhile, the work of [DL21, DLM20] implies that *approximately* counting the induced k-paths in a graph is fine-grained equivalent to detecting a single k-path. However, induced k-paths can be found (and hence can be approximately counted) *faster* than counting k-cliques: combinatorially, there's an $O(n^{k-2})$ time algorithm [BKS18], whereas k-clique counting is believed to require $n^{k-o(1)}$ time; with the use of fast matrix multiplication, if $k < 7$, k-paths can be found and approximately counted in the best known running time for $(k-1)$-clique detection [VWWY15, BKS18], faster than k-clique.

To reiterate Question Q3 in this context, it asks whether the core FGC problems are like induced k-path above, or perhaps surprisingly, are equivalent to their counting versions?

The fine-grained problems of interest such as 3SUM, Exact-Triangle, Orthogonal Vectors and more, admit efficient self-reductions. For such problems, prior work [VW18] has given generic techniques to show that the problems are fine-grained equivalent to the problem of *listing* any "small" number of solutions. For example, 3SUM is subquadratically equivalent to listing any truly subquadratic[2] number of 3SUM solutions. Thus, as long as the count is small, the listing problem is equivalent to the decision problem. This technique has become an important technique in FGC, used in many subsequent works (e.g. [HKNS15, BIS17, KPS17, CMWW19]). The issue is, however, that when the count is actually "large", say $\Omega(n^2)$ for 3SUM, the listing approach is too expensive. The question becomes, how do we count faster than listing when the count is large? Until now (more than 10 years after the conference version of [VW18]) there has been no technique to do this.

- Recently, wide classes of structured instances of Min-Plus matrix products, Min-Plus convolutions, and 3SUM have been identified which can be solved faster. Chan and Lewenstein [CL15] obtained the first truly subquadratic algorithm for Min-Plus convolution for monotone sequences over $[n] := \{1, \ldots, n\}$ (and also integer sequences with bounded differences), and Bringmann, Grandoni, Saha and Vassilevska W. [BGSV16] obtained the first truly subcubic algorithm for Min-Plus product for integer matrices with bounded differences. The latter result is in some sense more general, as bounded-difference Min-Plus convolution reduces to (row/column) bounded-difference Min-Plus matrix product. Both results have subsequently been generalized or improved [VX20b, GPVX21, CDX22, CDXZ22]. Interestingly, Chan and Lewenstein's approach made use of a famous result in additive combinatorics known as the *Balog–Szemerédi–Gowers Theorem* (the "BSG Theorem"), whereas [BGSV16]'s approach and all subsequent

[2]Truly subquadratic means $O(n^{2-\varepsilon})$ for some constant $\varepsilon > 0$; truly subcubic means $O(n^{3-\varepsilon})$ for constant $\varepsilon > 0$, etc.

algorithms use more direct techniques without the need for additive combinatorics. So far, there has been no explanation for why there are two seemingly different approaches. This leads to our last main question:

Q4: What is the relationship between the BSG-based approach and the direct approach to monotone Min-Plus product/convolution, and could they be unified?

Question Q4 might appear unrelated to the preceding questions, and is more vague or conceptual. But the hope is that by understanding the relationship better, we may obtain new improved algorithms, since Chan and Lewenstein's approach has several applications, e.g. to 3SUM with preprocessed universes and 3SUM for d-dimensional monotone sets in $[n]^d$, which are not handled by the subsequent approaches.

1.1 Summary of Our Contributions and New Tool

1. **Conditional hardness for u-dir-APSP.** One of the main hypotheses of fine-grained complexity, known as the "APSP Hypothesis", is that APSP in n-node graphs with polynomial integer edge weights requires $n^{3-o(1)}$ time. Meanwhile, the fastest algorithms for APSP [Sei95, Zwi02, Wil18] still fail to solve the problem in truly subcubic time when the edge weights are in $[n]$. In fact, even Min-Plus product on $n \times n$ matrices with entries in $[n]$ is not known to be solvable in truly subcubic time: the known algorithm for small integer entries runs in $\tilde{O}(Mn^{\omega})$ time [AGM97], and even if $\omega = 2$, this is no better than the brute-force cubic time algorithm when $M = n$. Moreover, Zwick [Zwi02] explicitly asks whether there is a truly subcubic time algorithm for APSP with weights in $[n]$.

   We formulate a version of the APSP Hypothesis, which we call the Strong APSP Hypothesis, stating (for $\omega = 2$) that APSP in graphs with edge weights in $[n]$ requires $n^{3-o(1)}$ time. Under this hypothesis we show that u-dir-APSP requires $n^{7/3-o(1)}$ time, even if $\omega = 2$.

   Thus, we conditionally resolve question Q1: either *unweighted directed* APSP requires super-quadratic time and is harder than unweighted undirected APSP (when $\omega = 2$), or APSP in weighted graphs with weights in $[n]$ is in truly subcubic time.

   This is the first fine-grained connection between unweighted and weighted APSP.

2. **Conditional hardness for Min-Witness-Prod.** We present the first fine-grained reduction from APSP in unweighted directed graphs to Min-Witness-Prod (which was left open by [LPV20]). Our reduction implies that, when $\omega = 2$, Min-Witness-Prod requires $n^{7/3-o(1)}$ time unless the current best algorithm for u-dir-APSP [Zwi02] can be improved. Alternatively, working under the Strong APSP Hypothesis, we also obtain an $n^{11/5-o(1)}$ lower bound for Min-Witness-Prod.

   Thus, either Min-Witness-Prod is truly harder than BMM (if $\omega = 2$), or there is a breakthrough in unweighted or weighted APSP algorithms. This gives an answer to question Q2.

   See Section 1.2 for further discussion and details of our results on Min-Witness-Prod and u-dir-APSP.

3. **More hardness results.** We give many more fine-grained lower bounds under the Strong APSP Hypothesis for problems such as Batched Range Mode, All Pairs Shortest Lightest Paths, Min Witness Equality Product, dynamic shortest paths in unweighted planar graphs and more.

4. **Counting is equivalent to detection.** We resolve question Q3 for several core problems in FGC. We show that the Min-Plus-Product problem is subcubically equivalent to its counting version, the 3SUM problem is subquadratically equivalent to its counting version, and the Exact-Weight Triangle problem

(Exact-Tri) is subcubically equivalent to its counting version. These are the first fine-grained equivalences between exact counting and decision problem variants, to our knowledge. See Section 1.3 for further results and discussion.

5. **New variants of the BSG Theorem and new algorithms.** We formulate a new decomposition theorem for zero-weight triangles of weighted graphs, which may be viewed as a substitute to the BSG Theorem but has a simple direct proof, providing an answer to question Q4. Besides being applicable to monotone Min-Plus convolution/products, 3SUM with preprocessed universes and 3SUM with d-dimensional monotone sets, the theorem yields the first truly subquartic algorithm for the counting version of the general weighted APSP problem. Our ideas also lead to a new bound on the BSG Theorem itself which is an improvement for a certain range of parameters. As a result, we obtain an improved new algorithm for 3SUM with preprocessed universes. See Sections 1.4 and 10 for more details.

Surprisingly, we are able to achieve all of these results with a **single new tool**, a careful combination of two known techniques in tackling shortest paths problems in graphs: Fredman's trick [Fre76] and Matoušek's approach for dominance product [Mat91].

It has long been known that APSP in general n-node graphs is equivalent to computing the Min-Plus product of two $n \times n$ matrices A and B (Min-Plus-Product), defined as the matrix C with $C_{ij} = \min_k(A_{ik} + B_{kj})$. Fredman [Fre76] introduced the following powerful "trick" for dealing with the above minimum: to determine if $A_{ik} + B_{kj} \leq A_{i\ell} + B_{\ell j}$, we simply need to check if

$$A_{ik} - A_{i\ell} \leq B_{\ell j} - B_{kj}.$$

While seemingly trivial, this idea of comparing a left-hand side that is purely in terms of entries of A to a right-hand side that is purely in terms of entries of B, leads to a variety of amazing results. Fredman used it to show that the decision tree complexity of Min-Plus-Product is $O(n^{2.5})$, and not cubic as previously thought. Practically all subcubic algorithms for APSP (e.g. [Fre76, Tak98, Cha10, Wil18]) including the current fastest $n^3/\exp(\Theta(\sqrt{\log n}))$ time algorithm by Williams [Wil18] use Fredman's trick. Several new truly subcubic time algorithms for variants of APSP (e.g. [CDX22] and, implicitly, [BGSV16]) and recent algorithms for 3SUM (e.g. [GP18, Cha20]) also use it.

A completely different technique is Matoušek's truly subcubic time algorithm [Mat91] for Dominance Product (Dominance-Product). The dominance product of two $n \times n$ matrices A and B is the $n \times n$ matrix C such that for all $i, j \in [n]$, $C_{ij} = |\{k \in [n] : A_{ik} \leq B_{kj}\}|$. Matoušek gave an approach that combined fast matrix multiplication with brute-force to obtain an $\widetilde{O}(n^{(3+\omega)/2})$ time algorithm for Dominance-Product. Dominance-Product is known to be equivalent to the so-called Equality Product (Equality-Product) problem[3] which asks to compute $C_{ij} = |\{k \in [n] : A_{ik} = B_{kj}\}|$, so we sometimes refer to Equality-Product instead.

Matoušek's subcubic time techniques have been used to obtain truly subcubic time algorithms for All-Pairs Bottleneck Paths [VWY07, DP09], All-Pairs Nondecreasing Paths [Vas10, DJW19], APSP in node-weighted graphs [Cha10] and more. Unfortunately, the technique has fallen short when applied directly to the general APSP problem.

In this paper we give a combination of these two techniques that allows us to obtain reductions that exploit fast matrix multiplication in a new way, thus allowing us to overcome many difficulties, such as counting solutions when the number of solutions is large. The main ideas involve: (i) division into

[3]The equivalence was proven e.g. by [LUWG19, Vas15], but also Matoušek's algorithm almost immediately works for Equality-Product.

"few-witnesses" and "many-witnesses" cases, (ii) standard witness-finding techniques to handle the "few-witnesses" case, (iii) hitting sets to hit the "many-witnesses", (iv) Fredman's trick (the obvious equivalence of $a+b=a'+b'$ with $a-a'=b'-b$), and lastly (v) Matoušek's technique for dominance or equality products (which also involves a division into "low-frequency" and "high-frequency" cases). Steps (iv) and (v) crucially allow us to handle the "many-witnesses" case, delicately exploiting the small hitting set from (iii). Individually, each of these ideas is simple and has appeared before. But the novelty lies in how they are pieced together (see Sections 3, 4, and 5), and the realization that these ideas are powerful enough to yield all the new results in the above bullets!

In the remainder of the introduction we give more details on each of the bullets above.

1.2 Conditional Lower Bounds for Unweighted Directed APSP, Min-Witness Product and Other Problems with Intermediate Complexity

u-dir-APSP and Min-Witness-Prod are both "intermediate" problems as dubbed by Lincoln, Polak and Vassilevska W. [LPV20], a class of matrix product and all-pairs graph problems whose running times are $\widetilde{O}(n^{2.5})$ when $\omega=2$ (and hence right in the middle between the brute-force n^3 and and the desired optimal n^2), and for which no $O(n^{2.5-\varepsilon})$ time algorithms are known. More examples of intermediate problems include Min-Equality Product and Max-Min Product. A similar class of "intermediate" *convolution* problems that are known to be solvable in $\widetilde{O}(n^{1.5})$ time but not much faster [LPV20] includes Max-Min Convolution, Minimum-Witness Convolution, Min-Equality Convolution, and pattern-to-text Hamming distances. For instance, $\widetilde{O}(n^{1.5})$-time algorithms were known since the 1980s for Max-Min convolution [Kos89] and pattern-to-text Hamming distances [Abr87], and these remain the fastest algorithms for these problems.

None of these intermediate problems currently have nontrivial conditional lower bounds under standard hypotheses in FGC. (Two exceptions are All-Edges Monochromatic Triangle (AE-Mono-Tri) and Monochromatic Convolution (Mono-Convolution), where a near-$n^{2.5}$ lower bound is known for the former under the 3SUM or the APSP Hypothesis [LPV20, VX20a] and a near-$n^{1.5}$ lower bound is known for the latter under the 3SUM Hypothesis [LPV20], but one may argue that these monochromatic problems are not true matrix product or convolution problems since their inputs involve *three* matrices or sequences rather than two.)

Thus, an important research direction is to prove super-quadratic conditional lower bounds for intermediate matrix product problems, and super-linear lower bounds for intermediate convolution problems. Some relationships are known between intermediate problems, some of which are illustrated in Figure 1. However, many questions remain; for instance, it was open whether u-dir-APSP and Min-Witness-Prod are related.

As u-dir-APSP and Min-Witness-Prod are among the "lowest" problems in this class, proving conditional lower bounds for these two problems is especially fundamental. (Besides, Min-Witness-Prod has been extensively studied, arising in many applications [KL05, VWY10, SYZ11, CY14, KL21].) The precise time bounds of the current best algorithms for u-dir-APSP by Zwick [Zwi02] and Min-Witness-Prod by Czumaj, Kowaluk and Lingas [CKL07] are both $\widetilde{O}(n^{2+\rho})$, where $\rho\in[1/2,0.529)$ satisfies[4] $\omega(1,\rho,1)=1+2\rho$.

To explain the hardness of u-dir-APSP, Min-Witness-Prod and more, we introduce a strong version of the APSP Hypothesis:

Hypothesis 1 (Strong APSP Hypothesis)**.** *In the Word-RAM model with $O(\log n)$-bit words, computing APSP for an undirected*[5] *graph with edge weights in $[n^{3-\omega}]$ requires $n^{3-o(1)}$ randomized time.*

[4] $\omega(a,b,c)$ denotes the rectangular matrix multiplication exponent between an $n^a\times n^b$ matrix and an $n^b\times n^c$ matrix.

[5] The version of this hypothesis for directed graphs is equivalent: see Remark 6.4.

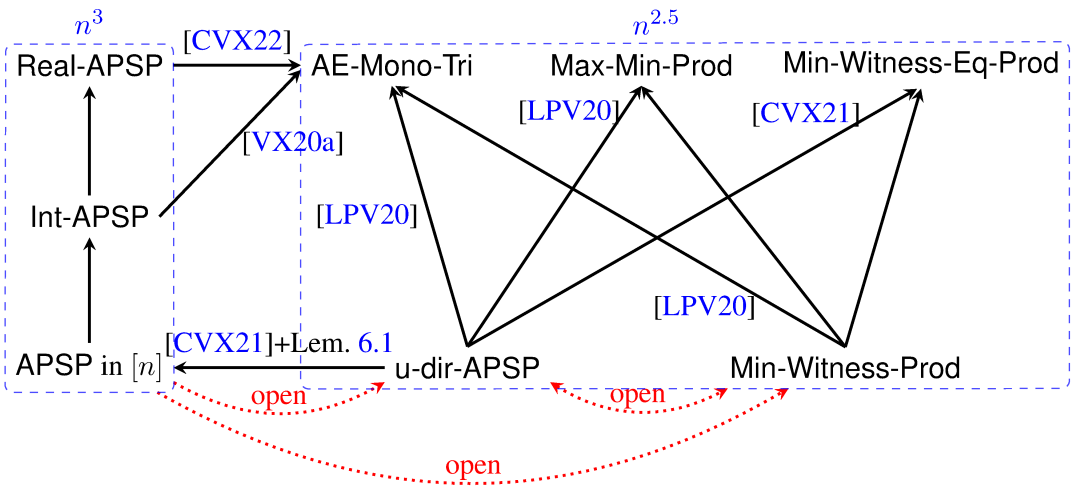

Figure 1: Previous work on intermediate matrix product problems, assuming $\omega = 2$. Here, APSP in $[n]$ denotes APSP with edge weights in $[n]$, Int-APSP denotes APSP with arbitrary $O(\log n)$-bit integer weights, and Real-APSP denotes APSP with real weights. All unlabelled arrows follow by trivial reductions.

For $\omega = 2$, the above asserts that the standard textbook cubic time algorithms for APSP are near-optimal when the edge weights are integers in $[n]$. Due to a tight equivalence [SZ99] between undirected APSP with edge weights in $[M]$ and Min-Plus-Product of two matrices with entries in $[O(M)]$, the above hypothesis is equivalent to the following:

Hypothesis 1′ (Strong APSP Hypothesis, restated)**.** *In the Word-RAM model with $O(\log n)$-bit words, computing the Min-Plus product between two $n \times n$ matrices with entries in $[n^{3-\omega}]$ requires $n^{3-o(1)}$ randomized time.*

The current upper bound for Min-Plus-Product between two $n \times n$ matrices with entries in $[M]$ is $\widetilde{O}(\min\{Mn^\omega, n^3\})$ [AGM97], which is cubic for $M = n^{3-\omega}$, and this has not been improved for over 30 years.

The above strengthening of the APSP Hypothesis is reminiscent of a strengthening of the 3SUM Hypothesis proposed by Amir, Chan, Lewenstein and Lewenstein [ACLL14], which they called the "Strong 3SUM-hardness assumption", asserting that the 3SUM Convolution problem requires near-quadratic time for integers in $[n]$.

We prove the following results:

Theorem 1.1. *Under the Strong APSP Hypothesis,* u-dir-APSP *requires $n^{7/3-o(1)}$ time, and* Min-Witness-Prod *requires $n^{11/5-o(1)}$ time (on a Word-RAM with $O(\log n)$-bit words).*

In fact, for u-dir-APSP, we can obtain a conditional lower bound of $n^{2+\beta-o(1)}$ for graphs with $n^{1+\beta}$ edges, for any $0 < \beta \leq \frac{1}{3}$. This time bound is *tight* for graphs of such sparsity. In other words, under the Strong APSP Hypothesis, the naive $O(mn)$ time algorithm for u-dir-APSP with m edges by repeated BFSs is essentially optimal for sufficiently sparse graphs, and fast matrix multiplication helps only for sufficiently large densities. Such lower bounds for sparse graphs were known only for *weighted* APSP [LWW18, AR18].

Our technique also yields new conditional lower bounds for many other problems, such as All-Pairs Shortest Lightest Paths (undir-APSLP) or All-Pairs Lightest Shortest Paths (undir-APLSP) for undirected

small-weighted graphs (which was first studied by Zwick [Zwi99]), a batched version of the range mode problem (Batch-Range-Mode) (which has received considerable recent attention [CDL+14, VX20b, GPVX21, GH22, JX22]), and dynamic shortest paths in planar graphs [AD16]. See Table 1 for some of the specific results, and Section 6 for more discussion on all these problems. As demonstrated by the applications to range mode and dynamic planar shortest paths, our new technique will likely be useful to proving conditional lower bounds for other data structure problems, serving as an alternative to existing techniques based on the Combinatorial BMM Hypothesis (see e.g. [AV14]) or the OMv Hypothesis [HKNS15].

We also consider conditional lower bounds based on the hardness of u-dir-APSP itself. The following hypothesis has been proposed by Chan, Vassilevska W. and Xu [CVX21]:

Hypothesis 2 (u-dir-APSP Hypothesis)**.** *In the Word-RAM model with $O(\log n)$-bit words, computing APSP for an n-node unweighted directed graph requires at least $n^{2+\rho-o(1)}$ randomized time where ρ is the constant satisfying $\omega(1,\rho,1) = 1+2\rho$.*

As noted in Remark 6.3, the u-dir-APSP Hypothesis implies the Strong APSP Hypothesis if $\omega = 2$. The Strong APSP Hypothesis is thus more believable in some sense, but the u-dir-APSP Hypothesis allows us to prove higher lower bounds. For example, we prove the following conditional lower bound for Min-Witness-Prod:

Theorem 1.2. *Under the u-dir-APSP Hypothesis,* Min-Witness-Prod *requires $n^{2.223}$ time, or $n^{7/3-o(1)}$ time if $\omega = 2$ (on a Word-RAM with $O(\log n)$-bit words).*

Earlier papers [LPV20, CVX21] were unable to obtain such a reduction from u-dir-APSP to Min-Witness-Prod (Chan, Vassilevska W. and Xu [CVX21] were only able to reduce from u-dir-APSP to Min-Witness-Eq-Prod, but not Min-Witness-Prod). We similarly obtain higher lower bounds for all the other problems, as indicated in Table 1. Our results thus show that the u-dir-APSP Hypothesis is far more versatile for proving conditional lower bounds than what previous papers [LPV20, CVX21] were able to show.

Lastly, to deal with convolution-type problems, we introduce a similar strong version of the Min-Plus Convolution Hypothesis:

Hypothesis 3 (Strong Min-Plus Convolution Hypothesis)**.** *In the Word-RAM model with $O(\log n)$-bit words, Min-Plus convolution between two length n arrays with entries in $[n]$ requires $n^{2-o(1)}$ randomized time.*

The best algorithm for Min-Plus-Convolution with numbers in $[M]$ runs in $\widetilde{O}(Mn)$ time (by combining Alon, Galil and Margalit's technique [AGM97] and Fast Fourier transform), and no truly subquadratic time algorithm is known for $M = n$.

We do not know any direct relationship between the Strong APSP Hypothesis and the Strong Min-Plus Convolution Hypothesis (although when there are no restrictions on weights, it was known that Min-Plus-Convolution reduces to Min-Plus-Product [BCD+14]).

We prove the first conditional lower bounds for one intermediate convolution problem, Min-Equal-Convolution, under the Strong APSP Hypothesis, the u-dir-APSP Hypothesis or the Strong Min-Plus Convolution Hypothesis:

Theorem 1.3. Min-Equal-Convolution *requires $n^{1+1/6-o(1)}$ time under the Strong APSP Hypothesis; or $n^{1+\rho/2-o(1)}$ time under the u-dir-APSP Hypothesis; or $n^{1+1/11-o(1)}$ time under the Strong Min-Plus Convolution Hypothesis.*

In the above theorem, the lower bound under the Strong Min-Plus Convolution Hypothesis is the most interesting, requiring the use of one of our new variants of the BSG Theorem.

We should emphasize that the significance of Theorem 1.3 stems from the relative rarity of nontrivial lower bounds for convolution and related string problems. Some problems may have $n^{\omega/2}$ lower bounds by reduction from BMM (e.g. there was a well-known reduction from BMM to pattern-to-text Hamming distances, attributed to Indyk – see also [GU18]), but such bounds become meaningless when $\omega = 2$. A recent paper [CVX22] obtained an $n^{1.5-o(1)}$ lower bound under the OV Hypothesis for a problem called "pattern-to-text distinct Hamming similarity", which is far less natural and basic than Min-Equal-Convolution.

Lower Bounds / Hypotheses / Problems	Strong APSP		u-dir-APSP		Strong Min-Plus Convolution	
u-dir-APSP	$n^{7/3-o(1)}$	Cor. 3.5	$n^{5/2-o(1)}$	trivial	N/A	
Min-Witness-Prod	$n^{11/5-o(1)}$	Cor. 6.6	$n^{7/3-o(1)}$	Cor. 7.2	N/A	
undir-APSLP	$n^{11/5-o(1)}$	Cor. 6.8	$n^{7/3-o(1)}$	Cor. 7.3	N/A	
Batch-Range-Mode	$n^{7/6-o(1)}$	Cor. 6.10	$n^{5/4-o(1)}$	Cor. 7.4	N/A	
Min-Equal-Convolution	$n^{7/6-o(1)}$	Thm. 11.1	$n^{5/4-o(1)}$	Thm. 11.2	$n^{12/11-o(1)}$	Thm. 11.3

Table 1: New conditional lower bounds obtained under the Strong APSP Hypothesis, the u-dir-APSP Hypothesis and the Strong Min-Plus Convolution Hypothesis, assuming $\omega = 2$. (We also have nontrivial lower bounds even if $\omega > 2$; see the corresponding theorems/corollaries for the precise bounds.)

1.3 Equivalence of Counting and Detection

Here we summarize our main results on counting vs detection.

Equivalence between counting and detection for 3SUM. In #3SUM we are given three sets of numbers A, B, C and we want to count the number of triples $a \in A, b \in B, c \in C$ such that $a + b = c$.[6] More complex variants include #All-Nums-3SUM in which we want to count for every $c \in C$, the number of $a \in A, b \in B$ such that $a + b = c$. All of 3SUM, #3SUM and #All-Nums-3SUM can be solved in $O(n^2)$ time, via the folklore algorithm. The 3SUM Hypothesis (see [Vas18]) asserts that (in the word-RAM model of computation with $O(\log n)$-bit words), $n^{2-o(1)}$ time is needed to solve 3SUM. As no reduction from #3SUM to 3SUM is known till now, a priori it could be that the 3SUM Hypothesis is false, but that #3SUM still requires $n^{2-o(1)}$ time. We prove:

Theorem 1.4. #3SUM *and* 3SUM *are equivalent under truly subquadratic randomized fine-grained reductions.*

Counting of APSP and Min-Plus-Product. It is known that APSP in n-node graphs is equivalent to Min-Plus-Product of $n \times n$ matrices [FM71]. The counting variant #APSP of APSP asks to count for every pair of nodes in the given graph, the number of shortest paths between them. The counting variant of Min-Plus-Product asks, given two matrices A, B to count for all pairs i, j, the number of k such that $A_{ik} + B_{kj} = \min_\ell(A_{i\ell} + B_{\ell j})$, i.e. the number of witnesses of the Min-Plus-Product.

While APSP and Min-Plus-Product are equivalent, #APSP and #Min-Plus-Product are not known to be. The main issue is that the shortest paths counts can be exponential, and operations on such large numbers are costly in any reasonable model of computation such as the Word-RAM (see the discussion in Section 1.4). Because of this the fastest known algorithm for #APSP is actually *quartic* in the number of nodes, and

[6] There are several equivalent definitions of 3SUM: the predicate can be $a + b = c$, or $a + b + c = 0$; the integers could come from the same set A, or the input could consist of three sets A, B, C and we require $a \in A, b \in B, c \in C$. We work with the version with three input sets and predicate $a + b = c$.

not cubic. While we are able to improve upon the quartic running time (see Section 1.4), the running time is still supercubic, so it is unclear whether a tight reduction is possible from #APSP to #Min-Plus-Product.

Chan, Vassilevska W. and Xu [CVX21] defined several variants of #APSP that mitigate the existence of exponential counts. One variant, $\#_{\bmod U}$APSP computes the counts modulo any $O(\text{poly}\log n)$-bit integer U, and thus no computations with large integers are necessary. The problem can be solved in $\widetilde{O}(n^3)$ time, and can be reduced to #Min-Plus-Product (see Appendix A.3). We prove:

Theorem 1.5. #Min-Plus-Product *and* Min-Plus-Product *are equivalent under truly subcubic fine-grained reductions. For any* $O(\text{poly}\log n)$*-bit integer* $U \geq 2$*,* $\#_{\bmod U}$APSP *and* APSP *are equivalent under truly subcubic fine-grained reductions.*

Counting Exact Triangles. The Exact Triangle Problem (Exact-Tri) is: given an n-node graph with $O(\log n)$-bit integer edge weights, to determine whether the graph contains a triangle whose edge weights sum to some target value t. This problem is known to be at least as hard as both 3SUM and APSP [Păt10, VW13, VW18], so that if its brute-force cubic time algorithm can be improved upon, then both the 3SUM Hypothesis and the APSP Hypothesis would be false. Exact-Tri is among the hardest problems in fine-grained complexity. The counting variant #Exact-Tri asks for the number of triangles with weight t. We prove:

Theorem 1.6. Exact-Tri *and* #Exact-Tri *are equivalent under truly subcubic fine-grained reductions.*

Abboud, Feller and Weimann [AFW20] previously considered the problem of computing the *parity* of the number of zero-weight triangles, and it is equivalent to the problem of computing the parity of the number of triangles with weight t for a given t. Let's call this Parity-Exact-Tri. They showed that Exact-Tri can be reduced to Parity-Exact-Tri via a randomized subcubic fine-grained reduction. They were *not* able to obtain an equivalence, as it is not obvious at all that the decision problem should be able to solve the parity problem. Since #Exact-Tri can easily solve Parity-Exact-Tri (if one knows the count, one can take it mod 2), we also get that Parity-Exact-Tri is equivalent to Exact-Tri.

More equivalence results can be found in Section 8 and Appendix A. See Section 8.4 for further discussion on possible implications of our results.

New nondeterministic and quantum counting algorithms. Using our new techniques, we can also provide efficient nondeterministic algorithms for #Neg-Tri (counting the number of triangles with negative weights), #Exact-Tri and #3SUM even when there are real-valued inputs. Ours are the first nondeterministic algorithms for these problems that beat their essentially cubic and quadratic deterministic algorithms. See Section 9.4 for the complexity-theoretic implications of these results.

Our equivalence results can be used to obtain new quantum algorithms. It is known that 3SUM can be solved in $\widetilde{O}(n)$ quantum time by an algorithm that uses Grover search (see e.g. [AL20]). However, it is not clear how to adapt that algorithm to solve #3SUM since Grover search cannot count the number of solutions exactly. By combining the $\widetilde{O}(n)$ quantum time 3SUM algorithm with our subquadratic equivalence between 3SUM and #3SUM in a black box way, we immediately obtain the first truly subquadratic time quantum algorithm for #3SUM.

1.4 New Variants of the BSG Theorem with Algorithmic Applications

As a way to address question Q4, we formulate a new "Triangle Decomposition Theorem", which follows from our techniques. The theorem can be stated roughly as follows:

> For any weighted n-node graph G and parameter s, there exist $O(s^3)$ subgraphs $G^{(\lambda)}$ such that the set of all triangles of weight zero (or any fixed target value) in G can be decomposed as the

union of the set of all triangles in $G^{(\lambda)}$, plus a small remainder set of $O(n^3/s)$ triangles. (All triangles in each $G^{(\lambda)}$ are guaranteed to have weight zero.)

Thus, the set of all zero-weight triangles in a graph is highly structured in some sense (in particular, if there are many zero-weight triangles, one can extract large subgraphs in which all triangles are zero-weight triangles). See Theorem 5.1 for the precise statement. From this theorem, we can easily rederive a subcubic algorithm for monotone Min-Plus product (as shown in Appendix B), if we don't care about optimizing the exponent in the running time. The theorem also leads to several new algorithms, as listed below.

This Triangle Decomposition Theorem resembles a covering version of the BSG Theorem, as formulated by Chan and Lewenstein [CL15], which can be roughly stated as follows:

> For any sets A, B, C in an abelian group and parameter s, there exist $O(s)$ pairs of subsets $(A^{(\lambda)}, B^{(\lambda)})$ such that $\{(a,b) \in A \times B : a+b \in C\}$ can be covered by the union of $A^{(\lambda)} \times B^{(\lambda)}$, plus a small remainder set of $O(n^2/s)$ pairs, where the total size of the sum sets[7] $A^{(\lambda)} + B^{(\lambda)}$ is $O(s^6 n)$.

Thus, the set of all solutions to 3SUM is highly structured in some sense (in particular, if there are many solutions to 3SUM, one can extract pairs of large subsets $(A^{(\lambda)}, B^{(\lambda)})$ whose sum sets are small). See Section 10 for the precise statement and for more background on the BSG Theorem.

We show that our approach, combined with some extra ideas, can actually prove a form of the BSG Theorem, namely, with the above $O(s^6 n)$ bound replaced by $\widetilde{O}(s^2 n^{3/2})$. At first, this new bound appears weaker—indeed, researchers in additive combinatorics were concerned more with obtaining a linear bound in n on the sum set size, and less with the dependency on s. However, Chan and Lewenstein's algorithmic applications require nonconstant choices of s, and the new bound lowering the s^6 factor to s^2 turns out to yield better results in at least one application below (namely, 3SUM with preprocessed universes).

We now mention a few applications of the new theorems:

#APSP for arbitrary weighted graphs. Recall the #APSP problem (counting the number of shortest paths from u to v, for every pair of nodes u, v in a weighted graph). This basic problem has applications to *betweenness centrality*. As mentioned earlier, because counts may be exponentially big, known algorithms run in near-n^4 time in the standard Word-RAM model with $O(\log n)$-bit words. An $\widetilde{O}(n^3)$-time algorithm for #APSP was given by Chan, Vassilevska W. and Xu [CVX21], but only for *unweighted* graphs. We give the first truly subquartic #APSP algorithm for arbitrary weighted graphs with running time $O(n^{3.83})$ (or $\widetilde{O}(n^{15/4})$ if $\omega = 2$).

3SUM in preprocessed universes. Chan and Lewenstein [CL15] studied a "preprocessed universe" setting for the 3SUM problem: preprocess three sets A, B, C of n integers (the *universes*) so that for any given query subsets $A' \subseteq A$, $B' \subseteq B$, and $C' \subseteq C$, we can solve 3SUM on the instance (A', B', C') more quickly than from scratch. (The problem was first stated by Bansal and Williams [BW12].) Chan and Lewenstein showed intriguingly that after preprocessing the universes in $\widetilde{O}(n^2)$ expected time, 3SUM on the given query subsets can be solved in time truly subquadratic in n, namely, $\widetilde{O}(n^{13/7})$. Our techniques yield a new, simpler solution with $\widetilde{O}(n^2)$ expected preprocessing time and an improved query time of $\widetilde{O}(n^{11/6})$. Furthermore, with the same $\widetilde{O}(n^{11/6})$ query time, the new algorithm can solve a slight generalization where the query subset C' can be an arbitrary set of n integers (i.e., the universe C needs not be specified). Here, our improvement is even bigger: Chan and Lewenstein's previous solution for this generalized case required $\widetilde{O}(n^{19/10})$ query time.

[7] Throughout the paper, $A + B$ denotes the sum set $\{a + b : a \in A,\ b \in B\}$, and $A - B$ denotes the difference set $\{a - b : a \in A,\ b \in B\}$.

We also obtain the first result for the analogous problem of Exact-Tri in preprocessed universes: we can preprocess any weighted n-node graph in $\widetilde{O}(n^3)$ time, so that for any given query subgraph, we can solve Exact-Tri in time truly subcubic in n, namely, $O(n^{2.83})$ (or $\widetilde{O}(n^{11/4})$ if $\omega = 2$). This result can be viewed as a generalization of the result for 3SUM, by known reductions from Exact-Tri to 3SUM (for integers).

3SUM for monotone sets in $[n]^d$. Chan and Lewenstein [CL15] also studied a special case of 3SUM for monotone sets A, B, C in $[n]^d$ for a constant dimension d, where a set of points is *monotone* if it is of the form $\{a_1, \ldots, a_n\}$ where the j-th coordinates of $a_1, \ldots, a_n$ is a monotone sequence for each j. The problem is related to an important special case of Min-Plus convolution for monotone sequences in $[n]$ (or integer sequences with bounded differences), which reduces to monotone 3SUM in 2 dimensions. This is also related to a data structure problem for strings known as *jumbled indexing*, where d corresponds to the alphabet size. Chan and Lewenstein gave the first truly subquadratic algorithm for the problem, with running time of the form $O(n^{2-1/(d+O(1))})$ using randomization. (See [ACLL14, HU17] for conditional lower bounds on this and related problems.) However, they obtained subquadratic deterministic algorithms only for $d \leq 7$ under the current fast matrix multiplication bounds. Our techniques give the first *deterministic* algorithm for all constant d, with running time $O(n^{2-1/O(d)})$. Although the new bound is not better (and for monotone Min-Plus convolution, a recent paper by Chi, Duan, Xie and Zhang [CDXZ22] presented even faster randomized algorithms), the new approach is simpler, besides being deterministic.

1.5 Paper Organization

In Section 2, we define notations and problems. The rest of the paper has three main threads:

- *Conditional lower bounds for problems with intermediate complexity*: In Section 3, we illustrate our approach by proving the first superquadratic lower bound for u-dir-APSP under the Strong APSP Hypothesis. In Sections 6–7, we prove lower bounds for other problems, including Min-Witness-Prod, under both the Strong APSP Hypothesis and u-dir-APSP Hypothesis.

- *Equivalences between counting and detection problems*: In Section 4, we illustrate our basic idea by proving the subcubic equivalence between #Exact-Tri and Exact-Tri. In Section 8, we prove more results of this kind, including the subquadratic equivalence between #3SUM and 3SUM. (Still more examples can be found in Appendix A.) In Section 9, we further adapt these ideas to obtain new nondeterministic and quantum algorithms for counting problems.

- *BSG-related theorems*: In Section 5, we present our new Triangle Decomposition Theorem and describe its applications. In Section 10, we describe our new variants of the BSG theorem. (Still more applications and variants can be found in Appendices B and C.) Finally, in Section 11, we prove the conditional lower bounds for Min-Equal-Convolution, the most sophisticated of which use one of our new BSG theorems.

Although the paper is lengthy, Sections 3, 4 and 5 should suffice to give the readers an overview of our main proof techniques. (Readers interested in diving deeper into any of the above threads may proceed to the subsequent sections.)

2 Problem Definitions

For integer $n \geq 1$, we use $[n]$ to denote $\{1, 2, \ldots, n\}$ and use $[\pm n]$ to denote $\{-n, -(n-1), \ldots, n\}$. For a predicate X, we use $[X]$ to denote a $\{0, 1\}$ value which evaluates to 1 if X is true, and 0 otherwise.

We use $M(n_1, n_2, n_3)$ to denote the (randomized) running time to multiply an $n_1 \times n_2$ matrix with an $n_2 \times n_3$ matrix. We consider the Word-RAM model of computation with $O(\log n)$-bit words, and all input numbers are $O(\log n)$-bit integers, unless otherwise stated.

Problem 2.1 (Min-Plus-Product)**.** *Given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, compute the $n_1 \times n_3$ matrix C where $C_{ij} = \min_{k \in [n_2]}(A_{ik} + B_{kj})$. We denote C by $A \star B$.*

Furthermore, we use $M^*(n_1, n_2, n_3 \mid \ell)$ to denote the (randomized) running time to compute the Min-Plus product between an $n_1 \times n_2$ matrix and an $n_2 \times n_3$ matrix where the entries of both matrices are from $[\ell] \cup \{\infty\}$.

Problem 2.2 (APSP)**.** *Given an edge-weighted n-node graph, compute its all-pairs shortest path distances.*

Problem 2.3 (Min-Plus-Convolution)**.** *Given two length n arrays A, B, compute the length n array C, where $C_i = \min_{k \in [i-1]}(A_k + B_{i-k})$.*

Problem 2.4 (Exact-Tri)**.** *Given an edge-weighted n-node graph G and a target value t, determine whether G contains a triangle whose edge weights sum up to t. In its All-Edges version (*AE-Exact-Tri*), we need to determine for every edge e, whether e is contained in a triangle whose edge weights sum up to t.*

Problem 2.5 (Neg-Tri)**.** *Given an edge-weighted n-node graph G, determine whether G contains a triangle whose edge weights sum up to a negative value. In its All-Edges version (*AE-Neg-Tri*), we need to determine for every edge e, whether e is contained in a triangle whose edge weights sum up to a negative value.*

Problem 2.6 (Exact-k-Clique)**.** *Given an edge-weighted n-node graph G and a target value t, determine whether G contains a k-clique whose edge weights sum up to t.*

Problem 2.7 (Min-k-Clique)**.** *Given an edge-weighted n-node graph G, determine the minimum edge weight sum over all k-cliques in G.*

Problem 2.8 (3SUM)**.** *Given three sets of numbers A, B, C of size n, determine if there exist $a \in A, b \in B, c \in C$ such that $a + b = c$. In its All-Numbers version (*All-Nums-3SUM*), we need to determine for each $c \in C$, whether there exist $a \in A, b \in B$ such that $a + b = c$.*

Problem 2.9 (3SUM-Convolution)**.** *Given three arrays of numbers A, B, C of lengths n, determine if there exist $i, j \in [n]$ such that $A_i + B_j = C_{i+j}$. In its All-Numbers version (*All-Nums-3SUM-Convolution*), we need to determine for each $k \in [n]$, whether there exist $i, j \in [n]$ such that $i + j = k$ and $A_i + B_j = C_k$.*

Problem 2.10 (Mono-Convolution)**.** *Given three arrays of numbers A, B, C of length n, determine for every $k \in [n]$, whether there exist $i, j \in [n]$ such that $i + j = k$ and $A_i = B_j = C_k$.*

All the problems defined above have natural counting variants, which will be denoted by adding a "#" prefix to the problems' names. For a detection problem that outputs a single bit or multiple bits, each bit represents whether an object that satisfies some requirements exists among a set of candidates; in its counting variant, we need to output a number in place of each bit to denote the number of objects that satisfy the requirements among the same set of candidates. For instance, #Exact-Tri asks to count the number of triangles in the graph whose edge weights sum up to t, and #AE-Exact-Tri asks to count, for each edge e, the number of triangles containing e whose edge weights sum up to t. For a minimization problem that outputs one or multiple optimal values, its counting variant needs to output the number of optimal solutions in place of each optimal value. For instance, #Min-Plus-Product asks for each $(i, j) \in [n_1] \times [n_3]$, the

number of k where $A_{ik} + B_{kj} = (A \star B)_{ij}$. One special problem is the $\#_{\mod U}$APSP problem, in which we need to compute the number of shortest paths mod U for every pair of nodes.

Clearly, for any of the detection problems, its counting variant is (not necessarily strictly) harder than its original version. The same is not obviously true for minimization problems.

Problem 2.11 (u-dir-APSP)**.** *Given an unweighted directed graph, compute its all-pairs shortest path distances.*

Problem 2.12 (Min-Witness-Prod)**.** *Given an $n_1 \times n_2$ Boolean matrix A and an $n_2 \times n_3$ Boolean matrix B, compute $\min\{k \in [n_2] : A_{ik} \wedge B_{kj}\}$ for every $(i, j) \in [n_1] \times [n_3]$.*

Problem 2.13 (Equality-Product)**.** *Given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, compute the number of $k \in [n_2]$ such that $A_{ik} = B_{kj}$ for every $(i, j) \in [n_1] \times [n_3]$.*

Problem 2.14 (Min-Witness-Eq-Prod)**.** *Given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, compute $\min\{k \in [n_2] : A_{ik} = B_{kj}\}$ for every $(i, j) \in [n_1] \times [n_3]$.*

Problem 2.15 (Min-Equal-Prod)**.** *Given an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, compute $\min\{A_{ik} : k \in [n_2] \wedge A_{ik} = B_{kj}\}$ for every $(i, j) \in [n_1] \times [n_3]$.*

Problem 2.16 (undir-APSLP$_{1,2}$)**.** *Given an n-node undirected graph whose edge weights are either 1 or 2, compute for every pair of nodes s, t, the smallest number of edges required to travel from s to t, and the minimum weight over all paths using the smallest number of edges.*

Problem 2.17 (undir-APLSP$_{1,2}$)**.** *Given an n-node undirected graph whose edge weights are either 1 or 2, compute for every pair of nodes s, t, the shortest path distance from s to t, and the smallest number of edges among all shortest paths.*

Problem 2.18 (Batch-Range-Mode)**.** *Given an length N array and Q intervals of the array, compute the element that appears the most (breaking ties arbitrarily) for each of the intervals.*

Problem 2.19 (Min-Equal-Convolution)**.** *Given two length n arrays A, B, compute the length n array C, where $C_i = \min\{A_j : j \in [i-1] \wedge A_j = B_{i-j}\}$.*

3 Conditional Lower Bounds for Problems with Intermediate Complexity: u-dir-APSP under the Strong APSP Hypothesis

In this section, we apply the idea of combining Fredman's trick with Equality Product to prove conditional lower bounds for problems with intermediate complexity. To illustrate the idea, we focus on lower bounds for the u-dir-APSP problem under the Strong APSP Hypothesis, but the approach also leads to lower bounds for Min-Witness-Prod and other problems, under the Strong APSP Hypothesis as well as the u-dir-APSP Hypothesis, as we will explain later in Sections 6–7. As noted earlier, the Strong APSP Hypothesis is equivalent to the hypothesis that $M^*(n, n, n \mid n^{3-\omega})$ is not truly subcubic. Thus, it suffices to describe fine-grained reductions from the Min-Plus-Product problem for two $n \times n$ matrices with bounded integer entries from $[n^{3-\omega}]$, to the u-dir-APSP problem.

When devising a reduction from one problem to another (as in typical NP-hardness proofs), we often concentrate on understanding the power of the latter problem. In our new reductions, we will focus mostly on the former problem instead, interestingly: we will attempt to design an algorithm to solve the Min-Plus-Product problem for bounded integers in subcubic time, and only at the end, reveal how an oracle for u-dir-APSP (or Min-Witness-Prod and other problems) could help.

3.1 Preliminaries: Generalized Equality Products

By Matoušek's technique for dominance product [Mat91, Yus09], the equality product of an $n_1 \times n_2$ and an $n_2 \times n_3$ matrix can be computed in time $\widetilde{O}\big(\min_r(n_1n_2n_3/r + M(n_1, rn_2, n_3))\big)$. For example, in the case $n_1 = n_2 = n_3 = n$, the bound is at most $\widetilde{O}(\min_r(n^3/r + rn^\omega)) = \widetilde{O}(n^{(3+\omega)/2})$, as we have stated before, if we don't use rectangular matrix multiplication exponents. We begin with the following lemma describing a straightforward generalization, which will be useful later:

Lemma 3.1. *Given $n_1 \times n_2$ matrices A and A', and $n_2 \times n_3$ matrices B and B', define the* generalized equality product *of (A, A') and (B, B') to be the following $n_1 \times n_3$ matrix E:*

$$E_{ij} := \min_{k:\, A_{ik}=B_{kj}} (A'_{ik} + B'_{kj}).$$

Suppose that all matrix entries of A' and B' are in $[\pm\ell] \cup \{\infty\}$. For any r, we can compute E in time

$$\widetilde{O}\big(n_1n_2n_3/r + M^*(n_1, rn_2, n_3 \mid \ell)\big).$$

Proof. Sort $B_k = (B_{kj})_{j\in[n_3]}$, i.e., the k-th column of B. Let F_k be the set of elements that have frequency more than n_3/r in B_k. Note that $|F_k| \le r$. We divide into two cases, computing two $n_1 \times n_3$ matrices E^L and E^H; the answers will be $E_{ij} = \min\{E^L_{ij}, E^H_{ij}\}$.

– **Low-frequency case: computing $E^L_{ij} = \min_{k:\, A_{ik}=B_{kj}\notin F_k} (A'_{ik} + B'_{kj})$.** Initially set $E^L_{ij} = \infty$. For each $i \in [n_1]$ and $k \in [n_2]$, if $A_{ik} \notin F_k$, we examine each of the at most n_3/r indices j with $A_{ik} = B_{kj}$, and reset $E^L_{ij} = \min\{E^L_{ij},\ A'_{ik} + B'_{kj}\}$. All this takes $O(n_1n_2 \cdot n_3/r)$ time.

– **High-frequency case: computing $E^H_{ij} = \min_{k:\, A_{ik}=B_{kj}\in F_k} (A'_{ik} + B'_{kj})$.** For each $i \in [n_1]$, $k \in [n_3]$, and $p \in F_k$, let $\hat{A}_{i,(k,p)} = A'_{ik}$ if $A_{ik} = p$, and $\hat{A}_{i,(k,p)} = \infty$ otherwise. For each $k \in [n_3]$, $j \in [n_2]$, and $p \in F_k$, let $\hat{B}_{(k,p),j} = B'_{kj}$ if $B_{kj} = p$, and $\hat{B}_{(k,p),j} = \infty$ otherwise. We let $E^H_{ij} = \min_{k\in[n_2],p\in F_k}(\hat{A}_{i,(k,p)} + \hat{B}_{(k,p),j})$. This can be computed by a Min-Plus product in $O(M^*(n_1, rn_2, n_3 \mid \ell))$ time.

□

3.2 The Key Reduction

We now present an approach to solve the Min-Plus-Product problem for two matrices with integer entries in $[\ell]$, by reducing it to instances that have simultaneously a smaller inner dimension s and smaller integer entries in $[t]$ with $t \le \ell$:

Theorem 3.2. *For any r, s, t with $s \le n_2$ and $t \le \ell$,*

$$M^*(n_1, n_2, n_3 \mid \ell) = \widetilde{O}\big((n_2/s)M^*(n_1, s, n_3 \mid t) + sn_1n_2n_3/r + sM^*(n_1, rn_2, n_3 \mid \ell/t)\big).$$

Proof. Let $g = \lceil \ell/t \rceil$. Let A be an $n_1 \times n_2$ matrix and B be an $n_2 \times n_3$ matrix, where all matrix entries are in $[\ell] \cup \{\infty\}$. We describe an algorithm to compute the Min-Plus product of A and B. Without loss of generality, we may assume that $(A_{ik} \bmod g) < g/2$ for all i, k with A_{ik} finite, since we can separate the problem into two instances $(A^<, B)$ and $(A^\ge, B)$ for two matrices $A^<$ and $A^\ge$, where $A^<_{ik} = A_{ik}$ and $A^\ge_{ik} = \infty$ if $(A_{ik} \bmod g) < g/2$, and $A^<_{ik} = \infty$ and $A^\ge_{ik} = A_{ik} - g/2$ if $(A_{ik} \bmod g) \ge g/2$. Similarly, we may assume that $(B_{kj} \bmod g) < g/2$ for all k, j with B_{kj} finite.

For each i, k, write A_{ik} as $A'_{ik}g + A''_{ik}$ with $0 \le A'_{ik} \le t$ and $0 \le A''_{ik} < g/2$. Similarly, for each k, j, write B_{kj} as $B'_{kj}g + B''_{kj}$ with $0 \le B'_{kj} \le t$ and $0 \le B''_{kj} < g/2$. (Set $A'_{ik} = A''_{ik} = \infty$ if $A_{ik} = \infty$, and $B'_{kj} = b''_{kj} = \infty$ if $B_{kj} = \infty$.)

We first compute the Min-Plus product C' of A' and B' (i.e., $C'_{ij} = \min_k(A'_{ik} + B'_{kj})$), in time $O(M^*(n_1, n_2, n_3 \mid t)) \le O((n_2/s) \cdot M^*(n_1, s, n_3 \mid t))$.

Let $W_{ij} = \{k \in [n_2] : A'_{ik} + B'_{kj} = C'_{ij}\}$; the elements in W_{ij} are the *witnesses* for C'_{ij}. The Min-Plus product C of A and B is given by $C_{ij} = C'_{ij}g + C''_{ij}$, where

$$C''_{ij} := \min_{k \in W_{ij}} (A''_{ik} + B''_{kj}).$$

It suffices to describe how to compute C''. We divide into two cases:

- **Few-witnesses case: computing C''_{ij} for all i, j with $|W_{ij}| \le n_2/s$.** For each such (i, j), we will explicitly enumerate all witnesses in W_{ij}. This can be done by standard techniques for witness finding [AGMN92, Sei95]: first, observe that if the witness is unique (i.e., $|W_{ij}| = 1$), it can be found by performing $O(\log n_2)$ Min-Plus products (namely, for each $\ell \in [\log n_2]$, the ℓ-th bit of the witness for C'_{ij} is 1 iff $\min_{k \in K_\ell}(A'_{ik} + B'_{kj}) = C_{ij}$, where $K_\ell := \{k \in [n_2] : \text{the } \ell\text{-th bit of } k \text{ is } 1\}$). We take a random subset $R \subseteq [n_2]$ of s indices, and find witnesses for the Min-Plus product of $(A'_{ik})_{i \in [n_1], k \in R}$ and $(B'_{kj})_{k \in R, j \in [n_3]}$ if the witnesses are unique. This takes $\widetilde{O}(M^*(n_1, s, n_3 \mid t))$ time. Fix i, j with $|W_{ij}| \le n_2/s$. For a fixed element $w \in W_{ij}$, the probability that w is found, i.e., w is in R but no other element of W_{ij} is in R, is $\Omega((s/n_2) \cdot (1 - s/n_2)^{n_2/s}) = \Omega(s/n_2)$. By repeating $O((n_2/s)\log(n_1 n_2 n_3))$ times (with different choices of R), all witnesses in W_{ij} would be found w.h.p. Once the entire witness set W_{ij} is found, we can compute each C''_{ij} naively in $O(|W_{ij}|)$ time. The total running time is $\widetilde{O}((n_2/s) \cdot M^*(n_1, s, n_3 \mid t))$.

- **Many-witnesses case: computing C''_{ij} for all i, j with $|W_{ij}| > n_2/s$.** Pick a random subset H of size $c_0 s \log(n_1 n_2 n_3)$ for a sufficiently large constant c_0. Then H hits (i.e., intersects) every witness set W_{ij} with $|W_{ij}| > n_2/s$ w.h.p.

  We do the following: for each $k_0 \in H$ and for each $i \in [n_1], j \in [n_3]$, if $A'_{ik_0} + B'_{k_0 j} = C'_{ij}$ (i.e., $k_0 \in W_{ij}$), set

$$C''_{ij} = \min_{k:\, A'_{ik} - A'_{ik_0} = B'_{k_0 j} - B'_{kj}} (A''_{ik} + B''_{kj}). \tag{1}$$

  Correctness of (1) follows immediately from "Fredman's trick": $A'_{ik} - A'_{ik_0} = B'_{k_0 j} - B'_{kj}$ is equivalent to $A'_{ik} + B'_{kj} = A'_{ik_0} + B'_{k_0 j}$, which is equivalent to $k \in W_{ij}$, assuming $A'_{ik_0} + B'_{k_0 j} = C'_{ij}$. Thus, the above correctly computes C''_{ij} for every i, j with $|W_{ij}| > n_2/s$, since H hits W_{ij}.

  Finally, we observe that for a fixed k_0, the right-hand side in (1) corresponds precisely to a generalized equality product! By Lemma 3.1, they can be computed in $\widetilde{O}(n_1 n_2 n_3/r + M^*(n_1, r n_2, n_3 \mid g))$ time, for each of the $\widetilde{O}(s)$ choices of k_0.

□

Note that Fredman's trick was originally introduced to solve APSP or compute Min-Plus products for arbitrary *real*-valued inputs. It is interesting that the trick is useful even when input values are in a restricted integer range (in $[t]$).

Note also that a more naive attempt to prove the above theorem is to just bound $M^*(n_1, n_2, n_3 \mid \ell)$ by $(n_2/s)M^*(n_1, s, n_3 \mid \ell)$, and once the inner dimension n_2 is reduced to s in the subproblems, try to use hashing to reduce the range of the integers to, say, $[\widetilde{O}(s^2)]$. However, while such a hashing approach might work for equality-type problems (e.g., AE-Exact-Tri), it does not work at all for Min-Plus-Product.

3.3 Consequences

In Theorem 3.2, we can directly bound the third term by using existing matrix multiplication results [AGMN92], leading to the following corollary:

Corollary 3.3. *For any constant* $0 < \beta \leq (3-\omega)/2$*, if* $M^*(n, n^\beta, n \mid n^{2\beta}) = O(n^{2+\beta-\varepsilon})$*, then* $M^*(n,n,n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

Proof. By Theorem 3.2,

$$\begin{aligned} M^*(n,n,n \mid n^{3-\omega}) &= \widetilde{O}\left((n/s)M^*(n,s,n \mid t) + sn^3/r + (sn^{3-\omega}/t)M(n,rn,n)\right) \\ &\leq \widetilde{O}\left((n/s)M^*(n,s,n \mid t) + sn^3/r + rsn^3/t\right). \end{aligned}$$

Setting $r = n^\beta$, $s = n^{\beta-\varepsilon'}$, and $t = n^{2\beta}$ with $\varepsilon' = \varepsilon/2$ yields $M^*(n,n,n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$. □

The above corollary establishes a conditional lower bound of $n^{2+\beta-\varepsilon}$ for the subproblem of Min-Plus product for rectangular matrices of dimension $n \times n^\beta$ and $n^\beta \times n$ for integers bounded by $n^{2\beta}$, under the Strong APSP Hypothesis. This lower bound is tight in the sense that $O(n^{2+\beta})$ is an obvious upper bound (though the range of allowed integer values $[n^{2\beta}]$ may not be tight). We will now use this corollary to derive conditional lower bounds for u-dir-APSP.

Let U-DIR-APSP(n) be the time complexity of u-dir-APSP on n-node graphs. More generally, let U-DIR-APSP(n,m) be the time complexity of APSP on an unweighted directed graph with n nodes and m edges. Chan, Vassilevska W., and Xu [CVX21] have given a simple reduction of Min-Plus product for rectangular matrices of certain inner dimensions and integer ranges to u-dir-APSP, as summarized by the following lemma (it is easy to check that the graph in their reduction has $O(nx)$ edges).

Lemma 3.4. *For any* x, y*, we have* $M^*(n,x,n \mid y) = O(\text{U-DIR-APSP}(n, nx))$ *if* $xy \leq n$.

Combining Corollary 3.3 and Lemma 3.4 immediately gives the following:

Corollary 3.5. *For any constant* $\beta \leq \min\{1/3, (3-\omega)/2\}$*, if* U-DIR-APSP$(n, n^{1+\beta}) = O(n^{2+\beta-\varepsilon})$*, then* $M^*(n,n,n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

By setting $\beta = 1/3$, we have thus proved that u-dir-APSP cannot be solved in $O(n^{7/3-\varepsilon})$ time under the Strong APSP Hypothesis, assuming $1/3 \leq (3-\omega)/2$ (this assumption can be removed, as we observe later in Section 7). In particular, if $\omega = 2$, this implies that u-dir-APSP is strictly harder than unweighted undirected APSP, as the latter problem can be solved in $\widetilde{O}(n^\omega)$ time [Sei95].

Furthermore, u-dir-APSP for a graph with $n^{1+\beta}$ edges cannot be solved in $O(n^{2+\beta-\varepsilon})$ time for any $\beta \leq 1/3$ under the same hypothesis, assuming $1/3 \leq (3-\omega)/2$ (again this assumption can be removed). In other words, the naive algorithm by repeated BFSs is essentially *optimal* for sufficiently sparse graphs.

If we assume a weaker hypothesis that APSP does not have truly subcubic algorithms for edge weights in $[n]$ instead of $[n^{3-\omega}]$, it can be checked that we still get a lower bound near $n^{2+(3-\omega)/3}$ for u-dir-APSP. In fact, assuming that APSP does not have truly subcubic algorithms for edge weights in $[n^\lambda]$, we can still obtain a super-quadratic lower bound for u-dir-APSP for λ as large as 1.99, if $\omega = 2$. For simplicity, we will concentrate only on the version of the Strong APSP Hypothesis with $\lambda = 3-\omega$ throughout the paper.

In Sections 6–7, we will use the same approach to derive further conditional lower bounds for Min-Witness-Prod, undir-APLSP$_{1,2}$, and Batch-Range-Mode, from both the Strong APSP Hypothesis and the u-dir-APSP Hypothesis.

4 Equivalences Between Counting and Detection Problems: #Exact-Triangle vs. Exact-Triangle

In this section, we describe a simple approach to proving equivalence between counting and detection problems, by combining Fredman's trick with Equality Product. To illustrate the basic idea, we focus on the equivalence of #AE-Exact-Tri and AE-Exact-Tri. With more work and additional ideas, the approach can also establish the equivalence of #3SUM and 3SUM, as we will later explain in Section 8.

We use G to denote the input graph of an AE-Exact-Tri or #AE-Exact-Tri instance, we use w to denote the weight function, and we use W_{ij} to denote the set of k where (i,j,k) forms a triangle whose edge weights sum up to the target value t.

Lemma 4.1. *Given an n node graph G, a target value t, and a subset $S \subseteq V(G)$, we can compute a matrix D in $\widetilde{O}(|S| \cdot n^{(3+\omega)/2})$ time such that $D_{ij} = |W_{ij}|$ whenever $S \cap W_{ij} \neq \emptyset$.*

Proof. For every $s \in S$, we do the following. Let $A^{(s)}$ be a matrix where $A^{(s)}_{ik} = w(i,k) - w(i,s)$ and $B^{(s)}_{kj} = w(s,j) - w(k,j)$. Then we use compute the equality product $C^{(s)}$ of $A^{(s)}$ and $B^{(s)}$ in $\widetilde{O}(n^{(3+\omega)/2})$ time for each s [Mat91]. Finally, if there exists $s \in S$ such that $A_{is} + B_{sj} + w(i,j) = t$, we let D_{ij} be $C^{(s)}_{ij}$ for an arbitrary s with the property; otherwise, we let D_{ij} be 0 (we don't care about its value in this case). The running time for computing D is clearly $\widetilde{O}(|S| \cdot n^{(3+\omega)/2})$.

Suppose $S \cap W_{ij} \neq \emptyset$ for some (i,j). Then D_{ij} equals $C^{(s)}_{ij}$ for some s where $A_{is} + B_{sj} + w(i,j) = t$. By Fredman's trick, $A^{(s)}_{ik} = B^{(s)}_{kj}$ if and only if $w(i,k) + w(k,j) + w(i,j) = w(i,s) + w(s,j) + w(i,j) = t$. Therefore, $D_{ij} = C^{(s)}_{ij} = |W_{ij}|$. □

Theorem 4.2. *If* AE-Exact-Tri *for n-node graphs has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #AE-Exact-Tri *for n-node graphs has an $O(n^{3-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$*

Proof. Given a #AE-Exact-Tri instance on an n-node graph G, we first list up to $n^{0.99}$ elements in W_{ij} for every i,j. By well-known techniques (e.g. [VW18]), an $O(n^{3-\varepsilon})$ time AE-Exact-Tri algorithm implies an $O(n^{3-\varepsilon''})$ for $\varepsilon'' > 0$ time algorithm for listing up to $n^{0.99}$ witnesses for each (i,j) in an AE-Exact-Tri instance.

If we list less than $n^{0.99}$ elements for some (i,j), we can output the number of elements we list as the exact witness count for (i,j). By the standard greedy algorithm for hitting set, in $\widetilde{O}(n^{2.99})$ time, we can find a set S of size $\widetilde{O}(n^{0.01})$ that intersect with W_{ij} for the remaining pairs (i,j). Therefore, we can apply Lemma 4.1 to compute the number of witnesses for these remaining (i,j) pairs in $\widetilde{O}(|S| \cdot n^{(3+\omega)/2}) \leq O(n^{2.70})$ time.

The total running time for the #AE-Exact-Tri instance is thus $\widetilde{O}(n^{3-\varepsilon''} + n^{2.99} + n^{2.70})$, which is truly subcubic. □

The reduction from AE-Exact-Tri to #AE-Exact-Tri is trivial.

Remark 4.3. Given Theorem 4.2, it is simple to derive a subcubic equivalence between Exact-Tri and #Exact-Tri. First, the reduction from Exact-Tri to #Exact-Tri is trivial. To reduce #Exact-Tri to Exact-Tri, we first reduce #Exact-Tri to #AE-Exact-Tri in the trivial way, then use Theorem 4.2 to further reduce it to AE-Exact-Tri, and finally reduce it to Exact-Tri by known reductions [VW18].

In Section 8, we will use a similar approach to obtain other equivalence results between counting and detection problems. In particular, the proof of subquadratic equivalence between #3SUM and 3SUM will require further technical ideas: we will need to exploit or modify known reductions from All-Nums-3SUM-Convolution to AE-Exact-Tri, and 3SUM to 3SUM-Convolution.

5 Alternative to BSG: A Triangle Decomposition Theorem and Its Applications

In this section, we introduce a decomposition theorem for zero-weight triangles, which encapsulates some of the key ideas we have used, and which may be viewed as an alternative to the BSG Theorem. We will describe applications of this decomposition theorem to some algorithmic problems that were previously solved via the BSG Theorem by Chan and Lewenstein [CL15], as well as a new application to the #APSP problem for arbitrary weighted graphs.

In a weighted tripartite graph G with node sets U, X, and V, let Triangles(G) denote the set of all triangles in $U \times X \times V$ in G, and let Zero-Triangles(G) denote the set of all zero-weight triangles in $U \times X \times V$ in G.

Theorem 5.1. (Triangle Decomposition) *Given a real-weighted tripartite graph G with n_1, n_2, and n_3 nodes in its three parts U, X, and V, and given a parameter s, there exist a collection of $\ell = \widetilde{O}(s^3)$ subgraphs $G^{(1)}, \ldots, G^{(\ell)}$ of G, and a set R of $\widetilde{O}(n_1 n_2 n_3/s)$ triangles, such that*

$$\text{Zero-Triangles}(G) \;=\; R \cup \bigcup_{\lambda=1}^{\ell} \text{Triangles}(G^{(\lambda)}).$$

The subsets in the union above are disjoint. The $G^{(\lambda)}$'s and R can be constructed in $\widetilde{O}(n_1 n_2 n_3 + s n_1 n_2 + s n_2 n_3 + s^2 n_1 n_3)$ deterministic time. And if the edge weights between U and V later change, then the $G^{(\lambda)}$'s and R can be updated in $\widetilde{O}(n_1 n_2 n_3/s + s^2 n_1 n_3)$ time.

Furthermore, the subgraphs can be grouped into $\widetilde{O}(s)$ categories of $\widetilde{O}(s^2)$ subgraphs each, such that if an edge $uv \in U \times V$ is present in one subgraph, it is present in all subgraphs of the same category.

Proof. Let $\{u_i : i \in [n_1]\}$, $\{x_k : k \in [n_2]\}$, and $\{v_j : j \in [n_3]\}$ be the nodes of the three parts. Let the weight of $u_i x_k$ be a_{ik}, the weight of $x_k v_j$ be b_{kj}, and the weight of $u_i v_j$ be $-c_{ij}$.

As a preprocessing step, we sort the multiset $\{a_{ik} + b_{kj} : k \in [n_2]\}$ for each $i \in [n_1]$ and $j \in [n_3]$, in $\widetilde{O}(n_1 n_2 n_3)$ total time. Let $W_{ij} = \{k \in [n_3] : a_{ik} + b_{kj} = c_{ij}\}$. Note that we can generate the elements in W_{ij} by searching in the sorted lists in $\widetilde{O}(1)$ time per element.

- **Few-witnesses case.** For each i, j with $|W_{ij}| \le n_2/s$, add the triangle $u_i x_k v_j$ to R for every $k \in W_{ij}$. The number of triangles added to R is $O(n_1 n_3 \cdot n_2/s)$. The running time of this step is also $\widetilde{O}(n_1 n_3 \cdot n_2/s)$.

- **Many-witnesses case.** Find a set $H \subseteq [n_2]$ of $O(s \log(n_1 n_2 n_3))$ nodes that hit all W_{ij} with $|W_{ij}| > n_2/s$. Here, we can use the standard greedy algorithm for hitting sets, which is deterministic and takes time linear in the total size of the sets W_{ij}; as we can reduce each set's size to n_2/s before running the hitting set algorithm, this takes $\widetilde{O}(n_1 n_2 n_3/s)$ time. For each i, j with $|W_{ij}| > n_2/s$, let $k_0[i, j]$ be some $k_0 \in W_{ij} \cap H$.

  For each $k_0 \in H$ and $k \in [n_2]$, let $L_{k_0 k}$ be the multiset $\{b_{k_0 j} - b_{kj} : j \in [n_3]\}$, and let $F_{k_0 k}$ be the elements that have frequency more than n_3/r in $L_{k_0 k}$. Note that $|F_{k_0 k}| \le r$.

– **Low-frequency case.** For each $k_0 \in H$ and $i \in [n_1]$ and $k \in [n_2]$, if $a_{ik} - a_{ik_0} \notin F_{k_0k}$, we examine each of the at most n_3/r indices j with $a_{ik} - a_{ik_0} = b_{k_0j} - b_{kj}$, and add $u_ix_kv_j$ to R if it is a zero-weight triangle and $|W_{ij}| > n_2/s$. The number of triangles added to R is $\widetilde{O}(sn_1n_2 \cdot n_3/r) = \widetilde{O}(n_1n_2n_3/s)$, by choosing $r := s^2$. The running time of this step is also bounded by $\widetilde{O}(n_1n_2n_3/s)$.

– **High-frequency case.** For each $k_0 \in H$ and $p \in [r]$, create a subgraph $G^{(k_0,p)}$ of G:

  * For each $i \in [n_1]$ and $j \in [n_3]$, keep edge u_iv_j iff $k_0[i,j] = k_0$ (in particular, $a_{ik_0} + b_{k_0j} = c_{ij}$).
  * For each $i \in [n_1]$ and $k \in [n_2]$, keep edge u_ix_k iff $a_{ik} - a_{ik_0}$ is the p-th element of F_{k_0k}.
  * For each $j \in [n_3]$ and $k \in [n_2]$, keep edge x_kv_j iff $b_{k_0j} - b_{kj}$ is the p-th element of F_{k_0k}.

  Note that if $u_ix_kv_j$ is a triangle in $G^{(k_0,p)}$, then $a_{ik} - a_{ik_0} = b_{k_0j} - b_{kj}$ and $a_{ik_0} + b_{k_0j} = c_{ij}$, and by Fredman's trick, $a_{ik} + b_{kj} = a_{ik_0} + b_{k_0j} = c_{ij}$, implying that $u_ix_kv_j$ is a zero-weight triangle. The running time of this step is bounded by $\widetilde{O}(sn_1n_2 + sn_2n_3 + rn_1n_3)$.

The number of subgraphs created is $\widetilde{O}(sr) = \widetilde{O}(s^3)$.

Correctness. Consider a zero-weight triangle $u_ix_kv_j$ in G. If $|W_{ij}| \le n_2/s$, the triangle is in R due to the "few-witnesses" case. So assume $|W_{ij}| > n_2/s$. Let $k_0 = k_0[i,j]$. We know that $a_{ik} + b_{kj} = c_{ij} = a_{ik_0} + b_{k_0j}$ and by Fredman's trick, $a_{ik} - a_{ik_0} = b_{k_0j} - b_{kj}$. If $a_{ik} - a_{ik_0} \notin F_{k_0k}$, then the triangle is in R due to the "low-frequency" case. Otherwise, it is a triangle in $G^{(k_0,p)}$ for some $p \in [r]$ due to the "high-frequency" case.

Update edge weights. If the edge weights between U and V change, we only need to rerun the parts of our algorithms that use the weights c. In particular, we need to rerun the low-frequency case, which has running time $\widetilde{O}(n_1n_2n_3/s)$, and the first subcase of the high-frequency case, which has running time $\widetilde{O}(rn_1n_3) = \widetilde{O}(s^2n_1n_3)$. Overall, the running time for updating the edge weights between U and V is $\widetilde{O}(n_1n_2n_3/s + s^2n_1n_3)$. □

5.1 Application 1: Exact Triangle in Preprocessed Universes

As one simple application of the decomposition theorem, we can solve AE-Exact-Tri in truly subcubic time in a "preprocessed universe" setting, where the input is a subgraph of a preprocessed graph. It is convenient to define a variant of the problem, AE-Exact-Tri', which is AE-Exact-Tri where the input graph is tripartite with three parts U, X, and V, but we only need to report if each edge between U and V is in an exact triangle. If $|U| = |V| = |X|$, AE-Exact-Tri' is equivalent to AE-Exact-Tri.

Corollary 5.2. *Given a real-weighted tripartite graph G with n_1, n_2, and n_3 nodes in its three parts U, X, and V, and given s, we can preprocess G in $\widetilde{O}(n_1n_2n_3 + sn_1n_2 + sn_2n_3 + s^2n_1n_3)$ time, so that for any given subgraph G' of G, we can solve* AE-Exact-Tri' *on G' in $\widetilde{O}(n_1n_2n_3/s + sM(n_1, s^2n_2, n_3))$ time.*

For example, for $n_1 = n_2 = n_3 = n$, after preprocessing in $\widetilde{O}(n^3)$ time, we can solve AE-Exact-Tri *on G' in time $O(n^{2.83})$, or if $\omega = 2$, $\widetilde{O}(n^{11/4})$.*

Proof. During preprocessing, we apply Theorem 5.1 to compute the subgraphs $G^{(\lambda)}$ and R in $\widetilde{O}(n_1n_2n_3/s + sn_1n_2 + sn_2n_3 + s^2n_1n_3)$ time.

During a query for a given subgraph G' and a target value t, if t has changed, we first subtract t from all the edge weights between U and V to transform the problem to detecting zero-weight triangles. We can update the $G^{(\lambda)}$'s and R in $\widetilde{O}(n_1n_2n_3/s + s^2n_1n_3)$ time.

Next, for each λ, we check whether after removing edges not present in G', the subgraph $G^{(\lambda)}$ has a triangle (which would automatically be a zero-weight triangle) through each edge in $U \times V$. Since triangle finding (without weights) reduces to matrix multiplication, the running time is $\widetilde{O}(s^3M(n_1, n_2, n_3))$.

We can do slightly better using the grouping of the subgraphs: for each category Λ, define a tripartite graph $G^{(\Lambda)}$ with parts U, $X \times \Lambda$, and V, and for each $\lambda \in \Lambda$, include an edge between u and (x, λ) if $ux \in G^{(\lambda)} \cap G'$, and between (x, λ) and v if $xv \in G^{(\lambda)} \cap G'$. For each category Λ, we check whether the subgraph $G^{(\Lambda)}$ has a triangle through each edge. The running time becomes $\widetilde{O}(sM(n_1, s^2 n_2, n_3))$. Also, the term $O(s^2 n_1 n_3)$ for the update cost can be lowered to $O(n_1 n_3)$ now when working with the $G^{(\Lambda)}$'s instead of the $G^{(\lambda)}$'s, as can be checked from our construction (since each edge $u_i v_j$ occurs in one category).

For $n_1 = n_2 = n_3 = n$, we choose $s = n^{0.17+\varepsilon}$ (using the fact that $\omega(1, 1.34, 1) < 2.657$ [LU18]), or if $\omega = 2$, $s = n^{1/4}$. □

5.2 Application 2: 3SUM in Preprocessed Universes

The above implies also a new algorithm for 3SUM with a preprocessed universe, previously studied by Chan and Lewenstein [CL15], who obtained $\widetilde{O}(n^{13/7})$ query time, after preprocessing in $\widetilde{O}(n^2)$ randomized time or $\widetilde{O}(n^\omega)$ deterministic time. Our query time is strictly better if ω is 2 (or is sufficiently close to 2), and we also obtain *deterministic* $\widetilde{O}(n^2)$ preprocessing time regardless of ω.

Corollary 5.3. *We can preprocess sets A, B, C of n integers in $[n^{O(1)}]$ in $\widetilde{O}(n^2)$ deterministic time, so that given any subsets $A' \subseteq A$, $B' \subseteq B$, and $C' \subseteq C$, we can solve* All-Nums-3SUM *on (A', B', C') in time $O(n^{1.891})$, or if $\omega = 2$, $\widetilde{O}(n^{11/6})$.*

Proof. We use a known reduction from (All-Nums-) 3SUM to (All-Nums-) 3SUM-Convolution (one of the reductions by Chan and He [CH20] is deterministic and increases running time only by polylogarithmic factors when the input numbers are polynomially bounded), in combination with a known reduction from All-Nums-3SUM-Convolution to AE-Exact-Tri [VW09]. The problem is reduced to $\widetilde{O}(1)$ instances of the problem in Corollary 5.2 with $n_1 = n/q$, $n_2 = q$ and $n_3 = n$ for a parameter q (the original reduction [VW09] has $q = \sqrt{n}$, but we will do better with a different choice of q). It is straightforward to check that these reductions carry over to the preprocessed universe setting. We then obtain preprocessing time $\widetilde{O}(n^2 + snq + s^2 n^2/q) = \widetilde{O}(n^2 + s^3 n^{1.5})$ and query time $\widetilde{O}(n^2/s + sM(n/q, s^2 q, n)) = \widetilde{O}(n^2/s + sM(s\sqrt{n}, s\sqrt{n}, n))$ by setting $q = \sqrt{n}/s$. We choose $s = n^{0.109+\varepsilon}$ (using the fact that $\omega(0.609, 0.609, 1) < 1.781$ [LU18]), or if $\omega = 2$, $s = n^{1/6}$. □

Remark 5.4. Corollary 5.2 and Corollary 5.3 can also be used to solve #AE-Exact-Tri and #All-Nums-3SUM in the preprocessed universe. This is because Theorem 5.1 provides a decomposition, so we can sum up the counts in all the cases (In Corollary 5.3, we also have to be careful when applying the reduction in [CH20], to make sure we do not over count, by using inclusion-exclusion). Prior method for 3SUM in the preprocessed universe [CL15] cannot compute counts, because it relies on the BSG theorem, which only provides a covering.

In Section 10.2, we will describe a still better solution to 3SUM in preprocessed universes, with randomization, using FFT instead of fast matrix multiplication.

5.3 Application 3: A Deterministic 3SUM Algorithm for Bounded Monotone Sets

Another interesting application is the following:

Corollary 5.5. *Given monotone sets $A, B, C \subseteq [n]^d$ for any constant $d \geq 2$, we can solve* All-Nums-3SUM *on A, B, C in $O(n^{2-1/O(d)})$ deterministic time.*

Proof. Chan and Lewenstein [CL15] observed that 3SUM for bounded monotone sets in any constant dimension d reduces to 3SUM for "clustered" sets, which in turn reduces to 3SUM with preprocessed

universes on $O(n/\ell)$ elements and $O(\ell^{3d})$ queries. Using Corollary 5.3 gives total deterministic time $\widetilde{O}((n/\ell)^2 + \ell^{3d}(n/\ell)^{1.891})$, which is $\widetilde{O}(n^{2-0.218/(3d+0.109)})$ by setting $\ell = n^{0.109/(3d+0.109)}$. (The exponent here is certainly improvable, by solving the problem using our techniques more directly, instead of applying a black-box reduction to 3SUM with preprocessed universes.) □

The above result provides the first truly subquadratic *deterministic* algorithm for bounded monotone 3SUM in arbitrary constant dimensions—Chan and Lewenstein [CL15] gave subquadratic randomized algorithms with $O(n^{2-1/(d+O(1))})$ running time, but they had nontrivial deterministic algorithms only for $d \leq 7$ under the current matrix multiplication bounds.

We can also apply the triangle decomposition theorem to obtain subquadratic algorithms for monotone or bounded-difference Min-Plus convolution (which were first obtained by Chan and Lewenstein [CL15], and followed by [CDXZ22]), and subcubic algorithms for monotone or bounded-difference Min-Plus products (which were first obtained by Bringmann et al. [BGSV16], and followed by [VX20b, GPVX21, CDX22, CDXZ22]). Since previous algorithms have been found for these problems, we will omit the details here and refer to Appendix B.

The main message is that many of the results in Chan and Lewenstein's paper can be obtained alternatively using our decomposition theorem, which is simpler and more elementary than the BSG Theorem, if we are interested in subquadratic algorithms but don't care about the precise values in the exponents. The advantage is simplicity—additive combinatorics is not needed after all! (However, the BSG Theorem is still potentially useful in optimizing those exponents.)

5.4 Application 4: A Truly Subquartic #APSP Algorithm

As another simple, interesting application of the triangle decomposition theorem, we obtain the first truly subquartic algorithm for #APSP for arbitrary weighted graphs:

Theorem 5.6. #APSP *for n-node graphs with positive edge weights has an algorithm running in $O(n^{3.83})$ time, or if $\omega = 2$, $\widetilde{O}(n^{15/4})$ time.*

Proof. We define the following "funny" matrix product $\otimes$: if $(C, C') = (A, A') \otimes (B, B')$, then $C = A \star B$ and $C'_{ij} = \sum_{k \in [n]: C_{ij} = A_{ik} + B_{kj}} A'_{ik} B'_{kj}$.

Claim 5.7. *Let (A, A') and (B, B') be two pairs of $n \times n$ matrices where the entries of A' and B' are (large) ℓ-bit integers. Then we can compute $(A, A') \otimes (B, B')$ in time $\widetilde{O}(n^3 + \ell n^{2.83})$, or if $\omega = 2$, $\widetilde{O}(n^3 + \ell n^{11/4})$.*

Proof. First compute $C = A \star B$ naively in $O(n^3)$ time. Initialize the entries of C' to 0. Consider the tripartite graph with nodes $\{u_i : i \in [n]\}$, $\{x_k : k \in [n]\}$, and $\{v_j : j \in [n]\}$, where $u_i x_k$ has weight A_{ik}, and $x_k v_j$ has weight B_{kj}, and $u_i v_j$ has weight $-C_{ij}$. Apply Theorem 5.1 to obtain subgraphs $G^{(\lambda)}$ and a set R in $\widetilde{O}(n^3 + s^2 n^2)$ time.

We first examine each triangle $u_i x_k v_j \in R$ and add $A'_{ik} B'_{kj}$ to C'_{ij}. This takes $\widetilde{O}(\ell n^3/s)$ time.

Next, for each λ, we compute $\sum_k [u_i x_k \in G^{(\lambda)}] A'_{ik} \cdot [x_k v_j \in G^{(\lambda)}] B'_{kj}$ and add it to C'_{ij} for every $u_i v_j \in G^{(\lambda)}$. This reduces to a standard matrix product on ℓ-bit integers and takes $\widetilde{O}(\ell n^\omega)$ time for each λ. The total time is $\widetilde{O}(\ell \cdot s^3 n^\omega)$. As before, we can do slightly better using the grouping of the subgraphs, which improve the running time to $\widetilde{O}(\ell \cdot sM(n, s^2 n, n))$.

As in Corollary 5.2, we choose $s = n^{0.17+\varepsilon}$, or if $\omega = 2$, $s = n^{1/4}$. □

Given an input graph G with positive edge weights, let $D^{(=2^i)}$ be the distance matrix for paths of (unweighted) length exactly 2^i, and $D'^{(=2^i)}$ be the number of paths of (unweighted) length exactly 2^i that

match the distance in $D^{(=2^i)}$. Similarly, we define $D^{(<2^i)}$ as the distance matrix for paths of (unweighted) length less 2^i, and define $D'^{(<2^i)}$ similarly.

For $i = 0$, it is easy to see that $D^{(=1)}$ is exactly the weight matrix of G, and $D'^{(=1)}$ is the adjacency matrix of G. Also, $D^{(<1)}$ is the matrix whose diagonal entries are all 0, and other entries are all ∞. Finally $D'^{(<1)}$ equals the $n \times n$ identity matrix.

For $i > 0$, we can use the following recurrences:

$$(D^{(=2^i)}, D'^{(=2^i)}) = (D^{(=2^{i-1})}, D'^{(=2^{i-1})}) \otimes (D^{(=2^{i-1})}, D'^{(=2^{i-1})}),$$
$$(D^{(<2^i)}, D'^{(<2^i)}) = (D^{(<2^{i-1})}, D'^{(<2^{i-1})}) \otimes (D^{(=2^{i-1})}, D'^{(=2^{i-1})}).$$

It is not difficult, though a bit tedious, to verify the correctness of these recurrences.

The matrix $D'^{(<2^i)}$ gives the result for #APSP when $2^i > n$. Therefore, #APSP reduces to $O(\log n)$ instances of the funny product $\otimes$, when the matrices A' and B' are $\widetilde{O}(n)$-bit numbers. Then applying Claim 5.7 with $\ell = \widetilde{O}(n)$ yields the theorem. □

In Section 10, we will return to the BSG Theorem and describe more variants and applications.

6 More Lower Bounds under the Strong APSP Hypothesis

Continuing the approach in Section 3, we now derive conditional lower bounds for more problems with intermediate complexity from the Strong APSP Hypothesis (which as noted before is equivalent to the hypothesis that $M^*(n, n, n \mid n^{3-\omega})$ is not truly subcubic). We begin with a useful lemma:

Lemma 6.1. *For any positive constants β, γ, c with $0 < \gamma < \beta$, if $M^*(n, n^\beta, n \mid n^{c\beta}) = O(n^{2+\beta-\varepsilon})$, then $M^*(n, n^\gamma, n \mid n^{c\gamma}) = O(n^{2+\gamma-\Omega(\varepsilon)})$.*

Proof.

$$\begin{aligned} M^*(n, n^\gamma, n \mid n^{c\gamma}) &\leq O(n^{2(1-\gamma/\beta)} M^*(n^{\gamma/\beta}, n^\gamma, n^{\gamma/\beta} \mid n^{c\gamma})) \\ &\leq \widetilde{O}(n^{2(1-\gamma/\beta)} \cdot (n^{\gamma/\beta})^{2+\beta-\varepsilon}) = O(n^{2+\gamma-\Omega(\varepsilon)}). \end{aligned}$$

□

Lemma 6.1 allows us to remove the assumption $\beta \leq (3-\omega)/2$ in Corollary 3.3 (since we can replace β with any sufficiently small positive constant γ). Consequently, Corollary 3.5 holds for all $\beta \leq 1/3$, regardless of the value of ω. For convenience, we repeat the statement of Corollary 3.3 below, with the assumption removed:

Corollary 6.2. *For any constant $\beta > 0$, if $M^*(n, n^\beta, n \mid n^{2\beta}) = O(n^{2+\beta-\varepsilon})$, then $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.*

Remark 6.3. By applying Lemma 6.1 with $\beta = 1$, $\gamma = 1/2$, and $c = 1$, we see that $M^*(n, n, n \mid n) = O(n^{3-\varepsilon})$ implies $M^*(n, \sqrt{n}, n \mid \sqrt{n}) = O(n^{5/2-\Omega(\varepsilon)})$. If $\omega = 2$, Chan, Vassilevska W. and Xu [CVX21] showed that the u-dir-APSP Hypothesis is equivalent to the claim that $M^*(n, \sqrt{n}, n \mid \sqrt{n})$ is not in $O(n^{5/2-\varepsilon})$ for any $\varepsilon > 0$. Consequently, the u-dir-APSP Hypothesis implies the Strong APSP Hypothesis when $\omega = 2$.

Remark 6.4. In the definition of the Strong APSP Hypothesis, it does not matter whether the input graph is undirected or directed—the directed version is also equivalent to the statement that $M^*(n, n, n \mid n^{3-\omega})$ is not truly subcubic: Directed APSP for integer weights in $[n^{3-\omega}]$ can be solved by Zwick's algorithm [Zwi02,

CVX21] in time $\widetilde{O}(\max_\ell M^*(n, n/\ell, n \mid \ell n^{3-\omega}))$. If $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\varepsilon})$, then for $\ell \le n^\delta$, we have $M^*(n, n/\ell, n \mid \ell n^{3-\omega}) \le \widetilde{O}(M^*(n, n, n \mid n^{3-\omega+\delta}) \le O(n^{(3-\varepsilon)(3-\omega+\delta)/(3-\omega)}) = O(n^{3-\Omega(\varepsilon)})$ for a sufficiently small δ. On the other hand, for $\ell > n^\delta$, we trivially have $M^*(n, n/\ell, n \mid \ell n^{3-\omega}) \le O(n^3/\ell) = O(n^{3-\delta})$.

6.1 Min-Witness Product

Let MIN-WITNESS-PROD(n) be the time complexity of computing min-witness product for two $n \times n$ Boolean matrices. More generally, let MIN-WITNESS-PROD(n_1, n_2, n_3) be the time complexity of Min-Witness-Prod for an $n_1 \times n_2$ Boolean matrix and an $n_2 \times n_3$ Boolean matrix. We observe that Min-Plus product for rectangular matrices of sufficiently small inner dimensions and integer ranges can be reduced to Min-Witness-Prod:

Lemma 6.5. *For any x, y, we have $M^*(n, x, n \mid y) = O(\text{MIN-WITNESS-PROD}(n, xy^2, n))$.*

Proof. Suppose that we are given an $n \times x$ matrix A and an $x \times n$ matrix B where all matrix entries are in $[y]$. For each $i \in [n]$, $k \in [x]$, and $u, v \in [y]$, let $A'_{i,(k,u,v)} = 1$ if $A_{ik} = u$, and $A'_{i,(k,u,v)} = 0$ otherwise. For each $j \in [n]$, $k \in [x]$, $u, v \in [y]$, let $B'_{(k,u,v),j} = 1$ if $B_{kj} = v$, and $B'_{(k,u,v),j} = 0$ otherwise. The number of triples $\tau = (k, u, v)$ is $xy^2 \le n$. By sorting the triples in increasing order of $u + v$, the Min-Plus product of A and B can be computed from the Min-Witness product of $(A'_{i,\tau})_{i\in[n],\tau\in[x]\times[y]^2}$ and $(B'_{\tau,j})_{\tau\in[x]\times[y]^2,j\in[n]}$. □

Combining Corollary 6.2 and Lemma 6.5 immediately gives the following:

Corollary 6.6. *If* MIN-WITNESS-PROD$(n, n^{5\beta}, n) = O(n^{2+\beta-\varepsilon})$, *then* $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

By setting $\beta = 1/5$, we have thus proved that Min-Witness-Prod of two $n \times n$ Boolean matrices cannot be computed in $O(n^{11/5-\varepsilon})$ time under the Strong APSP Hypothesis. In particular, if $\omega = 2$, this implies that Min-Witness-Prod is strictly harder than Boolean matrix multiplication.

Furthermore, by setting $\beta = \gamma/5$, Min-Witness-Prod of an $n \times n^\gamma$ and an $n^\gamma \times n$ Boolean matrix cannot be computed in $O(n^{2+\gamma/5-\varepsilon})$ time for any γ under the same hypothesis. This implies that for any $\gamma \le 0.3138$, Min-Witness-Prod is strictly harder than Boolean matrix multiplication, as $\omega(1, 0.3138, 1) = 2$ [LU18]. This result interestingly rules out the possibility that the polynomial method [Wil18, AVY15, CW21] could be used to transform the Min-Witness product of an $n \times d$ and a $d \times n$ matrix into a standard product of an $n \times d^{O(1)}$ and a $d^{O(1)} \times n$ matrix. It also contrasts Min-Witness-Prod with, for example, Dominance-Product or Equality-Product, which does have near-quadratic time complexity when the inner dimension d is smaller than $n^{0.1569}$ (since as mentioned in Section 3.1, Matoušek's technique [Mat91, Yus09] yields a time bound of $\widetilde{O}(\min_r(dn^2/r + M(n, dr, n)) \le \widetilde{O}(n^2 + M(n, d^2, n)))$.

6.2 All-Pairs Shortest Lightest Paths

Let UNDIR-APSLP$_{1,2}(n, m)$ be the time complexity of undir-APSLP$_{1,2}$ on graphs with n nodes and m edges. We observe the following:

Lemma 6.7. *For any x, y, we have $M^*(n, x, n \mid y) = O(\text{UNDIR-APSLP}_{1,2}(n, nx))$ if $xy^2 \le n$.*

Proof. Suppose that we are given an $n \times x$ matrix A and an $x \times n$ matrix B where all matrix entries are in $[y]$. We construct an undirected graph as follows:

- For each $i \in [n]$, create a node $s[i]$.
- For each $k \in [x]$ and $u \in [y]$, create a node $w_1[k, u]$.

- For each $k \in [x]$, create a node $w_2[k]$.
- For each $k \in [x]$ and $v \in [y]$, create a node $w_3[k, v]$.
- For each $j \in [n]$, create a node $t[j]$.
- For each $i \in [n]$ and $k \in [x]$, create an edge $s[i]w_1[k, u]$ of weight 1 where $u = a_{ik}$.
- For each $k \in [x]$ and $u \in [y]$, create a path between $w_1[k, u]$ and $w_2[k]$ that has u edges of weight 2 and $y - u$ edges of weight 1 (so that the path has y edges and weight $y + u$).
- For each $k \in [x]$ and $v \in [y]$, create a path between $w_2[k]$ and $w_3[k, v]$ that has v edges of weight 2 and $y - v$ edges of weight 1 (so that the path has y edges and weight $y + v$).
- For each $j \in [n]$ and $k \in [x]$, create an edge $w_3[k, v]t[j]$ of weight 1 where $v = b_{kj}$.

The number of nodes in the graph is $O(n+xy^2) = O(n)$, and the number of edges is $O(nx+xy^2) = O(nx)$. For each $i, j \in [n]$, all lightest paths from $s[i]$ to $t[j]$ have $2y + 2$ edges, and among them, the shortest has weight $2y + 2 + \min_k(a_{ik} + b_{kj})$. □

Combining Corollary 6.2 and Lemma 6.7 immediately gives the following:

Corollary 6.8. *For any constant $\beta \leq 1/5$, if* UNDIR-APSLP$_{1,2}(n, n^{1+\beta}) = O(n^{2+\beta-\varepsilon})$, *then* $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

By setting $\beta = 1/5$, we have thus proved that undir-APSLP$_{1,2}$ cannot be solved in $O(n^{11/5-\varepsilon})$ time under the Strong APSP Hypothesis. If $\omega = 2$, this implies that undir-APSLP$_{1,2}$ is strictly harder than undirected APSP for such weighted graphs. (The current best algorithm for undir-APSLP$_{1,2}$ has running time $\widetilde{O}(n^{2.58})$, or if $\omega = 2$, $\widetilde{O}(n^{5/2})$ [CVX21].) The same result holds for undir-APLSP$_{1,2}$.

Furthermore, undir-APSLP$_{1,2}$ with $n^{1+\beta}$ edges cannot be solved in $O(n^{2+\beta-\varepsilon})$ time for any $\beta \leq 1/5$ under the same hypothesis. Thus, the naive algorithm is essentially *optimal* for sufficiently sparse graphs.

The same results hold for the similar problem of undir-APLSP$_{1,2}$ (in the proof of Lemma 6.7, we just modify the path from $w_1[k, u]$ and $w_2[k]$ to use $y - u$ edges of weight 2 and $2u$ edges of weight 1, so that the path has $y + u$ edges and weight $2y$).

6.3 Batched Range Mode

For a different application about data structures, we next consider the range mode problem, which has received considerable attention and has been extensively studied in the literature [KMS05, CDL+14, VX20b, GH22, JX22]. Let BATCHED-RANGE-MODE$(N, Q \mid \sigma)$ be the total time complexity of Batch-Range-Mode for Q range mode queries in an array of N elements in $[\sigma]$.

Lemma 6.9. *For any x, y, we have* $M^*(n, x, n \mid y) = O(\text{BATCHED-RANGE-MODE}(nxy, n^2 \mid x))$.

Proof. Suppose that we are given an $n \times x$ matrix A and an $x \times n$ matrix B where all matrix entries are in $[y]$. We create an array holding a string S over the alphabet $[x]$, defined as follows:

- Let $\sigma_i = 1^{A_{i1}}2^{A_{i2}} \cdots x^{A_{ix}}$ and $\sigma'_i = 1^{y-A_{i1}}2^{y-A_{i2}} \cdots x^{y-A_{ix}}$, which have length $O(xy)$.
- Let $\tau_j = 1^{B_{1j}}2^{B_{2j}} \cdots x^{B_{xj}}$ and $\tau'_i = 1^{y-B_{1j}}2^{y-B_{2j}} \cdots x^{y-B_{xj}}$, which have length $O(xy)$.
- Let $S = \sigma_n\sigma'_n \cdots \sigma_2\sigma'_2\sigma_1\sigma'_1\tau'_1\tau_1\tau'_2\tau_2 \cdots \tau'_n\tau_n$, which has length $O(nxy)$.

For each $i, j \in [n]$, consider the substring $S_{ij} = \sigma'_i\sigma_{i-1}\sigma'_{i-1}\cdots\sigma_1\sigma'_1\tau'_1\tau_1\cdots\tau'_{j-1}\tau_{j-1}\tau'_j$. For each $k \in [x]$, the frequency of k in S_{ij} is precisely $iy + jy - A_{ik} - B_{kj}$. Thus, the mode of S_{ij} is an index k minimizing $A_{ik} + B_{kj}$. So, the Min-Plus product can be computed by answering $O(n^2)$ range mode queries on S. □

Combining Corollary 6.2 and Lemma 6.9 immediately gives the following:

Corollary 6.10. *For any constant β, if* BATCHED-RANGE-MODE$(n^{1+3\beta}, n^2 \mid n^\beta) = O(n^{2+\beta-\varepsilon})$, *then* $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

By setting $\beta = 1/3$ and $n = \sqrt{N}$, we have thus proved that Batch-Range-Mode for N queries on N elements cannot be solved in $O(N^{7/6-\varepsilon})$ time under the Strong APSP Hypothesis. Previously, Chan et al. [CDL+14] gave a reduction from Boolean matrix multiplication implying a better, near-$N^{3/2}$ conditional lower bound for *combinatorial* algorithms (under the Combinatorial BMM Hypothesis); this matched upper bounds of known combinatorial algorithms [KMS05, CDL+14]. However, for noncombinatorial algorithms, their lower bound was near $N^{\omega/2}$, which is trivial if $\omega = 2$. The distinction between combinatorial vs. noncombinatorial algorithms is especially important for the range mode problem, as it is actually possible to beat $N^{3/2}$ using fast matrix multiplication, as first shown by Vassilevska W. and Xu [VX20b]. The current fastest algorithm by Gao and He [GH22] runs in $O(N^{1.4797})$ time. Our new lower bound reveals that there is a limit on how much fast matrix multiplication can help.

(For still more recent work on range mode, see [JX22] for a conditional lower bound for the dynamic version of the range mode problem, but again this is only for combinatorial algorithms.)

Furthermore, by using the fact that BATCHED-RANGE-MODE$(n^{1+3\beta}, n^2 \mid n^\beta) \le O(n^{1-3\beta})\cdot$ BATCHED-RANGE-MODE$(n^{1+3\beta}, n^{1+3\beta} \mid n^\beta)$ and setting $N = n^{1+3\beta}$ and $\gamma = \beta/(1+3\beta)$, we see that Batch-Range-Mode for N queries on N elements in a universe of size $\sigma = N^\gamma$ cannot be answered in $O(N^{1+\gamma-\varepsilon})$ time for any $\gamma \le 1/6$, under the same hypothesis. This lower bound is *tight*, since an $O(N\sigma)$ upper bound is known [CDL+14].

Furthermore, by setting $n = \sqrt{Q}$ and $n^\beta = (N/\sqrt{Q})^{1/3}$, Batch-Range-Mode for Q queries on N elements cannot be solved in $O(Q^{5/6}N^{1/3-\varepsilon})$ time for any $Q \le N^2$ under the same hypothesis. For example, for $Q = N^{1.6}$, the lower bound is near $N^{1.666}$ (in other words, we need at least $N^{0.066}$ time per query). In contrast, the previous reduction by Chan et al. [CDL+14] from Boolean matrix multiplication gives a lower bound of $M(\sqrt{Q}, N/\sqrt{Q}, \sqrt{Q})$, which is only near linear in Q when $Q = N^{1.6}$, as $\omega(0.8, 0.2, 0.8) = 1.6$. (Known combinatorial algorithms have running time near $O(\sqrt{Q}N)$ as a function of N and Q [KMS05, CDL+14].)

The same results hold for the similar problem of *range minority* [CDSW15] (finding a least frequent element in a range).

6.4 Dynamic Shortest Paths in Unweighted Planar Graphs

As another example of an application to data structure problems, we now consider the dynamic shortest path problem for unweighted planar graphs. Let PLANAR-DYN-SP(N, Q, U) be the time complexity of performing an offline sequence of Q shortest path distance queries and U edge updates on an unweighted, undirected planar graph with N nodes.

Lemma 6.11. *For any $\alpha, \beta \le 1$,*

$$M^*(n, n^\beta, n \mid X) = O(n^{1-\alpha} \cdot \text{PLANAR-DYN-SP}((n^{2\alpha+\beta} + n^{\alpha+2\beta})X, n^{1+\alpha}, n^{1+\beta})).$$

Proof. Abboud and Dahlgaard [AD16, Proof of Theorem 1] reduced the computation of the Min-Plus product of an $n \times n^\beta$ and an $n^\beta \times n^\alpha$ matrix with entries from $[X]$, to the problem of performing an offline sequence of $O(n^{1+\alpha})$ shortest path queries and $O(n^{1+\beta})$ edge-weight changes on a *weighted* planar graph with $O(n^{\alpha+\beta})$ nodes. In their graph construction, all edges have integer weights bounded by $O(n^\alpha X)$, except for $O(n^\beta)$ edges having integer weights bounded by $O(n^{\alpha+\beta}X)$. (The X factor was stated as X^2 in their paper, but as they remarked at the end of their Section 2, X^2 can be lowered to $X+1$.) The weighted graph can be turned into an unweighted graph, simply by subdividing each edge. More precisely, we create a path π_e of length ℓ for an edge e with weight upper-bounded by ℓ. Whenever the weight of e changes, we can redirect an endpoint of e to an appropriate node in the path π_e, using $O(1)$ updates in this unweighted graph. The resulting unweighted planar graph has $O((n^{2\alpha+\beta}+n^{\alpha+2\beta})X)$ nodes. Hence, Abboud and Dahlgaard's reduction implies that $M^*(n, n^\beta, n^\alpha \mid X) = O(\text{PLANAR-DYN-SP}((n^{2\alpha+\beta}+n^{\alpha+2\beta})X, n^{1+\alpha}, n^{1+\beta}))$.

The lemma then follows, as $M^*(n, n^\beta, n \mid X) = O(n^{1-\alpha} \cdot M^*(n, n^\beta, n^\alpha \mid X))$. □

Combining Corollary 6.2 and Lemma 6.11 (with $\alpha = \beta$) immediately gives the following:

Corollary 6.12. *For any constant β, if* $\text{PLANAR-DYN-SP}(n^{5\beta}, n^{1+\beta}, n^{1+\beta}) = O(n^{1+2\beta-\varepsilon})$, *then* $M^*(n, n, n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

By setting $\beta = 1/4$ and $N = n^{5/4}$, we have thus proved that an offline sequence of N shortest path queries and N updates on an unweighted, undirected N-node planar graph cannot be processed in $O(N^{6/5-\varepsilon})$ time, under the Strong APSP Hypothesis. This rules out the existence of data structures with $N^{o(1)}$ time per operation. Abboud and Dahlgaard [AD16] proved a better lower bound near $N^{3/2}$ in the weighted case under the APSP Hypothesis, or near $N^{4/3}$ in the unweighted case under the OMv Hypothesis [HKNS15], but the latter bound under the OMv Hypothesis holds only for *online* queries and updates, when considering general noncombinatorial algorithms. We have obtained the first conditional lower bounds for the unweighted case that hold in the offline setting.

Gawrychowski and Janczewski [GJ21] have adapted Abboud and Dahlgaard's technique to prove conditional lower bounds for certain dynamic data structure versions of the *longest increasing subsequence (LIS)* problem. In the unweighted case, their reduction was again based on the OMv Hypothesis and applicable only for the online setting. Our approach should similarly yield new conditional lower bounds in the offline setting for their problem.

The preceding applications are not meant to be exhaustive, but the applications to Batch-Range-Mode and dynamic planar shortest paths should suffice to illustrate the potential usefulness of our technique to (unweighted) data structure problems in general. A common way to obtain conditional lower bounds for such data structure problems is via reduction from Boolean matrix multiplication, which is useful only for combinatorial algorithms, or via the OMv Hypothesis, which is only for online settings. Our technique provides a new avenue, allowing us to obtain (weaker, but still nontrivial) lower bounds for general noncombinatorial algorithms in offline or batched settings: namely, it suffices to reduce from rectangular Min-Plus products when the inner dimension and the integer range are both small.

6.5 Min-Witness Equality Product

Lastly, we revisit the Min-Witness-Eq-Prod problem. Let $\text{MIN-WITNESS-EQ-PROD}(n)$ be the time complexity of Min-Witness-Eq-Prod for $n \times n$ matrices. Chan, Vassilevska W., and Xu [CVX21] showed that $\text{U-DIR-APSP}(n) = \widetilde{O}(\text{MIN-WITNESS-EQ-PROD}(n))$, so we immediately obtain a near-$n^{7/3}$ lower bound for Min-Witness-Eq-Prod under the Strong APSP Hypothesis. However, the following corollary gives an alternative lower bound, which is worse if $\omega = 2$, but is better if the current bound of ω turns out to be close to tight. Note that $(2\omega+5)/4$ is strictly larger than ω for all $\omega \in [2, 2.373)$.

Corollary 6.13. *If* MIN-WITNESS-EQ-PROD$(n) = O(n^{(2\omega+5)/4-\varepsilon})$, *then* $M^*(n,n,n \mid n^{3-\omega}) = O(n^{3-\Omega(\varepsilon)})$.

Proof. Let GEN-EQ-PROD$(n_1,n_2,n_3 \mid \ell)$ be the time complexity of the generalized equality product problem in Lemma 3.1. Let EQ-PROD(n_1,n_2,n_3) be the time complexity of ∃Equality-Product, which is a variant of Equality-Product where we only need to determine if each of the outputs of the standard Equality-Product is nonzero or not. It is easy to see that GEN-EQ-PROD$(n_1,n_2,n_3 \mid \ell) \le O(\ell^2 \cdot$ EQ-PROD$(n_1,n_2,n_3))$.

The proof of Theorem 3.2 shows that for any $s \le n_2$ and $t \le \ell$,

$$M^*(n_1,n_2,n_3 \mid \ell) \;=\; \widetilde{O}\big((n_2/s)M^*(n_1,s,n_3 \mid t) \;+\; s \cdot \text{GEN-EQ-PROD}(n_1,n_2,n_3 \mid \ell/t)\big).$$

Consequently, for any $t \le n^{3-\omega}$,

$$M^*(n,n,n \mid n^{3-\omega}) \;=\; \widetilde{O}\big((n/s)M^*(n,s,n \mid t) \;+\; s(n^{3-\omega}/t)^2 \cdot \text{EQ-PROD}(n,n,n)\big).$$

Set $t = n/s$. Chan, Vassilevska W. and Xu [CVX21] gave a reduction showing that $M^*(n,s,n \mid n/s) = O(\text{MIN-WITNESS-EQ-PROD}(n))$. Trivially, EQ-PROD$(n,n,n) = O(\text{MIN-WITNESS-EQ-PROD}(n))$. It follows that $M^*(n,n,n \mid n^{3-\omega}) \;=\; O((n/s + s^3/n^{2\omega-4}) \cdot \text{MIN-WITNESS-EQ-PROD}(n))$. The result follows by setting $s = n^{(2\omega-3)/4}$. □

7 Lower Bounds under the u-dir-APSP Hypothesis

We can similarly apply our key reduction in Theorem 3.2 to obtain (better) lower bounds under the u-dir-APSP Hypothesis, using the following corollary:

Corollary 7.1. *Let ρ be such that $\omega(1,\rho,1) = 1+2\rho$. Fix any constant $\sigma > \rho$, and let $\kappa = \frac{\omega(1,\sigma,1)-1-2\rho}{\sigma-\rho}$. For any constant β, if $M^*(n,n^\beta,n \mid n^{(1+\kappa)\beta}) = O(n^{2+\beta-\varepsilon})$, then* U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$.

Proof. Chan, Vassilevska W. and Xu [CVX21] have shown that U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$ is equivalent to $M^*(n,n^\rho,n \mid n^{1-\rho}) = O(n^{2+\rho-\Omega(\varepsilon)})$.

By Theorem 3.2,

$$M^*(n,n^\rho,n \mid n^{1-\rho}) \;=\; \widetilde{O}\left((n^\rho/s)M^*(n,s,n \mid t) + sn^{2+\rho}/r + (sn^{1-\rho}/t)M(n,rn^\rho,n)\right).$$

Setting $s = n^{\beta-\varepsilon'}$, $t = n^{(1+\kappa)\beta}$, and $r = n^\beta$ with $\varepsilon' = \varepsilon/2$ yields

$$M^*(n,n^\rho,n \mid n^{1-\rho}) \;=\; \widetilde{O}\left(n^{\rho-\beta+\varepsilon'}M^*(n,n^\beta,n \mid n^{(1+\kappa)\beta}) + n^{2+\rho-\varepsilon'} + n^{1-\rho-\kappa\beta+\omega(1,\rho+\beta,1)-\varepsilon'}\right),$$

which is $\widetilde{O}(n^{2+\rho-\Omega(\varepsilon)})$, since $\omega(1,\rho+\beta,1) \le \omega(1,\rho,1) + \frac{\omega(1,\sigma,1)-\omega(1,\rho,1)}{\sigma-\rho}\beta = 1+2\rho+\kappa\beta$, by convexity of $\omega(1,\cdot,1)$.

The above assumes $(1+\kappa)\beta \le 1-\rho$ and $\beta \le \sigma-\rho$, but this assumption may be removed, since Lemma 6.1 allows us to replace β with any sufficiently small positive constant γ. □

We pick $\sigma = 0.85$. By known bounds [LU18], $\omega(1,0.85,1) < 2.258317$. Since $\rho \ge 0.5$, we have $\kappa \le \frac{1.258317-2\rho}{0.85-\rho} < 0.7381$. If $\omega = 2$, then $\kappa = 0$.

7.1 Min-Witness Product

Combining Corollary 7.1 and Lemma 6.5 immediately gives the following:

Corollary 7.2. *Let ρ and κ be as in Corollary 7.1. For any constant β, if* MIN-WITNESS-PROD$(n, n^{(3+2\kappa)\beta}, n) = O(n^{2+\beta-\varepsilon})$, *then* U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$.

By setting $\beta = 1/(3+2\kappa)$ (which is $1/3$ if $\omega = 2$, or < 0.223 regardless), we have thus proved that Min-Witness-Prod of two $n \times n$ Boolean matrices cannot be computed in $O(n^{7/3-\varepsilon})$ time if $\omega = 2$, or $O(n^{2.223})$ time regardless of the value of ω, under the u-dir-APSP Hypothesis. (This is better than the near-$n^{11/5}$ lower bound we obtained from the Strong APSP Hypothesis.) The question of proving lower bounds for Min-Witness-Prod from the u-dir-APSP Hypothesis was left open in the paper by Chan, Vassilevska W. and Xu [CVX21] (they were only able to do so for Min-Witness-Eq-Prod).

Furthermore, by setting $\beta = \gamma/(3+2\kappa)$, Min-Witness-Prod of an $n \times n^\gamma$ and an $n^\gamma \times n$ Boolean matrix cannot be computed in $O(n^{2+0.223\gamma-\varepsilon})$ time for any $\gamma \leq 1$ under the same hypothesis.

7.2 All-Pairs Shortest Lightest Paths

Combining Corollary 7.1 and Lemma 6.7 immediately gives the following:

Corollary 7.3. *Let ρ and κ be as in Corollary 7.1. For any constant $\beta \leq \frac{1}{3+2\kappa}$ and* UNDIR-APSLP$_{1,2}(n, n^{1+\beta}) = O(n^{2+\beta-\varepsilon})$, *then* U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$.

By setting $\beta = 1/(3+2\kappa)$, we have thus proved that undir-APSLP$_{1,2}$ cannot be solved in $O(n^{7/3-\varepsilon})$ time if $\omega = 2$, or $O(n^{2.223})$ time regardless of the exact value of ω, under the u-dir-APSP Hypothesis. (This is better than the near-$n^{11/5}$ lower bound we obtained from the Strong APSP Hypothesis.) The same result holds for undir-APLSP$_{1,2}$. Previously, Chan, Vassilevska W. and Xu [CVX21] proved a still better near-n^ρ lower bound for $\{0,1\}$-weighted undir-APLSP from the same hypothesis, but their proof crucially relied on zero-weight edges and also did not work for undir-APSLP (leaving open the question of finding nontrivial conditional lower bounds for both undir-APLSP$_{1,2}$ and undir-APSLP$_{1,2}$, which we answer here).

7.3 Batched Range Mode

By combining Chan, Vassilevska W. and Xu's observation that U-DIR-APSP$(n) = \widetilde{O}(\max_\ell M^*(n, n/\ell, n \mid \ell))$ [CVX21, Zwi02] with Lemma 6.9, and setting $n = \sqrt{N}$, we see that Batch-Range-Mode for N queries on N elements cannot be solved in $O(N^{5/4-\varepsilon})$ time if $\omega = 2$, or in $O(N^{1+\rho/2-\varepsilon})$ time regardless, under the u-dir-APSP Hypothesis.

To obtain a lower bound for general Q, we combine Corollary 7.1 and Lemma 6.9:

Corollary 7.4. *Let ρ and κ be as in Corollary 7.1. For any constant β, if* BATCHED-RANGE-MODE$(n^{1+(2+\kappa)\beta}, n^2 \mid n^\beta) = O(n^{2+\beta-\varepsilon})$, *then* U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$.

Thus, by setting $n = \sqrt{Q}$ and $n^\beta = (N/\sqrt{Q})^{1/(2+\kappa)}$, Batch-Range-Mode of Q queries on N elements cannot be answered in $O(Q^{3/4}N^{1/2-\varepsilon})$ time if $\omega = 2$, or $O(Q^{1-0.365/2}N^{0.365-\varepsilon})$ time regardless, for any $Q \leq N^2$, under the u-dir-APSP Hypothesis. For example, for $Q = N^{1.6}$, the lower bound is near $N^{1.7}$ if $\omega = 2$, or near $N^{1.673}$ regardless. (This is slightly better than the lower bound we obtained in previous section from the Strong APSP Hypothesis.)

7.4 Dynamic Shortest Paths in Planar Graphs

Combining Corollary 7.1 and Lemma 6.11 with $\alpha = \beta$ immediately gives the following:

Corollary 7.5. *Let ρ and κ be as in Corollary 7.1. For any constant β, if* PLANAR-DYN-SP$(n^{(4+\kappa)\beta}, n^{1+\beta}, n^{1+\beta}) = O(n^{1+2\beta-\varepsilon})$*, then* U-DIR-APSP$(n) = O(n^{2+\rho-\Omega(\varepsilon)})$.

By setting $\beta = 1/(3+\kappa)$ and $N = n^{1+\beta}$, we have thus proved that an offline sequence of N shortest path queries and N updates on an unweighted, undirected N-node planar graph cannot be processed in $O(N^{5/4-\varepsilon})$ time if $\omega = 2$, of $O(N^{1.211})$ time regardless, under the u-dir-APSP Hypothesis. (This is slightly better than the lower bound we obtained in previous section from the Strong APSP Hypothesis.)

7.5 An Equivalence Result

We also obtain an interesting equivalence result:

Corollary 7.6. *Let α be such that $\omega(1,\alpha,1) = 2$. For any constants $\beta, \gamma \in (0,\alpha)$, there exists $\varepsilon > 0$ such that $M^*(n, n^\beta, n \mid n^\beta) = O(n^{2+\beta-\varepsilon})$ if and only if there exists $\varepsilon' > 0$ such that $M^*(n, n^\gamma, n \mid n^\gamma) = O(n^{2+\gamma-\varepsilon'})$.*

Proof. W.l.o.g., assume $\gamma < \beta$. The "only if" direction is shown in Lemma 6.1. For the "if" direction, suppose $M^*(n, n^\gamma, n \mid n^\gamma) = O(n^{2+\gamma-\varepsilon'})$. By Theorem 3.2,

$$M^*(n, n^\beta, n \mid n^\beta) \;=\; \widetilde{O}\left((n^\beta/s)M^*(n, s, n \mid t) + sn^{2+\beta}/r + (sn^\beta/t)M(n, rn^\beta, n)\right).$$

Setting $s = n^{\gamma-\varepsilon}$, $t = n^\gamma$, and $r = n^\gamma$, with $\varepsilon = \varepsilon'/2$ yields $M^*(n, n^\beta, n \mid n^\beta) = O(n^{2+\beta-\Omega(\varepsilon')})$, assuming that $\gamma \le \alpha - \beta$.

This assumption may be removed, since Lemma 6.1 allows us to replace γ with a sufficiently small positive constant γ'. □

Corollary 7.7. *If $\omega = 2$, then for any constant $\beta \in (0,1)$, there exists $\varepsilon > 0$ such that* U-DIR-APSP$(n) = O(n^{2.5-\varepsilon})$ *iff there exists $\varepsilon' > 0$ such that $M^*(n, n^\beta, n \mid n^\beta) = \widetilde{O}(n^{2+\beta-\varepsilon'})$.*

Proof. If $\omega = 2$, then $\rho = 1/2$ and then Chan, Vassilevska W. and Xu's result [CVX21] showed that U-DIR-APSP$(n) = O(n^{2.5-\varepsilon})$ for some $\varepsilon > 0$ is equivalent to $M^*(n, \sqrt{n}, n \mid \sqrt{n}) = O(n^{2.5-\varepsilon'})$ for some $\varepsilon' > 0$. Since $\omega = 2$ implies $\alpha = 1$, we can apply the preceding corollary for any $\beta \in (0,1)$ and $\gamma = 1/2$. □

Let $\Phi(\beta)$ be the claim that $M^*(n, n^\beta, n \mid n^\beta)$ is not in $O(n^{2+\beta-\varepsilon})$ for any $\varepsilon > 0$. If $\omega = 2$, $\Phi(1)$ is just the Strong APSP Hypothesis, but intriguingly, by the above corollary, $\Phi(0.99)$ is equivalent to the u-dir-APSP Hypothesis, which has given us strictly better conditional lower bound results.

8 More Equivalences Between Counting and Detection Problems

Continuing the approach in Section 4 for proving equivalence between #AE-Exact-Tri and AE-Exact-Tri, we now derive more equivalence results between other counting and detection problems.

8.1 Min-Plus Product

In this section, we use A and B to denote the inputs to a Min-Plus-Product or #Min-Plus-Product instance, and we use W_{ij} to denote the set of k where $A_{ik} + B_{kj} = (A \star B)_{ij}$, i.e., the set of witnesses for (i, j).

Lemma 8.1. *Given two $n \times n$ matrices A, B and a subset $S \subseteq [n]$, we can compute a matrix D in $\widetilde{O}(|S| \cdot n^{(3+\omega)/2})$ time such that $D_{ij} = |W_{ij}|$ for pairs of (i, j) where $S \cap W_{ij} \neq \emptyset$.*

Proof. For every $s \in S$, we do the following. Let $A^{(s)}$ be a matrix where $A^{(s)}_{ik} = A_{ik} - A_{is}$ and $B^{(s)}_{kj} = B_{sj} - B_{kj}$. Then we compute the equality product $C^{(s)}$ of $A^{(s)}$ and $B^{(s)}$ in $\widetilde{O}(n^{(3+\omega)/2})$ time for each s using Matoušek's algorithm [Mat91]. Finally, let D_{ij} be $C^{(s)}_{ij}$ where $A_{is} + B_{sj}$ is the smallest over all $s \in S$ (breaking ties arbitrarily). The running time for computing D is clearly $\widetilde{O}(|S| \cdot n^{(3+\omega)/2})$.

Suppose $S \cap W_{ij} \neq \emptyset$ for some (i, j). Then D_{ij} equals $C^{(s)}_{ij}$ where $A_{is} + B_{sj} = (A \star B)_{ij}$. For any k, $A^{(s)}_{ik} = B^{(s)}_{kj}$ if and only if $A_{ik} + B_{kj} = A_{is} + B_{sj} = (A \star B)_{ij}$ by Fredman's trick. Therefore, $D_{ij} = C^{(s)}_{ij} = |W_{ij}|$. □

Theorem 8.2. *If* Min-Plus-Product *for $n \times n$ matrices has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #Min-Plus-Product *for $n \times n$ matrices has an $O(n^{3-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$.*

Proof. Given a #Min-Plus-Product instance on $n \times n$ matrices A, B, we first list up to $n^{0.99}$ elements in W_{ij} for every i, j. By well-known techniques (e.g. [VW18]), an $O(n^{3-\varepsilon})$ time Min-Plus-Product algorithm implies an $O(n^{3-\varepsilon''})$ for $\varepsilon'' > 0$ time algorithm for listing up to $n^{0.99}$ witnesses for each (i, j) in a Min-Plus-Product instance.

If we list less than $n^{0.99}$ elements for some (i, j), then these elements are all the elements in W_{ij}. Thus we can output the number of elements we list as the exact witness count for (i, j). For each of the remaining pairs of (i, j), we have found $n^{0.99}$ witnesses. By the standard greedy algorithm for hitting set, in $\widetilde{O}(n^{2.99})$ time, we can find a set S of size $\widetilde{O}(n^{0.01})$ that intersects with W_{ij} for each of these remaining (i, j) pairs. Therefore, we can apply Lemma 8.1 to compute the number of witnesses for these remaining (i, j) pairs in $\widetilde{O}(|S| \cdot n^{(3+\omega)/2}) \leq O(n^{2.70})$ time.

The total running time for the #Min-Plus-Product instance is thus $\widetilde{O}(n^{3-\varepsilon''} + n^{2.99} + n^{2.70})$, which is truly subcubic. □

We then show the reduction in the other direction. The proof is similar to the reduction from a certain version of Min-Plus product to certain versions of APSP counting in unweighted directed graphs [CVX21].

Theorem 8.3. *If* #Min-Plus-Product *for $n \times n$ matrices has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* Min-Plus-Product *for $n \times n$ matrices has an $O(n^{3-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$.*

Proof. Let A, B be two $n \times n$ matrices of a Min-Plus-Product instance. Let A' be another $n \times n$ matrix where $A'_{ik} = M \cdot A_{ik} + k$ for some large enough integer M (say $M > n$). Similarly, we create an $n \times n$ matrix B' where $B'_{kj} = M \cdot B_{kj}$. This way, for every i, j, there exists exactly one k_{ij} such that $A'_{ik_{ij}} + B'_{k_{ij}j} = (A' \star B')_{ij}$. Furthermore, we clearly also have $A_{i,k_{ij}} + B_{k_{ij},j} = (A \star B)_{ij}$.

Then for each integer $p \in [\lceil \log(n) \rceil]$, we do the following. Let $A'^{(p)}$ be a copy of A', but we duplicate all columns k where the p-th bit in k's binary representation is 1. We similarly create $B'^{(p)}$ which is a copy of B' but we duplicate all rows k where the p-th bit in k's binary representation is 1. Then we run the $O(n^{3-\varepsilon})$

time #Min-Plus-Convolution algorithm for $A'^{(p)}$ and $B'^{(p)}$. Suppose for some i, j, the number of witnesses is 2, then we know that the p-th bit of k_{ij} is 1; otherwise, the p-th bit of k_{ij} is 0.

After all $[\lceil \log(n) \rceil]$ rounds, we can compute k_{ij} for every i, j. Since $A_{i,k_{ij}} + B_{k_{ij},j} = (A \star B)_{ij}$, we can then compute the Min-Plus product between A and B in $\widetilde{O}(n^2)$ time. □

8.2 3SUM Convolution and Min-Plus Convolution

Let A, B and C be the inputs of an All-Nums-3SUM-Convolution or #All-Nums-3SUM-Convolution instance, and let W_k be the set of i where $A_i + B_{k-i} = C_k$, i.e., the set of witnesses for k.

Lemma 8.4. *Given a* #All-Nums-3SUM-Convolution *instance for length n arrays A, B, C, we can compute $|W_k|$ for every k such that $|W_k| \geq L$ in $\widetilde{O}(n^{(9+\omega)/4}/L)$ randomized time.*

Proof. First, we find an arbitrary prime p between $2n$ and $4n$, which can be done in $\widetilde{O}(n)$ time. Also, let x be a uniformly random number sampled from $\mathbb{F}_p \setminus \{0\}$ and let y be a uniformly random number sampled from $\mathbb{F}_p$. Then we create three arrays A', B' and C', indexed by $\mathbb{F}_p$ as follows:

$$A'_i = \begin{cases} A_{x^{-1}(i-y) \bmod p} & : x^{-1}(i-y) \bmod p \in [n] \\ M & : \text{otherwise} \end{cases},$$

$$B'_i = \begin{cases} B_{x^{-1}(i+y) \bmod p} & : x^{-1}(i+y) \bmod p \in [n] \\ M & : \text{otherwise} \end{cases},$$

$$C'_i = \begin{cases} C_{x^{-1}i \bmod p} & : x^{-1}i \bmod p \in [n] \\ 3M & : \text{otherwise} \end{cases},$$

where M is a large enough number (say M is larger than 10 times the largest absolute value of the input numbers).

If we use W'_k to denote the set of i such that $A'_i + B'_{(k-i) \bmod p} = C'_k$, then it is not difficult to verify that $|W'_{xk \bmod p}| = |W_k|$. Thus, from now on, we aim to compute $|W'_k|$ for indices k where $|W'_k| \geq L$.

We start with the following claim.

Claim 8.5. *Let $\mathcal{I} \subseteq \mathbb{F}_p$ be any fixed interval of length $\Theta(\sqrt{n})$ and let $1 \leq L' \leq \sqrt{n}$ be a fixed value. Then there exists an $\widetilde{O}(n^{(5+\omega)/4}/L')$ time algorithm that computes $|W'_k \cap \mathcal{I}|$ for every k such that $|W'_k \cap \mathcal{I}| \geq L'$, with high probability. Furthermore, for other values of k, we either also compute $|W'_k \cap \mathcal{I}|$ correctly, or declare that we don't know the value of $|W'_k \cap \mathcal{I}|$.*

Proof. We first reduce the problem of computing $|W'_k \cap \mathcal{I}|$ to an instance of #AE-Exact-Tri. Similar reductions from convolution problems to matrix-product type problems were known before [BCD+14, VW13]. Without loss of generality, we assume $\mathcal{I} = \{0, 1, \ldots, \ell-2, \ell-1\}$ for some $\ell = \Theta(\sqrt{n})$, by subtracting $\min \mathcal{I}$ from all indices of A' and adding $\min \mathcal{I}$ to all indices of B'.

We then create the following tripartite weighted graph G with three parts I, J, T, where $|I| = \ell, |J| = [\lceil p/\ell \rceil]$ and $|T| = 2\ell - 1$. We use I_i to denote the i-th node in I, J_j to denote the j-th node in J and T_t to denote the t-th node in T. We then add the following edges to the graph:

- For every $i \in [|I|], t \in [|T|]$ such that $i - t + \ell - 1 \in \mathcal{I}$, we add an edge between I_i and T_t with weight $w(I_i, T_t) = A'_{i-t+\ell-1}$.
- For every $t \in [|T|]$ and $j \in [|J|]$, we add an edge between T_t and J_j with weight $w(T_t, J_j) = B'_{((j-1)\ell+t-\ell) \bmod p}$.

- For every $i \in [|I|], j \in [|J|]$ such that $(j-1)\ell + i - 1 < p$, we add an edge between I_i and J_j with weight $w(I_i, J_j) = -C'_{(j-1)\ell+i-1}$.

Consider any $(i, j) \in [\ell] \times [\lceil p/\ell \rceil]$ such that $(j-1)\ell + i - 1 < p$. The nodes T_t such that $i - t + \ell - 1 \in \mathcal{I}$ form triangles with edge (I_i, J_j). The multiset of the weights of these triangles is

$$\begin{aligned}\{w(I_i, T_t) + w(J_j, T_t) + w(I_i, J_j)\}_{t=i}^{i+\ell-1} &= \left\{A'_{i-t+\ell-1} + B'_{((j-1)\ell+t-\ell) \bmod p} - C'_{(j-1)\ell+i-1}\right\}_{t=i}^{i+\ell-1} \\ &= \left\{A'_r + B'_{((j-1)\ell+(i-1)-r) \bmod p} - C'_{(j-1)\ell+i-1}\right\}_{r=0}^{\ell-1}.\end{aligned}$$

Thus, the number of triangles with weight 0 containing edge (I_i, J_j) in G is exactly $|W'_k \cap \mathcal{I}|$ for $k = (j-1)\ell+i-1$. In particular, if $|W'_k \cap \mathcal{I}| \geq L'$, then the number of witnesses for (I_i, J_j) in the #AE-Exact-Tri instance on graph G and target value 0 is also at least L'. Now let S be a random subset of $V(G)$ of size $Cn^{0.5} \log n / L'$ for a sufficiently large constant C. Then with high probability, S intersects with the set of witnesses for every edge (I_i, J_j) for which $k = (j-1)\ell + (i-1)$ has at least L' witnesses in $\mathcal{I}$. Now we can apply Lemma 4.1 on graph G, target value 0 and set S to compute the number of witnesses for these edges (I_i, J_j) in $\widetilde{O}(|S|\sqrt{n}^{(3+\omega)/2}) = \widetilde{O}(n^{(5+\omega)/4}/L')$ time.

If S does not intersect with the witnesses for some edge (I_i, J_j) (which is easy to check in $O(|S|n)$ total time), we declare that we don't know the value of $|W'_k \cap \mathcal{I}|$ for $k = (j-1)\ell + (i-1)$. □

Claim 8.6. *Let $\mathcal{I} \subseteq \mathbb{F}_p$ be any fixed interval and $k \in [n]$ be any fixed index. If $|W_k| \geq L$, then*

$$\Pr_{\substack{x \sim \mathbb{F}_p \setminus \{0\} \\ y \sim \mathbb{F}_p}} \left[\left|W'_{xk \pmod p} \cap \mathcal{I}\right| \leq \frac{L|\mathcal{I}|}{2p}\right] = O(\frac{n}{L|\mathcal{I}|}).$$

Proof. First, note that $W'_{xk \pmod p} = \{xw + y \pmod p : w \in W_k\}$. Let X be the random variable denoting $\left|W'_{xk \pmod p} \cap \mathcal{I}\right|$. First, since for any $w \in W_k$, $xw + y \pmod p$ is uniformly at random, $\Pr[xw + y \pmod p \in \mathcal{I}] = \frac{|\mathcal{I}|}{p}$, and consequently $\mathbb{E}[X] = \frac{|W_k||\mathcal{I}|}{p}$.

For any $w, w' \in W_k$ where $w \neq w'$, the probability that both $xw + y \pmod p$ and $xw' + y \pmod p$ fall in $\mathcal{I}$ can be expressed as

$$\sum_{i_1, i_2 \in \mathcal{I}} \Pr[xw + y \equiv i_1 \pmod p \wedge xw' + y \equiv i_2 \pmod p].$$

If $i_1 = i_2$, then $xw + y \equiv i_1 \pmod p$ and $xw' + y \equiv i_2 \pmod p$ cannot both happen; otherwise, there exists at most one pair $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ for which $xw + y \equiv i_1 \pmod p$ and $xw' + y \equiv i_2 \pmod p$ are both true. Thus, $\Pr[xw + y \pmod p \in \mathcal{I} \wedge xw' + y \pmod p \in \mathcal{I}] \leq \frac{|\mathcal{I}|(|\mathcal{I}|-1)}{p(p-1)} \leq \frac{|\mathcal{I}|^2}{p^2}$. Thus,

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \leq \left(|W_k|(|W_k| - 1)\frac{|\mathcal{I}|^2}{p^2} + \mathbb{E}[X]\right) - \mathbb{E}[X]^2 \leq \mathbb{E}[X].$$

Also,

$$\Pr\left[X \leq \frac{L|\mathcal{I}|}{2p}\right] \leq \Pr\left[X \leq \frac{|W_k||\mathcal{I}|}{2p}\right] \leq \Pr\left[|X - \mathbb{E}[X]| \leq \frac{1}{2}\mathbb{E}[X]\right].$$

By Chebyshev's inequality, this probability can be upper bounded by $\frac{\mathrm{Var}[X]}{(\frac{1}{2}\mathbb{E}[X])^2} = O(\frac{n}{L|\mathcal{I}|})$. □

We now describe our algorithm for computing $|W_k|$. First, we split $\mathbb{F}_p$ into $\ell = \Theta(\sqrt{n})$ intervals $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\ell$, each of size $\Theta(\sqrt{n})$. Then it suffices to compute $|W'_{xk \bmod p} \cap \mathcal{I}_i|$ for each $i \in [\ell]$, since $|W_k| = |W'_{xk \bmod p}| = \sum_{i=1}^{\ell} |W'_{xk \bmod p} \cap \mathcal{I}_i|$.

We first run the algorithm in Claim 8.5 for each i with $L' = \frac{L|\mathcal{I}_i|}{2p}$, which takes $\widetilde{O}(n^{(5+\omega)/4}/L') = \widetilde{O}(n^{(7+\omega)/4}/L)$ time. Claim 8.5 computes $|W'_{xk \bmod p} \cap \mathcal{I}_i|$ as long as $|W'_{xk \bmod p} \cap \mathcal{I}_i| \geq L'$. For each fixed k, it fails to compute $|W'_{xk \bmod p} \cap \mathcal{I}_i|$ with probability $O(\sqrt{n}/L)$ by Claim 8.6. For these k, we enumerate over $j \in \mathcal{I}_i$, check if $j \in W'_{xk \bmod p}$, and then compute $|W'_{xk \bmod p} \cap \mathcal{I}_i|$. In expectation, the cost of these k is $O(\frac{n \cdot \sqrt{n}}{L} \cdot |\mathcal{I}_i|) = O(n^2/L)$.

Summing over all $i \in [\ell]$, the total expected running time of the algorithm is $\widetilde{O}(n^{(9+\omega)/4}/L)$. □

Theorem 8.7. *If* All-Nums-3SUM-Convolution *for length n arrays has an $O(n^{2-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #All-Nums-3SUM-Convolution *for length n arrays has an $O(n^{2-\varepsilon'})$ time randomized algorithm for some $\varepsilon' > 0$*

Proof. Similar as before, given an #All-Nums-3SUM-Convolution instance on length n arrays A, B, C, we can count the number of witnesses for C_k that have at most $n^{0.99}$ witnesses in $O(n^{2-\varepsilon''})$ time by well-known techniques [VW18] when All-Nums-3SUM-Convolution has a truly subquadratic algorithm.

For the rest values of k, we run the algorithm in Lemma 8.4 which runs in $\widetilde{O}(n^{(9+\omega)/4}/n^{0.99}) = O(n^{1.86})$ time.

Overall, the algorithm for #All-Nums-3SUM-Convolution runs in $O(n^{2-\varepsilon''} + n^{1.86})$ time, which is truly subquadratic. □

As in Remark 4.3, Theorem 8.7 implies that 3SUM-Convolution is subquadratically equivalent to #3SUM-Convolution.

Theorem 8.8. *If* Min-Plus-Convolution *for length n arrays has an $O(n^{2-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #Min-Plus-Convolution *for length n arrays has an $O(n^{2-\varepsilon'})$ time randomized algorithm for some $\varepsilon' > 0$.*

Proof. Given an #Min-Plus-Convolution instance for length n arrays A, B, we first run the assumed Min-Plus-Convolution algorithm to compute the Min-Plus convolution C of A and B in $O(n^{2-\varepsilon})$ time.

Similar as before, given the $O(n^{2-\varepsilon})$ time algorithm for Min-Plus-Convolution, we can count the number of witnesses for C_k that have at most $n^{0.99}$ witnesses in $O(n^{2-\varepsilon''})$ time by well-known techniques [VW18].

For the rest values of k, we run the algorithm in Lemma 8.4 with arrays A, B, C and $L = n^{0.99}$ which runs in $\widetilde{O}(n^{(9+\omega)/4}/n^{0.99}) = O(n^{1.86})$ time.

Overall, the algorithm for #Min-Plus-Convolution runs in $O(n^{2-\varepsilon} + n^{2-\varepsilon''} + n^{1.86})$ time, which is truly subquadratic. □

We then show a reduction from Min-Plus-Convolution to #Min-Plus-Convolution.

Theorem 8.9. *If* #Min-Plus-Convolution *for length n arrays has an $O(n^{2-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* Min-Plus-Convolution *for length n arrays has an $O(n^{2-\varepsilon'})$ time randomized algorithm for some $\varepsilon' > 0$.*

Proof. Let A and B be two length n arrays for a Min-Plus-Convolution instance. Let C be their Min-Plus convolution. As in proof of Theorem 8.3, we can assume $|W_k| = 1$ for every k, i.e., there exists a unique i_k such that $A_{i_k} + B_{k-i_k} = C_k$.

For each $p \in [\lceil \log(n) \rceil]$, we perform the following round. Let A' be a length $2n$ array such that $A'_{2i-1} = A_i$ for every $i \in [n]$, $A'_{2i} = A_i$ for every $i \in [n]$ whose p-th bit in its binary representation is 1, and $A'_{2i} = \infty$ for the rest of i. Also, let B' be a length $2n$ array such that $B'_{2i-1} = B'_{2i} = B_i$ for every $i \in [n]$. Now we use the #Min-Plus-Convolution algorithm for arrays A' and B'. Suppose C' is the Min-Plus convolution between A' and B'. Clearly, for any $k \in [n]$, $C'_{2k-1} = C_k$. Also, suppose C'_{2k-1} has 2 witnesses, then we know that $A'_{2i_k} = A_{i_k}$ and thus the p-th bit in i_k-th binary representation is 1; otherwise the p-th bit in i_k-th binary representation is 0.

Thus, after the $\lceil \log(n) \rceil$ rounds, we can compute i_k for each $k \in [n]$, which can then be used to compute the Min-Plus convolution C between A and B in $O(n)$ time. □

8.3 All-Numbers 3SUM

Theorem 8.10. *If* All-Nums-3SUM *for sets of n numbers has an $O(n^{2-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #All-Nums-3SUM *for sets of n numbers has an $O(n^{2-\varepsilon'})$ time randomized algorithm for some $\varepsilon' > 0$.*

Proof. If All-Nums-3SUM for sets of n numbers has an $O(n^{2-\varepsilon})$ time algorithm for $\varepsilon > 0$, then so does All-Nums-3SUM-Convolution for length n arrays, since All-Nums-3SUM-Convolution is not harder than All-Nums-3SUM. Then by Theorem 8.7, #All-Nums-3SUM-Convolution for length n arrays has an $O(n^{2-\varepsilon''})$ time algorithm for some $\varepsilon'' > 0$. Therefore, it suffices to reduce #All-Nums-3SUM to #All-Nums-3SUM-Convolution. Some previous reductions from 3SUM to 3SUM-Convolution actually work for the counting variants as well [Păt10, CH20]. Arguably the simplest such reduction is given in [CH20, Section 3]. Applying their reduction finishes the proof. □

As in Remark 4.3, Theorem 8.10 implies that 3SUM is subquadratically equivalent to #3SUM.

Still more equivalence results for other counting and detection problems are given in Appendix A.

8.4 Discussion

Abboud, Feller and Weimann [AFW20] showed that counting the number of Negative Triangles in a graph (even mod 2) can solve Exact-Tri, thus presenting a barrier to showing that the Negative Triangle problem (Neg-Tri) is equivalent to its counting variant: Vassilevska W. and Williams [VW18] showed that Neg-Tri is equivalent to APSP under subcubic fine-grained reductions; then if #Neg-Tri can be reduced to Neg-Tri, one can also reduce it to APSP, and since there are fine-grained reductions from 3SUM to Exact-Tri [VW13], and from Exact-Tri to #Neg-Tri [AFW20], one would get a very surprising reduction from 3SUM to APSP. There is some evidence that such a reduction would be difficult to obtain: for instance, while APSP has a superlogarithmic improvement over its simple cubic algorithm [Wil18], the best improvement over the simple quadratic algorithm of 3SUM only shaves two logarithmic factors (e.g. [BDP08])!

Our equivalences between Min-Plus-Product (and thus Minimum Weight Triangle) and Exact-Tri respectively with their counting variants exhibit a strange phenomenon: Neg-Tri seems different from these problems! Or, perhaps, if we believe that Neg-Tri is like these problems and is equivalent to #Neg-Tri, then we should be more optimistic about the existence of a fine-grained reduction from 3SUM to APSP.

Another line of work in which counting variants of fine-grained problems have been considered is in worst-case to average-case reductions and *fine-grained cryptography* [BRSV17, BRSV18, BBB19, GR20, DLV20, LLV19, Mer78]: building cryptographic primitives from worst-case fine-grained assumptions that might still hold even if P = NP. The known techniques for worst-case to average-case reductions for fine-grained problems only work for counting problems, whereas the design of fine-grained public key protocols [LLV19, Mer78] seem to require that the decision variants are hard on average.

Suppose that one can use the known toolbox for worst-case to average-case reductions for counting problems to show that #Exact-Tri or #3SUM is hard on average. Then via our reductions back to Exact-Tri and 3SUM, one would get some distributions for which these decision problems are actually hard. This could pave the way to new public-key protocols.

9 Counting Algorithms in Other Models

9.1 Co-Nondeterministic Algorithms for Counting Problems with Integer Inputs

First, we show how to modify the co-nondeterministic algorithms in [CGI+16] to work for the counting versions of AE-Exact-Tri, AE-Neg-Tri and All-Nums-3SUM.

Before we show these algorithms, we start with the following simple observation.

Claim 9.1. *For any integers a, b, c, $a + b + c \leq 0$ if and only if one of the followings is true:*

- $\lceil \frac{a}{2} \rceil + \lceil \frac{b}{2} \rceil + \lceil \frac{c}{2} \rceil \leq 0$;
- *At least 2 of a, b, c are odd and $\lceil \frac{a}{2} \rceil + \lceil \frac{b}{2} \rceil + \lceil \frac{c}{2} \rceil = 1$.*

It has the following immediate consequence:

Lemma 9.2. *Given a* #AE-Neg-Tri *instance on an n-node tripartite graph G with node partitions A, B, and C and with edge weights in $[\pm n^{O(1)}]$, we can reduce it to $O(\log n)$ instances of* #AE-Exact-Tri *on graphs with the same vertex set and with edge weights in $[\pm n^{O(1)}]$ deterministically in $\widetilde{O}(n^2)$ time. Furthermore, the count for an edge in the original* #AE-Neg-Tri *instance can be written as a sum of the count for the corresponding edge among the* #AE-Exact-Tri *instances (if this edge does not exist in a certain instance, then the count is simply 0).*

Proof. Without loss of generality, we assume that the instance G has weight function w and we would like to compute for every $a \in A, c \in C$, the number of $b \in B$ such that $w(a,b) + w(b,c) + w(a,c) < 0$. Then we add 1 to the weight of every edge $(a,c) \in A \times C$. Now we need to count the number of number of $b \in B$ such that $w(a,b) + w(b,c) + w(a,c) \leq 0$.

Consider recursing with respect to the bound $[\pm W]$ on the edge weights, initially $W = n^{O(1)}$.

1. If $W = O(1)$. Then we enumerate all possible combinations of 3 weights $w_1, w_2, w_3 \in [\pm W]$ such that $w_1 + w_2 + w_3 \leq 0$, and for each combination, we create an #AE-Exact-Tri instance with all edges $(a,b) \in A \times B$ s.t. $w(a,b) = w_1$, all edges $(b,c) \in B \times C$ s.t. $w(b,c) = w_2$ and all edges $(c,a) \in C \times A$ s.t. $w(c,a) = w_3$. In these instances, we make all edge weights 0, so that all triangles have weights 0. Thus, the total number of exact triangles through edge (a,c) in these $O(1)$ instances is exactly the number of non-positive-weight triangles through edge (a,c) in the original instance.

2. Otherwise, consider Claim 9.1. A triangle (a,b,c) has $w(a,b) + w(b,c) + w(c,a) \leq 0$ if and only if $\lceil \frac{w(a,b)}{2} \rceil + \lceil \frac{w(b,c)}{2} \rceil + \lceil \frac{w(c,a)}{2} \rceil \leq 0$ or at least 2 of $w(a,b), w(b,c), w(c,a)$ are odd and $\lceil \frac{w(a,b)}{2} \rceil + \lceil \frac{w(b,c)}{2} \rceil + \lceil \frac{w(c,a)}{2} \rceil = 1$. Note that these two cases are disjoint, so we can separately consider them and sum up the counts. For the first case, we can create a graph G' by replacing the weight $w(u,v)$ of each edge with $\lceil \frac{w(u,v)}{2} \rceil$ and recursively solve the problem on G'. For the second case, we enumerate which subset of $w(a,b), w(b,c)$ and $w(c,a)$ are odd (there should be at least two of them) and keep the corresponding edges in G' only if they meet the parity condition. This way, we will create 4 sub-problems, where each sub-problem is an exact triangle instance with target value 1.

Overall, we will create $O(\log n)$ #AE-Exact-Tri instances as initially $W = n^{O(1)}$. □

Remark 9.3. Lemma 9.2 implies that #AE-Neg-Tri subcubically reduces to #AE-Exact-Tri. As shown in [AFW20], #Exact-Tri reduces to #Neg-Tri, and the same reduction works from #AE-Exact-Tri to #AE-Neg-Tri. Thus, #AE-Neg-Tri and #AE-Exact-Tri are subcubically equivalent. By Theorem 4.2, they are also subcubically equivalent to AE-Exact-Tri.

Theorem 9.4. #AE-Exact-Tri *for graphs with integer weights in* $[\pm n^\alpha]$ *for any constant* α *is in* $(N \cap coN)TIME[\widetilde{O}(n^{(3+\omega)/2})]$. *The same bound holds for* #AE-Neg-Tri.

Proof. Suppose we are given an instance of #AE-Exact-Tri. Without loss of generality, we assume the instance is a weighted graph G with node partitions A, B, and C and weight function w, and we would like to compute for every $a \in A, c \in C$, the number of $b \in B$ such that $w(a,b)+w(b,c)+w(a,c) = 0$. We will actually more generally count the number of negative-weight, zero-weight, and positive-weight triangles through each edge in $A \times C$.

Both the prover and the verifier run the reduction in Lemma 9.2 on G to get #AE-Exact-Tri instances $G_<^{(1)}, \ldots, G_<^{(O(\log n))}$. They also similarly run the reduction in Lemma 9.2 on G but with all edge weights negated to get #AE-Exact-Tri instances $G_>^{(1)}, \ldots, G_>^{(O(\log n))}$. Without loss of generality, we can assume these #AE-Exact-Tri instances all have target value 0. By construction in Lemma 9.2, all these graphs have the same vertex set $A \cup B \cup C$. Also, let q be a constant to be fixed later.

The prover provides the following for each of the graphs $G, G_<^{(1)}, \ldots, G_<^{(O(\log n))}, G_>^{(1)}, \ldots, G_>^{(O(\log n))}$ (we refer to it by G'):

- A prime p in the interval $[n^{1-q}, Cn^{1-q}\log n]$ for a large enough constant C.

  The prime p is supposed to be such that the number of triangles in G' that are zero mod p but are nonzero otherwise is at most $O(n^{2+q})$. Such a prime is guaranteed to exist since each triangle that has nonzero weight in $[-n^\alpha, n^\alpha]$ is zero mod at most $\log(3n^\alpha)/\log(n^{1-q})$ primes in the interval $[n^{1-q}, Cn^{1-q}\log n]$. The interval contains at least n^{1-q} primes, so some prime must give rise to at most $\frac{n^3\log(3n^\alpha)}{\log(n^{1-q})\cdot n^{1-q}} = O(n^{2+q})$ fake zero triangles.

- A set R of $O(n^{2+q})$ triangles. These are supposed to be the triangles in G' that are nonzero but are zero mod p.

The verifier first checks that R contains only triangles whose weights are nonzero but are zero mod p. This takes $O(|R|) = O(n^{2+q})$ time. Then it counts in $\tilde{O}(n^{1-q+\omega})$ time (using [AGM97]) the number of triangles that are zero mod p through each edge (a,c). Call it $t_{G'}(a,c)$. For each edge (a,c), the verifier subtracts the number of triangles through it in R from $t_{G'}(a,c)$. Now, notice that $t_{G'}(a,c)$ must be an upper bound on the number of zero triangles through (a,c); also, if R contains all the triangles whose weights are nonzero but are zero mod p as it is supposed to, $t_{G'}(a,c)$ will be equal to the number of zero triangles through (a,c).

Now, the verifier applies the second part of Lemma 9.2 to sum up the corresponding counts. For every edge $(a,c) \in A \times C$, it will get $T_<(a,c) = \sum_i t_{G_<^{(i)}}(a,c)$, which is supposed to be the number of negative-weight triangles through (a,c); $T_>(a,c) = \sum_i t_{G_>^{(i)}}(a,c)$, which is supposed to be the number of positive-weight triangles through (a,c); and $t_G(a,c)$, which is supposed to be the number of zero-weight triangles through (a,c).

Then the algorithm counts the number of triangles $s(a,c)$ through each edge (a,c) in $\widetilde{O}(n^\omega)$ time.

Finally, the algorithm verifies $s(a,c) = T_<(a,c) + T_>(a,c) + t_G(a,c)$ and then outputs $t_G(a,c)$ for every edge (a,c).

The correctness of the algorithm relies on the following: Suppose $s_<(a,c), s_=(a,c), s_>(a,c)$ are the actual number of negative-weight, zero-weight, and positive-weight triangles through each edge (a,c). Then by previous discussion, the algorithm is sure that $s_<(a,c) \leq T_<(a,c), s_=(a,c) \leq t_G(a,c)$ and $s_>(a,c) \leq T_>(a,c)$, and trivially $s(a,c) = s_<(a,c) + s_=(a,c) + s_>(a,c)$. These two combined with $s(a,c) = T_<(a,c) + T_>(a,c) + t_G(a,c)$ implies $s_<(a,c) = T_<(a,c), s_=(a,c) = t_G(a,c)$ and $s_>(a,c) = T_>(a,c)$.

The running time is $\widetilde{O}(n^{2+q} + n^{1-q+\omega})$ and is minimized for $q = (\omega-1)/2$. The verifier's running time is thus $\widetilde{O}(n^{(3+\omega)/2})$.

The above algorithm clearly also works for #AE-Neg-Tri. □

Similarly, we can design a sub-quadratic time co-nondeterministic algorithm for #All-Nums-3SUM.

Theorem 9.5. #All-Nums-3SUM *for size-n sets A, B, C of integers in $[\pm n^\alpha]$ for any constant α is in $(N \cap coN)TIME[\widetilde{O}(n^{1.5})]$.*

Proof sketch. The proof is essentially the same as the proof of Theorem 9.4, with the following modifications.

Without loss of generality, we assume #All-Nums-3SUM asks to count the number of $(a,b) \in A \times B$ where $a+b+c=0$ for every fixed $c \in C$.

Then, similar to Lemma 9.2, we can reduce counting the number of $(a,b) \in A \times B$ where $a+b+c<0$ and counting the number of $(a,b) \in A \times B$ where $a+b+c>0$ to $O(\log n)$ instances of #All-Nums-3SUM.

For each of these instances on (A', B', C'), the prover provides a prime p in the interval $[n^{1+q}, Cn^{1+q}\log n]$ for a large enough constant C and a constant q to be fixed. It also provides a set of R of $O(n^3/n^{1+q}) = O(n^{2-q})$ triples (a,b,c), which are supposed to be the triples in $A' \times B' \times C'$ whose sums are nonzero but are zero mod p.

For each of these instances, the verifier checks that R contains only triples whose sum is nonzero but is zero mod p. Then it counts in $\widetilde{O}(p) = \widetilde{O}(n^{1+q})$ time (using FFT) the number of triples (a,b,c) whose sum is zero mod p for every $c \in C'$, and then subtracts the corresponding count in R from this count.

Similar to Theorem 9.4, by summing up the counts in all these instances for every $c \in C$, and checking whether that equals $|A||B|$, the verifier will be sure to get the correct count of zero-sum triples involving each $c \in C$.

The running time of the verifier is $\widetilde{O}(n^{1+q} + n^{2-q})$, which is $\widetilde{O}(n^{1.5})$ by setting $q = 0.5$.

□

9.2 Co-Nondeterministic Algorithms for Counting Problems with Real Inputs

The co-nondeterministic algorithms in Section 9.1 heavily rely on the idea of modulo a prime p, which will no longer be possible if the input numbers are reals. In this section, we study algorithms for #AE-Real-Exact-Tri, #AE-Real-Neg-Tri and #All-Nums-Real-3SUM.

For problems with real inputs in the nondeterministic model, we assume the verifier has access to a "Reasonable" Real RAM model, which was discussed in [CVX22]. In such models, only a restricted subset of operations are allowed on the real-valued inputs, while any operations in the Word RAM model with $O(\log n)$-bit words are allowed for the integer parts of the computation. In particular, our algorithms work in the Real RAM with low-degree predicates model. Also, if we slightly change the definition of Exact-Tri to finding a triangle (a,b,c) such that $w(a,b) + w(b,c) = w(a,c)$ (and similarly for Neg-Tri), our algorithms will work in the Real RAM with 4-linear comparisons model. See [CVX22] for more details about "Reasonable" Real RAM models.

Our ideas for proving equivalence between counting and detection problems can be used to obtain new nondeterministic algorithms for counting problems with real inputs, as we show in this subsection.

We start with the following co-nondeterministic algorithm for Equality-Product and Dominance-Product.

Lemma 9.6. Equality-Product *between an $n_1 \times n_2$ matrix A and an $n_2 \times n_3$ matrix B, where we only need to determine the results on a given subset $X \subseteq [n_1] \times [n_3]$, is in*

$$(N \cap coN)TIME\left[\widetilde{O}\left(|X|n_2^s + M(n_1, n_2^{2-s}, n_3)\right)\right]$$

time for any $s \in [0, 1]$. The same bound holds for Dominance-Product.

Proof. Without loss of generality, we can assume all entries of A and B are integers in $O(n_1n_2 + n_2n_3)$, by replacing each entry by its rank.

For every $(i, j) \in [n_1] \times [n_3]$, we use $c_=(i, j)$ to denote the number of k where $A_{ik} = B_{kj}$, $c_<(i, j)$ to denote the number of k where $A_{ik} < B_{kj}$ and $c_>(i, j)$ to denote the number of k where $A_{ik} > B_{kj}$. Instead of only computing $c_=(i, j)$ for $(i, j) \in X$, we will more generally compute $c_<(i, j)$ and $c_>(i, j)$ as well.

By known reductions from Dominance-Product to Equality-Product [LUWG19, Vas15], we can create $O(\log n)$ instances of Equality-Product on matrices of the same dimensions, and use the sum of resulting values on entry (i, j) over these $O(\log n)$ instances to compute $c_<(i, j)$. It similarly holds for $c_>(i, j)$.

For each of the $O(\log n)$ instances of Equality-Product (the $O(\log n)$ instances generated above, and the original instance), the prover provides the following (say the instance is on matrices A', B'):

- A prime p in the interval $[n_2^{1-s}, Cn_2^{1-s} \log(n_2)]$ for a large enough constant C.

  The prime p is supposed to be such that the number of triples (i, k, j) where $(i, j) \in X$, $k \in [n_2]$ and $A'_{ik} \neq B'_{kj}$ while $A'_{ik} \equiv B'_{kj} \pmod p$ is at most $\widetilde{O}(|X|n_2^s)$. Such a prime is guaranteed to exist since for every triple (i, k, j) with $A'_{ik} \neq B'_{kj}$, A'_{ik} and B'_{kj} are congruent mod at most $\widetilde{O}(1)$ primes in the interval $[n_2^{1-s}, Cn_2^{1-s} \log(n_2)]$. The interval contains at least n_2^{1-s} primes, so some prime must give rise to at most $\widetilde{O}\left(\frac{|X|n_2}{n_2^{1-s}}\right) = \widetilde{O}(|X|n_2^s)$ fake zero triangles.

- A set R of $\widetilde{O}(|X|n_2^s)$ triples (i, k, j) where $(i, j) \in X$ and $k \in [n_2]$. These are supposed to be the triples where $A'_{ik} \neq B'_{kj}$ while $A'_{ik} \equiv B'_{kj} \pmod p$.

The verifier is similar to the verifier in the proof of Theorem 9.4. For each Equality-Product instance, after checking that the set R only contains triples (i, k, j) where $(i, j) \in X$, $k \in [n_2]$ and $A'_{ik} \neq B'_{kj}$ while $A'_{ik} \equiv B'_{kj} \pmod p$ in $\widetilde{O}(|R|)$ time, it computes C'_{ij}, the number of $k \in [n_2]$ such that $A'_{ik} \equiv B'_{kj} \pmod p$, for each pair of $(i, j) \in [n_1] \times [n_3]$ in $\widetilde{O}(M(n_1, pn_2, n_3))$ time, by packing p instances of matrix multiplications of dimensions $n_1 \times n_2 \times n_3$ together. Similar as before, by subtracting the number of triples involving (i, j) in R from C'_{ij}, C'_{ij} becomes an upper bound on the number of k such that $A'_{ik} = B'_{kj}$.

Finally, by checking that the sum of C'_{ij} over all the Equality-Product instances equals n_2 for every $(i, j) \in X$, the verifier will be sure that C'_{ij} equals exactly the number of k such that $A'_{ik} = B'_{kj}$ in each of the instances. In particular, it can compute $c_=(i, j), c_<(i, j)$ and $c_>(i, j)$ for every $(i, j) \in X$.

The running time of the verifier is $\widetilde{O}(|R| + M(n_1, pn_2, n_3)) = \widetilde{O}(|X|n_2^s + M(n_1, n_2^{2-s}, n_3))$, as desired. □

Next, we are ready to present our co-nondeterministic algorithm for #AE-Real-Exact-Tri and #AE-Real-Neg-Tri.

Theorem 9.7. #AE-Real-Exact-Tri *for n-node graphs is in $(N \cap coN)TIME[\widetilde{O}(n^{\frac{6+\omega}{3}})]$. The same bound holds for* #AE-Real-Neg-Tri.

Proof. Without loss of generality, we assume the instance is a weighted graph G with node partitions A, B, and C and weight function w, and we would like to compute for every $a \in A, c \in C$, the number of $b \in B$ such that $w(a,b) + w(b,c) + w(a,c) = 0$. Also, let q be a constant to be fixed later.

The prover provides the following:

- A subset $R \subseteq [n]$ of size $\widetilde{O}(n^q)$.

  For every $(a,c) \in A \times C$, let $L_{ac} = \{w(a,b) + w(b,c) : b \in R\}$. Let p_{ac} be the index of the predecessor of $-w(a,c)$ (including $-w(a,c)$) in L_{ac}, i.e., it is $\arg\max_{b \in R, w(a,b)+w(b,c) \leq -w(a,c)}(w(a,b) + w(b,c))$, and let s_{ac} be the index of the successor of $-w(a,c)$ (excluding $-w(a,c)$) in L_{ac}, i.e., it is $\arg\min_{b \in R, w(a,b)+w(b,c) > -w(a,c)}(w(a,b) + w(b,c))$. The set R is supposed to be that, if $-w(a,c) \notin L_{ac}$, then the number of $b \in B$ where $w(a,p_{ac}) + w(p_{ac},c) < w(a,b) + w(b,c) < w(a,s_{ac}) + w(s_{ac},c)$ is $O(n^{1-q})$. Such R exists because a random R satisfies these properties with high probability.

- For every $b_0 \in R$, the prover uses the protocol in Lemma 9.6 to create outputs for the purpose of counting $|\{w(a,b) - w(a,b_0) = w(b_0,c) - w(b,c) : b \in B\}|$ and $|\{w(a,b) - w(a,b_0) \leq w(b_0,c) - w(b,c) : b \in B\}|$, where X is the set of (a,c) such that $b_0 = p_{ac}$ or $b_0 = s_{ac}$.

- Finally, for every (a,c) with $-w(a,c) \notin L_{ac}$, it provides a list $\ell_{ac} \subseteq [n]$, which is supposed to contain all indices b with $w(a,p_{ac}) + w(p_{ac},c) < w(a,b) + w(b,c) < w(a,s_{ac}) + w(s_{ac},c)$. Note that $|\ell_{ac}| = O(n^{1-q})$ by the choice of R.

The verifier does the following. First, it computes L_{ac}, p_{ac}, s_{ac} in $\widetilde{O}(n^2|R|) = \widetilde{O}(n^{2+q})$ time. Next, it uses the algorithm in Lemma 9.6 to correctly count, for every $b_0 \in R$, the values of

$$|\{w(a,b) - w(a,b_0) = w(b_0,c) - w(b,c) : b \in B\}|$$

and

$$|\{w(a,b) - w(a,b_0) \leq w(b_0,c) - w(b,c) : b \in B\}|$$

for (a,c) such that $b_0 = p_{ac}$ or $b_0 = s_{ac}$. Let x_i be the size of X for the i-th call of Lemma 9.6. The running time of the i-th call of Lemma 9.6 can be bounded by

$$\widetilde{O}\left(|X|n^s + M(n, n^{2-s}, n)\right) = \widetilde{O}\left(|X|n^s + n^{1-s}n^\omega\right) = \widetilde{O}(n^{\frac{\omega+1}{2}}\sqrt{x_i} + n^\omega),$$

by setting $n^s = \min\left(n, \sqrt{\frac{n^{1+\omega}}{|X|}}\right)$. Then we notice that $\sum_i x_i = O(n^2)$. Therefore, by convexity, the running time can be bounded as

$$\sum_i \widetilde{O}\left(n^{\frac{\omega+1}{2}}\sqrt{x_i} + n^\omega\right) = \widetilde{O}\left(|R| \cdot n^{\frac{\omega+1}{2}}\sqrt{\frac{n^2}{|R|}} + |R| \cdot n^\omega\right) = \widetilde{O}(n^{\frac{\omega+3+q}{2}} + n^{\omega+q}).$$

For some (a,c), if $w(a,p_{ac}) + w(p_{ac},c) = -w(a,c)$, then by Fredman's trick,

$$\begin{aligned}
&|\{w(a,b) - w(a,b_0) = w(b_0,c) - w(b,c) : b \in B\}| \\
=&|\{w(a,b) + w(b,c) = w(a,b_0) + w(b_0,c) : b \in B\}| \\
=&|\{w(a,b) + w(b,c) + w(a,c) = 0 : b \in B\}|
\end{aligned}$$

for $b_0 = p_{ac}$, which is exactly the count we seek. Otherwise, the algorithm computes the number of b where $w(a, p_{ac}) + w(p_{ac}, c) < w(a, b) + w(b, c) < w(a, s_{ac}) + w(s_{ac}, c)$ via

$$\begin{aligned} &|\{w(a,b) - w(a,s_{ac}) \le w(s_{ac},c) - w(b,c) : b \in B\}| \\ -&|\{w(a,b) - w(a,s_{ac}) = w(s_{ac},c) - w(b,c) : b \in B\}| \\ -&|\{w(a,b) - w(a,p_{ac}) \le w(p_{ac},c) - w(b,c) : b \in B\}|, \end{aligned}$$

where all three counts are computed earlier. Next, the verifier checks that the length of ℓ_{ac} equals this count, and for every $b \in \ell_{ac}$, the verifier checks $w(a, p_{ac}) + w(p_{ac}, c) < w(a, b) + w(b, c) < w(a, s_{ac}) + w(s_{ac}, c)$. If these checks pass, then ℓ_{ac} contains exactly the set of b where $w(a, p_{ac}) + w(p_{ac}, c) < w(a, b) + w(b, c) < w(a, s_{ac}) + w(s_{ac}, c)$. By the definition of p_{ac} and s_{ac}, it must be the case that $w(a, p_{ac}) + w(p_{ac}, c) < -w(a, c) < w(a, s_{ac}) + w(s_{ac}, c)$. Therefore, by reading the list ℓ_{ac} and count how many $b \in \ell_{ac}$ has $w(a, b) + w(b, c) = -w(a, c)$, the verifier can correctly compute the number of exact triangles through edge (a, c). Overall, the cost of this step is the total length of ℓ_{ac}, which is $O(n^{3-q})$.

The running time of the verifier is $\widetilde{O}(n^{2+q} + n^{\frac{\omega+3+q}{2}} + n^{\omega+q} + n^{3-q})$. By setting $q = \frac{3-\omega}{3}$, it becomes $\widetilde{O}(n^{\frac{6+\omega}{3}})$.

To adapt the algorithm to #AE-Real-Neg-Tri, for every (a, c), we also count the number of b such that $w(a, b) + w(b, c) < w(a, p_{ac}) + w(p_{ac}, c)$ via

$$\begin{aligned} &|\{w(a,b) - w(a,p_{ac}) \le w(p_{ac},c) - w(b,c) : b \in B\}| \\ -&|\{w(a,b) - w(a,p_{ac}) = w(p_{ac},c) - w(b,c) : b \in B\}|. \end{aligned}$$

If $-w(a, c) \in L_{ac}$, then $w(a, p_{ac}) + w(p_{ac}, c) = w(a, c)$, so the above count is exactly the number of negative triangles through (a, c). Otherwise, every $b \in B$ with $w(a, b) + w(b, c) < w(a, p_{ac}) + w(p_{ac}, c) < -w(a, c)$ forms a negative triangle with (a, c). All other negative triangles can be found via searching ℓ_{ac}. □

Before we present our co-nondeterministic algorithm for #All-Nums-Real-3SUM, we first introduce the following lemma.

Lemma 9.8. *Given an $n_1 \times n_2$ matrix A, an $n_3 \times n_2$ matrix B, and a subset $X \subseteq [n_1] \times [n_3]$, counting the number of (k, ℓ) where $A_{ik} = B_{j\ell}$ for every $(i, j) \in X$ is in*

$$(N \cap coN)TIME\left[\widetilde{O}\left(|X| n_2^s + M(n_1, n_2^{2-s}, n_3)\right)\right]$$

time for any $s \in [0, 1]$. Consequently, counting the number of (k, ℓ) where $A_{ik} \le B_{j\ell}$ for every $(i, j) \in X$ is also in

$$(N \cap coN)TIME\left[\widetilde{O}\left(|X| n_2^s + M(n_1, n_2^{2-s}, n_3)\right)\right]$$

time for any $s \in [0, 1]$.

Proof. First, we can assume all entries of A and B are integers bounded by $O(n_1 n_2 + n_2 n_3)$, by replacing each entry by its rank.

We enumerate $b_1, b_2 \in \{0\} \cup [\lfloor \log(n_2) \rfloor]$. If the occurrence of a number on the i-th row of A has a 1 in its binary representation on the bit corresponding to 2^{b_1}, we keep one copy of this number on the i-th row of A; otherwise, we drop this number. Similarly, if the occurrence of a number on the j-th row of B has a 1 in its binary representation on the bit corresponding to 2^{b_2}, we keep one copy of this number on the j-th row

of B; otherwise, we drop this number. We then solve the original problem on these two modified matrices. Finally, we can sum up the results obtained on these modified matrices, weighted by $2^{b_1+b_2}$. This way, we can assume all numbers on the i-th row of A are distinct for any i, and all numbers on the j-th row of B are distinct for any j.

The prover provides the following:

- A function $h : [O(n_1n_2 + n_2n_3)] \to [n_2]$.

  The function is supposed to be that, for every $i \in [n_1]$, each value in $[n_2]$ matches at most $\widetilde{O}(1)$ numbers in the multiset $\{h(A_{ik}) : k \in [n_2]\}$. Similarly, for every $j \in [n_3]$, each value in $[n_2]$ matches at most $\widetilde{O}(1)$ numbers in the multiset $\{h(B_{jk}) : k \in [n_2]\}$. Such a function exists because a random function satisfies these constraints with high probability.

- Two matrices A', B'. A' is an $n_1 \times \widetilde{O}(n_2)$ matrix, where the columns of it is indexed by $[n_2] \times [\widetilde{O}(1)] \times [\widetilde{O}(1)]$. For every $i \in [n_1], k \in [n_2], s_2 \in [\widetilde{O}(1)]$, we set $A'_{i,(h(A_{ik}),s_1,s_2)}$ to A_{ik}, where A_{ik} is the s_1-th number on row i that get mapped to $h(A_{ik})$. Similarly, B' is an $n_3 \times \widetilde{O}(n_2)$ matrix, where the columns of it is indexed by $[n_2] \times [\widetilde{O}(1)] \times [\widetilde{O}(1)]$. For every $j \in [n_3], k \in [n_2], s_1 \in [\widetilde{O}(1)]$, we set $B'_{j,(h(B_{jk}),s_1,s_2)}$ to B_{jk}, where B_{jk} is the s_2-th number on row j that get mapped to $h(B_{jk})$. All other entries of A' and B' are set to values distinct from any other values.

- Use the protocol in Lemma 9.6 to create outputs for the purpose of computing the equality product between A' and B'^T with the output set X.

The verifier does the following. First, it checks that h, A', B' are valid. Then it runs the algorithm in Lemma 9.6 to compute, for every $(i, j) \in X$, the number of (k, s_1, s_2) where $A'_{i,(k,s_1,s_2)} = B'_{j,(k,s_1,s_2)}$. Note that this is exactly the number of (k, ℓ) where $A_{ik} = B_{j\ell}$. The running time of the algorithm is same as that of Lemma 9.6.

If we instead want to count the number of (k, ℓ) where $A_{ik} \le B_{j\ell}$ for every $(i, j) \in X$, we can use the idea that reduces Dominance-Product to Equality-Product [LUWG19, Vas15] to create $\widetilde{O}(1)$ instances of the previous problem, so it only incurs an additional $\widetilde{O}(1)$ factor. □

Given an All-Nums-3SUM instance on size-n sets A, B, C, we can first sort A and B and then divide them to consecutive sub-lists $A_1, \ldots, A_{n/d}$ and $B_1, \ldots, B_{n/d}$ of size d. It is a well-known observation that, in order to determine whether some $c \in C$ is in a 3SUM solution, it suffices to search for c in $A_i + B_j$ for $O(n/d)$ pairs of $(i, j) \in [n/d]^2$. For #All-Nums-Real-3SUM, the observation is still true: For every $c \in C$, it suffices to count the number of c in $A_i + B_j$ for $O(n/d)$ pairs of $(i, j) \in [n/d]^2$. We can thus determinitically reduce #All-Nums-Real-3SUM to the following problem (see its non-counting version in, e.g., [Cha20]).

Problem 9.9. *We are given two real $(n/d) \times d$ matrices A and B, and a set C_{ij} of real numbers for every (i, j). For every $c \in C_{ij}$, we are asked to count the number of (k, ℓ) such that $A_{i,k} + B_{j,\ell} = c$. Additionally $\sum_{i,j} |C_{ij}| = O(n^2/d)$.*

Theorem 9.10. #All-Nums-Real-3SUM *for size-n sets is in $(N \cap coN)TIME[\widetilde{O}(n^{\frac{3\omega+3}{\omega+3}})]$.*

Proof. We first deterministically reduce #All-Nums-Real-3SUM to Problem 9.9. The proof then proceeds similarly to the proof of Theorem 9.7.

The prover provides the following:

- A subset $R \subseteq [d] \times [d]$ of size $\widetilde{O}(r)$.

  For every i, j, let $L_{ij} = \{A_{ik} + B_{j\ell} : (k, \ell) \in R\}$. Let $(pk_{ijc}, p\ell_{ijc})$ be the index of the predecessor of c in L_{ij} (including c) and let $(sk_{ijc}, s\ell_{ijc})$ be the index of the successor of c in L_{ij} (excluding c). The set R is supposed to be that, for some $c \in C_{ij}$ and $c \notin L_{ij}$, the number of (k, ℓ) with $A_{i,pk_{ijc}} + B_{j,p\ell_{ijc}} < A_{ik} + B_{j\ell} < A_{i,sk_{ijc}} + B_{j,s\ell_{ijc}}$ is $O(d^2/r)$. Such a set exists because a random R satisfies these properties with high probability.

- For every $(k_0, \ell_0) \in R$, it uses the protocol in Lemma 9.8 to create outputs for the purpose of counting $|\{A_{ik} - A_{ik_0} = B_{j\ell_0} - B_{j\ell}\}|$ and $|\{A_{ik} - A_{ik_0} \leq B_{j\ell_0} - B_{j\ell}\}|$, where X is the set of (i, j) such that there exists $c \in C_{ij}$ with $(pk_{ijc}, p\ell_{ijc}) = (k_0, \ell_0)$ or $(sk_{ijc}, s\ell_{ijc}) = (k_0, \ell_0)$.

- Finally, for every (i, j) and $c \in C_{ij}$ with $c \notin L_{ij}$, it provides a list $\ell\ell_{ijc} \subseteq [d] \times [d]$, which is supposed to contain all pairs (k, ℓ) with $A_{i,pk_{ijc}} + B_{j,p\ell_{ijc}} < A_{i,k} + B_{j,\ell} < A_{i,sk_{ijc}} + B_{j,s\ell_{ijc}}$. Note that $|\ell\ell_{ijc}| = O(d^2/r)$ by the choice of R.

□

The verifier is almost identical to the verifier in Theorem 9.7, so we omit its details for conciseness. The running time of the verifier is

$$\widetilde{O}\left(\frac{d^2}{r} \cdot \sum_{i,j} |C_{ij}| + \sum_i \left(x_i d^s + M\left(\frac{n}{d}, d^{2-s}, \frac{n}{d}\right)\right)\right),$$

where x_i is the size of X in the i-th call of Lemma 9.8. By picking $d = n^{\frac{1}{3-s}}$, the running time can be simplified to

$$\widetilde{O}\left(\frac{n^{\frac{2}{3-s}}}{r} \cdot \sum_{i,j} |C_{ij}| + \sum_i \left(x_i n^{\frac{s}{3-s}} + n^{\frac{2-s}{3-s} \cdot \omega}\right)\right).$$

We know $\sum_i x_i = O(\sum_{i,j} |C_{ij}|) = O(n^2/d) = O(n^{\frac{5-2s}{3-s}})$, and we call Lemma 9.8 a total of $O(|R|) = O(r)$ times. Thus, we can further upper bound the running time by

$$\widetilde{O}\left(\frac{n^{\frac{7-2s}{3-s}}}{r} + n^{\frac{5-s}{3-s}} + r \cdot n^{\frac{2-s}{3-s} \cdot \omega}\right).$$

By setting $s = \frac{2\omega - 3}{\omega}$ and $r = n^{\frac{3}{\omega+3}}$, the running time becomes $\widetilde{O}(n^{\frac{3\omega+3}{\omega+3}})$.

Remark 9.11. The above approach also leads to new consequences in a certain unrealistic model of computation: an *unrestricted Real RAM*, supporting standard arithmetic operation on real numbers with unbounded precision, but without the floor function. With the floor function, it is known that the model enables PSPACE-hard problems to be solved in polynomial time [Sch79]. In a recent paper [CVX22], it was noted that even without the floor function, the model may still be unreasonably powerful; for example, there is a truly subcubic time algorithm for APSP for *integer* input under this model. But the question of whether there are similarly subcubic algorithms for APSP and other related problems for *real* input was not answered.

Our proof of Theorem 9.7 implies a truly subcubic randomized time algorithm for #AE-Real-Exact-Tri in this unrestricted Real RAM without the floor function. This is because we use nondeterminism mainly to generate all the witnesses. But using large numbers, we can do standard matrix product and have all the witnesses represented as a long bit vector. When witnesses are needed for an output entry, we can generate them one by one using a most-significant-bit operation, which can be simulated by binary search. Small adaptation of the proof also shows a truly subcubic randomized time algorithm for the real-valued version of #Min-Plus-Product. Similarly, Theorem 9.10 implies a truly subquadratic randomized time algorithm for #All-Nums-Real-3SUM in the unrestricted Real RAM without the floor function.

9.3 Quantum Algorithms for Counting Problems

Our equivalences between counting and detection problems immediately imply faster quantum algorithms for counting problems. For instance, we obtain the following

Corollary 9.12. *There exists an $\widetilde{O}(n^{2-\varepsilon})$ time quantum algorithm for* #3SUM *for some $\varepsilon > 0$.*

Proof. It is known that 3SUM can be solved in $\widetilde{O}(n)$ quantum time (see e.g. [AL20]). Applying Theorem 8.10 finishes the proof. □

9.4 Discussion

#CNF-SAT asks to count the number of satisfying assignments to a CNF formula, and it is considered harder than CNF-SAT: the counting version #SETH of the Strong Exponential Time Hypothesis (SETH) [IPZ01, CIP10] is considered even more believable than SETH (see [CM16]). Williams' [Wil05] reduction from CNF-SAT to OV preserves the counts, and thus is also a fine-grained reduction from #CNF-SAT to #OV. Similar to the situation for CNF-SAT, there's no known fine-grained reduction from #OV to OV.

Our techniques do not yet give equivalences between the decision and counting variants of CNF-SAT and OV. If such equivalences do not exist, this would indicate that OV and CNF-SAT are different from the other core problems in FGC.

Such an indication was already observed by Carmosino et al. [CGI+16] who studied the nondeterministic and co-nondeterministic complexity of fine-grained problems. They formulated NSETH that asserts that there is no $O((2-\varepsilon)^n)$ time nondeterministic algorithm which can verify that a given CNF formula has no satisfying assignment. They also exhibited nondeterministic algorithms for verifying the YES and NO solutions of Exact-Tri, APSP in truly subcubic time and 3SUM in truly subquadratic time, and concluded that if NSETH holds, then there can be no deterministic fine-grained reduction from CNF-SAT or OV to any of Exact-Tri, APSP or 3SUM.

Because of our efficient nondeterministic algorithms for #Exact-Tri, #Neg-Tri and #3SUM, we then get that under NSETH, there can be no deterministic fine-grained reductions from CNF-SAT or OV to #Exact-Tri, #APSP or #3SUM.

Recently, Akmal, Chen, Jin, Raj and Williams [ACJ+22] showed, among other things, that #Exact-Tri has an $\widetilde{O}(n^2)$ time Merlin-Arthur protocol. Their protocol crucially uses polynomial identity testing, and hence it is not known how to derandomize it and make it nondeterministic. Our results yield a truly subcubic nondeterministic protocol.

10 BSG Theorems Revisited

In Section 5, we have seen how our techniques may replace some of the previous uses of the BSG Theorem in algorithmic applications. In this section, we show how our techniques can actually prove variants

of the BSG Theorem itself.

We begin with a quick review of the BSG Theorem. Many different versions of the theorem can be found in the literature, and the following is one version that is easy to state:

Theorem 10.1. (BSG Theorem) *Given subsets A, B, and C of size n of an abelian group, and a parameter s, if $|\{(a,b) \in A \times B : \ a+b \in C\}| \geq n^2/s$, then there exist subsets $A' \subseteq A$ and $B' \subseteq B$ both of size $\Omega(n/s)$, such that*

$$|A' + B'| \ = \ O(s^5 n).$$

The earliest version of the theorem, with super-exponential factors in s, was obtained by Balog and Szemerédi [BS94], via the regularity lemma. Gowers [Gow01] was the first to obtain a version with polynomial dependency on s. The version stated above was proved by Balog [Bal07] and Sudakov, Szemerédi and Vu [SSV05]. Although the proof is not long and does not need advanced tools, it is clever and not easy to think of; see [TV06, Lov17, Vio11] for various different expositions.

Chan and Lewenstein [CL15] gave algorithmic applications using the following variant which we will call the "BSG Covering Theorem" (it was called the "BSG Corollary" in their paper). Instead of extracting a single pair of large subsets (A', B'), the goal is to construct a cover by multiple pairs of subsets $(A^{(i)}, B^{(i)})$:

Theorem 10.2. (BSG Covering) *Given subsets A, B, and C of size n of an abelian group, and a parameter s, there exist a collection of $\ell = O(s)$ subsets $A^{(1)}, \ldots, A^{(\ell)} \subseteq A$ and $B^{(1)}, \ldots, B^{(\ell)} \subseteq B$, and a set R of $O(n^2/s)$ pairs in $A \times B$, such that*

(i) $\{(a,b) \in A \times B : a+b \in C\} \ \subseteq \ R \cup \bigcup_\lambda (A^{(\lambda)} \times B^{(\lambda)})$, *and*

(ii) $|A^{(\lambda)} + B^{(\lambda)}| = O(s^5 n)$ *for each λ (and so $\sum_\lambda (|A^{(\lambda)} + B^{(\lambda)}|) = O(s^6 n)$).*

The BSG Covering Theorem is not implied by the BSG Theorem as stated, but the known proofs by Balog [Bal07] and Sudakov et al. [SSV05] established an extension of the BSG Theorem that involves an input graph, and repeated applications of this theorem indeed provide multiple pairs of subsets satisfying the stated properties.

10.1 A New Simpler Version

We will now show how our techniques, combined with some new extra ideas, can derive a version of the BSG Covering Theorem where the $O(s^6 n)$ bound is weakened to $\widetilde{O}(s^2 n^{3/2})$. Although the new bound is superlinear in n, the lower polynomial dependency on s actually compensates to yield improved results in some algorithmic applications of the Covering Theorem. In particular, $\widetilde{O}(s^2 n^{3/2})$ is better when $s \gg n^{1/8}$. A key advantage of the new proof is its simplicity: it constructs a cover directly, instead of repeatedly extracting subsets one at a time (thus, it avoids the need to extend the BSG Theorem with an input graph, and thereby simplifies the algorithm considerably).

We focus on the setting where each input set A is of the form $\{(i, a_i) : i \in [n]\}$ for a sequence of integers or reals $a_1, \ldots, a_n$. We call such a set an *indexed set*. This case turns out to be sufficient for applications involving integer input (because known hashing-based techniques used in reductions from 3SUM to 3SUM-Convolution (e.g. [KPP16]) can map integer sets to indexed sets—3SUM-Convolution is just 3SUM for indexed sets). Focusing on indexed sets makes the proof more intuitive, and also makes the construction more efficient. (In Appendix C, we present a variant of the proof for general sets in an abelian group.)

Note that in the theorem stated below, we work with "monochromatic" difference sets of the form $A-A$, instead of bichromatic sum sets $A+B$. This form is actually more general, since given A and B, we can

reset A to $A \cup (-B)$. The reduction does not go the other way, since knowing that $|A^{(\lambda)} + B^{(\lambda)}|$ is small does not mean $|(A^{(\lambda)} \cup (-B^{(\lambda)})) - (A^{(\lambda)} \cup (-B^{(\lambda)}))|$ is small. (The proofs for the known $O(s^6 n)$ bound work only for the bichromatic sum sets but not for monochromatic difference sets, whereas Gower's earlier proof works for monochromatic difference sets.)

Theorem 10.3. (Simpler BSG Covering) *Given indexed sets A and C of size n and a parameter s, there exist a collection of $\ell = \widetilde{O}(s^3)$ subsets $A^{(1)}, \ldots, A^{(\ell)} \subseteq A$, and a set R of $\widetilde{O}(n^2/s)$ pairs in $A \times A$, such that*

(i) $\{(a,b) \in A \times A : a - b \in C\} \subseteq R \cup \bigcup_\lambda (A^{(\lambda)} \times A^{(\lambda)})$, *and*

(ii) $\sum_\lambda |A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^2 n^{3/2})$.

The $A^{(\lambda)}$'s and R can be constructed in $\widetilde{O}(n^2)$ Las Vegas randomized time.

Proof. Let $A = \{(i, a_i) : i \in [n]\}$ and $C = \{(k, c_k) : k \in [n]\}$. As a preprocessing step, we sort the multiset $\{a_{i+k} - a_i : i \in [n]\}$ for each i, in $\widetilde{O}(n^2)$ total time. Let $W_k = \{i \in [n] : a_{i+k} - a_i = c_k\}$.

- **Few-witnesses case.** For each k with $|W_k| \le n/s$, add $\{((i+k, a_{i+k}), (i, a_i)) : i \in W_k\}$ to R. The number of pairs added to R is $O(n \cdot n/s)$. The running time of this step is also $\widetilde{O}(n \cdot n/s)$.
- **Many-witnesses case.** Pick a random subset $H \subseteq [\pm n]$ of size $c_0 s \log n$ for a sufficiently large constant c_0. Let $L^{(h)}$ be the multiset $\{a_{i+h} - a_i : i \in [n]\}$. Let $F^{(h)}$ be the elements of frequency more than n/r in $L^{(h)}$. Note that $|F^{(h)}| \le r$. These can be computed in $\widetilde{O}(sn)$ time.
  - **Low-frequency case.** For each $h \in H$ and $i \in [n]$, if $a_{i+h} - a_i \notin F^{(h)}$, we examine each of the at most n/r indices j with $a_{i+h} - a_i = a_{j+h} - a_j$ and add $((j, a_j), (i, a_i))$ to R. The number of pairs added to R is $\widetilde{O}(sn \cdot n/r) = \widetilde{O}(n^2/s)$ by choosing $r := s^2$. The running time of this step is also bounded by $\widetilde{O}(n^2/s)$.
  - **High-frequency case.** For each $h \in H$ and $f \in F^{(h)}$, add the following subset to the collection:
    $$A^{(h,f)} = \{(i, a_i) \in A : a_{i+h} - a_i = f\}.$$
    The number of subsets is $\widetilde{O}(s \cdot r) = \widetilde{O}(s^3)$. The running time of this step is
    $$O\left(\sum_{h \in H,\ f \in F^{(h)}} |A^{(h,f)}|\right) = \widetilde{O}(sn).$$

Correctness. To verify (i), consider a pair $((i+k, a_{i+k}), (i, a_i)) \in A \times A$ with $a_{i+k} - a_i = c_k$. If $|W_k| \le n/s$, then $((i+k, a_{i+k}), (i, a_i)) \in R$ due to the "few-witnesses" case. So assume $|W_k| > n/s$. Then H hits $W_k - i$ w.h.p., so there exists $h \in H$ with $i + h \in W_k$, i.e., $a_{i+h+k} - a_{i+h} = c_k$. Since $a_{i+k} - a_i = c_k$, we have $a_{i+h+k} - a_{i+k} = a_{i+h} - a_i$ (by Fredman's trick). Let $f = a_{i+h} - a_i$. If $f \notin F^{(h)}$, then $((i+k, a_{i+k}), (i, a_i)) \in R$ due to the "low-frequency" case. If $f \in F^{(h)}$, then $((i+k, a_{i+k}), (i, a_i)) \in A^{(h,f)} \times A^{(h,f)}$ due to the "high-frequency" case.

Thus far, the proof ideas are similar to what we have seen before. However, to verify (ii), we will propose a new probabilistic argument. Consider a fixed h. We want to bound the sum $S^{(h)} := \sum_{f \in F^{(h)}} |A^{(h,f)} - A^{(h,f)}|$. This is equivalent to bounding the number of triples (k, c, f) such that $f \in F^{(h)}$ and $\exists i$ with $a_{i+k} - a_i = c$ and $a_{i+h} - a_i = a_{i+k+h} - a_{i+k} = f$. Note that we also have $a_{i+k+h} - a_{i+h} = c$ (by Fredman's trick). Let $Y_{k,c} = \{i : a_{i+k} - a_i = c\}$ and q be a parameter. Then $S^{(h)}$ is upper-bounded by the number of triples (k, c, f) satisfying

1. $|Y_{k,c}| > q$ and $f \in F^{(h)}$, or

2. $|Y_{k,c}| \le q$ and $\exists i$ with $i \in Y_{k,c}$ and $i+h \in Y_{k,c}$ and $f = a_{i+h} - a_i$.

Since $\sum_{k,c} |Y_{k,c}| = O(n^2)$, the number of triples of type 1 is $O((n^2/q) \cdot r) = \widetilde{O}(s^2 n^2/q)$. On the other hand, the expected number of triples of type 2 for a random $h \in [\pm n]$ is

$$O\left(\sum_{k,c:\, |Y_{k,c}| \le q} |Y_{k,c}| \cdot |Y_{k,c}|/n\right) \;=\; O(n^2 \cdot q/n) \;=\; O(nq).$$

Hence, $\mathbb{E}[S^{(h)}] = \widetilde{O}(s^2 n^2/q + nq) = \widetilde{O}(sn^{3/2})$ by choosing $q = s\sqrt{n}$.

The expected total sum $\sum_{h \in H} \sum_{f \in F^{(h)}} |A^{(h,f)} - A^{(h,f)}|$ is $\widetilde{O}(s^2 n^{3/2})$. Thus, the probability that the desired bounds are not met is less than an arbitrarily small constant (by Markov's inequality and a union bound).

Note that the algorithm can be converted to Las Vegas, because we can verify correctness of the construction by examining all $O(n^2)$ pairs and computing all difference sets $A^{(h,f)} - A^{(h,f)}$ using known output-sensitive algorithms [CH02, CL15, BFN22] in total time $\widetilde{O}(n^2 + s^2 n^{3/2}) = \widetilde{O}(n^2)$ (we may assume $s < n^{1/4}$, for otherwise the theorem is trivial). □

10.2 Application: Improved 3SUM in Preprocessed Universes

As one immediate application, we can solve 3SUM with preprocessed universe, improving Chan and Lewenstein's previous solution which required $\widetilde{O}(n^{13/7})$ query time [CL15], and also improving Corollary 5.3 regardless of the value of ω: the query algorithm does not use fast matrix multiplication but uses FFT instead, though randomization is now needed in the preprocessing algorithm.

Corollary 10.4. *We can preprocess sets A, B, and C of n integers in $\widetilde{O}(n^2)$ Las Vegas randomized time, so that given any subsets $A' \subseteq A$, $B' \subseteq B$, and $C' \subseteq C$, we can solve* All-Nums-3SUM *on (A', B', C') in $\widetilde{O}(n^{11/6})$ time.*

Proof. Kopelowitz, Pettie and Porat [KPP16] gave a simple randomized reduction from 3SUM to $O(\log n)$ instances of 3SUM-Convolution via hashing. The same approach works in the preprocessed universe setting, and transform the input into $O(\log n)$ instances where A, B, and C are indexed sets.

During preprocessing, we apply Theorem 10.3 to $A \cup (-B)$, producing subsets $A^{(\lambda)}$ and a set R of pairs.

During a query with given subsets $A' \subseteq A$, $B' \subseteq B$, and $C' \subseteq C$, we first examine each pair $(a, -b) \in R$ and check whether $a \in A'$, $b \in B'$, and $a + b \in C'$. This takes $\widetilde{O}(n^2/s)$ time.

Next, for each λ, we compute $(A^{(\lambda)} \cap A') + ((-A^{(\lambda)}) \cap B')$ by known FFT-based algorithms [CH02, CL15, BFN22]; the running time is near-linear in the output size, which is bounded by $|A^{(\lambda)} - A^{(\lambda)}|$. For each output value c, we check whether $c \in C'$.

The total query time is $\widetilde{O}(n^2/s + s^2 n^{3/2})$. Choosing $s = n^{1/6}$ yields the theorem. □

We remark that the same $\widetilde{O}(n^{11/6})$ bound holds for a slightly more general case when C' is an arbitrary set of n integers, i.e., the preprocessing does not need the set C. This is because in the proof of Theorem 10.3, the $\widetilde{O}(n^2)$-time preprocessing step is independent of C, and the rest of the construction takes $\widetilde{O}(n^2/s + sn)$ time. In contrast, Chan and Lewenstein's paper obtained a weaker $\widetilde{O}(n^{19/10})$ bound for the same case without C.

10.3 Reinterpreting Gower's Version

In this section, we show that Gower's proof [Gow01] (see also a related recent proof by Schoen [Sch15]) can also be modified to construct a cover directly.[8] Gower's proof requires more clever arguments; our new presentation highlights the similarities and differences with the proof of Theorem 10.3. This variant of the proof will be needed in a later application in Section 11 to conditional lower bounds—so ideas from additive combinatorics will be useful after all!

In the following, we let $\mathrm{pop}_A(x)$ (the *popularity of* x) denote the number of pairs $(a, b) \in A \times A$ with $x = a - b$; in other words, $\mathrm{pop}_A(x) = |\{a \in A : a - x \in A\}|$.

Theorem 10.5. *Given indexed sets A and C of size n and a parameter s, there exist a collection of $\ell = \widetilde{O}(s^3)$ subsets $A^{(1)}, \ldots, A^{(\ell)} \subseteq A$, and a set R of $\widetilde{O}(n^2/s)$ pairs in $A \times A$, such that*

(i) $\{(a, b) \in A \times A : a - b \in C\} \subseteq R \cup \bigcup_\lambda (A^{(\lambda)} \times A^{(\lambda)})$, *and*

(ii) $|A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^6 n)$ *for each λ (and so $\sum_\lambda |A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^9 n)$).*

The $A^{(\lambda)}$'s and R can be constructed in $\widetilde{O}(n^2)$ Las Vegas randomized time.

Proof. We follow the proof in Theorem 10.3 but modify the handling of the "high-frequency" case. Let t be a parameter. Define

$$\begin{aligned} G^{(h,f)} &= \{(a, b) \in A^{(h,f)} \times A^{(h,f)} : \mathrm{pop}_A(a - b) \le n/t\} \\ Z^{(h,f)} &= \{a \in A^{(h,f)} : \deg_{G^{(h,f)}}(a) > |A^{(h,f)}|/4\}. \end{aligned}$$

Here, $\deg_{G^{(h,f)}}(a)$ refers to the degree of a in $G^{(h,f)}$ when viewed as a graph. For each $h \in H$ and $f \in F^{(h)}$, add $Z^{(h,f)} \times A^{(h,f)}$ and $A^{(h,f)} \times Z^{(h,f)}$ to R. For each $h \in H$ and $f \in F^{(h)}$, instead of adding the subset $A^{(h,f)}$, we add the subset $\widetilde{A}^{(h,f)} := A^{(h,f)} \setminus Z^{(h,f)}$ to the collection.

Correctness. To analyze this modified construction, first observe that every pair previously covered by $A^{(h,f)} \times A^{(h,f)}$ is now covered by $\widetilde{A}^{(h,f)} \times \widetilde{A}^{(h,f)}$ or by the extra pairs added to R (i.e., $Z^{(h,f)} \times A^{(h,f)}$ or $A^{(h,f)} \times Z^{(h,f)}$).

We bound the expected number of extra pairs added to R. Consider a fixed h. Note that $|Z^{(h,f)}| \le O\left(\frac{|G^{(h,f)}|}{|A^{(h,f)}|/4}\right)$, and so $\sum_f |Z^{(h,f)}||A^{(h,f)}| = O\left(\sum_f |G^{(h,f)}|\right)$. The sum $\sum_f |G^{(h,f)}|$ is bounded by the number of triples (i, j, f) such that $a_{i+h} - a_i = a_{j+h} - a_j = f$ and $\mathrm{pop}_A((i - j, a_i - a_j)) \le n/t$. This is bounded by the number of pairs (i, j) such that $\mathrm{pop}_A((i - j, a_i - a_j)) \le n/t$ and $a_i - a_j = a_{i+h} - a_{j+h}$ (by Fredman's trick). For a fixed (i, j) with $\mathrm{pop}_A((i - j, a_i - a_j)) \le n/t$, the number of h's with $a_i - a_j = a_{i+h} - a_{j+h}$ is at most n/t, and so the probability that $a_i - a_j = a_{i+h} - a_{j+h}$ for a random $h \in [\pm n]$ is $O(1/t)$. It follows that

$$\mathbb{E}_h \left[\sum_f |Z^{(h,f)}||A^{(h,f)}| \right] = O(n^2 \cdot 1/t).$$

Consequently, the expected number of extra pairs added to R is $\widetilde{O}(s \cdot n^2/t) = \widetilde{O}(n^2/s)$ by setting $t := s^2$.

[8] There are multiple different exposition of Gower's and subsequent proofs of the BSG Theorem in the literature. For example, some presentations [Bal07, SSV05, TV06, Vio11] cleanly separate the algebraic from the combinatorial components, by reducing the problem to some combinatorial lemma about graphs (counting paths of length 2 or 3 or 4). But these versions of the proof do not achieve our goal of computing a cover directly and efficiently. Our reinterpretation is nontrivial and requires examining Gower's proof from the right perspective.

Finally, we consider a fixed h and fixed $f \in F^{(h)}$ and provide an upper bound on $|\widetilde{A}^{(h,f)} - \widetilde{A}^{(h,f)}|$. For each $c \in \widetilde{A}^{(h,f)} - \widetilde{A}^{(h,f)}$, pick a lexicographically smallest $(a, b) \in \widetilde{A}^{(h,f)} \times \widetilde{A}^{(h,f)}$ with $c = a - b$. Consider all $y \in A^{(h,f)}$ with $(a, y), (b, y) \notin G^{(h,f)}$; the number of such y's is at least $|A^{(h,f)}| - |A^{(h,f)}|/4 - |A^{(h,f)}|/4 = \Omega(|A^{(h,f)}|) = \Omega(n/r)$ (since $|A^{(h,f)}| > n/r$ for $f \in F^{(h)}$). For each such y, examine each $(a', a'') \in A \times A$ with $a - y = a' - a''$ and each $(b', b'') \in A \times A$ with $b - y = b' - b''$, and mark the quadruple (a', a'', b', b''). Since $(a, y), (b, y) \notin G^{(h,f)}$, there are at least n/t choices of (a', a'') and at least n/t choices of (b', b'') for each such y. Letting Q be the number of quadruples marked, we obtain

$$Q \;=\; \Omega\left(|\widetilde{A}^{(h,f)} - \widetilde{A}^{(h,f)}| \cdot (n/r) \cdot (n/t)^2\right).$$

On the other hand, each quadruple (a', a'', b', b''), is marked once, since it uniquely determines the element $c = (a' - a'') - (b' - b'')$, from which (a, b) is uniquely determined and $y = a - (a' - a'')$ is uniquely determined. Thus, $Q = O(n^4)$. We conclude that

$$|\widetilde{A}^{(h,f)} - \widetilde{A}^{(h,f)}| \;=\; O\left(\frac{n^4}{(n/r) \cdot (n/t)^2}\right) \;=\; O(rt^2 n) \;=\; \widetilde{O}(s^6 n).$$

Construction time. One could naively construct the sets $Z^{(h,f)}$ and $\widetilde{A}^{(h,f)}$ in $O(\sum_{h,f} |A^{(h,f)}|^2)$ time, but a faster way is to use random sampling. Given any value c, we can approximate $\mathrm{pop}_A(c)$ with additive error $\delta n/t$ w.h.p. by taking a random subset $A' \subseteq A$ of $O((1/\delta^2)t \log n) = \widetilde{O}(t)$ elements and computing $|\{a \in A' : a - c \in A\}| \cdot |A|/|A'|$ (by a standard Chernoff bound). We will not construct $G^{(h,f)}$ explicitly. Instead, given any (a, b), we can test for membership in $G^{(h,f)}$ in $\widetilde{O}(t)$ time. Furthermore, given a, we can approximate $\deg_{G^{(h,f)}}(a)$ with additive error $\delta|A^{(h,f)}|$ w.h.p. by taking a random subset $A'' \subseteq A^{(h,f)}$ of $O((1/\delta^2)\log n) = \widetilde{O}(1)$ elements and computing $|\{y \in A'' : (a, y) \in G^{(h,f)}\}| \cdot |A^{(h,f)}|/|A''|$. This way, $Z^{(h,f)}$ (and thus $\widetilde{A}^{(h,f)}$) can be generated in $\widetilde{O}(t|A^{(h,f)}|)$ time for each h and f. The total time bound is $\widetilde{O}(t \sum_{h,f} |A^{(h,f)}|) = \widetilde{O}(tsn) = \widetilde{O}(s^3 n)$, which is dominated by other costs (we may assume that $s < n^{1/6}$, for otherwise the theorem is trivial). Due to these approximations, our earlier analysis needs small adjustments in the constant factors, but is otherwise the same.

As before, the algorithm can be converted to Las Vegas in $\widetilde{O}(n^2)$ additional time. □

One important advantage of the above proof is that the running time is actually *subquadratic*, excluding the $\widetilde{O}(n^2)$-time preprocessing step, which is needed only in the "few-witnesses" case (ignoring the conversion to Las Vegas). In particular, we immediately obtain subquadratic running time for the following variant of the theorem, which requires only the "many-witnesses" case (where we reset r and t to $s\hat{s}$ instead of s^2). This variant will be useful later.

Theorem 10.6. *Given an indexed set A of size n and parameters s and $\hat{s}$, there exist a collection of $\ell = \widetilde{O}(s^2\hat{s})$ subsets $A^{(1)}, \ldots, A^{(\ell)} \subseteq A$, and a set R of $\widetilde{O}(n^2/\hat{s})$ pairs in $A \times A$, such that*

(i) $\{(a, b) \in A \times A : \mathrm{pop}_A(a - b) > n/s\} \;\subseteq\; R \cup \bigcup_\lambda (A^{(\lambda)} \times A^{(\lambda)})$, *and*

(ii) $|A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^3\hat{s}^3 n)$ *for each λ (and so $\sum_\lambda |A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^5\hat{s}^4 n)$).*

The $A^{(\lambda)}$'s and R can be constructed in $\widetilde{O}(n^2/\hat{s} + s^2\hat{s}n)$ Monte Carlo randomized time.

Although we are able to reinterpret Gower's proof, we are unable to modify the proof by Balog [Bal07] or Sudakov et al. [SSV05] to achieve similar subquadratic construction time.

11 Lower Bounds for Min-Equality Convolution

In this section, we prove conditional lower bounds for the Min-Equal-Convolution problem under Strong APSP Hypothesis, the u-dir-APSP Hypothesis, and the Strong Min-Plus Convolution Hypothesis. The lower bound under the Strong APSP Hypothesis or u-dir-APSP Hypothesis follows just by combining Corollary 3.5 with the known reduction from u-dir-APSP to Min-Witness-Eq-Prod [CVX21] (and noticing that Min-Witness-Eq-Prod is easier than Min-Equal-Prod), and then using known ideas for reducing matrix product problems to convolution problems (more specifically, the unpublished reduction from BMM to pattern-to-text Hamming distances, attributed to Indyk – see e.g. [GU18]). The lower bound under the Strong Min-Plus Convolution Hypothesis is more delicate: interestingly, we will combine ideas that we have developed for conditional lower bounds for intermediate matrix product problems, with one of our new versions of the BSG Theorem from the previous section.

Theorem 11.1. *Under the Strong APSP Hypothesis,* Min-Equal-Convolution *for length n arrays requires $n^{1+1/6-o(1)}$ time.*

Proof. First, by Corollary 3.5, u-dir-APSP requires $n^{7/3-o(1)}$ time under the Strong Integer-APSP Hypothesis. Zwick's algorithm [Zwi02] can be seen as a reduction from u-dir-APSP to $\widetilde{O}(1)$ instances of Min-Plus-Product between $n \times n^\alpha$ matrices and $n^\alpha \times n$ matrices with weights in $[n^{1-\alpha}]$ for various values of $\alpha \in [0,1]$. As shown in [CVX21], each instance of Min-Plus-Product in this form can be reduced to an instance of Min-Witness-Eq-Prod for $O(n) \times O(n)$ matrices (they only stated the reduction for $\alpha = \rho$ for a particular value of ρ, but their proof works for any $\alpha \in [0,1]$). Therefore, Min-Witness-Eq-Prod requires $n^{7/3-o(1)}$ time under the Strong Integer-APSP Hypothesis.

We can easily reduce a Min-Witness-Eq-Prod instance to a Min-Equal-Prod instance. Suppose the input of a Min-Witness-Eq-Prod instance is A, B. Assume all entries are in $[2n]$ without loss of generality. We can create two $n \times n$ matrices A' and B', where $A'_{ik} = A_{ik} + 2nk$ and $B'_{kj} = B_{kj} + 2nk$, then the Min-Witness Equality product between A and B can be computed in $\widetilde{O}(n^2)$ time given the Min-Equality product between A' and B'.

Finally, we reduce Min-Equal-Prod to Min-Equal-Convolution, following the strategy of the unpublished reduction from Boolean matrix multiplication to pattern-to-text Hamming distances, attributed to Indyk, see e.g. [GU18].

Let A and B be the inputs of Min-Equal-Prod. W.l.o.g., we can assume all entries of A and B are integers in $[2n^2]$. We first create two length $2n^2$ arrays a and b, where initially all entries of a and b are ∞. For every $(i,k) \in [n] \times [n]$, we set $a_{(n+1)(i-1)+k}$ to $nA_{ik} + k - 1$; for every $(k,j) \in [n] \times [n]$, we set b_{jn-k} to $nB_{kj} + k - 1$.

Suppose the Min-Equality product between A and B is C and the Min-Equality convolution between a and b is c, we will show that $C_{ij} = \lfloor c_{(n+1)(i-1)+jn}/n \rfloor$ (and $C_{ij} = \infty$ if $c_{(n+1)(i-1)+jn}$ is ∞), which will complete the reduction.

To show the equality, first notice that

$$\begin{aligned} &\min\left\{a_{(n+1)(i-1)+k} : k \in [n] \wedge a_{(n+1)(i-1)+k} = b_{jn-k}\right\} \\ =&\min\left\{nA_{ik} + k - 1 : k \in [n] \wedge A_{ik} = B_{kj}\right\} \end{aligned} \tag{2}$$

contributes to the minimization of $c_{(n+1)(i-1)+jn}$. Also, no other terms less than ∞ can contribute: suppose there exists some x such that $a_x = b_y < \infty$ and $x + y = (n+1)(i-1) + jn$, then $a_x \bmod n$ must match $b_y \bmod n$. Thus, there must exist $i', j', k' \in [n]$ such that $x = (n+1)(i'-1) + k'$ and $y = j'n - k'$,

so $(n+1)(i'-1)+j'n = (n+1)(i-1)+jn$. Then it must be the case that $i = i'$ and $j = j'$, so a_x corresponds to one of the terms in Equation (2). □

The reduction in the proof of Theorem 11.1 from u-dir-APSP to Min-Equal-Convolution also easily imply the following:

Theorem 11.2. *Under the u-dir-APSP Hypothesis,* Min-Equal-Convolution *for length n arrays requires $n^{1+\rho/2-o(1)}$ time, where ρ is the constant satisfying $\omega(1,\rho,1) = 1+2\rho$, or $n^{1.25-o(1)}$ time if $\omega = 2$.*

We finally show the lower bound of Min-Equal-Convolution under the Strong Min-Plus Convolution Hypothesis.

Theorem 11.3. *Under the Strong Min-Plus Convolution Hypothesis,* Min-Equal-Convolution *for length n arrays requires $n^{1+1/11-o(1)}$ time.*

Proof. We will show that if Min-Equal-Convolution for length n arrays has an $\widetilde{O}(n^{1+\delta})$ time algorithm for some $\delta \geq 0$, then Min-Plus-Convolution for length n arrays of entries that are bounded by $O(n)$ has an $\widetilde{O}(n^{2-\frac{1-11\delta}{21}})$ time randomized algorithm.

Let A and B be the input arrays of a Min-Plus-Convolution instance. Let t be a parameter to be fixed later, and let $g = \lceil n/t \rceil$. Similar to the proof of Theorem 3.2, we can assume $(A_i \bmod g) < g/2$ and $(B_i \bmod g) < g/2$ for each $i \in [n]$. Also, for each i, we can write A_i as $A'_i g + A''_i$, for $0 \leq A'_i \leq t$ and $0 \leq A''_i < g/2$. Similarly, we can write B_i as $B'_i g + B''_i$.

We first compute the Min-Plus convolution C' of A' and B' in $\widetilde{O}(tn)$ time. Let $W_k = \{i \in [n] : k-i \in [n] \wedge C'_k = A'_i + B'_{k-i}\}$. Suppose we can compute C'', which is defined as $C''_k = \min_{i \in W_k}(A''_i + B''_{k-i})$, then we can compute the Min-Plus convolution C of A and B as $C_k = C'_k g + C''_k$.

We then compute C''_k by two methods depending on whether $|W_k|$ is greater than n/s or not, for some parameter s to be determined.

Claim 11.4. *For any parameter $\hat{s}$, we can compute C''_k for every k where $|W_k| > n/s$ in $\widetilde{O}(n^2/\hat{s} + s^5\hat{s}^4 n^2/t)$ time.*

Proof. We create the following indexed set $\mathcal{A}$ of size $O(n)$:

$$\left\{(i, A'_i) : i \in [n]\right\} \cup \left\{(i+n, \infty) : i \in [n]\right\} \cup \left\{(3n+1-i, -B'_i) : i \in [n]\right\}.$$

Note that for any $k \in [n]$, $|W_k| = \mathrm{pop}_{\mathcal{A}}((k-3n-1, C'_k))$. Then we apply Theorem 10.6 with the index set $\mathcal{A}$ and parameters $s, \hat{s}$ to find a collection of $\ell = \widetilde{O}(s^2\hat{s})$ subsets $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(\ell)}$, and a set R of $\widetilde{O}(n^2/\hat{s})$ pairs in $\mathcal{A} \times \mathcal{A}$ in $\widetilde{O}(n^2/\hat{s} + s^2\hat{s}n)$ randomized time. Furthermore, Theorem 10.6 guarantees that

(i) $\{(a,b) \in \mathcal{A} \times \mathcal{A} : \mathrm{pop}_{\mathcal{A}}(a-b) > n/s\} \subseteq R \cup \bigcup_\lambda (\mathcal{A}^{(\lambda)} \times \mathcal{A}^{(\lambda)})$. This further means that, for every k where $|W_k| > n/s$,

$$\left\{\left((i, A'_i), (3n+1-(k-i), -B'_{k-i})\right) : i \in W_k\right\} \subseteq R \cup \bigcup_\lambda (\mathcal{A}^{(\lambda)} \times \mathcal{A}^{(\lambda)}).$$

(ii) $\sum_\lambda |\mathcal{A}^{(\lambda)} - \mathcal{A}^{(\lambda)}| = \widetilde{O}(s^5\hat{s}^4 n)$.

Then we first enumerate $((i_1, v_1), (i_2, v_2)) \in R$. If this pair corresponds to some A'_i and B'_j (i.e., this pair has $i_1 = i, i_2 = 3n+1-j$), we use $A''_i + B''_j$ to update C''_{i+j} if $A'_i + B'_j = C'_{i+j}$. This takes $\widetilde{O}(|R|) = \widetilde{O}(n^2/\hat{s})$ time.

For each $\lambda \in [\ell]$, we consider the possible witnesses in $\mathcal{A}^{(\lambda)} \times \mathcal{A}^{(\lambda)}$. We prepare a map f from $\mathcal{A}^{(\lambda)}$ to $[g] \cup \{\infty\}$ as follows: if $a \in \mathcal{A}^{(\lambda)}$ corresponds to some A'_i (i.e., $a = (i, A'_i)$), we set $f(a) = A''_i$; if a corresponds to some B'_j (i.e., $a = (3n+1-j, -B'_j)$), we set $f(a) = B''_j$; otherwise, we set $f(a) = \infty$. Then we compute the following Min-Plus "convolution" $\mathcal{C}^{(\lambda)}$:

$$\mathcal{C}^{(\lambda)}_c = \min_{\substack{(a,b)\in\mathcal{A}^{(\lambda)}\times\mathcal{A}^{(\lambda)} \\ a-b=c}} (f(a) + f(b)).$$

By known techniques for solving Min-Plus convolution with small integer weights [AGM97], we can instead solve a normal convolution with weights bounded by $2^{\widetilde{O}(g)}$. More specifically, let $h(a) = M^{f(a)}$ if $f(a) \neq \infty$ and $f(a) = 0$ otherwise, where $M = |\mathcal{A}^{(\lambda)}| + 1$. Then it suffices to compute the following for every c:

$$\sum_{\substack{(a,b)\in\mathcal{A}^{(\lambda)}\times\mathcal{A}^{(\lambda)} \\ a-b=c}} h(a) \cdot h(b).$$

By known output-sensitive algorithms [CH02, CL15, BFN22], it takes $\widetilde{O}(|\mathcal{A}^{(\lambda)} - \mathcal{A}^{(\lambda)}|)$ arithmetic operations to compute the above convolution, and each arithmetic operation takes $\widetilde{O}(g)$ time. Thus, it takes $\widetilde{O}(g|\mathcal{A}^{(\lambda)} - \mathcal{A}^{(\lambda)}|)$ time to compute $\mathcal{C}^{(\lambda)}$.

After we compute $\mathcal{C}^{(\lambda)}$ for every λ, we use the value of $\mathcal{C}^{(\lambda)}_{(k-3n-1,C'_k)}$ to update C''_k for every k, λ.

Overall, the running time is $\widetilde{O}(n^2/\hat{s} + s^2\hat{s}n + g\sum_i |\mathcal{A}^{(\lambda)} - \mathcal{A}^{(\lambda)}|) = \widetilde{O}(n^2/\hat{s} + s^5\hat{s}^4ng) = \widetilde{O}(n^2/\hat{s} + s^5\hat{s}^4n^2/t)$. □

Next we show the following algorithm for the rest values of k where $|W_k|$ is small. Recall that we assumed Min-Equal-Convolution for length n arrays has an $\widetilde{O}(n^{1+\delta})$ time algorithm.

Claim 11.5. *We can compute C''_k for every k where $|W_k| \leq n/s$ in $\widetilde{O}(\frac{n}{s} \cdot n^{1+\delta})$ time as long as $t\sqrt{s} = O(\sqrt{n})$.*

Proof. Let $I \subseteq [n]$ be a random subset of indices for which each index is kept in I independently with probability $\frac{\sqrt{s}}{\sqrt{n}}$. Similarly let $J \subseteq [n]$ be such a random subset as well. With high probability, $|I|, |J| = O(\sqrt{sn})$.

In the sparse Min-Plus convolution between the two sparse arrays A'_I and B'_J, we need to compute a length n array D where $D_k = \min_{i\in I, k-i\in J}(A'_i + B'_j)$.

For every $k \in [n]$ and $i \in W_k$, the probability that $i \in I$ and $k - i \in J$ is $\frac{s}{n}$. Thus, for any particular $i \in W_k$, the probability that i is the unique witness for D_k in the sparse Min-Plus convolution between A'_I and B'_J is $\frac{s}{n} \cdot (1 - \frac{s}{n})^{|W_k|-1}$, which is $\Theta(\frac{s}{n})$ if $|W_k| \leq n/s$. Thus, if we keep sampling I and J for $\widetilde{O}(\frac{n}{s})$ times, all indices in W_k will be the unique witness for D_k in at least one time with high probability, as in standard sampling techniques (see e.g. [AGMN92, Sei95]).

Suppose i is indeed the unique witness for D_k, then we can find i by repeatedly computing some instances of sparse Min-Plus convolutions. More specifically, in the p-th round, let $I^{(p)}$ be I but only keeping the indices whose p-th bit in the binary representation is 1. Say the sparse Min-Plus convolution between $A'_{I^{(p)}}$ and B'_J is $D^{(p)}$. Then if $D^{(p)}_k = D_k$, then we know the p-th bit of i is 1, and otherwise it is 0. Thus, we can recover i after $O(\log n)$ rounds. After we have i, we can use $A''_i + B''_{k-i}$ to update C''_k.

Therefore, it remains to show how to compute the sparse Min-Plus convolution between A'_I and B'_J for $|I|, |J| = O(\sqrt{sn})$.

Let $F : [t] \times [t] \to [n]$ be a random function (independent to the choice of I, J). We then create two arrays X, Y each of length $O(n)$ as follows. Initially, all entries in X and Y are set to some distinct values out side of $[t] \times [t]$. For every $i \in I$ and $y \in [t]$, we set $X_{i+F(A'_i,y)}$ to (A'_i, y); for every $j \in J$ and $x \in [t]$, we set $Y_{j+n-F(x,B'_j)}$ to (x, B'_j). Suppose all entries are only set at most once. Then we compute the Min-Equality convolution Z between X and Y, where we compare two pairs (x, y) and (x', y') by comparing $x + y$ and $x' + y'$ and breaking ties arbitrarily. Then the sum of the two integers in the pair Z_{k+n} equals

$$\min_{\substack{(i,j,x,y)\in I\times J\times[t]\times[t]\\ i+F(A'_i,y)+j+n-F(x,B'_j)=k+n\\ (A'_i,y)=(x,B'_j)}} (A'_i + y) = \min_{\substack{(i,j)\in I\times J\\ i+j=k}} (A'_i + B'_j).$$

Thus, computing a Min-Equal-Convolution instance gives the result of sparse Min-Plus convolution between A'_I and B'_J.

We then remove the assumption that each entry of X and Y is only set once by standard techniques. Consider a fixed entry X_q. For every $(x, y) \in t$, we set X_q to (x, y) if and only if $q - F(x, y) \in I$ and $A'_{q-F(x,y)} = x$. Since $F(x, y)$ is sampled from $[n]$ uniformly at random, the probability that we set X_q to (x, y) is at most $\frac{|\{i\in I:A'_i=x\}|}{n}$. Summing over all x, y, the expected number of times that X_q is set is $O(\frac{|I|t}{n}) = O(\frac{t\sqrt{s}}{\sqrt{n}}) = O(1)$ since $t\sqrt{s} = O(\sqrt{n})$. Since the values of $F(x, y)$ are independent for different (x, y), by Chernoff bound, we conclude that X_q is set only $O(\log n)$ times with high probability. Similarly, all indices in Y are set only $O(\log n)$ times with high probability. Thus, we can create $O(\log n)$ arrays $X^{(a)}$, where $X^{(a)}_p$ equals the value of X_p when we attempt to set it the a-th time. We can similarly create $O(\log n)$ arrays $Y^{(b)}$. Then it suffices to compute the Min-Equal-Convolution between $X^{(a)}$ and $Y^{(b)}$ for every $(a, b) \in [O(\log n)]^2$.

Overall, the running time is $\widetilde{O}(\frac{n}{s} \cdot n^{1+\delta})$ as long as $t\sqrt{s} = O(\sqrt{n})$, assuming Min-Equal-Convolution for length n arrays has an $\widetilde{O}(n^{1+\delta})$ time algorithm. □

By Claim 11.4 and Claim 11.5, we can compute C'' and thus C in $\widetilde{O}(n^2/\hat{s} + s^5\hat{s}^4n^2/t + \frac{n}{s} \cdot n^{1+\delta})$ time as long as $t\sqrt{s} = O(\sqrt{n})$. We can set $t = n^{\frac{10-5\delta}{21}}$, $s = n^{\frac{1+10\delta}{21}}$ and $\hat{s} = n^{\frac{1-11\delta}{21}}$ to get the $\widetilde{O}(n^{2-\frac{1-11\delta}{21}})$ randomized running time. □

Note that in order for the proof for Theorem 11.3 to work, the more difficult BSG covering of Theorem 10.6 that builds on Gower's proof [Gow01] is necessary. If we instead use the simpler BSG covering of Theorem 10.3, we will have an $s^{O(1)}n^{2.5}/t$ term from Claim 11.4, which cannot give subquadratic running time considering the $t\sqrt{s} = O(\sqrt{n})$ requirement in Claim 11.5.

12 Acknowledgement

We would like to thank Lijie Chen for suggesting the quantum #3SUM problem.

References

[Abr87] Karl Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987. 5

[ACJ+22] Shyan Akmal, Lijie Chen, Ce Jin, Malvika Raj, and Ryan Williams. Improved Merlin-Arthur protocols for central problems in fine-grained complexity. In *Proc. 13th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 3:1–3:25, 2022. 43

[ACLL14] Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 114–125, 2014. 6, 11

[AD16] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 477–486, 2016. 7, 26

[AFW20] Amir Abboud, Shon Feller, and Oren Weimann. On the fine-grained complexity of parity problems. In *Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 5:1–5:19, 2020. 9, 34, 36

[AGM97] Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997. Preliminary version in FOCS 1991. 3, 6, 7, 36, 51

[AGMN92] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. Witnesses for Boolean matrix multiplication and for shortest paths. In *Proc. 33rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 417–426, 1992. 15, 16, 51

[AL20] Andris Ambainis and Nikita Larka. Quantum algorithms for computational geometry problems. In *Proc. 15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*, pages 9:1–9:10, 2020. 9, 43

[AR18] Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 239–252, 2018. 6

[AV14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014. 7

[AV21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proc. 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021. 1

[AVY15] Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 41–50, 2015. 23

[Bal07] Antal Balog. Many additive quadruples. In *Additive Combinatorics*, volume 43 of *CRM Proc. Lecture Notes*, pages 39–49. Amer. Math. Soc., 2007. 44, 47, 48

[BBB19] Enric Boix-Adserà, Matthew S. Brennan, and Guy Bresler. The average-case complexity of counting cliques in Erdős-Rényi hypergraphs. In *Proc. 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1256–1280, 2019. 34

[BCD+14] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and $X + Y$. *Algorithmica*, 69(2):294–314, 2014. Preliminary version in ESA 2006. 7, 31

[BDP08] Ilya Baran, Erik D. Demaine, and Mihai Patrascu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008. Preliminary version in WADS 2005. 34

[BFN22] Karl Bringmann, Nick Fischer, and Vasileios Nakos. Deterministic and Las Vegas algorithms for sparse nonnegative convolution. In *Proc. 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3069–3090, 2022. 46, 51

[BGSV16] Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly sub-cubic algorithms for language edit distance and RNA-folding via fast bounded-difference min-plus product. In *Proc. 57th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 375–384, 2016. 2, 4, 21, 63

[BIS17] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2215–2229, 2017. 2

[BKS18] Markus Bläser, Balagopal Komarath, and Karteek Sreenivasaiah. Graph pattern polynomials. In *Proc. 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 18:1–18:13, 2018. 2

[BRSV17] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 483–496, 2017. 34

[BRSV18] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Proofs of work from worst-case assumptions. In *Proc. 38th Annual International Cryptology Conference (CRYPTO)*, pages 789–819, 2018. 34

[BS94] Antal Balog and Endre Szemerédi. A statistical theorem of set addition. *Combinatorica*, 14:263–268, 1994. 44

[BW12] Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. *Theory Comput.*, 8(1):69–94, 2012. 10

[CDL+14] Timothy M. Chan, Stephane Durocher, Kasper Green Larsen, Jason Morrison, and Bryan T. Wilkinson. Linear-space data structures for range mode query in arrays. *Theory Comput. Syst.*, 55(4):719–741, 2014. 7, 24, 25

[CDM17] Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proc. 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 210–223, 2017. 2

[CDSW15] Timothy M. Chan, Stephane Durocher, Matthew Skala, and Bryan T. Wilkinson. Linear-space data structures for range minority query in arrays. *Algorithmica*, 72(4):901–913, 2015. 25

[CDX22] Shucheng Chi, Ran Duan, and Tianle Xie. Faster algorithms for bounded-difference min-plus product. In *Proc. 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1435–1447, 2022. 2, 4, 21, 63

[CDXZ22] Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In *Proc. 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1529–1542, 2022. 2, 11, 21, 63

[CGI+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proc. 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 261–270, 2016. 35, 43

[CH02] Richard Cole and Ramesh Hariharan. Verifying candidate matches in sparse and wildcard matching. In *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 592–601, 2002. 46, 51

[CH20] Timothy M. Chan and Qizheng He. Reducing 3SUM to convolution-3SUM. In *Proc. SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 1–7, 2020. 20, 34

[Cha10] Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010. 4

[Cha20] Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median,+)-convolution, and some geometric 3SUM-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020. Preliminary version in SODA 2018. 4, 41

[CIP10] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. On the exact complexity of evaluating quantified k-CNF. In *Proc. 5th International Symposium on Parameterized and Exact Computation (IPEC)*, pages 50–59, 2010. 43

[CKL07] Artur Czumaj, Miroslaw Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theor. Comput. Sci.*, 380(1-2):37–46, 2007. 1, 5

[CL09] Artur Czumaj and Andrzej Lingas. Finding a heaviest vertex-weighted triangle is not harder than matrix multiplication. *SIAM J. Comput.*, 39(2):431–444, 2009. 1

[CL15] Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 31–40, 2015. 2, 10, 11, 18, 20, 21, 44, 46, 51, 65

[CM16] Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1650–1669, 2016. 43

[CMWW19] Marek Cygan, Marcin Mucha, Karol Węgrzycki, and Michał Włodarczyk. On problems equivalent to (min,+)-convolution. *ACM Trans. Algorithms*, 15(1):1–25, 2019. 2

[CVX21] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Algorithms, Reductions and Equivalences for Small Weight Variants of All-Pairs Shortest Paths. In *Proc. 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 47:1–47:21, 2021. 6, 7, 9, 10, 16, 22, 23, 24, 26, 27, 28, 29, 30, 49

[CVX22] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Hardness for triangle problems under even more believable hypotheses: Reductions from real APSP, real 3SUM, and OV. In *Proc. 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1501–1514, 2022. 6, 8, 37, 42

[CW21] Timothy M. Chan and R. Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms*, 17(1):2:1–2:14, 2021. Preliminary version in SODA 2016. 23

[CY14] Keren Cohen and Raphael Yuster. On minimum witnesses for Boolean matrix multiplication. *Algorithmica*, 69(2):431–442, 2014. 1, 5

[DJW19] Ran Duan, Ce Jin, and Hongxun Wu. Faster algorithms for all pairs non-decreasing paths problem. In *Proc. 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 48:1–48:13, 2019. 4

[DL21] Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. *ACM Trans. Comput. Theory*, 13(2):8:1–8:24, 2021. 2

[DLM20] Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proc. 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2201–2211, 2020. 2

[DLV20] Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In *Proc. 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 774–785, 2020. 34

[DP09] Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–391, 2009. 4

[FM71] Michael J. Fischer and Albert R. Meyer. Boolean matrix multiplication and transitive closure. In *Proc. 12th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 129–131, 1971. 8

[Fre76] Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. 4

[GH22] Younan Gao and Meng He. Faster Path Queries in Colored Trees via Sparse Matrix Multiplication and Min-Plus Product. In *Proc. 30th Annual European Symposium on Algorithms (ESA)*, pages 59:1–59:15, 2022. 7, 24, 25

[GJ21] Pawel Gawrychowski and Wojciech Janczewski. Conditional lower bounds for variants of dynamic LIS. *CoRR*, abs/2102.11797, 2021. 26

[Gow01] William Timothy Gowers. A new proof of Szemerédi's theorem. *Geom. Funct. Anal.*, 11:465–588, 2001. 44, 47, 52, 65

[GP18] Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018. Preliminary version in FOCS 2014. 4

[GPVX21] Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster monotone min-plus product, range mode, and single source replacement paths. In *Proc. 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 75:1–75:20, 2021. 2, 7, 21, 63

[GR20] Oded Goldreich and Guy N. Rothblum. *Worst-Case to Average-Case Reductions for Subclasses of P*, pages 249–295. Springer, 2020. 34

[GU18] Paweł Gawrychowski and Przemysław Uznański. Towards unified approximate pattern matching for Hamming and L_1 distance. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 62:1–62:13, 2018. 8, 49

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. 2, 7, 26

[HU17] Chloe Ching-Yun Hsu and Chris Umans. On multidimensional and monotone k-SUM. In *Proc. 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 50:1–50:13, 2017. 11

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. 43

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004. 2

[JX22] Ce Jin and Yinzhan Xu. Tight dynamic problem lower bounds from generalized BMM and OMv. In *Proc. 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1515–1528, 2022. 7, 24, 25

[KL05] Miroslaw Kowaluk and Andrzej Lingas. LCA queries in directed acyclic graphs. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 241–248, 2005. 1, 5

[KL21] Miroslaw Kowaluk and Andrzej Lingas. Quantum and approximation algorithms for maximum witnesses of Boolean matrix products. In *Proc. 7th Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM)*, pages 440–451, 2021. 1, 5

[KMS05] Danny Krizanc, Pat Morin, and Michiel H. M. Smid. Range mode and range median queries on lists and trees. *Nord. J. Comput.*, 12(1):1–17, 2005. 24, 25

[Kos89] S. Rao Kosaraju. Efficient tree pattern matching. In *Proc. 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 178–183, 1989. 5

[KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016. 44, 46

[KPS17] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. On the fine-grained complexity of one-dimensional dynamic programming. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 21:1–21:15, 2017. 2

[LLV19] Rio LaVigne, Andrea Lincoln, and Virginia Vassilevska Williams. Public-key cryptography in the fine-grained setting. In *Proc. 39th Annual International Cryptology Conference (CRYPTO)*, pages 605–635, 2019. 34

[Lov17] Shachar Lovett. Additive combinatorics and its applications in theoretical computer science. *Theory Comput.*, 8:1–55, 2017. 44

[LPV20] Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. Monochromatic triangles, intermediate matrix products, and convolutions. In *Proc. 11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 53:1–53:18, 2020. 1, 3, 5, 6, 7, 61

[LU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1029–1046, 2018. 1, 20, 23, 27

[LUWG19] Karim Labib, Przemyslaw Uznanski, and Daniel Wolleb-Graf. Hamming distance completeness. In *Proc. 30th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 128, page 14, 2019. 4, 38, 41

[LWW18] Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1236–1252, 2018. 6

[Mat91] Jiří Matoušek. Computing dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278, 1991. 4, 14, 17, 23, 30

[Mer78] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978. 34

[NP85] Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Comment. Math. Univ. Carol.*, 026(2):415–419, 1985. 60

[Păt10] Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 603–610, 2010. 9, 34

[Sch79] Arnold Schönhage. On the power of random access machines. In *Proc. 6th Colloquium on Automata, Languages and Programming (ICALP)*, pages 520–529, 1979. 42

[Sch15] Tomasz Schoen. New bounds in Balog–Szemerédi–Gowers Theorem. *Combinatorica*, 35:695–701, 2015. 47, 65

[Sei95] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995. 1, 3, 15, 16, 51

[SSV05] Benny Sudakov, Endre Szemerédi, and Van Vu. On a question of Erdös and Moser. *Duke Math. J.*, 129:129–155, 2005. 44, 47, 48

[SYZ11] Asaf Shapira, Raphael Yuster, and Uri Zwick. All-pairs bottleneck paths in vertex weighted graphs. *Algorithmica*, 59(4):621–633, 2011. 1, 5

[SZ99] Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proc. 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 605–614, 1999. 6

[Tak98] Tadao Takaoka. Subcubic cost algorithms for the all pairs shortest path problem. *Algorithmica*, 20(3):309–318, 1998. 4

[TV06] Terence Tao and Van Vu. *Additive Combinatorics*. Cambridge University Press, 2006. 44, 47

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. 2

[Vas10] Virginia Vassilevska Williams. Nondecreasing paths in a weighted graph or: How to optimally read a train schedule. *ACM Trans. Algorithms*, 6(4):70:1–70:24, 2010. 4

[Vas15] Virginia Vassilevska Williams. Problem 2 on problem set 2 of CS367, October 15, 2015. 4, 38, 41

[Vas18] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018. 8

[Vio11] Emanuele Viola. *Selected Results in Additive Combinatorics: An Exposition*. Number 3 in Graduate Surveys. Theory of Computing Library, 2011. 44, 47

[VW09] Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 455–464, 2009. 20

[VW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. 51st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 645–654, 2010. 1

[VW13] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. Preliminary version in STOC 2009. 9, 31, 34

[VW18] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. Preliminary version in FOCS 2010. 1, 2, 9, 17, 30, 33, 34, 60, 61, 62

[VWWY15] Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding four-node subgraphs in triangle time. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1671–1680, 2015. 2

[VWY07] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All-pairs bottleneck paths for general graphs in truly sub-cubic time. In *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 585–589, 2007. 4

[VWY10] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding heaviest H-subgraphs in real weighted graphs, with applications. *ACM Trans. Algorithms*, 6(3):44:1–44:23, 2010. 1, 5

[VX20a] Virginia Vassilevska Williams and Yinzhan Xu. Monochromatic triangles, triangle listing and APSP. In *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 786–797, 2020. 1, 5, 6

[VX20b] Virginia Vassilevska Williams and Yinzhan Xu. Truly subcubic min-plus product for less structured matrices, with applications. In *Proc. 31st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 12–29, 2020. 2, 7, 21, 24, 25, 63

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. Preliminary version in ICALP 2004. 43

[Wil18] R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. Preliminary version in STOC 2014. 3, 4, 23, 34

[Yus09] Raphael Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 950–957, 2009. 14, 23

[Zwi99] Uri Zwick. All pairs lightest shortest paths. In *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 61–69, 1999. 7

[Zwi02] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. 1, 3, 5, 23, 28, 49

A Still More Equivalences Between Counting and Detection Problems

A.1 Exact k-Clique and Minimum k-Clique

Let G be the input graph of an Exact-k-Clique or #Exact-k-Clique instance, and let w be the weight function of the graph. For every set $I \subseteq V(G)$ of size $k-1$, we use W_I to denote the set of j where $I \cup \{j\}$ forms a k-clique whose edge weights sum up to the required value t.

Theorem A.1. *If* Exact-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #Exact-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$*

Proof. Similar as before, by well-known techniques [VW18], given a #Exact-k-Clique instance on a graph G with n nodes and a target t, we can use the $O(n^{k-\varepsilon})$ time algorithm for Exact-k-Clique to list up to $n^{0.99}$ witnesses for every set I of $k-1$ nodes, in $O(n^{k-\varepsilon''})$ time for some $\varepsilon'' > 0$.

Then we enumerate all possible subsets $J \subseteq V(G)$ of size $k-3$. For each J, we can reduce the problem of counting witnesses for sets $I \supseteq J$ of size $k-1$ to a #AE-Exact-Tri instance (G', t') in a standard way [NP85]. The set of nodes of G' corresponds to $V(G) \setminus J$. Let the edge weight between i_1 and i_2 be $w'(i_1, i_2) = 2w(i_1, i_2) + \sum_{j \in J}(w(j, i_1) + w(j, i_2))$. Also, let t' be $2t - 2\sum_{\substack{j_1, j_2 \in J \\ j_1 < j_2}} w(j_1, j_2)$. It is

not difficult to verify that (i_1, i_2, i_3) forms an exact triangle in G' if and only if $J \cup \{i_1, i_2, i_3\}$ forms an exact k-clique in G. Thus, the number of witnesses of (i_1, i_2) in G' equals the number of witnesses of $J \cup \{i_1, i_2\}$ in G. We then proceed similar to the proof of Theorem 4.2. If the number of witnesses of (i_1, i_2) in G' is less than $n^{0.99}$, then we have already listed all of their witnesses in the $O(n^{k-\varepsilon''})$ time step. For (i_1, i_2) that has at least $n^{0.99}$ witnesses, we can find a set S of size $\widetilde{O}(n^{0.01})$ that intersects with each of W_{i_1,i_2} in $\widetilde{O}(n^{2.99})$ time, and then apply Lemma 4.1 to compute the witness count for these (i_1, i_2) pairs in $\widetilde{O}(|S| \cdot n^{(3+\omega)/2}) \leq O(n^{2.70})$ time.

Finally, summing up W_I for every distinct set I of $k-1$ nodes gives the total exact triangle count of G. The overall running time for the #Exact-k-Clique instance is thus $\widetilde{O}(n^{k-\varepsilon''} + n^{k-3} \cdot (n^{2.99} + n^{2.70})) = \widetilde{O}(n^{k-\min\{\varepsilon'', 0.01\}})$. □

We can similarly show that #Min-k-Clique reduces to Min-k-Clique. Since the proof is essentially the same, we omit the proof of the following theorem for conciseness.

Theorem A.2. *If* Min-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #Min-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$*

We then reduce Min-k-Clique to #Min-k-Clique.

Theorem A.3. *If* #Min-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* Min-k-Clique *for n-node graphs has an $O(n^{k-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$*

Proof. Let G be the input graph for a Min-k-Clique instance. Without loss of generality, we can assume G is a k-partite graph on node parts $V_1 \cup \cdots \cup V_k$.

We first multiply all the edge weights of G by a large enough number $M \geq 10k \cdot n^k$. Then for each $i \in [k]$, $v_i \in V_i$, we add $v_i \cdot n^{i-1}$ to all edges adjacent to v_i. After these transformations, there will be a unique minimum weight k-clique in the graph. Furthermore, this k-clique must also be a minimum weight k-clique in the original graph. We denote this minimum weight k-clique by $(u_1, \ldots, u_k) \in V_1 \times \cdots \times V_k$.

Then for each $i \in [k]$ and each $p \in [\lceil \log(n) \rceil]$, we do the following. Let $G^{(i,p)}$ be a copy of the graph (after the weight changes), and we duplicate all nodes $v \in V_i$ whose p-th bit in its binary representation is 1. We use the assumed #Min-k-Clique algorithm the count the number of minimum weight k-cliques in in $G^{(i,p)}$. If the number of minimum weight k-clique in $G^{(i,p)}$ is 2, then we know the p-th bit of u_i is 1; otherwise, the p-th bit of u_i is 0.

After all $k\lceil \log(n) \rceil$ rounds, we can recover $(u_1, \ldots, u_k)$, and thus compute the weight of the minimum weight k-clique in the original graph. □

A.2 Monochromatic Convolution

Theorem A.4. *If* Mono-Convolution *for length n arrays has an $O(n^{1.5-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then* #Mono-Convolution *for length n arrays has an $O(n^{1.5-\varepsilon'})$ time randomized algorithm for some $\varepsilon' > 0$*

Proof. If Mono-Convolution has an $O(n^{1.5-\varepsilon})$ time algorithm, then 3SUM has a truly subquadratic time algorithm [LPV20], and consequently All-Nums-3SUM has a truly subquadratic time algorithm [VW18]. Furthermore, by Theorem 8.10, #All-Nums-3SUM has an $O(n^{2-\varepsilon''})$ time algorithm for some $\varepsilon'' > 0$.

Thus, it suffices to reduce #Mono-Convolution to #All-Nums-3SUM. Lincoln, Polak, and Vassilevska W. [LPV20] showed that a truly subquadratic time algorithm for All-Nums-3SUM implies a truly sub-$n^{1.5}$ time algorithm for Mono-Convolution. It is not difficult to check that their reduction preserves the number of solutions, and thus works for the counting versions as well. □

A.3 All-Pairs Shortest Paths

Theorem A.5. *If* APSP *for n-node graphs with positive edge weights has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then $\#_{\mathrm{mod}\, U}$APSP for n-node graphs with positive edge weights has an $O(n^{3-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$, for any $\widetilde{O}(1)$-bit integer $U \geq 2$.*

Proof. Let (A, A') and (B, B') be two pairs of $n \times n$ matrices where the entries of A' and B' are $\widetilde{O}(1)$-bit integers. We define two "funny" matrix products (one of them was defined in the proof of Theorem 5.6). If $(C, C') = (A, A') \oplus (B, B')$, then $C_{ij} = \min(A_{ij}, B_{ij})$, and $C'_{ij} = [A_{ij} = C_{ij}]A'_{ij} + [B_{ij} = C_{ij}]B'_{ij}$, where $[\cdot]$ denotes the indicator function. Clearly, we can compute $(A, A') \oplus (B, B')$ in $\widetilde{O}(n^2)$ time.

Claim A.6. *Suppose* APSP *on n-node graphs with positive edge weights has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, then we can compute $(A, A') \otimes (B, B')$ in $O(n^{3-\varepsilon''})$ time for some $\varepsilon'' > 0$ where the entries of A' and B' are $\widetilde{O}(1)$-bit integers.*

Proof. If APSP on n-node graphs with positive edge weights has an $O(n^{3-\varepsilon})$ time algorithm, then so does Min-Plus-Product [VW18]. By Theorem 8.2, #Min-Plus-Product also has a truly subcubic time algorithm. It remains to reduce computing $(A, A') \otimes (B, B')$ to #Min-Plus-Product.

For $p \in [O(\log n)]$, let $A^{(p)}$ be the matrix where $A^{(p)}_{ij} = A_{ij}$ if the p-th bit of A'_{ij} is 1, and $A^{(p)}_{ij} = \infty$ otherwise. We can similarly define $B^{(p)}$. Also, let J be the $n \times n$ matrix whose entries are all 1. It is then not difficult to verify that

$$(A, A') \otimes (B, B') = \bigoplus_{p,q} (A^{(p)}, 2^p J) \otimes (B^{(q)}, 2^q J).$$

To compute each term in the above "sum", say $(C^{(p,q)}, C'^{(p,q)}) = (A^{(p)}, 2^p J) \otimes (B^{(q)}, 2^q J)$, we first use the assumed Min-Plus-Product algorithm to compute $C^{(p,q)} = A^{(p)} \star B^{(q)}$ in $O(n^{3-\varepsilon})$ time. Then $C'^{(p,q)}_{ij}$ is exactly the number of witnesses for $C^{(p,q)}$ in the previous Min-Plus product, multiplied by 2^{p+q}, so we can use the truly subcubic time algorithm for #Min-Plus-Product to compute $C'^{(p,q)}$. □

As showed in the proof of Theorem 5.6, #APSP reduces to $O(\log n)$ instances of the funny matrix product; also, now we can mod all entries in A', B' by U after each funny matrix product to keep them $\widetilde{O}(1)$-bit integers. Thus, by Claim A.6, $\#_{\mathrm{mod}\, U}$APSP has an $\widetilde{O}(n^{3-\varepsilon''})$ time algorithm assuming APSP can be solved in truly subcubic time. □

Theorem A.7. *If $\#_{\mathrm{mod}\, c}$APSP for n-node graphs with positive edge weights has an $O(n^{3-\varepsilon})$ time algorithm for some $\varepsilon > 0$, where $c \geq 2$ is some $\widetilde{O}(1)$-bit integer, then* APSP *for n-node graphs with positive edge weights has an $O(n^{3-\varepsilon'})$ time algorithm for some $\varepsilon' > 0$,*

Proof. Suppose there is an $O(n^{3-\varepsilon})$ time algorithm for $\#_{\mathrm{mod}\, c}$APSP for n-node graphs positive edge weights, then there is also an $O(n^{3-\varepsilon})$ time algorithm for counting the number of witnesses modulo c for Min-Plus-Product for $n \times n$ matrices, by following the standard reduction from Min-Plus-Product to APSP [VW18].

Then by essentially the same proof to the proof of Theorem 8.3, there exists an $\widetilde{O}(n^{3-\varepsilon})$ time algorithm for Min-Plus-Product. Finally, there is a truly subcubic time algorithm for APSP since APSP and Min-Plus-Product are subcubically equivalent [VW18]. □

B Bounded-Difference or Monotone Min-Plus Product from the Triangle Decomposition Theorem

In this appendix, we note that our Triangle Decomposition Theorem (Theorem 5.1) implies a truly subcubic algorithm for bounded-difference Min-Plus product. Existing algorithms [BGSV16, VX20b, GPVX21, CDX22, CDXZ22] are faster, but our presentation is simpler, and so is interesting from the pedagogical perspective in our opinion. In a way, the Triangle Decomposition Theorem clarifies conceptually why a subcubic algorithm is possible (if we don't care too much about optimizing the exponent in the running time).

Theorem B.1. *There is a truly subcubic algorithm for computing the Min-Plus product of two integer $n \times n$ matrices A and B satisfying the* bounded difference property*, i.e., $|A_{i,k+1} - A_{ik}| \leq c_0$ for all i, k and $|B_{k+1,j} - B_{kj}| \leq c_0$ for all k, j for some constant c_0.*

Proof. Let $C = A \star B$ denote the matrix that we want to compute. Let ℓ be a parameter, and let D be the set of all indices in $[n]$ that are divisible by ℓ. Let $\ell' = 2c_0\ell + 1$. We first compute $\tilde{C}_{ij} = \min_{k \in D}(\lceil A_{ik}/\ell' \rceil + \lceil B_{kj}/\ell' \rceil)$ for every $i, j \in [n]$. By brute force, this takes $O(n^3/\ell)$ time. Note that $\tilde{C}_{ij} \geq C_{ij}/\ell'$. For each $r \in [\ell]$, $i, j \in [n]$ and $k \in D$, let $A^{(r)}_{ik} = A_{i,k+r} - \lceil A_{ik}/\ell' \rceil \ell'$ and $B^{(r)}_{kj} = B_{k+r,j} - \lceil B_{kj}/\ell' \rceil \ell'$. Note that $A^{(r)}_{ik}, B^{(r)}_{kj} \in [\pm O(c_0\ell)]$.

To compute $C = A \star B$, we use the following formula:

$$\begin{aligned} C_{ij} &= \min_{k \in D,\, r \in [\ell]} (A_{i,k+r} + B_{k+r,j}) \\ &= \min_{\Delta \in \{0,1,2\}} \left((\tilde{C}_{ij} + \Delta)\ell' + \min_{r \in [\ell],\, k \in D:\, \lceil A_{ik}/\ell' \rceil + \lceil B_{kj}/\ell' \rceil = \tilde{C}_{ij} + \Delta} (A^{(r)}_{ik} + B^{(r)}_{kj}) \right). \end{aligned}$$

The reason for restricting the range of Δ to $\{0, 1, 2\}$ is this: if $\lceil A_{ik}/\ell' \rceil + \lceil B_{kj}/\ell' \rceil \geq \tilde{C}_{ij} + 3$, then $A_{ik}/\ell' + B_{kj}/\ell' \geq \tilde{C}_{ij} + 1$, and so $A_{i,k+r} + B_{k+r,j} \geq A_{ik} + B_{kj} - 2c_0\ell > \tilde{C}_{ij}\ell'$; but we know that the true value of C_{ij} is at most $\tilde{C}_{ij}\ell'$.

To evaluate the above formula, let's fix $\Delta \in \{0, 1, 2\}$. We want to compute

$$C'_{ij} := \min_{r \in [\ell],\, k \in D:\, \lceil A_{ik}/\ell' \rceil + \lceil B_{kj}/\ell' \rceil = \tilde{C}_{ij} + \Delta} (A^{(r)}_{ik} + B^{(r)}_{kj}).$$

Initialize C'_{ij} to ∞. Consider the tripartite graph with nodes $\{u_i : i \in [n]\}$, $\{x_k : k \in D\}$, and $\{v_j : j \in [n]\}$, where $u_i x_k$ has weight $\lceil A_{ik}/\ell' \rceil$, and $x_k v_j$ has weight $\lceil B_{kj}/\ell' \rceil$, and $u_i v_j$ has weight $-\tilde{C}_{ij} - \Delta$. Apply Theorem 5.1 to get subgraphs $G^{(\lambda)}$ and a set R in $\widetilde{O}(n^3/s + s^2 n^2)$ time.

We first examine each triangle $u_i x_k v_j \in R$ and each $r \in [\ell]$ and reset C'_{ij} to $A^{(r)}_{ik} + B^{(r)}_{kj}$ if it is smaller than the current value. This takes $\widetilde{O}(\ell \cdot (n^3/\ell)/s) = \widetilde{O}(n^3/s)$ time.

Next, for each λ and each $r \in [\ell]$, we compute $\min_{k \in D:\, u_i x_k, x_k v_j \in G^{(\lambda)}} (A'_{ik} + B'_{kj})$ and reset C'_{ij} to this value if it is smaller, for every $u_i v_j \in G^{(\lambda)}$. This reduces to a Min-Plus product instance on integers in $[\pm O(c_0\ell)]$ and takes $\widetilde{O}(c_0 \ell n^\omega)$ time for each λ and r. The total time is $\widetilde{O}(s^3 \cdot \ell \cdot c_0 \ell n^\omega) = \widetilde{O}(c_0 \ell^2 s^3 n^\omega)$.

The overall running time is $\widetilde{O}(n^3/\ell + n^3/s + c_0\ell^2 s^3 n^\omega)$. Setting $\ell = s = n^{(3-\omega)/6}$ gives a bound of $\widetilde{O}(n^{(15+\omega)/6})$ for constant c_0 (which is improvable by using rectangular matrix multiplication). □

The above algorithm easily extends to the case where we allow $O(n^{2-\delta})$ exceptional pairs (i,k) to violate the bounded difference property $|A_{i,k+1} - A_{ik}| \le c_0$, and similarly $O(n^{2-\delta})$ exceptional pairs (k,j) to violate the bounded difference property $|B_{k+1,j} - B_{kj}| \le c_0$. We just need to add an extra cost of $O(\ell n^{2-\delta} \cdot n)$. The total time bound $\widetilde{O}(\ell n^{3-\delta} + n^3/\ell + n^3/s + c_0\ell^2 s^3 n^\omega)$ remains truly subcubic for an appropriate choice of ℓ and s.

The case of matrices with monotone rows/columns and integer entries in $[n]$ easily reduce to the bounded difference case with a nonconstant $c_0 = n^\delta$ and $O(n^{2-\delta})$ exceptional pairs. So, we also get a truly subcubic algorithm (with an appropriate choice of δ and a slightly worse final exponent) for the monotone case.

C More on BSG

In this appendix, we show that the proof of Theorem 10.3 can be modified to hold not just for indexed sets, but also for arbitrary subsets A and C of an abelian group, if we ignore the construction time. This requires a more clever argument in the "low-frequency" case. (The proof of Theorem 10.5 can also be modified in a similar way.)

Theorem C.1. (Simpler BSG Covering) *Given subsets A and C of size n of an abelian group and a parameter s, there exist a collection of $\ell = \widetilde{O}(s^3)$ subsets $A^{(1)}, \ldots, A^{(\ell)} \subseteq A$, and a set R of $\widetilde{O}(n^2/s)$ pairs in $A \times A$, such that*

(i) $\{(a,b) \in A \times A : a - b \in C\} \subseteq R \cup \bigcup_\lambda (A^{(\lambda)} \times A^{(\lambda)})$, *and*

(ii) $\sum_\lambda |A^{(\lambda)} - A^{(\lambda)}| = \widetilde{O}(s^2 n^{3/2})$.

Proof.

- **Few-witnesses case.** First add $\{(a,b) \in A \times A : \ a - b \in C, \ \mathrm{pop}_A(a-b) \le n/s\}$ to R. The number of pairs added to R is $\widetilde{O}(n \cdot n/s)$.
- **Many-witnesses case.** Let $F = \{h : \mathrm{pop}_A(h) > n/r\}$. Then $|F| = O(rn)$, since the total popularity is n^2. By adding extra elements to F, we may assume that $|F| = \Theta(rn)$. Pick a random subset $H \subseteq F$ of size $c_0 sr \log n$ for a sufficiently large constant c_0.
  - **Low-frequency case.** Add the following to R:

    $$\{(a,b) \in A \times A : \ |\{(a',b') \in A \times A : a - a' = b - b' \notin F\}| > n/(2s)\}.$$

    Since for each (a,a') with $a - a' \notin F$, there are at most n/r choices of (b,b') satisfying $a - a' = b - b'$, the number of pairs added to R is $\widetilde{O}(\frac{n^2 \cdot n/r}{n/(2s)}) = \widetilde{O}(n^2/s)$ by choosing $r := s^2$.
  - **High-frequency case.** For each $h \in H$, add the following subset to the collection:

    $$A^{(h)} = \{a \in A : \ a - h \in A\}.$$

    The number of subsets is $\widetilde{O}(sr) = \widetilde{O}(s^3)$.

Correctness. To verify (i), consider a fixed pair $(a,b) \in A \times A$ with $a - b \in C$. If $\mathrm{pop}_A(a-b) \le n/s$, then $(a,b) \in R$ due to the "few-witnesses" case. So assume $\mathrm{pop}_A(a-b) > n/s$. Furthermore, assume that $|\{(a',b') \in A \times A : a - a' = b - b' \notin F\}| \le n/(2s)$, for otherwise $(a,b) \in R$ due to the "low-frequency"

case. There are at least n/s pairs $(a', b') \in A \times A$ with $a' - b' = a - b$, which also satisfy $a - a' = b - b'$ by Fredman's trick. Among them, there are at least $n/(2s)$ pairs $(a', b') \in A \times A$ with $a - a' = b - b' \in F$. For $h = a - a' = b - b'$, we have $(a, b) \in A^{(h)} \times A^{(h)}$. So, the probability that $(a, b) \in A^{(h)} \times A^{(h)}$ for a random $h \in F$ is $\Omega(\frac{n/(2s)}{rn}) = \Omega(1/(sr))$. Thus, $(a, b) \in \bigcup_{h \in H}(A^{(h)} \times A^{(h)})$ w.h.p. for a random subset $H \subset F$ of size $c_0 sr \log n$.

To verify (ii), consider a fixed h. The set $A^{(h)} - A^{(h)}$ is equal to $\{c : \exists a, b, a', b' \in A,\ c = a - b,\ a - h = a',\ b - h = b'\}$. Let $Y_c = \{a \in A : a - c \in A\}$. Then $A^{(h)} - A^{(h)}$ is contained in $\{c : \exists a \in Y_c \text{ and } a - h \in Y_c\}$. The expected size of $A^{(h)} - A^{(h)}$ for a random $h \in F$ is thus bounded by

$$\sum_c \min\{|Y_c| \cdot |Y_c|/|F|,\ 1\} \ \le\ O(n^2/\sqrt{|F|}) \ =\ O(n^{3/2}/\sqrt{r}),$$

since $\sum_c |Y_c| = O(n^2)$.

The expected total sum $\sum_{h \in H} |A^{(h)} - A^{(h)}|$ is bounded by $\widetilde{O}(sr \cdot n^{3/2}/\sqrt{r}) = \widetilde{O}(s^2 n^{3/2})$. □

Chan and Lewenstein [CL15] posed the following interesting combinatorial question:

> Given sets A, B, C in an abelian group of size n, we want to cover $\{(a, b) \in A \times B : a + b \in C\}$ by bicliques $A^{(\lambda)} \times B^{(\lambda)}$, so as to minimize the cost function $\sum_\lambda |A^{(\lambda)} + B^{(\lambda)}|$. Prove worst-case bounds on the minimum cost as a function of n.

They observed that Theorem 10.2 implies an $O(n^{13/7})$ upper bound for this problem. Theorem C.1 implies an improved $\widetilde{O}(n^{11/6})$ upper bound (as we can use "singleton" bicliques to cover R and choose s to minimize $\widetilde{O}(n^2/s + s^2 n^{3/2})$).

If we just want an analog of the BSG Theorem that extracts a single subset rather than constructs a cover (in the style of Theorem 10.1), the proof of the above theorem becomes even simpler (the "few-witnesses" and "low-frequency" cases may be skipped):

Theorem C.2. (Simpler BSG) *Given subsets A and C of size n of an abelian group and a parameter s, if $|\{(a, b) \in A \times A : a - b \in C\}| \ge n^2/s$, then there exists a subset $A' \subseteq A$ of size $\Omega(n/s)$, such that*

$$|A' - A'| \ =\ O(s^{1/2} n^{3/2}).$$

Proof. Let $F = \{x : \text{pop}_A(x) > n/(2s)\}$. Then $|F| \ge n/(2s)$, because otherwise, $\sum_{c \in C} \text{pop}_A(c) < |F|n + n^2/(2s) < n^2/s$, contradicting the stated assumption.

Pick a random $h \in F$ and let $A^{(h)} = \{a \in A : a - h \in A\}$ as before. Then $|A^{(h)}| = \text{pop}_A(h) > n/(2s)$.

By the same argument as before, the expected size of $A^{(h)} - A^{(h)}$ for a random $h \in F$ is bounded by $O(n^2/\sqrt{|F|})$, which is now $O(s^{1/2} n^{3/2})$. □

The above $O(s^{1/2} n^{3/2})$ bound is better than the known $O(s^5 n)$ bound (Theorem 10.1) when $s \gg n^{1/9}$, though the latter holds only for the bichromatic sum set setting.[9]

Along the same lines, we can obtain the following combinatorial result from the Triangle Decomposition Theorem, which might be of independent interest:

[9] Bounds of the form $O(s^{O(1)} n)$ were also known in the setting of monochromatic difference sets, but direct comparisons require care: many previous work such as [Gow01, Sch15] started from a related but different assumption, namely, that the *energy* $|\{(a, b, a', b') \in A \times A \times A \times A : a + b = a' + b'\}|$ is at least n^3/s' for some parameter s'. This parameter s' is not identical to the parameter s in Theorem C.2, though they are related polynomially (or quadratically).

Theorem C.3. *Given a real-weighted tripartite graph G with n nodes, and given a parameter s, if G contains $\Omega(n^3/s)$ zero-weight triangles, then there exists a subgraph G' with $\widetilde{\Omega}(n^3/s^4)$ triangles (and thus $\widetilde{\Omega}(n^2/s^4)$ edges) such that all triangles in G' are zero-weight triangles.*

Proof. Apply Theorem 5.1 with s replaced by s'. The size of R is $\widetilde{O}(n^3/s')$, which can be made less than $n^3/(2s)$ for some choice of $s' = \widetilde{\Theta}(s)$. Then there exists $G^{(\lambda)}$ with $|\text{Triangles}(G^{(\lambda)})| = \widetilde{\Omega}((n^3/s) \cdot (1/s^3))$. □