Simpler Reductions from Exact Triangle

Timothy M. Chan*
UIUC
tmc@illinois.edu

Yinzhan Xu†
MIT
xyzhan@mit.edu

Abstract

In this paper, we provide simpler reductions from Exact Triangle to two important problems in fine-grained complexity: Exact Triangle with Few Zero-Weight 4-Cycles and All-Edges Sparse Triangle.

Exact Triangle instances with few zero-weight 4-cycles was considered by Jin and Xu [STOC 2023], who used it as an intermediate problem to show 3SUM hardness of All-Edges Sparse Triangle with few 4-cycles (independently obtained by Abboud, Bringmann and Fischer [STOC 2023]), which is further used to show 3SUM hardness of a variety of problems, including 4-Cycle Enumeration, Offline Approximate Distance Oracle, Dynamic Approximate Shortest Paths and All-Nodes Shortest Cycles. We provide a simple reduction from Exact Triangle to Exact Triangle with few zero-weight 4-cycles. Our new reduction not only simplifies Jin and Xu's previous reduction, but also strengthens the conditional lower bounds from being under the 3SUM hypothesis to the even more believable Exact Triangle hypothesis. As a result, all conditional lower bounds shown by Jin and Xu [STOC 2023] and by Abboud, Bringmann and Fischer [STOC 2023] using All-Edges Sparse Triangle with few 4-cycles as an intermediate problem now also hold under the Exact Triangle hypothesis.

We also provide two alternative proofs of the conditional lower bound of the All-Edges Sparse Triangle problem under the Exact Triangle hypothesis, which was originally proved by Vassilevska Williams and Xu [FOCS 2020]. Both of our new reductions are simpler, and one of them is also deterministic—all previous reductions from Exact Triangle or 3SUM to All-Edges Sparse Triangle (including Pătraşcu's seminal work [STOC 2010]) were randomized.

1 Introduction

In this paper, we present several new simpler reductions between core problems in fine-grained complexity, which lead to simpler conditional lower bound proofs for a plethora of problems, and at the same time strengthening some of these results by weakening the hypothesis assumed.

In the Exact Triangle problem (also known as Zero-Weight Triangle), we are given an n-node weighted graph G whose edge weights are from $[\pm n^{O(1)}]$,[1] and we need to determine whether it contains a triangle whose edge weights sum up to 0. It is one of the key problems in the field of fine-grained complexity, primarily due to its relationship between the 3SUM hypothesis and the APSP hypothesis, which are among the three central hypotheses in fine-grained complexity (the other one is the Strong Exponential Time hypothesis). It is known that under either the 3SUM hypothesis [VW13] or the APSP hypothesis [VW18], the Exact Triangle problem requires $n^{3-o(1)}$ time. As a result, the following Exact Triangle hypothesis holds as long as at least one of the 3SUM hypothesis and the APSP hypothesis holds.

*Supported by NSF Grant CCF-2224271.
†Partially supported by NSF Grants CCF-2129139 and CCF-2330048 and BSF Grant 2020356.
[1]For a nonnegative integer N, $[N]$ denotes $\{1, \ldots, N\}$ and $[\pm N]$ denotes $\{-N, \ldots, N\}$.

Hypothesis 1 (Exact Triangle hypothesis)**.** In the Word-RAM model with $O(\log n)$-bit words, Exact Triangle on n-node weighted graphs whose edge weights are from $[\pm n^{O(1)}]$ requires $n^{3-o(1)}$ time.

By designing reductions from the Exact Triangle problem to other problems, we can obtain conditional lower bounds that hold under the Exact Triangle hypothesis, which in turn also hold under either the 3SUM hypothesis or the APSP hypothesis. This makes it desirable to design reductions from the Exact Triangle problem.

Exact Triangle with few zero-weight 4-cycles. Recently, Jin and Xu [JX23] reduced 3SUM to 4-Cycle Enumeration, Offline Approximate Distance Oracle, Dynamic Approximate Shortest Paths and All-Nodes Shortest Cycles. As one of their intermediate steps, they showed 3SUM-hardness of a variant of Exact Triangle on instances with a small number of zero-weight 4-cycles. This part of their reduction is very technically involved, and it uses heavy machineries such as the Balog–Szemerédi–Gowers theorem from additive combinatorics. Independently, Abboud, Bringmann and Fischer [ABF23] also showed 3SUM hardness of several graph problems including 4-Cycle Enumeration and Offline Approximate Distance Oracle. They do not use aforementioned variant of Exact Triangle in their reduction, but their reduction also uses tools from additive combinatorics. (See Figure 1 (right).)

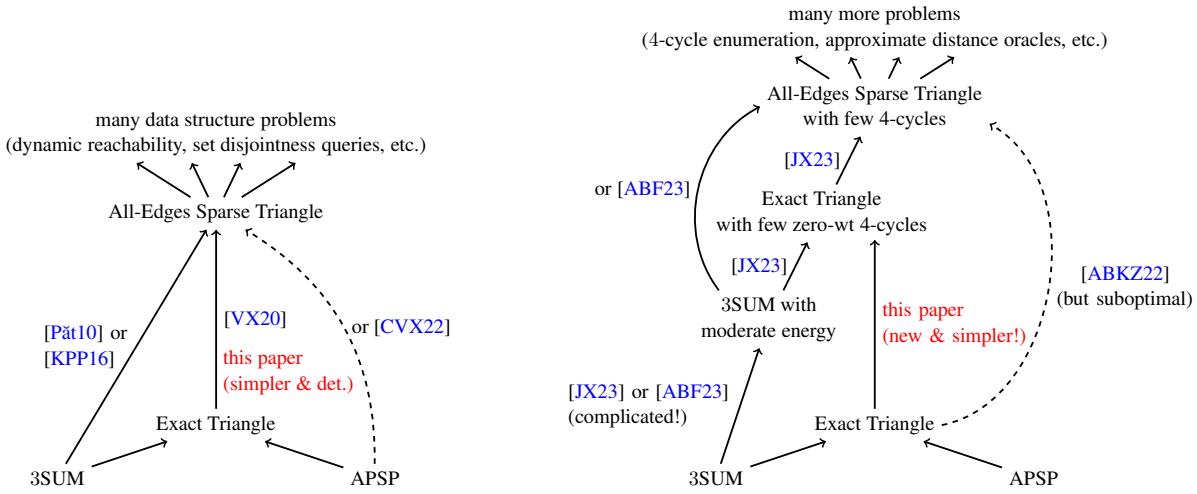

Figure 1: Summary of reductions. (See the cited references for precise definitions of some of the intermediate problems in the right diagram, which may have slight variations in different papers.)

In this paper, we *vastly* simplify Jin and Xu's reduction by designing a reduction from Exact Triangle (instead of 3SUM in their case) to the aforementioned variant of Exact Triangle, by a proof that is under two pages. Moreover, with our new reduction, the lower bounds for 4-Cycle Enumeration, Offline Approximate Distance Oracle, Dynamic Approximate Shortest Paths and All-Nodes Shortest Cycles in [JX23] now also hold under the Exact Triangle hypothesis; previously, they only hold under the 3SUM hypothesis. (We remark that our new reduction does not imply Exact Triangle hardness for problems such as Sidon Set Verification and 4-LDT, which were shown to be 3SUM hard in [JX23], because the reduction from 3SUM to them does not use the variant of Exact Triangle as an intermediate problem.)

Our new reduction is inspired by the recent work of Chan, Vassilevska Williams and Xu [CVX23]. In particular, they showed versions of the Balog–Szemerédi–Gowers theorem that have simpler proofs inspired

by the famous "Fredman's trick"[2] [Fre76]. We borrow their intuition that Fredman's trick can help simplifying proofs involving the Balog–Szemerédi–Gowers theorem, but our reduction does not use their results directly.

All-Edges Sparse Triangle. We also present two simplifications of the known reduction from Exact Triangle to the All-Edges Sparse Triangle problem [VX20], in which we are given an m-edge graph, and for each edge in the graph, we need to decide whether it is in a triangle. This problem can be solved in $O(m^{2\omega/(\omega+1)})$ time [AYZ97], where $\omega < 2.372$ is the square matrix multiplication exponent [DWZ23]. When $\omega = 2$, this running time becomes $O(m^{4/3})$. In a landmark paper, Pătraşcu [Păt10] showed that this problem requires $m^{4/3-o(1)}$ time under the 3SUM hypothesis, matching the running time if $\omega = 2$.[3] Vassilevska Williams and Xu [VX20] later strengthened this lower bound to make it also work under the Exact Triangle hypothesis. Another paper by Chan, Vassilevska Williams and Xu [CVX22] showed that conditional lower bounds for All-Edges Sparse Triangle also hold under the real-valued variants of fine-grained hypotheses, including an $m^{4/3-o(1)}$ time lower bound under the Real APSP hypothesis, an $m^{5/4-o(1)}$ time lower bound under the Real Exact Triangle hypothesis, and an $m^{6/5-o(1)}$ time lower bound under the Real 3SUM hypothesis. Since Pătraşcu's work [Păt10], All-Edges Sparse Triangle has been used as a bridge to show conditional lower bounds for a wide variety of dynamic problems, such as dynamic reachability and dynamic shortest paths [Păt10, AV14], and also static data structure problems, such as set disjointness and set intersection [KPP16] and certain types of generalized range queries [DKPW20]. (See Figure 1 (left).)

We design two simple new reductions from Exact Triangle to All-Edges Sparse Triangle, simplifying Vassilevska Williams and Xu's reduction [VX20]. In particular, our reductions do not use any hash function and are considerably shorter than theirs.

Our first reduction is adapted from Chan, Vassilevska Williams and Xu's reduction from real-valued problems to All-Edges Sparse Triangle [CVX22]. Even though they were only able to obtain a non-tight $m^{5/4-o(1)}$ time lower bound for All-Edges Sparse Triangle under the Real Exact Triangle hypothesis, we show that their ideas can be implemented more efficiently for the integer-valued version of Exact Triangle. Similar to [CVX22], our new reduction also relies on Fredman's trick. An additional advantage of this reduction is that it is deterministic, while the reduction in [VX20] is randomized. Note that Pătraşcu's original reduction [Păt10] from 3SUM to All-Edges Sparse Triangle and Kopelowitz, Pettie and Porat's later reduction [KPP16] were also randomized (and crucially relied on hashing). Our approach gives the first deterministic reduction from 3SUM to All-Edges Sparse Triangle as well, yielding the same $m^{4/3-o(1)}$ conditional lower bound.

Our second reduction borrows an idea from a recent conditional lower bound of Sparse Triangle Detection under the strong 3SUM hypothesis by Jin and Xu [JX23]. Sparse Triangle Detection is a problem closely-related to All-Edges Sparse Triangle, where now we only need to decide if a given graph contains a triangle or not. Jin and Xu [JX23] showed an $m^{6/5-o(1)}$ lower bound for Sparse Triangle Detection under the Strong 3SUM hypothesis, which states that 3SUM instances on n integers from $[\pm n^2]$ requires $n^{2-o(1)}$ time. Previously, Abboud, Bringmann, Khoury and Zamir [ABKZ22] showed an $m^{4/3-o(1)}$ lower bound for Sparse Triangle Detection under an unbalanced bounded-weight variant of the Exact Triangle hypothesis.

The method for our second reduction can actually lead to more consequences, including an $m^{5/4-o(1)}$ lower bound of the All-Nodes Sparse Triangle problem (deciding, for each node, whether it is in a triangle) under the Exact Triangle hypothesis, and an $m^{9/7-o(1)}$ lower bound of the Sparse Triangle Detection

[2]The simple observation that $a + b = a' + b'$ is equivalent to $a - a' = b' - b$.

[3]Technically, Pătraşcu proved the conditional lower bound for the problem of listing m triangles in an m-edge graph, but this problem is equivalent up to polylog factors to the All-Edges Sparse Triangle problem under randomized reductions [DKPW20].

problem under a (more natural) balanced bounded-weight variant of the Exact Triangle hypothesis.

2 Exact Triangle with Few Zero-Weight 4-Cycles

In this section, we will consider Exact Triangle instances where the edge weights w_{ij} can be different from w_{ji}, and the weight of a triangle (i, j, k) is defined as $w_{ij}+w_{jk}+w_{ki}$. The following property was considered by [JX23].

Property 2.1. In an Exact Triangle instance on edge set E and weight function w,

- **Antisymmetry**: For every $(i, j) \in E$, it holds that $(j, i) \in E$ and $w_{ij} = -w_{ji}$;
- **Few zero-weight 4-cycles**: The number of (i, j, k, ℓ) for which $(i, j), (j, k), (k, \ell), (\ell, i) \in E$ and $w_{ij} + w_{jk} + w_{kl} + w_{li} = 0$ is at most n^3 (we will call each (i, j, k, ℓ) a zero-weight 4-cycle, and here, i, j, k, ℓ are not necessarily distinct).

Jin and Xu [JX23] gave a long and complicated proof that Exact Triangle instances on graphs with Property 2.1 require $n^{3-o(1)}$ time under the 3SUM hypothesis. We present a much simpler proof of the same result under the Exact Triangle hypothesis (from which one can then obtain conditional lower bounds for a host of other problems, including 4-Cycle-Enumeration and Offline Approximate Distance Oracle, as shown in [JX23]):

Theorem 2.2. *Under the Exact Triangle hypothesis, Exact Triangle instances on n-node graphs with Property 2.1 requires $n^{3-o(1)}$ time.*

Proof. Suppose for the sake of contradiction that Exact Triangle instances on n-node graphs with Property 2.1 can be solved in $O(n^{3-\varepsilon})$ time for some $\varepsilon > 0$.

Given any Exact Triangle instance on $G = (V, E)$ (where $w_{ij} = w_{ji}$ for all edges ij), we can copy V three times to V_1, V_2, V_3, add E between any two copies of V, and direct the edges in the direction of $V_1 \to V_2 \to V_3 \to V_1$. This way, we can add in the edges in the reverse directions with negated weights. Hence, the Antisymmetry property is satisfied.

Let $\delta > 0$ be some constant to be fixed later. We consider the following two cases:

The number of zero-weight 4-cycles is at most $n^{4-\delta}$. In this case, we randomly partition the vertices to reduce the problem to multiple smaller Exact Triangle instances, reducing the overall number of zero-weight 4-cycles significantly while preserving all zero-weight triangles.

Let $x > 0$ be a constant. We create n^{1-x} buckets of vertices, and put each $v \in V$ into one of the buckets chosen uniformly at random. For every triple of buckets, we create an Exact Triangle instance on the subgraph induced by the vertices in the three buckets. Each instance contains $O(n^x)$ nodes w.h.p.[4] It suffices to solve all $O(n^{3-3x})$ instances of these small Exact Triangle instances.

For any triple of buckets B_1, B_2, B_3, and any zero-weight 4-cycle (i, j, k, ℓ) in the original graph with distinct i, j, k, ℓ, (i, j, k, ℓ) appears in the small instance induced by B_1, B_2, B_3 with probability $\frac{1}{n^{4-4x}}$. Therefore, in expectation, the number of zero-weight 4-cycles with distinct nodes in the instance induced by buckets B_1, B_2, B_3 is $O(n^{4x-\delta})$. By sampling random distinct tuples of nodes i, j, k, ℓ in the small instance and testing whether they form a zero-weight 4-cycle, we can estimate the number of zero-weight 4-cycles with distinct vertices up to n^{2x} additive error w.h.p. in $\widetilde{O}(\frac{n^{4x}}{n^{2x}}) = \widetilde{O}(n^{2x})$ time. If the estimated count is at

[4] With high probability, i.e., probability $1 - O(1/n^c)$ for an arbitrarily large constant c.

least $2n^{2x}$, we use brute-force to solve this small instance in $O(n^{3x})$ time. Note that this case happens with probability $O(\frac{n^{4x-\delta}}{n^{2x}}) = O(n^{2x-\delta})$ by Markov's inequality. Otherwise, the number of zero-weight 4-cycles in the instance is at most $O(n^{3x})$ w.h.p. ($O(n^{2x})$ from zero-weight 4-cycles with distinct nodes, and $O(n^{3x})$ from zero-weight 4-cycles with some repeated nodes). We can add some isolated nodes in the small instance to increase its number n' of nodes (n' will still be $O(n^x)$), so that the number of zero-weight 4-cycles can be bounded by $(n')^3$. Thus, we can call an algorithm for Exact Triangle with Property 2.1 on this small instance, which runs in $O(n^{(3-\varepsilon)x})$ time.

Overall, the expected running time is

$$O\left(n^{3-3x} \cdot \left(n^{2x-\delta} \cdot n^{3x} + n^{(3-\varepsilon)x}\right)\right) = O(n^{3+2x-\delta} + n^{3-3\varepsilon x}),$$

which is $O(n^{3-\frac{3\varepsilon}{2+3\varepsilon}\delta})$ by setting $x = \delta/(2+3\varepsilon)$.

The number of zero-weight 4-cycles is more than $\Omega(n^{4-\delta})$. In this case, there exists $(i_0, k_0) \in E$ that participates in $\Omega(n^{2-\delta})$ zero-weight 4-cycles. We can find such an (i_0, k_0) by testing all $O(n^2)$ pairs. To test a given pair (i_0, k_0), we can estimate the number of (i, k) such that (i_0, k_0, i, k) form a zero-weight 4-cycle with up to $o(n^{2-\delta})$ additive error w.h.p., by sampling random pairs (i, k), in $\widetilde{O}(\frac{n^2}{n^{2-\delta}}) = \widetilde{O}(n^\delta)$ time. The total time is $\widetilde{O}(n^{2+\delta})$.

For each k, define $r_k := w_{i_0,k} - w_{i_0,k_0}$. Define $E_0 := \{(i,k) \in E : w_{ik} - w_{i,k_0} = r_k\}$. Note that E_0 is exactly the set of (i, k) where (i_0, k_0, i, k) forms a zero-weight 4-cycle, because

$$w_{ik} - w_{i,k_0} = w_{i_0,k} - w_{i_0,k_0} \iff w_{i_0,k_0} + w_{k_0,i} + w_{ik} + w_{k,i_0} = 0$$

by the Antisymmetry property (and Fredman's trick). Thus, $|E_0| = \Omega(n^{2-\delta})$ w.h.p.

We can test whether there is a zero-weight triangle using at least one edge in E_0 as follows. Say $w_{ik} + w_{kj} + w_{ji} = 0$ for some edge $(i, k) \in E_0$. The condition is equivalent to $w_{i,k_0} + r_k + w_{kj} + w_{ji} = 0$, which further is equivalent to $w_{i,k_0} + w_{ji} = -w_{kj} - r_k$. We create two matrices A, B where $A_{ij} := w_{i,k_0} + w_{ji}$ and $B_{jk} := -w_{kj} - r_k$. Now the problem becomes the following: for every $(i, k) \in E_0$, test whether there exists j such that $A_{ij} = B_{jk}$. This can be solved by computing the so-called *Equality Product* of the two matrices A and B, which is known to be in $O(n^{(3+\omega)/2})$ time [Mat91] (see [CVX23] for more applications of Equality Product combined with Fredman's trick). After this step, we can remove E_0 from E, and by symmetry, we can also remove all edges that are reverses to the edges in E_0. Then we can iteratively solve the Exact Triangle instance on the remaining graph.

Putting things together. The overall algorithm is as follows. It determines which case we are at by estimating the number of zero-weight 4-cycles in $\widetilde{O}(n^\delta)$ time. We can call the second case at most $O(n^\delta)$ expected number of times because each time we remove a subset of $\Omega(n^{2-\delta})$ edges w.h.p. Also, we can call the first case at most once. Thus, the overall running time is

$$\widetilde{O}(n^{2\delta} + n^{3-\frac{3\varepsilon}{2+3\varepsilon}\delta} + n^\delta \cdot (n^{2+\delta} + n^{(3+\omega)/2})),$$

which is truly subcubic by setting δ properly (say $\delta = 0.1$). This would violate the Exact Triangle hypothesis, so Exact Triangle instances on n-node graphs with Property 2.1 requires $n^{3-o(1)}$ time. □

3 All-Edges Sparse Triangle: First Alternative Proof

In this section, we present the first alternative proof for the lower bound of All-Edges Sparse Triangle under the Exact Triangle hypothesis (from which one can then obtain conditional lower bounds for many data structure problems [Păt10, AV14, KPP16]). This proof is adapted from Chan, Vassilevska Williams and Xu's reduction from real-valued problems to All-Edges Sparse Triangle [CVX22]. We combine it with an extra simple idea where we iteratively halve the weights. The new reduction is entirely deterministic, unlike all previous reductions for All-Edges Sparse Triangle, and avoids any form of hashing.

Theorem 3.1. *Under the Exact Triangle hypothesis, All-Edges Sparse Triangle requires $m^{4/3-o(1)}$ time.*

Proof. Without loss of generality, we assume that the Exact-Triangle instance is on a tripartite graph with node partitions A, B, C where $|A| = |B| = n$ and $|C| = \sqrt{n}$ and with weight function w. Such an instance requires $n^{2.5-o(1)}$ time under the Exact-Triangle hypothesis. We also solve a stronger version, where for every $(a, b) \in A \times B$, we need to find a $c_{ab} \in C$ where $|w_{ab} + w_{b,c_{ab}} + w_{a,c_{ab}}| \leq 3$, if one exists. This version is stronger because if we multiply all initial edge weights by 4, then finding such (a, b, c_{ab}) is equivalent to finding zero-weight triangles.

First, we recursively solve the instance where the edge weights w_{ij} are replaced with $\lfloor w_{ij}/2 \rfloor$. The base case is when all the edge weights are in $[\pm O(1)]$. In this case, the instance can be solved in $\widetilde{O}(n^\omega)$ time, by enumerating all triples of edge weights that sum up to $[\pm 3]$, and then find triangles with the corresponding edge weights using known methods for finding triangles in unweighted graphs [AGMN92].

Suppose for every (a, b), we already found k_{ab} (via recursion) where $|\lfloor w_{ab}/2 \rfloor + \lfloor w_{b,k_{ab}}/2 \rfloor + \lfloor w_{a,k_{ab}}/2 \rfloor| \leq 3$ if one exists. Then we need to find c_{ab} with $|w_{ab} + w_{b,c_{ab}} + w_{a,c_{ab}}| \leq 3$ if one exists.

Let $L_{k,\delta} := \{(a, b) \in A \times B : k_{ab} = k, w_{ab} + w_{bk} + w_{ka} = \delta\}$ for $k \in C$ and $\delta \in \{-6, \ldots, 9\}$. If some $L_{k,\delta}$ has size greater than $n^{1.5}$, we split it to $O(\frac{|L_{k,\delta}|}{n^{1.5}})$ subsets of size $O(n^{1.5})$. The total number of subsets will remain $O(|C| + \frac{n^2}{n^{1.5}}) = O(n^{0.5})$. We also split the set C into n^ε equally-sized subsets $\{C_s\}_{s \in [n^\varepsilon]}$. For each subset $L = L_{k,\delta}$, each $\delta' \in \{-3, \ldots, 3\}$, and each subset C_s, we create the following All-Edges Sparse Triangle instance:

- The vertex sets are A', B', U, where A' is a copy of A, B' is a copy of B, and U is $C_s \times [\pm n^{O(1)}]$ (sparsely represented).
- For every $(a, b) \in L$, add an edge between $a \in A'$ and $b \in B'$.
- For every $(a, c) \in A \times C_s$, add an edge between a and $(c, w_{ac} - w_{ak} + \delta - \delta')$.
- For every $(b, c) \in B \times C_s$, add an edge between b and $(c, w_{bk} - w_{bc})$.

Then we detect for every edge (a, b) in the graph, whether it is contained in a triangle. If so, we exhaustively search c_{ab} in C_s (for each edge (a, b), we only perform this exhaustive search once).

Correctness. First, we show that for any triangle $(a, b, (c, u))$ that is in any of the created All-Edges Sparse Triangle instances, c is a valid candidate for c_{ab}. This is because $w_{ac} - w_{ak} + \delta - \delta' = u = w_{bk} - w_{bc}$, which implies $w_{ab} + w_{bc} + w_{ca} = \delta' + (w_{ab} + w_{bk} + w_{ak} - \delta)$ (by Fredman's trick). Now, $w_{ab} + w_{bk} + w_{ak} - \delta = 0$ because any edge (a, b) in this particular All-Edges Sparse Triangle instance has $w_{ab} + w_{bk} + w_{ak} = \delta$ by the definition of $L_{k,\delta}$. Therefore, $w_{ab} + w_{bc} + w_{ca} = \delta' \in [\pm 3]$ and thus c is a valid candidate for c_{ab}.

Then we show that if $c \in C_s$ is a valid candidate for c_{ab}, then $(a, b, (c, u))$ for some u is a triangle in some of the created All-Edges Sparse Triangle instances. Because c is a valid candidate for c_{ab}, and

$$\frac{1}{2}(w_{ab} + w_{bc} + w_{ca}) - \frac{3}{2} \leq \lfloor w_{ab}/2 \rfloor + \lfloor w_{bc}/2 \rfloor + \lfloor w_{ca}/2 \rfloor \leq \frac{1}{2}(w_{ab} + w_{bc} + w_{ca}).$$

we have $-3 \leq \lfloor w_{ab}/2 \rfloor + \lfloor w_{bc}/2 \rfloor + \lfloor w_{ac}/2 \rfloor \leq \frac{3}{2} \leq 3$, so k_{ab} must also exist (in particular, it can take value c, but it can be any other valid value as well). Therefore, $(a, b) \in L_{k_{ab},\delta}$, where $\delta = w_{ab} + w_{a,k_{ab}} + w_{b,k_{ab}} \in \{-6, \ldots, 9\}$. Also, let $\delta' = w_{ab} + w_{ac} + w_{bc}$. The edge (a, b) will be included in the All-Edges Sparse Triangle instance created for $L = L_{k_{ab},\delta}$, δ' and C_s. Because $w_{ab}+w_{ak}+w_{bk}-\delta = 0 = w_{ab}+w_{ac}+w_{bc}-\delta'$, we have $w_{ac} - w_{ak} + \delta - \delta' = w_{bk} - w_{bc}$, so we will find a triangle for the edge (a, b) (in particular, $(a, b, (c, w_{bk} - w_{bc}))$ is a valid triangle).

Running time. Suppose the running time for All-Edges Sparse Triangle is $O(m^{4/3-\varepsilon})$ for some $\varepsilon > 0$, then the overall running time can be bounded as follows. The time for handling the base case is $\widetilde{O}(n^\omega)$. In each recursive step, we need to solve $O(n^{0.5+\varepsilon})$ instances of All-Edges Sparse Triangle, which run in $\widetilde{O}(n^{0.5+\varepsilon} \cdot (n^{1.5})^{4/3-\varepsilon}) = \widetilde{O}(n^{2.5-0.5\varepsilon})$ time. For every edge (a, b), we also need to spend time performing the exhaustive search, which run in $\widetilde{O}(n^2 \cdot n^{0.5-\varepsilon}) = O(n^{2.5-\varepsilon})$ time. Overall, the running time is $\widetilde{O}(n^\omega + n^{2.5-0.5\varepsilon})$, which would violate the Exact Triangle hypothesis. □

4 All-Edges Sparse Triangle: Second Alternative Proof

In this section, we present the second alternative proof for the lower bound of All-Edges Sparse Triangle under the Exact-Triangle hypothesis, which is based on an idea from the recent conditional lower bound of Sparse Triangle Detection under the strong 3SUM hypothesis by Jin and Xu [JX23].

Let EXACT-TRI$(n \mid W)$ denote the time complexity of the problem of detecting a zero-weight triangle in an n-node tripartite graph with edge weights in $[\pm W]$. Let EXACT-TRI-LIST$(n, t \mid W)$ denote the time complexity of the problem of listing all t zero-weight triangles in an n-node tripartite graph with edge weights in $[\pm W]$ (note that the value of t is not given in advance).

Let SPARSE-TRI(m), AE-SPARSE-TRI(m), and AN-SPARSE-TRI(m) denote the time complexity of the Sparse Triangle Detection, All-Edges Sparse Triangle, and All-Nodes Sparse Triangle problems respectively for an m-edge unweighted tripartite graph.[5] Let SPARSE-TRI-LIST(m, t) denote the time complexity of the problem of listing all t triangles in an m-edge unweighted tripartite graph.

Our main idea for reducing Exact Triangle to Sparse Triangle Listing is encapsulated in the proof of the following lemma, which is short and simple.

Lemma 4.1. EXACT-TRI-LIST$(n, t \mid W) \leq O(1) \cdot$ SPARSE-TRI-LIST$(n^2 W^{1/3}, t)$.

Proof. Exact Triangle Listing for weights in $[\pm W]$ reduces to $O(1)$ instances of Exact Triangle Listing for weights in $[\pm q]^3$ with $q := W^{1/3}$, since we can map any number $x = x_1 q^2 + x_2 q + x_3 \in [\pm W]$ with $x_1 \in [\pm q]$, $x_2, x_3 \in \{0, \ldots, q-1\}$ to the triple $\varphi(x) = (x_1, x_2, x_3) \in [\pm q]^3$; then $x + y + z = 0$ iff $\varphi(x) + \varphi(y) + \varphi(z) \in \Delta$ for a constant-size set $\Delta = \{(0,0,0), (0,-1,q), (0,-2,2q)\} + \{(0,0,0), (-1,q,0), (-2,2q,0)\}$. For every $\delta \in \Delta$, we need to solve an instance where the target weight of the triangle is δ; by subtracting δ from all edges in one of the edge parts, the target can become 0 again (this will only increase the edge weight bound by a constant factor).

[5] As a reminder, in the Sparse Triangle Detection problem, we need to determine whether a given graph contains a triangle; in the All-Nodes Sparse Triangle problem, we need to decide whether each node in a given graph is contained in a triangle.

Let G be an n-node tripartite graph with node partition A, B, C and edge weights in $[\pm q]^3$. Construct a new unweighted tripartite graph G' with node partition $A \times [\pm q]^2$, $B \times [\pm q]^2$, $C \times [\pm q]^2$:

- Add an edge between $(a, x_1, z_3) \in A \times [\pm q]^2$ and $(b, y_3, x_2) \in B \times [\pm q]^2$ iff ab is an edge in G with weight $(x_1, x_2, -y_3 - z_3)$.
- Add an edge between $(b, y_3, x_2) \in B \times [\pm q]^2$ and $(c, z_2, y_1) \in C \times [\pm q]^2$ iff bc is an edge in G with weight $(y_1, -x_2 - z_2, y_3)$.
- Add an edge between $(c, z_2, y_1) \in C \times [\pm q]^2$ and $(a, x_1, z_3) \in A \times [\pm q]^2$ iff ca is an edge in G with weight $(-x_1 - y_1, z_2, z_3)$.

The number of edges in G' is $O(n^2 q) = O(n^2 W^{1/3})$ (for example, to count the number of edges added in the first bullet, note that there are $O(q)$ ways to write a given number in $[\pm q]$ as $-y_3 - z_3$). Furthermore, $(a, x_1, z_3), (b, y_3, x_2), (c, z_2, y_1)$ form a triangle in G' iff a, b, c form a triangle in G whose edges ab, bc, ca have weights (x_1, x_2, x_3), (y_1, y_2, y_3), (z_1, z_2, z_3) summing to $(0, 0, 0)$. Thus, we can solve Exact Triangle Listing on G by solving Triangle Listing on G'. □

For the above lemma to be effective, the universe size W needs to be subcubic. To lower the universe size, we use a standard hashing trick described in the next lemma. We only need the simplest type of linear hash functions (mod p for a random prime p), and this is the only place in this reduction where randomization is used.

Lemma 4.2. $\text{EXACT-TRI}(n \mid n^{O(1)}) \leq \widetilde{O}(1) \cdot \text{EXACT-TRI-LIST}(n, \widetilde{O}(t) \mid n^3/t)$ *for any* t.

Proof. Pick a random prime p in $[n^3/(2t), n^3/t]$. To solve Exact Triangle for a given weighted graph, we take the weights mod p and solve Exact Triangle Listing to find all triangles with weights congruent to $0 \bmod p$. The answer is yes iff one of these triangles actually has zero weight. For a fixed triangle with nonzero weight, the probability that its weight is congruent to $0 \bmod p$ is $O((t/n^3)\log n)$, since there are $O(\frac{n^3/t}{\log(n^3/t)})$ primes in $[n^3/(2t), n^3/t]$, and a fixed number in $[n^{O(1)}]$ has at most $O(\frac{\log n}{\log(n^3/t)})$ prime divisors in this range. Thus, the expected number of triangles with nonzero weights congruent to $0 \bmod p$ (i.e., number of false positives) is $O(t \log n)$. Thus, if the process takes more than $\text{EXACT-TRI-LIST}(n, \widetilde{O}(t) \mid n^3/t)$ time, we can terminate and infer that the answer is yes with probability $\Omega(1)$ (or w.h.p. by repeating logarithmically many times). □

The main result now immediately follows:

Theorem 4.3. *Under the Exact Triangle hypothesis, All-Edges Sparse Triangle requires* $m^{4/3-o(1)}$ *time.*

Proof. We first show that for listing all t triangles in an m-edge graph when $t = \Theta(m)$ requires $m^{4/3-o(1)}$ time, under the Exact Triangle hypothesis.

Combining Lemmas 4.1 and 4.2 gives

$$\begin{aligned}
\text{EXACT-TRI}(n \mid n^{O(1)}) &\leq \widetilde{O}(1) \cdot \text{SPARSE-TRI-LIST}(n^3/t^{1/3}, \widetilde{O}(t)) \\
&\leq \widetilde{O}(1) \cdot \text{SPARSE-TRI-LIST}(n^{9/4}, \widetilde{O}(n^{9/4}))) \qquad \text{by setting } t = n^{9/4} \\
&\leq \widetilde{O}(n^{3-(9/4)\varepsilon}) \qquad \text{if } \text{SPARSE-TRI-LIST}(m, m) \leq O(m^{4/3-\varepsilon}).
\end{aligned}$$

To prove hardness for All-Edges Sparse Triangle instead of Sparse Triangle Listing, we apply a known (simple) reduction from Sparse Triangle Listing to All-Edges Sparse Triangle from [DKPW20, Theorem 1.4]. □

4.1 Discussion and Further Consequences

Monte Carlo randomization is used in the proof of Lemma 4.2. With a little more effort, it is possible to modify it to require only Las Vegas randomization.[6] However, it is not clear how to completely derandomize this reduction without slow-down. If a deterministic reduction is desired, see our alternative proof in Section 3.

Compared to Vassilevska Williams and Xu's reduction [VX20] from Exact Triangle to Sparse Triangle Listing or All-Edges Sparse Triangle, the new reduction is not only simpler, but also better in a technical sense. Their reduction [VX20, Theorem 3.4] (also randomized) basically shows that

$$\textsc{Exact-Tri}(n \mid n^{O(1)}) \leq \widetilde{O}(n^{2\rho}) \cdot \textsc{Sparse-Tri-List}(n^{2-\rho}, n^{3-3\rho}).$$

Setting $\rho = 1/2$ gives $\textsc{Exact-Tri}(n \mid n^{O(1)}) \leq \widetilde{O}(n) \cdot \textsc{Sparse-Tri-List}(n^{3/2}, n^{3/2})$, which is $\widetilde{O}(n^{3-(3/2)\varepsilon})$ if $\textsc{Sparse-Tri-List}(m, m) \leq O(m^{4/3-\varepsilon})$. In contrast, the proof of Theorem 4.3 provides a slightly better bound of $\widetilde{O}(n^{3-(9/4)\varepsilon})$.

As an application, we derive a new conditional lower bound for All-Nodes Sparse Triangle, a natural problem "in between" All-Edges Sparse Triangle and Sparse Triangle Detection. (The previous reduction could also be used in combination of Lemma 4.5 below, but the dependence on ε would again be worse.)

Let $\textsc{Sparse-Tri-List}(m, t, d_{\max})$ denote the time complexity of the problem of listing all t triangles in an m-edge unweighted tripartite graph with maximum degree $d_{\max}$.

Lemma 4.4. $\textsc{Exact-Tri-List}(n, t \mid W) \leq \widetilde{O}(1) \cdot \textsc{Sparse-Tri-List}(n^2 W^{1/3}, t, O(n/W^{1/3}))$.

Proof. As in the proof of Lemma 4.1, we may assume the edge weights are in $[\pm q]^3$ with $q := W^{1/3}$. Pick a random function $h : A \cup B \cup C \to [q]^3$. For each edge uv, replace the weight w_{uv} by $\hat{w}_{uv} = w_{uv} + h(v) - h(u) \in [\pm 2q]^3$. Clearly, the weight of each triangle remains unchanged. Now, apply the same construction of G' as in the proof of Lemma 4.1 (with q adjusted by a factor of 2).

Consider a fixed node $(a, x_1, z_3) \in A \times [\pm 2q]^2$, and a fixed $b \in B$. Then (a, x_1, z_3) has a (unique) neighbor (b, y_3, x_2) in G' for some y_3 and x_2 only if the first component of $\hat{w}_{ab}$ is equal to x_1, i.e., $h(b)$ is equal to $h(a) - w_{ab} + x_1$ in the first component. This holds with probability at most $1/q$, since the first component of $h(b)$ is uniformly distributed in $[q]$ conditioned to any fixed $h(a)$. For a fixed $c \in C$, the probability that (a, x_1, z_3) has a (unique) neighbor (c, z_2, y_1) for some z_2 and y_1 is similarly at most $1/q$. Thus, the expected degree of (a, x_1, z_3) is at most $n/q = n/W^{1/3}$.

We now remove all nodes of degree more than $6n/W^{1/3}$ from G'. The probability that a fixed node is removed is at most $1/6$ by Markov's inequality. Thus, the probability that any fixed triangle is eliminated is at most $1/2$. To solve Exact Triangle Listing on G, we solve Sparse Triangle Listing on G', repeating logarithmically many times to ensure a high survival probability bound per triangle. □

Lemma 4.5. $\textsc{Sparse-Tri-List}(m, t, d_{\max}) \leq \widetilde{O}(\textsc{AN-Sparse-Tri}(m + t d_{\max}))$.

Proof. We describe a recursive algorithm for Sparse Triangle Listing: Given an m-edge tripartite graph G with node partition A, B, C, arbitrarily divide A into two subsets A_1 and A_2 of size $|A|/2$. For each $j \in [2]$, run the All-Nodes Sparse Triangle oracle to identify the subset B_j of nodes in B that participate in triangles

[6] For example, in the proof of Lemma 4.2, when the process takes more than the allotted time, we return "not sure" instead of yes. If there is at most one zero-weight triangle, the probability of returning "not sure" is small. By standard random-sampling arguments, we can run the algorithm on logarithmically many subgraphs where one of the subgraphs has a unique zero-weight triangle with good probability if there exists a zero-weight triangle. If "not sure" is reported for all such subgraphs, we restart from scratch.

in the subgraph of G induced by $A_j \cup B \cup C$; then recursively solve the Sparse Triangle Listing problem on the subgraph of G induced by $A_j \cup B_j \cup C$.

Observe that each edge in $A \times B$ or $A \times C$ participates in one subproblem per level in the recursion. Furthermore, an edge $(b, c) \in B \times C$ participates in at most $f(b)$ subproblems per level, where $f(b)$ denotes the number of triangles that b participates in. Thus, the sum of the number of edges over all the subproblems per level is $O(m + \sum_{b \in B} f(b) d_{\max}) = O(m + t d_{\max})$. There are $O(\log n)$ levels of recursion. By superlinearity of AN-SPARSE-TRI$(\cdot)$, the entire algorithm runs in $O(\text{AN-SPARSE-TRI}(m + t d_{\max}) \log n)$ time. □

Theorem 4.6. *Under the Exact Triangle hypothesis, All-Nodes Sparse Triangle requires $m^{5/4-o(1)}$ time.*

Proof. Combining Lemmas 4.2, 4.4 and 4.5 gives

$$\begin{aligned}
\text{EXACT-TRI}(n \mid n^{O(1)}) &\leq \widetilde{O}(1) \cdot \text{AN-SPARSE-TRI}(n^3/t^{1/3} + t^{4/3}) \\
&\leq \widetilde{O}(1) \cdot \text{AN-SPARSE-TRI}(n^{12/5}) \qquad \text{by setting } t = n^{9/5} \\
&\leq \widetilde{O}(n^{3-(12/5)\varepsilon}) \qquad \text{if AN-SPARSE-TRI}(m) \leq O(m^{5/4-\varepsilon}).
\end{aligned}$$

□

Unfortunately, a nontrivial lower bound for Sparse Triangle Detection is still out of reach under the Exact Triangle hypothesis. However, we have the following result if we assume the *Strong Exact Triangle hypothesis*, namely, that Exact Triangle for integer weights in $[\pm n]$ requires $n^{3-o(1)}$ time. (Recent work has considered similarly defined Strong APSP hypothesis, Strong 3SUM hypothesis, and so on [CVX23, JX23].) The proof here only needs Lemma 4.1, so the reduction is even simpler (and deterministic).

Theorem 4.7. *Under the Strong Exact Triangle hypothesis, Sparse Triangle Detection requires $m^{9/7-o(1)}$ time.*

Proof. The analog of Lemma 4.1 for detection instead of listing implies that EXACT-TRI$(n \mid W) \leq O(1) \cdot$ SPARSE-TRI$(n^2 W^{1/3})$, and so EXACT-TRI$(n \mid n) \leq O(\text{SPARSE-TRI}(n^{7/3}))$. □

Abboud et al. [ABKZ22] were the first to prove conditional lower bounds for Sparse Triangle Detection under a variant of the Exact Triangle hypothesis (which they call "the Strong Zero-Triangle Conjecture"). More precisely, they showed a better lower bound of $m^{4/3-o(1)}$ under an "unbalanced" version of the hypothesis, specifically, that Exact Triangle for a tripartite graph with n nodes in the first part, and $\sqrt{n}$ nodes in the second and third parts, and weights in $[\pm\sqrt{n}]$ requires $n^{2-o(1)}$ time. In contrast, Theorem 4.7 assumes the Strong Exact Triangle hypothesis in the balanced case, which appears more natural.

Compared to Abboud et al.'s reduction (which is extremely simple), ours is more powerful in some sense. Let EXACT-TRI$(n_1, n_2, n_3 \mid W)$ denote the time complexity of Exact Triangle for a tripartite graph with n_1, n_2, n_3 nodes in its three parts and edge weights in $[\pm W]$. They basically observed that

$$\text{EXACT-TRI}(n_1, n_2, n_3 \mid W) \leq O(1) \cdot \text{SPARSE-TRI}(n_1 n_2 + n_1 n_3 + n_2 n_3 W).$$

Setting $n_2 = n_3 = W = \sqrt{n}$ gives EXACT-TRI$(n, \sqrt{n}, \sqrt{n} \mid \sqrt{n}) \leq O(\text{SPARSE-TRI}(n^{3/2}))$, which is $O(n^{2-(3/2)\varepsilon})$ if SPARSE-TRI$(m) \leq O(m^{4/3-\varepsilon})$. In contrast, straightforward modification of the proof of Lemma 4.1 yields a more general bound:

$$\begin{aligned}
\text{EXACT-TRI}(n_1, n_2, n_3 \mid W) &\leq \min_{q_1, q_2, q_3 :\ q_1 q_2 q_3 = W} O(1) \cdot \text{SPARSE-TRI}(n_1 n_2 q_3 + n_1 n_3 q_2 + n_2 n_3 q_1) \\
&= O(1) \cdot \text{SPARSE-TRI}((n_1 n_2 n_3)^{2/3} W^{1/3} + n_1 n_2 + n_1 n_3 + n_2 n_3).
\end{aligned}$$

5 Acknowledgements

We would like to thank Ce Jin for helpful discussions.

References

[ABF23] Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-SUM lower bounds for approximate distance oracles via additive combinatorics. In *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 391–404, 2023. doi:10.1145/3564246.3585240. 2, 12, 13

[ABKZ22] Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in P via short cycle removal: Cycle detection, distance oracles, and beyond. In *Proc. 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1487–1500, 2022. doi:10.1145/3519935.3520066. 2, 3, 10

[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. doi:10.1109/SFCS.1992.267748. 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. doi:10.1109/FOCS.2014.53. 3, 6

[AYZ97] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189. 3

[CH20] Timothy M. Chan and Qizheng He. Reducing 3SUM to Convolution-3SUM. In *Proc. 3rd Symposium on Simplicity in Algorithms (SOSA)*, pages 1–7, 2020. doi:10.1137/1.9781611976014.1. 12, 13

[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. doi:10.1145/3519935.3520032. 2, 3, 6

[CVX23] Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Fredman's trick meets dominance product: Fine-grained complexity of unweighted APSP, 3SUM counting, and more. In *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 419–432, 2023. doi:10.1145/3564246.3585237. 2, 5, 10, 13

[DKPW20] Lech Duraj, Krzysztof Kleiner, Adam Polak, and Virginia Vassilevska Williams. Equivalences between triangle and range query problems. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 30–47, 2020. doi:10.1137/1.9781611975994.3. 3, 8, 13

[DWZ23] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *Proc. 64th IEEE Symposium on Foundations of Computer Science (FOCS)*, page to appear, 2023. 3

[Fre76] Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. doi:10.1137/0205006. 3

[JX23] Ce Jin and Yinzhan Xu. Removing additive structure in 3SUM-based reductions. In *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 405–418, 2023. doi:10.1145/3564246.3585157. 2, 3, 4, 7, 10, 12, 13

[KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016. doi:10.1137/1.9781611974331.ch89. 2, 3, 6, 12, 13

[Mat91] Jiří Matoušek. Computing dominances in E^n. *Inf. Process. Lett.*, 38(5):277–278, 1991. doi:10.1016/0020-0190(91)90071-O. 5

[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. doi:10.1145/1806689.1806772. 2, 3, 6, 12, 13

[VW13] Virginia Vassilevska Williams and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X. 1, 12

[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. doi:10.1145/3186893. 1

[VX20] 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. doi:10.1109/FOCS46700.2020.00078. 2, 3, 9

A A Simple Reduction from 3SUM to Sparse Triangle Listing

By combining with known reductions from 3SUM to Convolution-3SUM [Păt10, KPP16, CH20] and from Convolution-3SUM to Exact Triangle [VW13], the result in Section 4 implies a reduction from 3SUM to Sparse Triangle Listing. In this appendix, we describe a more direct modification to obtain a simple reduction from 3SUM to Sparse Triangle Listing, without going through Convolution-3SUM and Exact Triangle. The proof is very similar to the one in Section 4 (based on an idea in [JX23]), and is also similar to ideas contained in a proof by Abboud, Bringmann and Fischer [ABF23], but we feel it is worthwhile to write out the proof explicitly, since it is simpler (and more "symmetric") than Pătraşcu's original reduction [Păt10] or Kopelowitz, Pettie and Porat's later reduction [KPP16], and may have pedagogical value (being the easiest to teach).

Let $3\text{SUM}(n \mid W)$ denote the time complexity of the 3SUM problem for n numbers in $[\pm W]$. Let $3\text{SUM-LIST}(n, t \mid W)$ denote the time complexity of the problem of listing all t triples of numbers summing to 0 for n numbers in $[\pm W]$ (note that t is not given in advance).

We first adapt Lemma 4.1.

Lemma A.1. $3\text{SUM-LIST}(n, t \mid W) \leq O(1) \cdot \text{SPARSE-TRI-LIST}(nW^{1/3}, t)$.

Proof. 3SUM for numbers in $[\pm W]$ reduces to $O(1)$ instances of 3SUM for triples in $[\pm q]^3$ with $q := W^{1/3}$, by the same mapping φ.

Let A, B, C be three given sets of n triples in $[\pm q]^3$. Construct an unweighted tripartite graph G' with node partition $\{1\} \times [\pm q]^2$, $\{2\} \times [\pm q]^2$, $\{3\} \times [\pm q]^2$:

- Add edge between $(1, x_1, z_3) \in \{1\} \times [\pm q]^2$ and $(2, y_3, x_2) \in \{2\} \times [\pm q]^2$ iff $(x_1, x_2, -y_3 - z_3) \in A$.
- Add edge between $(2, y_3, x_2) \in \{2\} \times [\pm q]^2$ and $(3, z_2, y_1) \in \{3\} \times [\pm q]^2$ iff $(y_1, -x_2 - z_2, y_3) \in B$.
- Add edge between $(3, z_2, y_1) \in \{3\} \times [\pm q]^2$ and $(1, x_1, z_3) \in \{1\} \times [\pm q]^2$ iff $(-x_1 - y_1, z_2, z_3) \in C$.

The number of edges in G' is $O(nq) = O(nW^{1/3})$. Observe that $(1, x_1, z_3), (2, y_3, x_2), (3, z_2, y_1)$ form a triangle in G' iff $(x_1, x_2, x_3) \in A$, $(y_1, y_2, y_3) \in B$, $(z_1, z_2, z_3) \in C$ sum to $(0, 0, 0)$ for some x_3, y_2, z_1. Thus, we can solve 3SUM Listing on A, B, C by solving Triangle Listing on G'. □

Lemma A.2. $\textsc{3sum}(n \mid n^{O(1)}) \leq \widetilde{O}(1) \cdot \textsc{3sum-List}(n, \widetilde{O}(t) \mid n^3/t)$ *for any* t.

Proof. Similar to the proof of Lemma 4.2. □

Theorem A.3. *Under the 3SUM hypothesis, All-Edges Sparse Triangle requires* $m^{4/3-o(1)}$ *time.*

Proof. Combining Lemmas A.1 and A.2 gives

$$\begin{aligned}
\textsc{3sum}(n \mid n^{O(1)}) &\leq \widetilde{O}(1) \cdot \textsc{Sparse-Tri-List}(n^2/t^{1/3}, \widetilde{O}(t)) \\
&\leq \widetilde{O}(1) \cdot \textsc{Sparse-Tri-List}(n^{3/2}, \widetilde{O}(n^{3/2}))) \qquad \text{by setting } t = n^{3/2} \\
&\leq \widetilde{O}(n^{2-(3/2)\varepsilon}) \qquad \text{if } \textsc{Sparse-Tri-List}(m, m) \leq O(m^{4/3-\varepsilon}).
\end{aligned}$$

To prove hardness for All-Edges Sparse Triangle instead of Sparse Triangle Listing, we can again apply a known reduction from Sparse Triangle Listing to All-Edges Sparse Triangle [DKPW20]. □

We remark that the proof of Lemma A.1 is similar to a proof of Jin and Xu [JX23] that $\textsc{3sum}(n \mid n^2) \leq \widetilde{O}(1) \cdot \textsc{Sparse-Tri}(n^{5/3})$, implying an $m^{6/5-o(1)}$ lower bound for Sparse Triangle Detection under the Strong 3SUM hypothesis. In fact, it is basically a reinterpretation. The only main difference is that their proof used a different mapping φ via the Chinese remainder theorem with three primes. Their proof also used randomization, but the only reason was that they wanted the lower bound to hold for more structured graph instances with bounded maximum degree. A proof in Abboud, Bringmann and Fischer's work [ABF23] also contained similar ideas (they also used randomized hashing to bound vertex degrees).

Remark A.4. Our first approach in Section 3 can also be adapted to give a direct reduction from 3SUM to All-Edges Sparse Triangle, by combining the idea of halving the weights with the reduction from Real 3SUM to #All-Edges Sparse Triangle in [CVX23]. The resulting reduction is deterministic and completely avoids hashing, unlike all previous reductions. (Note that if we instead go through Convolution-3SUM and Exact Triangle, the part of the reduction from 3SUM to Convolution-3SUM would already need hashing [Păt10, KPP16], although this part has been derandomized [CH20].)