Deterministic 3SUM-Hardness

Nick Fischer
Weizmann Institute of Science, Rehovot, Israel

Piotr Kaliciak
Jagiellonian University in Kraków, Poland

Adam Polak
Max Planck Institute for Informatics, Saarbrücken, Germany
Jagiellonian University in Kraków, Poland

Abstract

As one of the three main pillars of fine-grained complexity theory, the 3SUM problem explains the hardness of many diverse polynomial-time problems via fine-grained reductions. Many of these reductions are either directly based on or heavily inspired by Pătraşcu's framework involving additive hashing and are thus *randomized*. Some selected reductions were derandomized in previous work [Chan, He; SOSA'20], but the current techniques are limited and a major fraction of the reductions remains randomized.

In this work we gather a toolkit aimed to derandomize reductions based on additive hashing. Using this toolkit, we manage to derandomize *almost all* known 3SUM-hardness reductions. As technical highlights we derandomize the hardness reductions to (offline) Set Disjointness, (offline) Set Intersection and Triangle Listing – these questions were explicitly left open in previous work [Kopelowitz, Pettie, Porat; SODA'16]. The few exceptions to our work fall into a special category of recent reductions based on structure-versus-randomness dichotomies.

We expect that our toolkit can be readily applied to derandomize future reductions as well. As a conceptual innovation, our work thereby promotes the theory of *deterministic 3SUM-hardness*.

As our second contribution, we prove that there is a *deterministic universe reduction* for 3SUM. Specifically, using additive hashing it is a standard trick to assume that the numbers in 3SUM have size at most n^3. We prove that this assumption is similarly valid for *deterministic* algorithms.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases 3SUM, derandomization, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.49

Related Version *Full Version*: `https://arxiv.org/abs/2310.12913`

Funding *Nick Fischer*: This work is part of the project CONJEXITY that has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101078482).

1 Introduction

In this paper we revisit the famous integer 3SUM problem, which involves checking for a given set A of n integers whether there is a triple $a, b, c \in A$ with $a + b = c$. It is well-known that the 3SUM problem can be solved in deterministic time $O(n^2)$ by a classic textbook algorithm, and it is an infamous open problem whether the running time can be substantially improved beyond quadratic time. While recent improvements only scored log-shavings [14, 38, 34, 36, 40, 20], it became a popular conjecture that there is no truly subquadratic algorithm (i.e., an algorithm running in time $O(n^{2-\epsilon})$ for some $\epsilon > 0$).

Over the course of the last decades, this conjecture has evolved into one the three main pillars of fine-grained complexity, explaining the hardness of a diverse set of polynomial-time problems via fine-grained reductions from the 3SUM problem. This trend was initiated in the 1990's by Gajentaan and Overmars [35] leading to a series of quadratic-time lower bounds for various geometric problems [35, 29, 12, 16, 32, 8, 15, 50, 11, 33, 25, 13]. Later, in 2010, Pătraşcu achieved another breakthrough [48]. Based on earlier ideas due to Baran, Demaine and Pătraşcu [14], Pătraşcu realized that the interplay between 3SUM and *additive hashing* can be exploited to derive *randomized reductions* from 3SUM to the related *Convolution 3SUM* and *Triangle Listing* problems. This result inspired an even bigger wave of 3SUM-based lower bounds for various combinatorial problems. Today it is known that the class of 3SUM-hard problems encompasses graph problems [4, 52, 3, 2, 39], string problems [5, 9], dynamic problems [48, 44, 28, 7, 24] and many more.

Randomized versus Deterministic. As noted before, many of these reductions (for non-geometric problems) use Pătraşcu's seminal reduction as a stepping stone, or at least reuse it's ideas. Consequently, a major fraction of the 3SUM-based hardness results are only achieved via *randomized* reductions. These reductions thus imply lower bounds only if we are willing to assume that the 3SUM problem requires quadratic time also for randomized algorithms.

Chan and He [22] took a first step in understanding the role of randomness in 3SUM-based hardness reductions. They proposed to replace the hash function in Pătraşcu's reduction from 3SUM to Convolution 3SUM with a deterministic (and simpler) one. This implied that Convolution 3SUM is hard even for deterministic reductions, and entails that there is no deterministic subquadratic algorithm for Convolution 3SUM under the following hypothesis:

▶ **Hypothesis 1** (Deterministic 3SUM). *For any $\epsilon > 0$ there is a constant c, such that there is no deterministic algorithm for the 3SUM problem over $[n^c]$ in time $O(n^{2-\epsilon})$.*

While their result entails derandomizations for some reductions [18, 21, 9, 52], for technical reasons it fails to address the full range of problems. Let us describe the issue in a few words. In essence, Chan and He propose to select a hash function $h(x) = x \bmod m$, where m is the product of small primes, and to select these primes one by one using the method of conditional expectations. With this idea one can easily construct a hash function h with few *collisions* ($h(a) = h(b)$) since the number of collisions is easily computable. However, for the majority of reductions we need the stronger property that there are few *false positives* ($h(a) + h(b) = h(c)$) which is more difficult to test. In the overview in Section 3 we elaborate on this in more detail.

Theory of Deterministic 3SUM-Hardness. Nevertheless, Chan and He's result marks an important first step towards a *theory of deterministic 3SUM-hardness.* Aiming at a more complete theory, in this work we study the following wide-open question:

Question 1: Can we derandomize all reductions from 3SUM?

Understanding the power of randomness is one of the central goals in theoretical computer science. However, in fine-grained complexity (and also parameterized complexity) it is common to neglect this aspect and allow for randomized algorithms and reductions by default. In light of Question 1 it would be exciting to see whether at least for 3SUM-hard problems we could gain some foundational knowledge about the necessity of randomness.

Answering Question 1 would also be particularly important if there turned out to be faster randomized than deterministic algorithms for 3SUM. This is very well possible – in fact, in the history of state-of-the-art 3SUM algorithms (with mildly subquadratic running times) the gap between randomized and deterministic algorithms was only recently closed [20].

Of course the net of 3SUM-based reductions is vast, so where should we begin to tackle Question 1? A reasonable starting point is the aforementioned Triangle Listing problem, which is one of the bottlenecks for the many still-randomized reductions. Recall that Pătraşcu [48] proved that Triangle Listing is hard under randomized reductions, and this reduction was later refined by Kopelowitz, Pettie and Porat [44]. They proposed, as appropriate intermediate problems, the (offline) Set Disjointness and (offline) Set Intersection problems, and proved that these problems are 3SUM-hard under randomized reductions. In order to answer Question 1, we should therefore focus on Set Disjointness and Set Intersection. In their paper [44], Kopelowitz et al. remark that "it would be surprising if [their] construction could be efficiently derandomized".

We also consider another question related to the role of randomness for the 3SUM problem: universe reductions. It is well known that for randomized algorithms we can assume without loss of generality that the input is over the universe $[O(n^3)]$ (simply replace each input $a \in A$ by $h(a)$ where $h : \mathbb{Z} \to [O(n^3)]$ is an appropriate additive hash function). For deterministic algorithms, on the other hand, there are no similar bounds. Chan and He [22] end their paper with the following open problem:

Question 2: Can 3SUM for arbitrary integers be reduced deterministically to 3SUM for integers bounded by $n^{O(1)}$?

Besides being an interesting question in its own right, obtaining a deterministic universe reduction would also constitute a handy tool in the design of deterministic 3SUM-based reductions, and thus contribute to our dream goal of a complete theory of deterministic 3SUM-hardness.

1.1 Our Results

Our main technical contribution is that we gather a toolset aimed to derandomize 3SUM-based reductions. It consists of two major tools: A deterministic algorithm to compute an additive hash function with few collisions (see Lemma 11), and a refined deterministic self-reduction for 3SUM (see Lemma 12). We establish these tools in Section 3 and give a short technical overview. Notably, none of these results use heavy black-box derandomization machinery, but instead are rather simple algorithms tailored to the 3SUM problem.

To our great surprise, by a combination of these tools we can completely answer Question 2, and answer Question 1 for *almost all* known 3SUM-based reductions.

Deterministic Universe Reduction. Let us start with our second driving question. We prove that the universe size can indeed be reduced to $n^{O(1)}$:

▶ **Theorem 2** (Deterministic Universe Reduction for 3SUM). *If 3SUM over $[n^3]$ can be solved in deterministic time $O(n^{2-\epsilon})$ for some $\epsilon > 0$, then 3SUM over $[U]$ can be solved in deterministic time $O(n^{2-\epsilon'} \log^c U)$ for some constants $\epsilon', c > 0$.*

It is satisfactory that the precise polynomial bound is n^3, which matches what is known in terms of randomized algorithms precisely. Interestingly though, we obtain our universe reduction not as a preprocessing step, but rather as a fine-grained reduction from 3SUM with large universe to 3SUM with small universe.

Let us note that Chan and He's original motivation to this question was replacing the dependence on U by a dependence on n (in a reduction from 3SUM to Convolution 3SUM) [22], which our result fails to achieve, because it involves $\log U$ factors in the running time. However, it is a typical assumption in fine-grained complexity theory that the problems involve weights in $[-n^c, n^c]$ (for some constant c), and in this setting the overhead is only polylogarithmic in n. In particular, Theorem 2 implies that unless the deterministic 3SUM hypothesis (Hypothesis 1) fails, there is no deterministic subquadratic-time algorithm for 3SUM over $[n^3]$. In all deterministic 3SUM-hardness reductions we can thus assume without loss of generality that the given 3SUM instance is over $[n^3]$. And indeed, for some problems [1] this assumption is necessary to recover the same lower bounds that are known via randomized reductions.

Derandomizing Almost All 3SUM-Based Reductions. Let us turn to our driving Question 1. Up to very few exceptions, we indeed manage to derandomize all remaining randomized 3SUM-based reductions. Our flagship result here is that we derandomize the reductions to Set Disjointness and Set Intersection, originally due to Kopelowitz, Pettie and Porat [44]:

▶ **Problem 3** (Set Disjointness)**.** Given sets $S_1, \ldots, S_N \subseteq [U]$ each of size at most s and a set of q queries $Q \subseteq [N]^2$, report for each query $(i, j) \in Q$ whether $S_i \cap S_j = \emptyset$.

▶ **Theorem 4** (Deterministic Set Disjointness Hardness)**.** *Let $0 \leq \alpha < 1$. Unless the deterministic 3SUM hypothesis fails, there is no deterministic algorithm for (offline) Set Disjointness with parameters $|U| = O(n^{2-2\alpha})$, $N = O(n)$, $s = O(n^{1-\alpha})$ and $q = O(n^{1+\alpha})$ that runs in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$.*

▶ **Problem 5** (Set Intersection)**.** Given sets $S_1, \ldots, S_N \subseteq [U]$ each of size at most s and a set of q queries $Q \subseteq [N]^2$, report for each query $(i, j) \in Q$ the set intersection $S_i \cap S_j$. Occasionally we specify a size threshold up to which the algorithm is supposed to list elements.

▶ **Theorem 6** (Deterministic Set Intersection Hardness)**.** *Let $0 \leq \alpha < 1$ and $0 \leq \beta \leq 1 - \alpha$. Unless the deterministic 3SUM hypothesis fails, there is no deterministic algorithm for (offline) Set Intersection with parameters $|U| = O(n^{1+\beta-\alpha})$, $N = O(n^{\frac{1}{2}+\frac{\alpha}{2}+\frac{\beta}{2}})$, $s = O(n^{1-\alpha})$ and $q = O(n^{1+\alpha})$ that in total lists up to $O(n^{2-\beta})$ elements and runs in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$.*

As announced before, Theorems 4 and 6 immediately entail various deterministic lower bounds for e.g. Triangle Listing [44], which in turn implies deterministic lower bounds e.g. for many database problems [47, 43, 17, 42].

We remark that the range of parameters for which our deterministic reduction to Set Intersection applies is limited in comparison to the original randomized reduction. While we can only treat cases where we expect to list at least $\Omega(1)$ elements per queried intersection (on average), Kopelowitz et al.'s reduction also applies to the setting where the number of listed elements is much smaller. For all tight follow-up reductions based on Set Intersection, our parameterization suffices though.

An answer to Question 1 does not only involve considering selected problems though – there are many scattered 3SUM-based reductions in the literature that have to be considered. We have made the effort to check for all these reductions (that we are aware of) whether our derandomization ideas apply. The number of papers is overwhelming – thus, in order to limit the scope of our project, we have only considered *tight* reductions. In the following subsection we summarize the landscape of 3SUM-based reductions, and summarize which reductions admit derandomizations in what way.

1.2 The Landscape of 3SUM-Based Reductions

We have depicted the complete landscape of 3SUM-based reductions (that we are aware of) in Figure 1 on Page 6. In this figure, we draw a solid arrow from P to Q if there is a deterministic (tight) reduction from P to Q, and a dotted arrow if the reduction is randomized. We have segmented the space of problems into seven regions depending on whether and how derandomizations are known. The light blue regions are deterministic prior to our work, our contribution is that we derandomize all problems in the yellow regions, and the dark blue region remains randomized.

(O) Originally Deterministic Reductions. Only a minority of 3SUM-based hardness results is deterministic out of the box – most problems rely directly or indirectly on a randomized reduction. This minority includes in particular the numerous geometric problems [35, 29, 12, 16, 32, 8, 15, 50, 11, 33, 25, 13]. These reductions usually build on simple algebraic transformations of the input. For instance, the reduction from 3SUM to the problem of finding a triple of colinear points simply creates point (a, a^3) for each input number $a \in A$. As a consequence all these reductions are deterministic.

Other two examples of similarly simple deterministic constructions are reductions to Hamming Pattern Matching under Polynomial Transformation [19], and to fully retroactive 3SUM data structures [24].

(C) Reductions via Convolution 3SUM. As mentioned before, the next wave of 3SUM-hard problems often involved reductions via the Convolution 3SUM problem. Except for the randomization in the reduction to Convolution 3SUM, many of these reductions were deterministic, and were thus fully derandomized by [22]. This list of problems includes Zero Triangle [52], Jumbled Indexing [9], Subset Sum for k-Enclosing Rectangle [21], and Hausdorff Distance under translation [18]. The Zero Triangle problem was then deterministically reduced to the Matching Triangle and Triangle Collection problems [7], which in turn led to deterministic reductions to a number of dynamic problems [7, 28].

(S) Reductions Based on the Self-Reduction. Another class of reductions that was already deranomized prior to our work is reductions that rely on an efficient *self-reduction* for 3SUM (i.e., a reduction that reduces 3SUM on n numbers to $n^{2\alpha}$ instances on $n^{1-\alpha}$ numbers each, for any $\alpha \in [0, 1]$). The first 3SUM self-reduction was given implicitly by Patrascu [48], and it was randomized. Later, deterministic self-reductions were proposed [46, 38]. Based on these deterministic self-reductions, one can immediately derandomize the reductions to All-Numbers 3SUM [51], and to detecting a 3-star or a 3-matching of total weight zero in edge-weighted graphs [4].

(D) Reductions via Set Disjointness, Set Intersection, and Triangle Problems. Another major class of reductions, as announced before, is via the Set Disjointness or Set Intersection problems. There are deterministic reductions from Set Intersection to Triangle Listing (in various parameterizations, also involving the arboricity of the given graph) [44], and via Triangle Listing to Dictionary Matching with one gap [10] and exists-connectivity queries in graph timelines [41]. Besides, Triangle Listing reduces to several database problems [47, 43, 17, 42]. The Set Disjointness problem reduces trivially to the All-Edges Triangle problem which in turn reduces to several range-query problems [31]. As a consequence of our Theorems 4 and 6, we have successfully derandomized all these reductions.

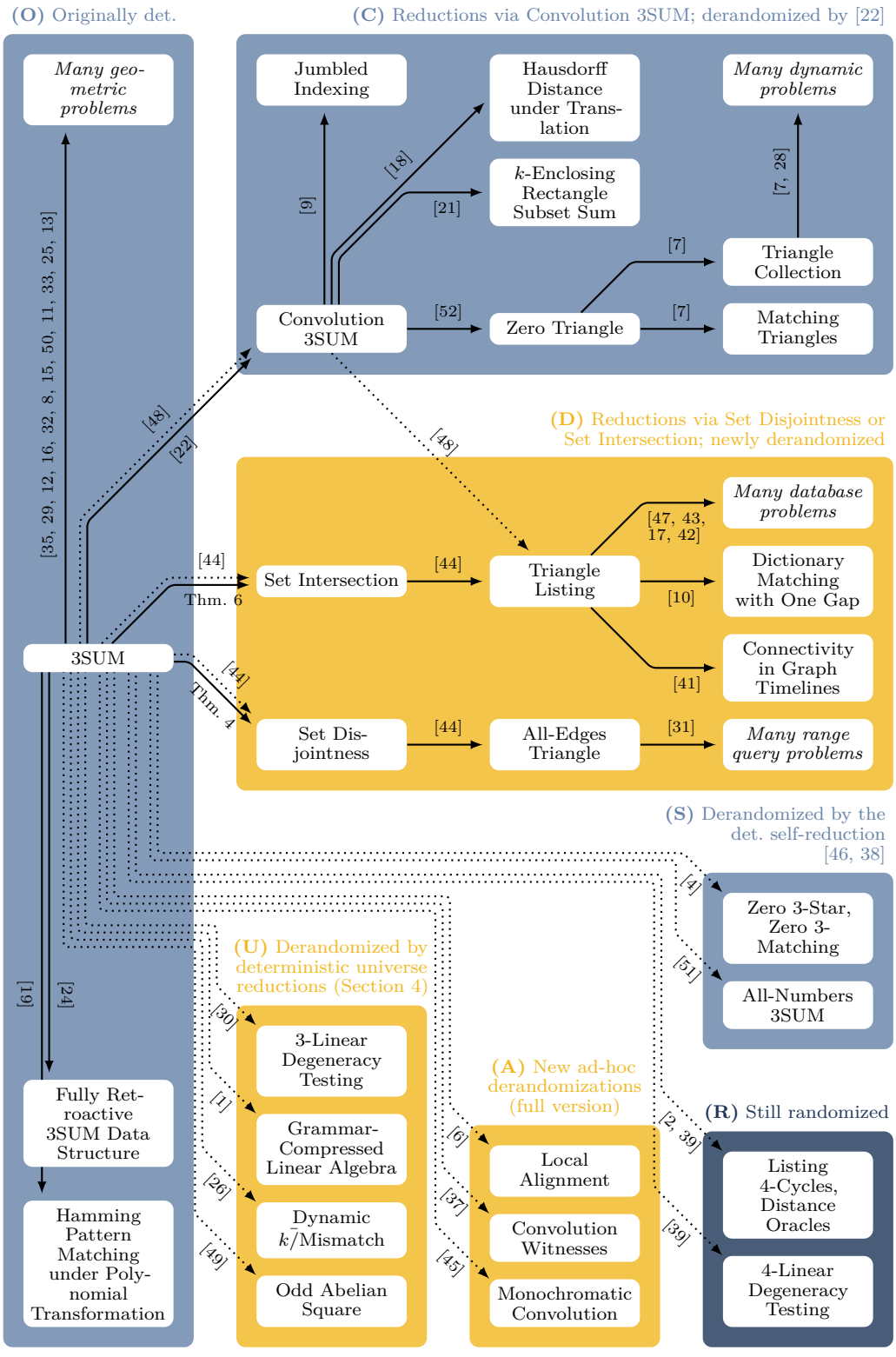

Figure 1 Fine-grained reductions from 3SUM. Solid edges depict deterministic reductions, and dotted edges depict randomized reductions. Our contribution is that we derandomize all yellow regions.

(U) Reductions Relying on Small Universe Size. Other reductions are not inherently randomized, but rather rely on a randomized universe reduction. We can derandomize all such reductions by replacing the randomized universe reduction with our deterministic one (Theorem 2). Reductions of this type have been established for 3-Linear Degeneracy Testing [30][1] and grammar-compressed variants of vector inner product and matrix-vector multiplication [1]. Similarly, the Dynamic k-Mismatch problem [26] relies on a quadratic-size universe reduction for All-Numbers 3SUM, and the Odd Abelian Square problem [49] relies on a quadratic-size universe reduction for Convolution 3SUM. To additionally derandomize these problems we extend our universe reduction for 3SUM to also cover these two cases (Lemmas 17 and 18).

(A) Reductions via Ad-Hoc Derandomizations. Fortunately, most hardness results have been derandomized using our general toolset. There are however a couple of reductions that required additional tailored arguments on top of that. The randomized reduction to Monochromatic Convolution [45] relies, among other things, on a trick of adding random offsets to merge several sparse sets into a dense one without (too many) collisions. We provide a deterministic variant of this trick, and successfully derandomize the reduction. Next, to derandomize the reduction to Local Alignment [6] we follow the approach of Chan an He [22] of controlling the number of collisions, and add the trick of Abboud, Lewi, and Williams [5] of replacing numbers with multidimensional vectors with small entries. Finally, the reduction to Convolution Witnesses [37] can be derandomized by plugging in our method for selecting a hash function with few false positives.

(R) Reductions that Remain Randomized. Unfortunately the list of known 3SUM-based reductions includes three more results which we did not manage to derandomize [3, 2, 39]. These papers rely on the recent "short cycle removal" technique that was originally proposed in [3] and optimized in [2, 39], and led to tight lower bounds for 4-Cycle Listing, Approximate Distance Oracles with stretch close to 2 or 3, and for 4-Linear Degeneracy Testing. At the heart of short cycle removal lies a structure-versus-randomness dichotomy which involves, in [2, 39], finding *structured* subsets via the algorithmic Balog-Szemerédi-Gowers (BSG) theorem [23]. Derandomizing the BSG algorithm constitutes the major challenge in derandomizing these reductions (though not the only one – the reductions also rely on a specialized kind of additive hashing which we have not attempted to derandomize here). We leave it as an important open problem to derandomize these few remaining reductions.

1.3 Outline

In Section 2 we fix some preliminaries. In Section 3 we introduce our derandomization toolkit – this section starts with a high-level technical overview before diving into the proofs. In Section 4 we combine our tools to obtain the deterministic universe reduction, and in Section 5 we provide the deterministic hardness results for Set Disjointness and Set Intersection. In the full version of the paper we provide even more derandomizations: for the Monochromatic Convolution, Local Alignment and Convolution Witness problems.

[1] In [30] the authors define 3SUM and the class of 3-Linear Degeneracy Testing problems over a universe of cubic size n^3, and thereby implicitly rely on a universe reduction.

2 Preliminaries

We use the standard notation $[n] := \{1, \ldots, n\}$ and $\widetilde{O}(T) = T(\log T)^{O(1)}$. Throughout this paper, we consider the 3SUM problem (and its variants) formally defined as follows:

▶ **Problem 7** (3SUM). Decide whether in a given set $A \subseteq [U]$ there is a triple $a, b, c \in A$ with $a + b = c$.

▶ **Problem 8** (All-Numbers 3SUM). Given a set $A \subseteq [U]$, decide for each $c \in A$ whether there is a pair $a, b \in A$ with $a + b = c$.

▶ **Problem 9** (Convolution 3SUM). Given a vector $X \in [U]^n$, decide for each $k \in [n]$ whether there is a pair $i, j \in [n]$ with $i + j = k$ and $X[i] + X[j] = X[k]$.

While we have stated the problems in a *monochromatic* form (i.e., for a single set A where we select $a, b, c \in A$), it is well-known and easy to check that they are equivalent to their *trichromatic* variants of these problems (i.e., for three sets A, B, C where we select $a \in A$, $b \in B$, $c \in C$) (by deterministic reductions). We occasionally rely on this statement in our proofs.

3 Our Toolkit

In this section we develop the toolkit that is necessary to prove our main derandomizations. We will start with a technical overview of the two tools in Sections 3.1 and 3.2, and provide the missing formal proofs in Section 3.3.

3.1 Tool 1: Deterministic Additive Hashing

A hash function h is *additive* if there is some modulus m, such that $h(a)+h(b) = h(a+b) \bmod m$ for all inputs a, b. Many 3SUM-hardness reductions involve additive hash functions (or *pseudo-additive* functions, which have the slightly weaker property that $h(a) + h(b) - h(a + b)$ takes only a constant number of values), and one of the simplest randomized constructions is the family of functions $h(x) = x \bmod m$ where m is a *random* prime of prescribed size.

In our setting we are required to select an additive hash function h *deterministically*. Previous work [23, 22] has already faced the same challenge, and provided the following simple solution. The insight is that function $h(x) = x \bmod m$ is still an effective hash function even if m is the product of several smaller primes $m = p_1 \cdot \ldots \cdot p_R$, $p_i \approx m^{1/R}$. In order to select m, one can follow the method of conditional expectations: Instead of picking all primes at the same time, we will fix $p_1, \ldots, p_R$ step by step. In each step, we pick a prime p_i which is *at least as good as what we would expect from a random prime*. In particular, it suffices to pick a locally optimal prime. To this end, we can exhaustively enumerate all primes of size $\approx m^{1/R}$ and test which one suits us best. All in all, this approach works to deterministically construct hash functions for all properties that can be efficiently tested.

Additive Hashing with Few Collisions. One such example is Chan and He's deterministic reduction from 3SUM to Convolution 3SUM [22]. Specifically, they require a hash function which has few *collisions* (i.e., pairs $a, b \in A$ with $h(a) = h(b)$). Since it is easy to count the number of collisions for a given hash function in linear time, the above recipe leads to the following derandomization:

▶ **Lemma 10** (Deterministic Additive Hashing with Few Collisions [23]). *Let $0 \leq \mu \leq 2, \delta > 0$. There is a deterministic algorithm that, given a set $A \subseteq [U]$ of size n, finds a modulus $m \in [n^\mu, 2n^\mu)$ such that*

$$\#\{(a,b) \in A^2 : a \equiv b \pmod m\} \leq n^{2-\mu}(\log U)^{O(1/\delta)}.$$

The algorithm runs in time $O(n^{1+\delta})$.

Additive Hashing with Few False Positives. Unfortunately, for most 3SUM-based reductions, bounding the number of collisions is not sufficient though. The property that most reductions rely on is that the number of triples $a, b, c \in A$ with $a + b \equiv c \pmod m$ is small – we often refer to such a triple as a *pseudo-solution* or a *false positive*. Unfortunately, counting the number of false positives is in general a hard problem – it is in fact another 3SUM instance $\{a \bmod m : a \in A\}$ which cannot be expected to be solvable in subquadratic time.

To circumvent this barrier, we exploit a simple observation: Note that the universe size of the reduced 3SUM instance is only m, hence we can count the 3SUM solutions in time $\widetilde{O}(m)$ using the Fast Fourier Transform. Building on that insight, we establish the following new lemma.

▶ **Lemma 11** (Deterministic Additive Hashing with Few False Positives). *Let $0 \leq \mu < 3, \delta > 0$. There is a deterministic algorithm that, given 3SUM instances $A_1, \ldots, A_g \subseteq [U]$ of size $O(n)$, either finds a positive modulus $m \in [n^\mu, 2n^\mu)$ such that*

$$\sum_{i=1}^{g} \#\{(a,b,c) \in A_i^3 : a + b \equiv c \pmod m\} \leq gn^{3-\mu+\delta}(\log U)^{O(1/\delta)},$$

or decides that at least one instance is a yes-instance. The algorithm runs in $\widetilde{O}(gn^{\max(\mu,1)+\delta})$ time.

We remark that while we state this lemma in a slightly more general form, in almost all applications we only need the lemma for the special case $g = 1$.

3.2 Tool 2: Deterministic Self-Reduction

As our second major tool we rely on a deterministic *self-reduction* for 3SUM. Here by a self-reduction we mean an algorithm that reduces a 3SUM instance A to some other 3SUM instances $A_1, \ldots, A_N$ with the property that A is a yes-instance if and only if there is a yes-instance A_i. This reduction is considered efficient only if $\sum_i |A_i|^2 \leq \widetilde{O}(|A|^2)$ (i.e., if the brute-force running times match).

Self-reductions for 3SUM are known based on two very different approaches. The first approach relies on additive hashing. The second approach, originally due to [46, 38] (with ideas borrowed from [27]), is conceptually simpler and also deterministic. The idea is to bucket A into smaller sets in such a way that only few triples of buckets can contain a solution. For our purposes we need a slightly stronger version than what was stated in the previous papers, where we also consider the universe size of the constructed subinstances:

▶ **Lemma 12** (Deterministic Self-Reduction). *Let $A \subseteq [U]$ be a set of size n, and let $g \geq 1$. We can deterministically construct a partition of A into subsets $A_1, \ldots, A_g$, and a set of triples $R \subseteq [g]^3$ of size $O(g^2)$ such that:*

- *Each set A_i has size $O(n/g)$ and can be covered by an interval of length $O(U/g)$.*
- *For all triples $a, b, c \in A$ with $a + b = c$, there is some $(i, j, k) \in R$ with $a \in A_i$, $b \in A_j$, $c \in A_k$.*

The algorithm runs in time $\widetilde{O}(n + g^2)$.

3.3 Proofs

We finally provide the missing proofs of Lemmas 11 and 12.

Proof of Lemma 11. For a modulus m, we call a triple $(a,b,c) \in A_i^3$ satisfying $a+b \equiv c \pmod m$ a *pseudo-solution* of m. Let us write

$$S(m) = \sum_{i=1}^{g} \#\{(a,b,c) \in A_i^3 : a+b \equiv c \pmod m\}$$

to denote the number of pseudo-solutions of m; our goal is to compute a modulus m that with few pseudo-solutions, $S(m) \leq gn^{3-\mu}(\log U)^{O(1/\delta)}$. As outlined before, our strategy is to let m be the product of several small primes, each of size roughly n^δ. These primes are selected one by one, such that each next prime narrows down the number of false positives by an appropriate amount. Specifically, consider the following process: We initialize $m \leftarrow 1$ and $P \subseteq [n^\delta, 2n^\delta)$ to be the subset of primes of size roughly n^δ, and run the following three steps:

1. For each prime $p \in P$, compute $S(m \cdot p)$ using FFT.
2. Select the prime $p \in P$ that minimizes $S(m \cdot p)$. Update $m \leftarrow m \cdot p$ and $P \leftarrow P \setminus \{p\}$.
3. If $m \geq \frac{1}{2}n^{\mu-\delta}$, then update $m \leftarrow m \cdot \lceil n^\mu/m \rceil$ and stop. Otherwise, go to step 1.

Running Time. The easier part of the proof is to bound the running time of this process. Note that we increase m by a factor of at least n^δ in each step. Hence, after at most $O(1)$ iterations we have increased m beyond the threshold $\frac{1}{2}n^{\mu-\delta}$ and the process stops. The running time is dominated by step 1: For each prime p, we compute $S(m \cdot p)$ using the Fast Fourier Transform. More precisely, for each 3SUM instance A_i we first prepare the (multi-)set $A_i \bmod (mp)$ obtained by hashing all numbers modulo mp. We can compute the number of 3SUM solutions in this reduced instance in time $O(mp \log(mp)) = \widetilde{O}(n+n^\mu)$ (using that $m \leq n^{\mu-\delta}$ and $p \leq n^\delta$). Repeating this call for all g instances and all $O(n^\delta)$ primes amounts for time $\widetilde{O}(n^\delta \cdot g \cdot (n+n^\mu)) = \widetilde{O}(gn^{\max(1,\mu)+\delta})$.

Correctness. It remains to bound $S(m)$, where is m is the modulus computed by the above process. Let m_i denote the modulus at the i-th step of the process (with $m_0 = 1$) and assume that all given 3SUM instances are unsatisfiable. We prove by induction that

$$S(m_i) \leq \frac{gn^3}{n^{i\delta}} \cdot (2\log U)^i.$$

For $i = 0$, this is clearly true. So let $i > 0$ and assume that the induction hypothesis holds for $i-1$. Our strategy is to show that for a *random* prime $p \in [n^\delta, 2n^\delta)$, with good probability we have

$$S(m_{i-1} \cdot p) \leq \frac{gn^3}{n^{i\delta}} \cdot O(\log U)^i.$$

Consider an arbitrary triple $a,b,c \in [U]$ with $a+b \neq c$. The event $a+b \equiv c \bmod p$ is equivalent to p being a divisor of the number $a+b-c \in [-U\,..\,2U]$, which happens with probability at most

$$\frac{\log_{n^\delta}(2U)}{\Omega(n^\delta \log^{-1}(n))} = O(n^{-\delta}\log U),$$

since $\log_{n^\delta}(2U)$ is the maximum number of divisors of size at least n^δ of $a+b-c$, and since there are $\Omega(n^\delta \log^{-1}(n))$ primes in $[n^\delta, 2n^\delta)$ by the Prime Number Theorem. Therefore, among the pseudo-solutions modulo m_{i-1}, only an expected $O(n^{-\delta}\log U)$-fraction is also a pseudo-solution modulo $m_{i-1}\cdot p$. Therefore, by Markov's inequality with probability at least $\frac{1}{2}$ we have that

$$\begin{aligned} S(m_{i-1}\cdot p) &\le 2\cdot S(m_{i-1})\cdot O(n^{-\delta}\log U) \\ &\le \frac{gn^3}{n^{(i-1)\delta}}\cdot O(\log U)^{i-1}\cdot O(n^{-\delta}\log U) \\ &= \frac{gn^3}{n^{i\delta}}\cdot O(\log U)^i. \end{aligned}$$

It follows that there is a successful prime $p\in[n^\delta,2n^\delta)$ with $S(m_{i-1}\cdot p)\le \frac{gn^3}{n^{i\delta}}\cdot O(\log U)^i$. Recall that the algorithm selects the prime that *minimizes* the number of pseudo-solutions $S(m_{i-1}\cdot p)$ and so indeed we have $S(m_i)\le \frac{gn^3}{n^{i\delta}}\cdot O(\log U)^i$.

Finally, recall that the process terminates no sooner than $m_i\ge\frac{1}{2}n^{\mu-\delta}$. Since $n^\delta\le m_i<(2n^\delta)^i$, it follows that the final iteration count is at least $i\ge\frac{\mu}{\delta}-O(1)$ and at most $i\le\frac{\mu}{\delta}$. Hence, for the final modulus m we have indeed

$$S(m)\le\frac{gn^3}{n^{\mu-O(\delta)}}\cdot O(\log U)^{\frac{\mu}{\delta}} = gn^{3-\mu+O(\delta)}\cdot(\log U)^{O(1/\delta)}.$$

By decreasing δ by a constant factor, we obtain the claimed bound. Recall that this bound is conditioned on the assumption that the given 3SUM instances are no-instances. However, if our algorithm computes a modulus m with unexpectedly many pseudo-solutions (which we can verify in time $\widetilde{O}(gn^{\max(1,\mu)+\delta})$), we can infer that at least one of the given instances is a yes-instance.

As a final detail, we show that the algorithm produces a modulus m in the range $[n^\mu,2n^\mu)$. Indeed, in line 3, we execute the final update $m\leftarrow m\cdot\lceil n^\mu/m\rceil$ only after we have increased m to be in the range $\frac{1}{2}n^{\mu-\delta}\le m<n^\mu$. It follows that then $m\cdot\lceil n^\mu/m\rceil\le m\cdot(n^\mu/m+1)<2n^\mu$. ◀

This completes the treatment of the deterministic additive hashing tool. Let us next prove the refined deterministic self-reduction.

Proof of Lemma 12. Our goal is to partition A into subsets $A_1,\dots,A_g$ with the following two properties:

- $|A_i|\le 2n/g$, and
- $\max(A_i)-\min(A_i)\le 2U/g$ (i.e., A_i can be covered by an interval of length $2U/g$).

By constructing this partition greedily, by scanning over the elements in A in sorted order including all elements until one of the two conditions is violated, it is easy to see that both rules lead to at most $g/2$ stops and therefore g groups suffice in total. We assume that the groups constructed in this way are ordered in the natural way (i.e., $\max(A_i)<\min(A_{i+1})$ for all i).

The next step is to construct the set $R\subseteq[g]^3$. Naively, including all g^3 triples (i,j,k) would be correct, if there is a 3-sum in the original instance it is certainly contained in one of the subinstance (A_i,A_j,A_k). However, many of these instances are *trivial* in the sense that either

1. $\min(A_i)+\min(A_j)>\max(A_k)$, or
2. $\max(A_i)+\max(A_j)<\min(A_k)$;

both conditions make it impossible for (A_i,A_j,A_k) to contain a 3-sum. To improve the reduction, we will therefore include in R only triples that are not trivial.

We argue that the number of remaining instances (that is, the number of instances that falsify both previous conditions) is at most $O(g^2)$. To see this, consider the partially ordered set $([g]^3, \prec)$ where we let $(i,j,k) \prec (i',j',k')$ if and only if $\max(A_i) < \min(A_{i'})$, $\max(A_j) < \min(A_{j'})$ and $\max(A_k) < \min(A_{k'})$. With this definition, if $(i,j,k) \prec (i',j',k')$, then (i,j,k) or (i',j',k') must be a trivial no-instance. Indeed, suppose that (i,j,k) and (i',j',k') both falsify the conditions (1) and (2). Then:

$$\begin{aligned} 0 &\le \max(A_i) + \max(A_j) - \min(A_k) && \text{(since } (i,j,k) \text{ falsifies (2))} \\ &< \min(A_{i'}) + \min(A_{j'}) - \max(A_{k'}) && \text{(since } (i,j,k) \prec (i',j',k')\text{)} \\ &\le 0. && \text{(since } (i',j',k') \text{ falsifies (1))} \end{aligned}$$

More generally, on any chain in the lattice there can be at most one nontrivial instance. Therefore, in order to bound the number of nontrivial instances, it suffices to cover $[g]^3$ by $O(g^2)$ chains. This cover is easy to construct: For all $a, b \in \{-g, \dots, g\}$, take the chain $\{(i, i+a, -i+b) : i \in [g]\}$. Any triple (i,j,k) is covered by the chain for $a = j - i$ and $b = k + i$.

It remains to analyze the running time. Construction the partition of A takes linear time after sorting A in time $O(n \log n)$. We can construct the set R by enumerating all $(i,j) \in [g]^2$, and enumerating for each such pair only the subset of k's with $\min(A_i) + \min(A_j) \le \max(A_k)$ and $\max(A_i) + \max(A_j) \ge \min(A_k)$. By preprocessing A with a range query data structure in near-linear time, this step runs in time $\widetilde{O}(g^2 + |R|) = \widetilde{O}(g^2)$. ◀

This lemma immediately implies the following deterministic self-reductions for 3SUM and the all-numbers variant of 3SUM. (Here, the running time increases to $\widetilde{O}(ng)$ since we explicitly write down all constructed instances, instead of concisely describing the instances via subsets of the original instance.)

▶ **Corollary 13** (Deterministic Self-Reduction for 3SUM). *For any $g \ge 1$, a given 3SUM instance of size n over the universe $[U]$ can deterministically be reduced to $O(g^2)$ 3SUM instances of size $O(n/g)$ over the universe $[O(U/g)]$. The running time of the reduction is $\widetilde{O}(ng)$.*

▶ **Corollary 14** (Deterministic Self-Reduction for All-Numbers 3SUM). *For any $g \ge 1$, a given AN-3SUM instance of size n over the universe $[U]$ can deterministically be reduced to $O(g^2)$ AN-3SUM instances of size $O(n/g)$ over the universe $[O(U/g)]$. The running time of the reduction is $\widetilde{O}(ng)$.*

4 Deterministic Universe Reduction for 3SUM

In this section we combine the tools established in Section 3 to deduce a deterministic universe reduction for 3SUM (Theorem 2). We later extend the universe reduction also to All-Numbers 3SUM and Convolution 3SUM (Lemmas 17 and 18). We start with a trivial reduction which we need in the proof of Theorem 2.

▶ **Observation 15** (Trivial Universe Reduction for 3SUM). *Let $U \ge U'$. A 3SUM instance over $[U]$ can be reduced to $O((\frac{U}{U'})^3)$ many 3SUM instances over $[U']$ (of the same size n) such that the solutions are in 1-to-1 correspondence. The reduction runs in time $O(n \cdot (\frac{U}{U'})^3)$.*

Proof. Chop the given set A into $\lceil \frac{U}{U'} \rceil$ subsets that can be covered by an interval of length U'. In this way we construct at most $\lceil \frac{U}{U'} \rceil^3$ triples each of which can be viewed as a 3SUM instance over the universe $[U']$. (In fact the number of relevant 3SUM solutions is only $O((\frac{U}{U'})^2)$, but we do not need this improvement here.) ◀

▶ **Theorem 2** (Deterministic Universe Reduction for 3SUM). *If 3SUM over $[n^3]$ can be solved in deterministic time $O(n^{2-\epsilon})$ for some $\epsilon > 0$, then 3SUM over $[U]$ can be solved in deterministic time $O(n^{2-\epsilon'} \log^c U)$ for some constants $\epsilon', c > 0$.*

Proof. Assume that 3SUM over the universe $[n^3]$ can be solved in time $O(n^{2-\epsilon})$ for some $\epsilon > 0$. We are designing a subquadratic algorithm for a given 3SUM instance $A \subseteq [U]$. Our reduction runs in four steps:

1. Let $\mu, \delta > 0$ be parameters to be determined later. We use the deterministic additive hashing lemma (Lemma 11 with parameters μ, δ and $g = 1$) to find a modulus m with $m = \Theta(n^\mu)$ and

$$S := \#\{(a,b,c) \in A^3 : a + b \equiv c \pmod m\} \leq n^{3-\mu+\delta}(\log U)^{O(1/\delta)}.$$

   Alternatively, if Lemma 11 reports that the instance contains a solution, we can immediately stop here. As before let us refer to triples (a,b,c) with $a + b \equiv c \pmod m$ as *pseudo-solutions.* Our strategy is to design a subquadratic algorithm that lists all pseudo-solutions. Afterwards, it is easy to check if a genuine solution is among them. Note that the pseudo-solutions stand in $1\overline{/}1$-to-$\overline{/}O(1)$-correspondence to the solutions of the 3SUM instance

$$A' := \{a \bmod m, (a \bmod m) + m : a \in A\},$$

   hence in the following steps it suffices to list all solutions in A'.
2. Let α be a parameter to be determined. We apply the deterministic self-reduction (Corollary 13 with parameter $g = n^\alpha$) to A' to construct $O(n^{2\alpha})$ 3SUM instances A'_i, each of size $O(n^{1-\alpha})$ and over a universe of size $O(m/n^\alpha) = O(n^{\mu-\alpha})$.
3. Using the trivial universe reduction (Observation 15 with parameters $U = \Theta(n^{\mu-\alpha})$ and $U' = \Theta(n^{3-3\alpha})$) we can further reduce each instance A'_i to $\max(1, (\frac{n^{\mu-\alpha}}{n^{3-3\alpha}})^3)$ 3SUM instances of the same size $O(n^{1-\alpha})$ and over a universe of cubic size $O(n^{3-3\alpha})$. We solve each such instance using the fast algorithm in time $O(n^{(1-\alpha)(2-\epsilon)})$. (Here we only detect whether A'_i contains a solution, but do not list all solutions.)
4. For each set A'_i for which we have detected a solution in the previous step, we exhaustively enumerate all solutions in time $O(|A'_i|^2) = O(n^{2-2\alpha})$.

Running Time. The correctness is clear, but it remains to analyze the running time. In what follows we choose the parameters $\delta = \frac{\epsilon}{32} > 0$, $\mu = 2 - 2\delta$, $\alpha = \frac{1}{2} + 2\delta$ and analyze the contributions step-by-step:

1. Running Lemma 11 takes time $O(n^{\max(1,\mu)+\delta}) = O(n^{2-\delta})$.
2. Running Corollary 13 takes time $\widetilde{O}(ng) = \widetilde{O}(n^{3/2+2\delta})$.
3. The overhead due to Observation 15 is negligible, and this step is dominated by solving the $O(n^{2\alpha} \cdot \max(1, \frac{n^{\mu-\alpha}}{n^{3-3\alpha}})^3)$ instances using the fast algorithm. Each call takes time $O(n^{(1-\alpha)(2-\epsilon)})$, hence in total this step takes time

$$\begin{aligned} &O(n^{2\alpha} \cdot \max(1, \tfrac{n^{\mu-\alpha}}{n^{3-3\alpha}})^3 \cdot n^{(1-\alpha)(2-\epsilon)}) \\ &\quad = O(n^{2-\epsilon(1-\alpha)} \cdot \max(1, \tfrac{n^{3/2-4\delta}}{n^{3/2-6\delta}})^3) \\ &\quad = O(n^{2-\epsilon(1-\alpha)+6\delta}) \\ &\quad = O(n^{2-2\delta}), \end{aligned}$$

   where in the last step we used that $1 - \alpha \geq \frac{1}{4}$.

4. Recall that we only brute-force 3SUM solutions A'_i which contain a solution. Moreover, since the solutions across the sets A'_i are in $O(1)$-to-1 correspondence to the at most S pseudo-solutions, we brute-force at most $O(S)$ instances in total time

$$O(S \cdot n^{2-2\alpha}) = O(n^{3-\mu+\delta+2-2\alpha}(\log U)^{O(1/\delta)}) = O(n^{2-\delta}(\log U)^{O(1/\delta)}).$$

Summing over all contributions, the total running time is bounded by $n^{2-\delta}(\log U)^{O(1/\delta)}$, which is as claimed. ◀

▶ **Corollary 16** (3SUM over Cubic Universes). *For any $\epsilon > 0$, the 3SUM problem over a universe of size n^3 cannot be solved in deterministic time $O(n^{2-\epsilon})$, unless the deterministic 3SUM hypothesis fails.*

4.1 Universe Reduction for All-Numbers 3SUM

For the All-Numbers 3SUM problem there exists an even better randomized universe reduction – to a universe of quadratic size n^2. To derandomize this universe reduction as well, we follow the same ideas as in Theorem 2:

▶ **Lemma 17** (All-Numbers 3SUM over Quadratic Universes). *For any $\epsilon > 0$, the AN-3SUM problem over a universe of size n^2 cannot be solved in deterministic time $O(n^{2-\epsilon})$, unless the deterministic 3SUM hypothesis fails.*

Proof. Suppose that AN-3SUM over quadratic universes can be solved in time $O(n^{2-\epsilon})$. We design a 3SUM algorithm for $A \subseteq [n^c]$ in time $O(n^{2-\epsilon'})$. The reduction again runs in four steps:

1. We first apply Lemma 11 (with parameters $\mu, \delta > 0$ to be determined later and $g = 1$) to find a modulus $m = \Theta(n^\mu)$ such that

$$S := \#\{(a,b,c) \in A^3 : a + b \equiv c \pmod m\} \leq n^{3-\mu+\delta}(\log n)^{O(1/\delta)} = \widetilde{O}(n^{3-\mu+\delta}).$$

   Let us again refer to all triples (a,b,c) with $a + b \equiv c \pmod m$ as *pseudo-solutions.* Then the pseudo-solutions of the original instance are in 1-to-$O(1)$ correspondence to the solutions of the 3SUM instance

$$A' = \{a \bmod m, (a \bmod m) + m : a \in A\}.$$

2. We apply Lemma 12 with parameter $g = n^\alpha$ to partition A' into subsets $A'_1, \ldots, A'_g$ such that each part has size at most $O(n^{1-\alpha})$, with only $O(n^{2\alpha})$ triples of parts that may contain a 3-sum. (This reduction even reduces the universes of the subinstances by a factor n^α, but we don't need this improvement here).
3. We apply the trivial universe reduction (Observation 15 with parameters $U = \Theta(n^\mu)$ and $U' = \Theta(n^{2-2\alpha})$) to further reduce each instance A'_i to $\max(1, \frac{n^\mu}{n^{2-2\alpha}})^3$ instances of the same size over a universe of quadratic size $O(n^{2-2\alpha})$. We solve each new instance using the fast AN-3SUM algorithm in total time $O(n^{2\alpha} \cdot \max(1, \frac{n^\mu}{n^{2-2\alpha}})^3 \cdot n^{(1-\alpha)(2-\epsilon)})$.
4. For each relevant triple (i,j,k), and for each $a \in A'_i$ that was found to be part of a pseudo-solution, we explicitly list all pseudo-solutions in $A'_i \cup A'_j \cup A'_k$. As this step lists all pseudo-solutions, we are guaranteed to find any proper 3-sum in this step.

Running Time. It is easy to see that this algorithm is correct, but it remains to analyze the running time. We choose the parameters $\delta := \frac{\epsilon}{16}$, $\mu := 2 - 2\delta$, and $\alpha = 4\delta$, and analyze the running time step by step:

1. Running Lemma 11 takes time $O(n^{\max(\mu,1)+\delta}) = O(n^{2-\delta})$.
2. Running Lemma 12 takes time $\widetilde{O}(n + g^2) = \widetilde{O}(n)$.
3. The trivial universe reduction is negligible, hence this step runs in total time

$$O(\max(1, \tfrac{n^\mu}{n^{2-2\alpha}})^3 \cdot n^{(1-\alpha)(2-\epsilon)}) = O(n^{2-(1-\alpha)\epsilon+6\delta}) = O(n^{2-2\delta}),$$

   using in the last step that $1 - \alpha \geq \frac{1}{2}$.
4. We brute-force $S = \widetilde{O}(n^{3-\mu+\delta})$ many instances of size $O(n^{1-\alpha})$, taking time $\widetilde{O}(n^{2-\delta})$.

Summing over all contributions we obtain that the total running time is $O(n^{2-\delta})$ as claimed. ◀

4.2 Universe Reduction for Convolution 3SUM

Finally we also prove that by means of deterministic reductions, the Convolution 3SUM problem is already hard over quadratic universes. This proof is essentially identical to [22], except that we start with our deterministic universe reduction for 3SUM.

▶ **Lemma 18** (Convolution 3SUM over Quadratic Universes)**.** *For any $\epsilon > 0$, the Convolution 3SUM problem over a universe of size n^2 cannot be solved in deterministic time $O(n^{2-\epsilon})$, unless the deterministic 3SUM hypothesis fails.*

Proof. Suppose that the Convolution 3SUM problem over a universe of size n^2 can be solved in deterministic time $O(n^{2-\epsilon})$ for some $\epsilon > 0$. In order to falsify the deterministic 3SUM hypothesis, it suffices to design a subquadratic algorithm for a given 3SUM instance $A \subseteq [n^3]$. Using the deterministic additive hashing construction with few *collisions* (Lemma 10 with parameters $\mu = 1$ and $\delta = \frac{1}{2}$, say), we can find a modulus $m = \Theta(n)$ such that

$$\#\{(a, b) \in A^2 : a \equiv b \pmod m\} \leq \widetilde{O}(n).$$

We continue with some terminology: Call an integer i is *α-heavy* if $|\{a \in A : a \equiv i \pmod m\}| > \alpha$, and *$\alpha$-light* otherwise.

▷ Claim 19. Let $\alpha > 1$. The number of α-heavy elements $a \in A$ is at most $\widetilde{O}(n/\alpha)$.

Proof. The proof is by a simple calculation. We express the number of α-heavy elements in A as

$$\begin{aligned}
&\sum_{\substack{i \in [m] \\ i \text{ is heavy}}} |\{a \in A : a \equiv i \pmod m\}| \\
&\quad \leq \sum_{\ell=\lfloor \log \alpha \rfloor}^{\infty} \sum_{\substack{i \in [m] \\ i \text{ is } 2^\ell\text{-heavy} \\ i \text{ is } 2^{\ell+1}\text{-light}}} |\{a \in A : a \equiv i \pmod m\}| \\
&\quad \leq \sum_{\ell=\lfloor \log \alpha \rfloor}^{\infty} \sum_{\substack{i \in [m] \\ i \text{ is } 2^\ell\text{-heavy}}} 2^{\ell+1}
\end{aligned}$$

Now observe that each bucket $i \in [m]$ which is 2^ℓ-heavy contributes at least $\binom{2^\ell}{2} = \Omega(2^{2\ell})$ many collisions. Since the total number of collisions is bounded by $\widetilde{O}(n)$, there can be at most $\widetilde{O}(n/2^{2\ell})$ many such heavy buckets, and thus

$$\leq \sum_{\ell=\lfloor \log \alpha \rfloor} \widetilde{O}(2^\ell \cdot \tfrac{n}{2^{2\ell}})$$
$$\leq \widetilde{O}(n/\alpha).$$ ◁

Coming back to the lemma, let us simply call i *heavy* if it is n^δ-heavy for some parameter $\delta > 0$ to be determined later, and *light* otherwise. We consider two cases, depending on whether the 3SUM solution contains a 3SUM solution with a heavy element, or whether all 3SUM solutions consist of light elements only. For the former case, since there are at most $\widetilde{O}(n^{1-\delta})$ many heavy elements $a \in A$, we can simply brute-force over all solutions involving a heavy element in time $\widetilde{O}(n^{1-\delta} \cdot n) = \widetilde{O}(n^{2-\delta})$. Hence, for the rest of the proof we focus on the latter case and prove that we can detect 3SUM solutions involving only light elements.

Let $(x, y, z) \in [n^\delta]^3$. For each such triple, we construct a trichromatic Convolution 3SUM instance (X_x, Y_y, Z_z) of length $2m$ as follows. (Reducing that trichromatic instance further to a monochromatic instance is standard.) For $i \in [2m]$, let $X_x[i] := \frac{a-i}{m}$ where a is the x-th element in A with $a \equiv i \pmod m$ (according to an arbitrary but fix order). Similarly, let $Y_y[j] := \frac{b-j}{m}$ where b is the y-th element in A with $b \equiv j \pmod m$ and let $Z_z[k] := \frac{c-k}{m}$ where c is the z-th element in A with $z \equiv k \pmod m$. If there happens to be no x-th (or y-th or z-th) element, simply put a dummy value that cannot be part of any solution. It is easy to verify that all entries are integers, and further have size at most $n^3/m = O(n^2)$.

Moreover, there is a solution in any of the Convolution 3SUM instances $(X_x, Y_y, Z_z)_{x,y,z \in [n^\delta]}$ if and only if the given 3SUM instance A has a light solution. For the one direction, suppose that there are light elements $a, b, c \in A$ with $a + b = c$. Let $i := x \bmod m$, $j := y \bmod m$ and $k := i + j$; clearly we have $k \equiv x \pmod m$. Then there are $x, y, z \in [n^\delta]$ such that a is the x-th element in A with $a \equiv i \pmod m$, b is the y-th element in A with $b \equiv j \pmod m$, and c is the z-th element in A with $c \equiv k \pmod m$. By definition we have that $i + j = k$ and that

$$X_x[i] + Y_y[j] = \frac{a - i + b - j}{m} = \frac{c - k}{m} = Z_z[k].$$

The other direction is symmetric.

We have constructed $n^{3\delta}$ many Convolution 3SUM instances over a quadratic universe, and solving all of these takes time $O(n^{3\delta} \cdot n^{2-\epsilon})$ by our initial assumption. The total time is thus $O(n^{2-\delta} + n^{2-\epsilon+3\delta})$ which is subquadratic by choosing $\delta = \frac{\epsilon}{4} > 0$. ◀

4.3 Universe Reduction for 3SUM Listing

As a final simple corollary we include the following deterministic universe reduction for the 3SUM Listing problem (where the goal is to list all solutions):

▶ **Lemma 20** (3SUM Listing over Small Universes). *For any $1 < \mu < 2$ and $\epsilon > 0$, there is no deterministic algorithm that given a size-n 3SUM instance $A \subseteq [n^\mu]$ with $O(n^{3-\mu})$ solutions lists all solutions in time $O(n^{2-\epsilon})$, unless the deterministic 3SUM hypothesis fails.*

Proof. This lemma is basically an immediate consequence of Lemma 11. The only minor complication is how to remove the overhead δ.

Let $A \subseteq [n^c]$ be a given 3SUM instance. Using Lemma 11 (with parameters μ, $0 < \delta \leq \mu - 1$ to be determined, and $g = 1$) we can compute a modulus $m \in [n^\mu, 2n^\mu]$ such that the number of pseudo-solutions is bounded by

$$S := \#\{(a,b,c) \in A^3 : a + b \equiv c \pmod m\} \leq \widetilde{O}(n^{3-\mu+\delta}).$$

We construct the 3SUM Listing instance $A' = \{a \bmod m, (a \bmod m) + m : a \in A\} \cup B$, where B is a set of $n^{1+\delta}$ dummy elements that cannot be part of a solution (e.g., $B = \{10m + i : i \in [n^{1+\delta}]\}$). Clearly the solutions in A' stand in $O(1)$-to-1 correspondence to the pseudo-solutions of A. Therefore, given a list of all $O(S)$ solutions of A' we can decide the original instance by testing whether one of the pseudo-solutions is proper. Note that $n' = |A'| = \Theta(n^{1+\delta})$, and thus $A' \subseteq [O(m)] \subseteq [(n')^\mu]$ and the number of solutions is $O(S) = \widetilde{O}(n^{3-\mu+\delta}) = O((n')^{3-\mu})$. Applying the efficient listing algorithm takes subquadratic time $O((n')^{2-\epsilon}) = O(n^{2+2\delta-\epsilon})$, by choosing $\delta := \min\{\frac{\epsilon}{3}, \mu - 1\} > 0$. ◀

5 Deterministic Lower Bounds for Set Disjointness and Set Intersection

In this section we give our deterministic lower bounds for the (offline) Set Disjointness and (offline) Set Intersection problems (Theorems 4 and 6). Let us recall their definitions first.

▶ **Problem 3** (Set Disjointness)**.** Given sets $S_1, \ldots, S_N \subseteq [U]$ each of size at most s and a set of q queries $Q \subseteq [N]^2$, report for each query $(i,j) \in Q$ whether $S_i \cap S_j = \emptyset$.

▶ **Problem 5** (Set Intersection)**.** Given sets $S_1, \ldots, S_N \subseteq [U]$ each of size at most s and a set of q queries $Q \subseteq [N]^2$, report for each query $(i,j) \in Q$ the set intersection $S_i \cap S_j$. Occasionally we specify a size threshold up to which the algorithm is supposed to list elements.

The Set Disjointness has two "trivial" solutions: On the one hand, we can solve each query in time $O(s)$ thus taking time $O(qs)$ in total. On the other hand, we can encode all queries as a matrix multiplication problem of size $N \times s \times N$. If matrix multiplication is in linear time (i.e., if $\omega = 2$), this leads to an algorithm in time $(N^2 + Ns)^{1+o(1)}$. While the first approach works out of the box also for the Set Intersection problem, the latter algorithm does not immediately apply.

Kopelowitz, Pettie and Porat [44] have proved that these algorithms are best-possible – at least if 3SUM cannot be solved in randomized subquadratic time. In what follows, we derandomize their reductions.[2]

We start with the following trivial reduction from Set Intersection (for small thresholds t) to Set Disjointness. We will later rely on this reduction for $t = n^\epsilon$.

▶ **Observation 21** (Trivial Reduction from Set Intersection to Set Disjointness)**.** *If the offline Set Disjointness problem is in deterministic time $f(U,N,s,q)$, then the offline Set Intersection problem with a threshold of t per query is in time $O(f(U, Nt^2, \frac{s}{t^2}, qt^4) + \frac{qs}{t}) = O(t^6 \cdot f(U,N,s,q) + \frac{qs}{t})$.*

Proof. Given a Set Intersection instance $S_1, \ldots, S_N$, let us arbitrarily partition each set S_i into t^2 subsets $S_{i,1}, \ldots, S_{i,t^2}$ of size $O(s/t^2)$. We replace each original by t^4 queries that respectively compare all possible combination of subsets. In this way we clearly obtain an

[2] Let us remark that in their paper [44] they even optimize their reductions with respect to logarithmic factors; here, since we are only interested in subquadratic-time 3SUM-hardness, we consider a coarser-grained view.

equivalent Set Intersection instance, with Nt^2 many sets of size $O(s/t^2)$, over the same universe size U and with qt^4 many queries. We run the fast Set Disjointness algorithm on that instance. Our task is to report, for each query $(i,j) \in Q$ in the original Set Intersection instance, up to t elements from the intersection $S_i \cap S_j$. The outcome of the Set Disjointness call yields a subset $Q_{i,j} \subseteq [t^2]^2$ of positions (i',j') such that $S_{i,i'} \cap S_{j,j'}$ is nonempty. Among these pairs in $Q_{i,j}$, select up to t arbitrarily. For each selected pair (i',j') we evaluate $S_{i,i'} \cap S_{j,j'}$ by scanning through all elements of $S_{i,i'}$. Each such scan reveals at least one element of the intersection $S_i \cap S_j$, and therefore indeed after t repetitions we have achieved our goal.

The algorithm takes time $f(U, Nt^2, \frac{s}{t^2}, qt^4)$ for the call to the Set Disjointness oracle. Afterwards, we explicitly compute the set intersections of $q \cdot t$ sets of size $O(s/t^2)$ each, taking time $O(q \cdot t \cdot s/t^2) = O(\frac{qs}{t})$ in total. ◀

▶ **Theorem 4** (Deterministic Set Disjointness Hardness)**.** *Let $0 \leq \alpha < 1$. Unless the deterministic 3SUM hypothesis fails, there is no deterministic algorithm for (offline) Set Disjointness with parameters $|U| = O(n^{2-2\alpha})$, $N = O(n)$, $s = O(n^{1-\alpha})$ and $q = O(n^{1+\alpha})$ that runs in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$.*

Let us remark that the case $\alpha = 0$ is trivial, since the input size is already $\Omega(n^2)$.

Proof. Suppose there is a Set Disjointness algorithm with parameters as specified in the lemma statement. We will design a subquadratic algorithm for 3SUM. Given a 3SUM instance A, we proceed in the following steps:

1. We apply the deterministic self-reduction for 3SUM. Specifically, we apply Lemma 12 to partition A into $g = n^\alpha$ groups $A_1, \ldots, A_g$ of size $O(n^{1-\alpha})$, such that only a subset $R \subseteq [g]^3$ of at most $|R| = O(n^{2\alpha})$ group triples is relevant.
2. We apply the deterministic hashing lemma to find a modulus m under which we have only few false positives in the relevant groups. Specifically, we apply Lemma 11 for the 3SUM instances $(A_i \cup A_j \cup A_k)_{(i,j,k)\in R}$ (of size $O(n^{1-\alpha})$ each) and for $\mu = 2 - 2\delta$ and some parameter $\delta > 0$ to be specified later. This either reports that one of the instances contains a 3SUM solution (in which case we can stop), or the lemma returns a modulus $m = O((n^{1-\alpha})^\mu) = O(n^{2-2\alpha})$ satisfying that

$$\sum_{(i,j,k)\in R} \#\{(a,b,c) \in A_i \times A_j \times A_k : a + b \equiv c \pmod m\} \leq |R| \cdot O((n^{1-\alpha})^{1+3\delta}) \cdot (\log n)^{O(1/\delta)} = \widetilde{O}(n^{1+\alpha+3\delta}).$$

   Let us for simplicity assume that $\sqrt{m}$ is an integer; otherwise replace all the following occurrences of $\sqrt{m}$ with an appropriate rounding of $\sqrt{m}$.
3. We are ready to construct the Set Disjointness instance: For each $i \in [g]$ and for all $x, y \in [\sqrt{m}]$, we construct the following two sets:

$$B_{i,x} = \{(b + x\sqrt{m}) \bmod m : b \in A_i\},$$
$$C_{i,y} = \{(c + y) \bmod m : c \in A_i\}.$$

   The Set Disjointness instance consists of the sets $\{B_{i,x}\}_{i,x} \cup \{C_{i,y}\}_{i,y}$. The queries are defined as follows: For each $(i,j,k) \in R$ and for each $a \in A_i$, we add a query. Namely, let $x, y \in [\sqrt{m}]$ be such that $a \equiv x\sqrt{m} - y \pmod m$; then we query whether the intersection $B_{j,x} \cap C_{k,y}$ is empty.

Let us take a break here from the algorithm, and for now give an explanation of the Set Disjointness instance. We claim that the elements in the queried set intersections $B_{j,x} \cap C_{k,y}$ are in one-to-one correspondence to the set of *pseudo-solutions*

$$\bigcup_{(i,j,k)\in R} \{(a,b,c) \in A_i \times A_j \times A_k : a + b \equiv c \pmod m\}.$$

Indeed, for any pseudo-solution $(a,b,c) \in A_i \times A_j \times A_k$, let $x, y \in [\sqrt{m}]$ be such that $a \equiv x\sqrt{m} - y \pmod m$. Then note that $(b + x\sqrt{m}) \bmod m \in B_{j,x}$ and that $(c+y) \bmod m \in C_{k,y}$. Moreover, since $x\sqrt{m} - y + b \equiv a + b \equiv c \pmod m$, these two elements are in fact identical. The converse direction is analogous.

With this in mind, our goal is to list *all* elements in the intersections $B_{j,x} \cap C_{k,y}$ of all queries. In this way we obtain the set of all pseudo-solutions and we can easily check whether a 3SUM solution is among them. However, the Set Disjointness algorithm is not required to report the elements in the intersections. Instead, we proceed as follows:

4. Using the reduction from Observation 21 with threshold $t = n^\rho$ (for some parameter ρ to be determined later), we list up to t elements from each queried intersection.
5. For each query $B_{j,x} \cap C_{k,y}$ with $|B_{j,x} \cap C_{k,y}| \geq t$ (i.e., for which we have exhausted the threshold in the previous listing step), we explicitly enumerate all elements in the intersection by scanning once through $B_{j,x}$.

After these two steps, we have clearly listed all elements in the queried intersections. As argued before, this completes the 3SUM algorithm.

It remains to argue that the running time is subquadratic as claimed. Let us analyze the five steps of the algorithm individually:

1. Takes time $\widetilde{O}(n + g^2) = \widetilde{O}(n + n^{2\alpha})$ by Lemma 12.
2. Takes time $\widetilde{O}(g(n^{1-\alpha})^{\max(1,\mu)+\delta}) = O(n^{2\alpha}(n^{1-\alpha+\delta} + n^{(1-\alpha)(2-\delta)})) = O(n^{1+\alpha+\delta} + n^{2-\delta(1-\alpha)})$ by Lemma 11.
3. The whole construction takes linear time in the size of the Set Disjointness instance. This is negligible in comparison to step 4 (where we actually solve the instance).
4. To analyze the running time of this step, we first analyze the parameters of the constructed Set Disjointness instance: Observe that

$$\begin{aligned} U &= m = O(n^{2-2\alpha}), \\ N &= 2g\sqrt{m} = O(n^{\alpha+1-\alpha}) = O(n), \\ s &= O(n^{1-\alpha}), \\ q &= |R| \cdot O(n^{1-\alpha}) = O(n^{1+\alpha}). \end{aligned}$$

   Since these are exactly as claimed in the lemma statement, the efficient Set Disjointness algorithm can solve this instance in time $O(n^{2-\epsilon})$ for some $\epsilon > 0$. Due to the overhead of Observation 21, this step runs in total time $O(t^6 \cdot n^{2-\epsilon} + \frac{qs}{t}) = O(n^{2-\epsilon+6\rho} + n^{2-\rho})$.
5. To analyze the running time of this step, first recall that there are at most $\widetilde{O}(n^{1+\alpha+3\delta})$ many pseudo-solutions (modulo m). Moreover, the pseudo-solutions are in one-to-one correspondence to the queried set intersections. Since we only query the intersections containing at least $t = n^\rho$ elements, there can be at most $\widetilde{O}(n^{1+\alpha+3\delta-\rho})$ that have not been completely answered in the previous step. Each such remaining query takes time $O(s) = O(n^{1-\alpha})$, hence this step takes time $\widetilde{O}(n^{2+3\delta-\rho})$ in total.

Summing over all contributions, the total running time is

$$\widetilde{O}(n + n^{2\alpha} + n^{1+\alpha+\delta} + n^{2-\delta(1-\alpha)} + n^{2-\epsilon+6\rho} + n^{2-\rho} + n^{2+3\delta-\rho}).$$

Recall that $0 \leq \alpha < 1$. Thus by picking e.g. $\rho = \frac{\epsilon}{7} > 0$ and $\delta = \min\{\frac{1-\alpha}{2}, \frac{\rho}{4}\} > 0$ this running time becomes subquadratic, $O(n^{2-\epsilon'})$ for some $\epsilon' > 0$. ◀

Next, let us turn to the analogous proof for Set Intersection. The idea is similar, but requires some tweaks (as in the original paper [44]).

▶ **Theorem 6** (Deterministic Set Intersection Hardness). *Let $0 \leq \alpha < 1$ and $0 \leq \beta \leq 1 - \alpha$. Unless the deterministic 3SUM hypothesis fails, there is no deterministic algorithm for (offline) Set Intersection with parameters $|U| = O(n^{1+\beta-\alpha})$, $N = O(n^{\frac{1}{2}+\frac{\alpha}{2}+\frac{\beta}{2}})$, $s = O(n^{1-\alpha})$ and $q = O(n^{1+\alpha})$ that in total lists up to $O(n^{2-\beta})$ elements and runs in time $O(n^{2-\epsilon})$, for any $\epsilon > 0$.*

With this theorem we prove that Set Intersection has a deterministic matching lower bound whenever the queries are expected to produce at least $\Omega(1)$ elements on average, possibly much more. We remark however that there are some non-trivial cases which are covered by the randomized reduction [44] which we cannot cover here – namely if $1 - \alpha < \beta \leq 1$. This parameterization describes a setting where we expect the total size of the intersections to be much smaller than the total number of queries.

Proof. This proof is overall very similar to the proof of Theorem 4. We will thus be more concise here. Assume that Set Disjointness with parameters as specified in the lemma statement is in subquadratic time; we design a subquadratic algorithm for a given 3SUM instance A:

1. We apply the deterministic self-reduction for 3SUM. Specifically, we apply Lemma 12 to partition A into $g = n^\alpha$ groups $A_1, \ldots, A_g$ of size $O(n^{1-\alpha})$, such that only a subset $R \subseteq [g]^3$ of at most $|R| = O(n^{2\alpha})$ group triples is relevant.
2. We apply the deterministic hashing lemma to find a modulus m under which we have only few false positives in the relevant groups. Specifically, we apply Lemma 11 for the 3SUM instances $(A_i \cup A_j \cup A_k)_{(i,j,k)\in R}$ (of size $O(n^{1-\alpha})$ each) and for $\mu = \frac{1-\alpha+\beta}{1-\alpha} - 2\delta$ and some parameter $\delta > 0$ to be specified later. This either reports that one of the instances contains a 3SUM solution (in which case we can stop), or the lemma returns a modulus $m = O((n^{1-\alpha})^\mu) = O(n^{1-\alpha+\beta})$ satisfying that

$$\sum_{(i,j,k)\in R} \#\{(a,b,c) \in A_i \times A_j \times A_k : a + b \equiv c \pmod m\}$$
$$\leq |R| \cdot O((n^{1-\alpha})^{3-\mu+\delta}) \cdot (\log n)^{O(1/\delta)} = \widetilde{O}(n^{2\alpha+3-3\alpha-1+\alpha-\beta+3\delta}) = \widetilde{O}(n^{2-\beta+3\delta}).$$

   Let us for simplicity assume that $\sqrt{m}$ is an integer; otherwise replace all the following occurrences of $\sqrt{m}$ with an appropriate rounding of $\sqrt{m}$.
3. We construct the Set Intersection instance exactly as in Theorem 4: For each $i \in [g]$ and for all $x, y \in [\sqrt{m}]$, we construct the following two sets:

$$B_{i,x} = \{(b + x\sqrt{m}) \bmod m : b \in A_i\},$$
$$C_{i,y} = \{(c + y) \bmod m : c \in A_i\}.$$

   The Set Intersection instance consists of the sets $\{B_{i,x}\}_{i,x} \cup \{C_{i,y}\}_{i,y}$. The queries are defined as follows: For each $(i,j,k) \in R$ and for each $a \in A_i$, we query the intersection $B_{j,x} \cap C_{k,y}$, where $x, y \in [\sqrt{m}]$ are such that $a \equiv x\sqrt{m} - y \bmod m$. As before, there is a natural one-to-one correspondence between the queried elements in the set intersections and the pseudo-solutions modulo m.
4. This is where the proof diverges. We can simply solve the Set Intersection instance using the oracle. For each reported element in a one of the set intersections, we test in $O(1)$ time whether the corresponding pseudo-solution is a proper 3SUM solution.

This completes the description of the algorithm. The correctness is immediate from the given in-text descriptions. It remains to bound the running time, for which we again analyze the four steps individually:

1. Takes time $\widetilde{O}(n+g^2) = \widetilde{O}(n+n^{2\alpha})$ by Lemma 12.
2. Takes time
$$\widetilde{O}(g(n^{1-\alpha})^{\max(1,\mu)+\delta}) = O(n^{2\alpha}(n^{1-\alpha+\delta}+n^{(1-\alpha)(\frac{1-\alpha+\beta}{1-\alpha}-\delta)})) = O(n^{1+\alpha+\delta}+n^{1+\alpha+\beta-\delta(1-\alpha)})$$
by Lemma 11.
3. The whole construction takes linear time in the size of the Set Intersection instance. This is negligible in comparison to step 4 (where we actually solve the instance).
4. To analyze the running time of this step, we first analyze the parameters of the constructed Set Intersection instance: Observe that
$$\begin{aligned} U &= m = O(n^{1-\alpha+\beta}),\\ N &= 2g\sqrt{m} = O(n^{\alpha+\frac{1}{2}-\frac{\alpha}{2}+\frac{\beta}{2}}) = O(n^{\frac{1}{2}+\frac{\alpha}{2}+\frac{\beta}{2}}),\\ s &= O(n^{1-\alpha}),\\ q &= |R|\cdot O(n^{1-\alpha}) = O(n^{1+\alpha}). \end{aligned}$$
These are exactly as claimed in the lemma statement. Recall further that the total number of elements in all set intersections (i.e., the output size) is exactly the number of pseudo-solutions modulo m. By the second step, this number is bounded by $\widetilde{O}(n^{2-\beta+3\delta})$. This is slightly more than we have claimed in the lemma statement, however, we can simply substitute n for $n' = n^{1+3\delta}$; then the constructed Set Intersection instance has the claimed parameters with respect to n'. The oracle thus takes time $O((n')^{2-\epsilon}) = O(n^{(1+3\delta)(2-\epsilon)}) = O(n^{2-\epsilon+6\delta})$ to solve the instance.

Summing over all contributions, the total running time is
$$\widetilde{O}(n+n^{2\alpha}+n^{1+\alpha+\delta}+n^{1+\alpha+\beta-\delta(1-\alpha)}+n^{2-\epsilon+6\delta}).$$
Recall that $0 \le \alpha < 1$ and $0 \le \beta \le 1-\alpha$. Thus, by picking e.g. $\delta = \min\{\frac{1-\alpha}{2}, \frac{\epsilon}{7}\}$ the running time becomes truly subquadratic, $O(n^{2-\epsilon'})$ for some $\epsilon' > 0$. ◀

References

1 Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Impossibility results for grammar-compressed linear algebra. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *33rd Advances in Neural Information Processing Systems (NeurIPS 2020)*, 2020. URL: `https://proceedings.neurips.cc/paper/2020/hash/645e6bfdd05d1a69c5e47b20f0a91d46-Abstract.html`.

2 Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-SUM lower bounds for approximate distance oracles via additive combinatorics. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 391–404. ACM, 2023. `doi:10.1145/3564246.3585240`.

3 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 Stefano Leonardi and Anupam Gupta, editors, *54th Annual ACM Symposium on Theory of Computing (STOC 2022)*, pages 1487–1500. ACM, 2022. `doi:10.1145/3519935.3520066`.

4 Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *40th International Colloquium on Automata, Languages, and Programming (ICALP 2013)*, volume 7965 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013. `doi:10.1007/978-3-642-39206-1_1`.

5 Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *22th Annual European Symposium on Algorithms (ESA 2014)*, volume 8737 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2014. doi:10.1007/978-3-662-44777-2_1.

6 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014*, volume 8572 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2014. doi:10.1007/978-3-662-43948-7_4.

7 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM J. Comput.*, 47(3):1098–1122, 2018. doi:10.1137/15M1050987.

8 Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. Smallest color-spanning objects. In Friedhelm Meyer auf der Heide, editor, *9th Annual European Symposium on Algorithms (ESA 2001)*, volume 2161 of *Lecture Notes in Computer Science*, pages 278–289. Springer, 2001. doi:10.1007/3-540-44676-1_23.

9 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2014. doi:10.1007/978-3-662-43948-7_10.

10 Amihood Amir, Tsvi Kopelowitz, Avivit Levy, Seth Pettie, Ely Porat, and B. Riva Shalom. Mind the gap! – online dictionary matching with one gap. *Algorithmica*, 81(6):2123–2157, 2019. doi:10.1007/s00453-018-0526-2.

11 Daniel Archambault, William S. Evans, and David G. Kirkpatrick. Computing the set of all the distant horizons of a terrain. *Int. J. Comput. Geom. Appl.*, 15(6):547–564, 2005. doi:10.1142/S0218195905001841.

12 Esther M. Arkin, Yi-Jen Chiang, Martin Held, Joseph S. B. Mitchell, Vera Sacristán, Steven Skiena, and Tae-Heng Yang. On minimum-area hulls. *Algorithmica*, 21(1):119–136, 1998. doi:10.1007/PL00009204.

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

14 Ilya Baran, Erik D. Demaine, and Mihai Patrascu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008. doi:10.1007/s00453-007-9036-3.

15 Gill Barequet and Sariel Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3SUM-hard. *Int. J. Comput. Geom. Appl.*, 11(4):465–474, 2001. doi:10.1142/S0218195901000596.

16 Prosenjit Bose, Marc J. van Kreveld, and Godfried T. Toussaint. Filling polyhedral molds. *Comput. Aided Des.*, 30(4):245–254, 1998. doi:10.1016/S0010-4485(97)00075-4.

17 Karl Bringmann and Nofar Carmeli. Unbalanced triangle detection and enumeration hardness for unions of conjunctive queries. *CoRR*, abs/2210.11996, 2022. doi:10.48550/arXiv.2210.11996.

18 Karl Bringmann and André Nusser. Translating hausdorff is hard: Fine-grained lower bounds for hausdorff distance under translation. In Kevin Buchin and Éric Colin de Verdière, editors, *37th Annual Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SoCG.2021.18.

19 Ayelet Butman, Peter Clifford, Raphaël Clifford, Markus Jalsenius, Noa Lewenstein, Benny Porat, Ely Porat, and Benjamin Sach. Pattern matching under polynomial transformation. *SIAM J. Comput.*, 42(2):611–633, 2013. doi:10.1137/110853327.

20 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. doi:10.1145/3363541.

21 Timothy M. Chan and Sariel Har-Peled. Smallest k-enclosing rectangle revisited. In Gill Barequet and Yusu Wang, editors, *35th Annual Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *LIPIcs*, pages 23:1–23:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.23.

22 Timothy M. Chan and Qizheng He. Reducing 3SUM to convolution-3SUM. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms (SOSA 2020)*, pages 1–7. SIAM, 2020. doi:10.1137/1.9781611976014.1.

23 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing (STOC 2015)*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

24 Lijie Chen, Erik D. Demaine, Yuzhou Gu, Virginia Vassilevska Williams, Yinzhan Xu, and Yuancheng Yu. Nearly optimal separation between partially and fully retroactive data structures. In *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018*, volume 101 of *LIPIcs*, pages 33:1–33:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SWAT.2018.33.

25 Otfried Cheong, Alon Efrat, and Sariel Har-Peled. Finding a guard that sees most and a shop that sells most. *Discret. Comput. Geom.*, 37(4):545–563, 2007. doi:10.1007/s00454-007-1328-5.

26 Raphaël Clifford, Pawel Gawrychowski, Tomasz Kociumaka, Daniel P. Martin, and Przemyslaw Uznanski. The dynamic k-mismatch problem. In *33rd Annual Symposium on Combinatorial Pattern Matching, CPM 2022*, volume 223 of *LIPIcs*, pages 18:1–18:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CPM.2022.18.

27 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. doi:10.1137/070695149.

28 Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 48:1–48:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.48.

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

30 Bartlomiej Dudek, Pawel Gawrychowski, and Tatiana Starikovskaya. All non-trivial variants of 3-LDT are equivalent. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 974–981. ACM, 2020. doi:10.1145/3357713.3384275.

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

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

33 Jeff Erickson, Sariel Har-Peled, and David M. Mount. On the least median square problem. *Discret. Comput. Geom.*, 36(4):593–607, 2006. doi:10.1007/s00454-006-1267-6.

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

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

36 Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *LIPIcs*, pages 42:1–42:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.42.

37 Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. How hard is it to find (honest) witnesses? In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *LIPIcs*, pages 45:1–45:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.45.

38 Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4):22:1–22:25, 2018. doi:10.1145/3185378.

39 Ce Jin and Yinzhan Xu. Removing additive structure in 3SUM-based reductions. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 405–418. ACM, 2023. doi:10.1145/3564246.3585157.

40 Daniel M. Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-sum and related problems. *J. ACM*, 66(3):16:1–16:18, 2019. doi:10.1145/3285953.

41 Adam Karczmarz and Jakub Lacki. Fast and simple connectivity in graph timelines. In Frank Dehne, Jörg-Rüdiger Sack, and Ulrike Stege, editors, *14th International Symposium on Algorithms and Data Structures (WADS 2015)*, volume 9214 of *Lecture Notes in Computer Science*, pages 458–469. Springer, 2015. doi:10.1007/978-3-319-21840-3_38.

42 Mahmoud Abo Khamis, George Chichirim, Antonia Kormpa, and Dan Olteanu. The complexity of boolean conjunctive queries with intersection joins. In Leonid Libkin and Pablo Barceló, editors, *41st ACM Symposium on Principles of Database Systems (PODS 2022)*, pages 53–65. ACM, 2022. doi:10.1145/3517804.3524156.

43 Mahmoud Abo Khamis, Hung Q. Ngo, Christopher Ré, and Atri Rudra. Joins via geometric resolutions: Worst case and beyond. *ACM Trans. Database Syst.*, 41(4):22:1–22:45, 2016. doi:10.1145/2967101.

44 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 1272–1287. SIAM, 2016. doi:10.1137/1.9781611974331.ch89.

45 Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. Monochromatic triangles, intermediate matrix products, and convolutions. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *LIPIcs*, pages 53:1–53:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.53.

46 Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-sum. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 58:1–58:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.58.

47 Hung Q. Ngo, Dung T. Nguyen, Christopher Ré, and Atri Rudra. Beyond worst-case analysis for joins with minesweeper. In *33rd ACM Symposium on Principles of Database Systems (PODS 2014)*, pages 234–245. ACM, 2014. doi:10.1145/2594538.2594547.

48 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing (STOC 2010)*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

49 Jakub Radoszewski, Wojciech Rytter, Juliusz Straszynski, Tomasz Walen, and Wiktor Zuba. Hardness of detecting abelian and additive square factors in strings. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 77:1–77:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.77.

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

51 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.

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