(Multivariate) k-SUM as barrier to succinct computation

Geri Gokaj
Karlsruhe Institute of Technology, Germany

Marvin Künnemann
Karlsruhe Institute of Technology, Germany

Sabine Storandt
University of Konstanz, Germany

Carina Truschel
University of Konstanz, Germany

Abstract

How does the time complexity of a problem change when the input is given *succinctly* rather than explicitly? We study this question for several geometric problems defined on a set X of N points in $\mathbb{Z}^d$. As succinct representation, we choose a sumset (or Minkowski sum) representation: Instead of receiving X explicitly, we are given sets A, B of n points that define X as $A + B = \{a + b \mid a \in A, b \in B\}$.

We investigate the fine-grained complexity of this succinct version for several $\tilde{O}(N)$-time computable geometric primitives. Remarkably, we can tie their complexity tightly to the complexity of corresponding k-SUM problems. Specifically, we introduce as All-ints 3-SUM(n, n, k) the following multivariate, multi-output variant of 3-SUM: given sets A, B of size n and set C of size k, determine for all $c \in C$ whether there are $a \in A$ and $b \in B$ with $a + b = c$. We obtain the following results:

1. Succinct closest L_∞-pair requires time $N^{1-o(1)}$ under the 3-SUM hypothesis, while succinct furthest L_∞-pair can be solved in time $\tilde{O}(n)$.
2. Succinct bichromatic closest L_∞-Pair requires time $N^{1-o(1)}$ iff the 4-SUM hypothesis holds.
3. The following problems are fine-grained equivalent to All-ints 3-SUM(n, n, k): succinct skyline computation in 2D with output size k and succinct batched orthogonal range search with k given ranges. This establishes conditionally tight $\tilde{O}(\min\{nk, N\})$-time algorithms for these problems. We obtain further connections with All-ints 3-SUM(n, n, k) for succinctly computing independent sets in unit interval graphs.

Thus, (Multivariate) k-SUM problems precisely capture the barrier for enabling sumset-succinct computation for various geometric primitives.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Fine-grained complexity theory, sumsets, additive combinatorics, succinct inputs, computational geometry

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Geri Gokaj*: Research supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 462679611.
Carina Truschel: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 251654672 – TRR 161.

Acknowledgements We thank all reviewers for their in-depth review and feedback.

1 Introduction

Consider an algorithmic problem on an input set X of N points in $\mathbb{Z}^d$. How does the time complexity of the problem change when the input is given *succinctly* rather than explicitly?

In this paper, we investigate this question when the succinct representation of X is chosen as a *sumset* (or *Minkowski sum*) representation $X = A + B \coloneqq \{a + b \mid a \in A, b \in B\}$.

Our motivation is threefold:

1. **Complexity on structured input:** How does the time complexity of the problem change when we are supplied with structural information about the input? As a case in point, this question has received considerable interest for the fundamental task of sorting. Indeed, the well-known Sorting $A + B$ problem studied by Fredman [20], and originally posed by Berlekamp[1], asks to determine whether we can compute, given A, B with $|A| = |B| = n$, the sorted order of $A + B$ in time $o(n^2 \log n)$. Since the succinct representation gives a large partial order essentially for free – simply presort A, B in time $O(n \log n)$ –, this question essentially boils to whether or not a partial solution structure helps significantly in determining the full output. The Sorting $A + B$ problem has several connections to problems in computational geometry such as polygon containment, computing the Minkowski sum of orthogonal polygons and minimizing the Hausdorff distance under translation for two segment sets, as described by the works of Barequet and Har-Peled [6], and Barrera [7].
   Another example that has received considerable attention is the *Pareto Sum* problem (see, e.g., [27, 24]), which asks to compute the skyline of $X = A + B$ (see Section 4 for a formal definition). It is only natural to ask analogous questions for other numerical/geometric problems. In particular, a succinct sumset representation reveals additive structure of the problem, which may or may not simplify the algorithmic task considerably.
2. **Algorithmic applications (sumset-compressed computational geometry):** Let $T(N)$ denote the optimal running time for solving a problem of interest on an input set of N points in $\mathbb{Z}^d$. For favorable inputs, it may be possible to find *much smaller* sets A, B with $X = A + B$ – in principle compressions to size $O(\sqrt{N})$ are possible. If we were able to solve the problem of interest in time $O(T(|A| + |B|))$ rather than $T(|X|)$ (aka *decompress-and-solve* running time), we could obtain significantly faster algorithms that reduce the running time from $T(N)$ to $O(T(\sqrt{N}))$. In fact, even small asymptotic improvements over $T(N)$ might be algorithmically interesting.
3. **k-SUM as precise barrier for succinct computations:** Our setup offers another perspective on the k-SUM problem. Specifically, for various problems, we shall prove that refuting a corresponding k-SUM hypothesis is *equivalent* to solving the sumset-succinct version of the problem faster than via decompress-and-solve. That is, despite its simple definition (asking a very basic question about sumsets), the k-SUM problem captures much more general primitives on succinct inputs. We believe that this yields valuable insights into the true expressiveness of k-SUM and the plausibility of the corresponding hardness hypothesis.

1.1 Our setup

For any algorithmic problem g defined on an input set $X \subseteq \mathbb{Z}^d$, we consider the following sumset-succinct version as *succinct-g*:

[1] In the literature, the name Sorting $X + Y$ is more common. However, throughout our paper X will denote the explicit input and $A + B$ its succinct representation.

Problem g		**Problem** succinct-g	
Input:	$X \subseteq \mathbb{Z}^d$ possibly additional inputs z	**Input:**	$A, B \subseteq \mathbb{Z}^d$ possibly additional inputs z
Output:	$g(X, z)$	**Output:**	$g(X, z)$ where $X = A + B$

As a convention, we set $N \coloneqq |X|$, $n \coloneqq |A|$ and $m \coloneqq |B|$. For ease of presentation, we will generally assume that $n = m$, but our results can be adapted to the general case. Throughout the paper, we will assume that all input numbers are chosen from a polynomial range, i.e., $X \subseteq \{N^c, \dots, N^c\}^d$ for some arbitrarily large constant c.

Let us consider a few (near-)linear time solvable problems g. We observe that their succinct versions may have very different time complexities:

A succintly easy problem: Selection

Consider the problem Selection: Given $X \subseteq \mathbb{Z}$ and $1 \leq k \leq N$, return the k-th largest element in X. Selection is well known to be solvable in time $O(N)$; see [8, 15]. Its succinct version succinct-Selection has received considerable attention as well [30, 29, 28, 19] and can be solved significantly faster than decompress-and-solve, specifically, in time $\tilde{O}(n)$.[2]

A succinctly hard problem: Skyline computation

Consider the problem of skyline computation (aka non-dominance filtering or Pareto front computation), see, e.g. [32, 14, 22]: Given $X \subseteq \mathbb{Z}^d$, compute the set Z of non-dominated points, i.e., all $x \in X$ for which there is no $x' \in X$ with $x' \neq x$ and $x \leq x'$ component-wise. This problem can be solved in near-linear time $\tilde{O}(N)$, see, e.g., [32]. Its succinct version is known as Pareto Sum and naturally occurs in several multi-objective optimization algorithms, see Hespe et al. [27]. Notably, a conditional lower bound of $N^{1-o(1)}$ that holds when the output size is $\Theta(\sqrt{N})$ has been proven under the $(\min, +)$-convolution hypothesis [21] and the 3-SUM hypothesis [24]. We shall investigate this problem in much more detail below.

On reductions to k-SUM

The 3-SUM problem[3] is one of the first problems used for conditional lower bounds. Initially introduced to explain various quadratic-time barriers observed in computational geometry [23], it has since been used to show quadratic-time hardness for a wealth of problems from various fields [41, 35, 5, 34, 13]. Its generalization, the k-SUM problem[4], has led to further conditional lower bounds beyond quadratic-time solvable problems [18, 4, 1, 2]. For a more comprehensive overview, we refer to the survey of Williams [43].

An important aspect of our investigations is that we establish *fine-grained equivalences* with k-SUM problems, which requires the somewhat unusual direction of reducing certain problems *to* k-SUM. Conveniently, we may exploit for this task a recent completeness theorem [24]: This work introduces general classes of problems which can be reduced to k-SUM, thereby simplifying the arguments needed to reduce sumset-related

[2] The authors consider a slight variant in which $A + B$ is viewed as a multiset (or $|A + B| = |A| \cdot |B|$).

[3] Given sets A, B, C do there exist $a \in A, b \in B, c \in C$ s.t $a + b + c = 0$.

[4] The k-SUM problem asks, given sets $A_1, \dots, A_k$ of n numbers, whether there exist $a_1 \in A_1, \dots, a_k \in A_k$ such that $\sum_{i=1}^{k} a_i = 0$. The k-SUM hypothesis states that for no $\epsilon > 0$ there exists a $O(n^{\lceil k/2 \rceil - \epsilon})$ time algorithm that solves k-SUM.

problems to k-SUM. Specifically, we shall need the existential fragment of the class $\mathsf{FOP}_{\mathbb{Z}}$: It consists of problems whose inputs are sets $A_1 \subseteq \mathbb{Z}^{d_1}, \ldots, A_k \subseteq \mathbb{Z}^{d_k}$, and free variables $t_1 \ldots t_\ell \in \mathbb{Z}$, and the task is to determine whether there exist $a_1 \in A_1 \ldots a_k \in A_k$ such that $\varphi(a_1[1], \ldots, a_1[d_1], \ldots, a_k[1], \ldots, a_k[d_k], t_1 \ldots, t_\ell)$ holds, where φ is a fixed, but arbitrary linear arithmetic formula over the vector entries of $a_1 \ldots a_k$ and the free variables. (In particular, the dimensions $d_1, \ldots, d_k$ are constant.) Each such problem is near-linear time reducible to k-SUM. We will frequently argue in our reductions that subtasks can be expressed as an existential $\mathsf{FOP}_{\mathbb{Z}}$ formula and exploit the completeness theorem.

Further related work

The authors in [9] explored connections of the $X + Y$ sorting problem with the $(\min, +)$-convolution problem. Other works consider the fine-grained complexity of other notions of succinct inputs, such as representing sequences succinctly as grammar-based compressions [2, 1], or succinctly defined graphs using the notion of factored graphs [25].

1.2 Our results

We investigate the succinct versions of many well-studied geometric primitives solvable in near-linear time. While some turn out to be easy, many others can be tightly connected to the k-SUM problem. In fact, we believe our results may serve as template to classify further problems according to their succinct complexity.

Some succinctly easy problems

For context, we observe that there is a large number of non-trivial geometric primitives (beyond selection) for which fast succinct computation is indeed possible.

As perhaps most prominent and well-known example, consider the convex hull problem: Given $A, B \subseteq \mathbb{Z}^2$, compute the convex hull $\mathsf{conv}(X)$ of $X = A + B$. By a folklore identity, $\mathsf{conv}(A + B) = \mathsf{conv}(A) + \mathsf{conv}(B)$, with complexity $|\mathsf{conv}(A)| + |\mathsf{conv}(B)|$. This yields $\tilde{O}(n)$-time algorithms for succinctly computing convex hulls, see also [17]. This fact leads us to the following general observation.

▶ **Observation 1.** *If we can compute $g(X)$ by a $T(m)$-time algorithm that receives as input only the convex hull of X of size m, then succinct-$g(X)$ can be solved in time $\tilde{O}(T(n))$.*

While the observation is simple, it leads to an interesting class of problems which can be solved in near-linear time in the size of the sumset compression. Some examples of problems which can be efficiently solved using only the convex hull of the input are the diameter of points, the maximum bounding box, the distance between two convex hulls and many more, described by Shamos in [36]. All these problems can be solved with a known technique originally introduced by Shamos [37, 36], which has subsequently been coined *rotating calipers* and further explored by Toussaint [39].

On Closest vs. Furthest Pair

Consider the problem Closest L_∞-Pair: Given $X \subseteq \mathbb{Z}^d$, determine $\min_{x_1, x_2 \in X, x_1 \neq x_2} \|x_1 - x_2\|_\infty$. Its natural maximization analogue is Furthest L_∞-Pair: Given $X \subseteq \mathbb{Z}^d$, determine $\max_{x_1, x_2 \in X} \|x_1 - x_2\|_\infty$. Both problems are solvable in near-linear time $\tilde{O}(N)$ whenever d is constant [22, 10].

However, we observe that the succinct versions differ substantially in their time complexity: Succinct-Furthest L_∞-Pair can be solved in time $\tilde{O}(n)$, while for succinct-Closest L_∞-Pair we obtain a 3-SUM-based lower bound of $N^{1-o(1)}$ already in 1D, using the fact that the EqDist problem is 3-SUM-hard [6]; see the full version of the paper.

Naturally, we may also consider the bichromatic case of the Closest L_∞-Pair problem: Given $X, Y \subseteq \mathbb{Z}^d$, determine $\min_{x\in X, y\in Y} \|x-y\|_\infty$. Here, the natural succinct version receives sets $A, B, C, D \subseteq \mathbb{Z}^d$ as inputs, and the task is to solve the problem for $X = A + B$ and $Y = C + D$. We prove that this succinct-Bichromatic L_∞-Pair problem is *fine-grained equivalent* to 4-SUM. The reduction to 4-SUM follows from observing that the decision problem can be formulated as an existential $\mathsf{FOP}_\mathbb{Z}$ formula, which can be reduced to 4-SUM [24]. It remains an interesting question whether the *monochromatic* succinct Closest-L_∞-Pair is fine-grained equivalent to 3-SUM (our results place it between 3-SUM and 4-SUM). For further fine-grained complexity results on Closest and Furthest Pair in higher dimensions; see e.g., [42] and [31].

Multivariate 3-SUM

The remaining problems that we study will exhibit a tight connection to a multivariate version of the 3-SUM problem. Specifically, we introduce the following problem:

▶ **Definition 2** (All-ints 3-SUM(n,n,k))**.** *Given sets $A, B \subseteq \mathbb{Z}$ of size n, and a set $C \subseteq \mathbb{Z}$ of size k. Determine for each c in C if there exists $(a,b) \in A \times B$ with $a + b = c$.*

It is easy to see that this problem can be solved in time $\tilde{O}(\min\{kn, n^2 + k\})$: The $O(kn)$ bound follows by solving, for each $c \in C$, the corresponding 2-SUM problem of detecting $a \in A, b \in B$ with $a + b = c$, which is well-known to be solvable in time $O(n)$. The $\tilde{O}(n^2 + k)$ upper bound follows from computing $A+B$ explicitly, and solving the corresponding dictionary problem for each $c \in C$. This time bound turns out to be conditionally tight; see Section 3: For any $k = \Theta(n^\alpha)$ with $0 < \alpha \leq 1$, we show using standard techniques that this problem cannot be solved in time $O((nk)^{1-\epsilon})$ with $\epsilon > 0$ unless the 3-SUM hypothesis is false. Note that this establishes a conditionally tight answer for all k: If $k = \Theta(n^\alpha)$ with $\alpha > 1$, then we obtain a $n^{2-o(1)}$ conditional lower bound via padding, and a $\Omega(k)$ lower bound follows from the input size of the problem.

Perhaps surprisingly, this problem naturally captures the precise fine-grained complexity of various succinct geometric primitives. Note that when $k = n$, All-ints 3-SUM(n,n,k) is well known to be subquadratic equivalent to 3-SUM [40], but a fine-grained equivalence for $k = \Theta(n^\alpha)$ with $\alpha < 1$ does not appear to follow from standard techniques. We will thus make use of a slight generalization of the completeness theorem for existential $\mathsf{FOP}_\mathbb{Z}$ formulas, which is implicit in [24]. Consider the following problems: Given $A \subseteq \mathbb{Z}^{d_1}, B \subseteq \mathbb{Z}^{d_2}, C \subseteq \mathbb{Z}^{d_3}$ of sizes n, n and k, respectively, compute for each $c \in C$, whether there exist $a \in A, b \in B$ such that a linear arithmetic formula $\varphi(a[1], \ldots, a[d_1], b[1], \ldots, b[d_2], c[1], \ldots, c[d_3])$ over the entries in the vectors a, b, c holds. For every linear arithmetic formula φ, we can reduce this problem to $\tilde{O}(1)$ instances of All-ints 3-SUM(n,n,k). A standalone proof of this fact can be found in the full version of the paper.

Succint orthogonal range queries

Consider a *batched* (or *offline*) version of answering orthogonal range queries over X, which we call Batched Orthogonal Range Searching: Given $X \subseteq \mathbb{Z}^d$ and orthogonal ranges $r_1, \ldots, r_k$, determine for each $j = 1, \ldots, k$ whether $X \cap r_j$ is empty. This problem can be solved in

time $\tilde{O}(N+k)$ for constant d, see, e.g. [12]. We observe that succinct-Batched Orthogonal Range Searching is fine-grained equivalent to All-ints 3-SUM(n,n,k). To reduce from All-ints 3-SUM(n,n,k), use A,B as succinct input sets for a 1-dimensional instance of succinct-Batched Orthogonal Range Searching and for $C=\{c_1,\ldots,c_k\}$ choose $r_i=[c_i,c_i]$. The other direction follows from observing that we may represent the succinct Orthogonal Range Searching problem as a suitable All-ints-$\mathsf{FOP}_{\mathbb{Z}}$ formula. The details can be found in the full version of the paper.

Succinct skyline (Pareto Sum)

Recall the prominent geometric problem of skyline computation (aka non-dominance filtering or maxima of point sets computation), see, e.g. [32, 14, 22]: Given $X\subseteq\mathbb{Z}^d$, compute the set Z of non-dominated points, i.e., all $x\in X$ for which there is no $x'\in X$ with $x'\neq x$ and $x\leq x'$ component-wise. This (explicit) problem can be solved in time $O(N\log k)$ [32], where $k=|Z|$ denotes the output size. For $d=2$, Hespe et al. show how to compute its succinct variant, the Pareto Sum, in time $O(n\log n+kn)$. We prove that computing all k non-dominated points of $X=A+B$ is fine-grained equivalent to All-ints 3-SUM(n,n,k).

Starting point of our reduction from All-ints 3-SUM(n,n,k) to the Pareto Sum is the subquadratic reduction from 3-SUM to Pareto Sum given in [24], which uses Conv3-SUM as an intermediate step. Notably, however, our multivariate setting poses a perhaps unexpected challenge: The standard reductions from 3-SUM to Conv3-SUM [35, 11] do not readily transfer to a multivariate setting. Instead, we introduce a variant of Conv3-SUM that we call All-ints-MixedConv3-SUM(n,n,k)[5]:

▶ **Definition 3** (All-ints-MixedConv3-SUM). *Given sequences $A[0],\ldots,A[k-1]$ and $B[0],\ldots,B[k-1]$, where each set $A[i],B[j]$ is a* set *of $O(n/k)$ integers, and a sequence of integers $C[0],\ldots,C[k-1]$, determine for each $0\leq i<k$ whether there exists some j and elements $a\in A[j],b\in B[i-j]$ such that $a+b=C[i]$.*

This problem can easily be solved in time $O(kn)$. We carefully adapt the reduction from 3-SUM to convolution 3-SUM given in [11] to reduce All-ints 3-SUM(n,n,k) to All-ints-MixedConv3-SUM(n,n,k). As a further technical challenge, our approach requires an equivalence between the problem of computing the complete Pareto Sum and computing the Pareto Sum inside a given query range. Specifically, in our lower bound, we need to allow as query ranges quadrants of the form $[x,\infty)\times[y,\infty)$; with this choice, even for sets A,B with a Pareto Sum as large as $\Omega(n^2)$ one may still give a query range with arbitrary small outputsize k, e.g., $k=O(1)$. Interestingly, using an involved construction detailed in Section 4.1, we show that this problem reduces to its specical case of computing the complete Pareto set under the promise that its size is $k+O(1)$. As a by-product, this also shows that for any $n\in\mathbb{N}$, there exist sets A,B of size n of incomparable vectors in $\mathbb{Z}^2$ with a Pareto Sum of constant size, which does not appear to have been proven before. After showing this equivalence, we reduce All-ints-MixedConv3-SUM to Pareto Sum, reminiscent of the reduction from Convolution 3-SUM to Pareto Sum in [24].

The reverse reduction from Pareto Sum to All-ints 3-SUM(n,n,k) is also technically interesting: while Hespe et al. [27] show how to compute each of the k output points successively in $\tilde{O}(n)$ time per point, unfortunately, we cannot follow this approach in our reduction (calling an All-ints 3-SUM(n,n,k)-oracle $\Omega(k)$ times would trivially exceed $\tilde{O}(kn)$

[5] The definition appears related in spirit (but different) to Factored 3-SUM defined in [16].

running time). Instead, we aim to identify the k output points essentially simultaneously. To this end, we perform a divide-and-conquer approach. We maintain a set of already identified Pareto Sum points S, as well as a list of up to k *active* x-intervals – an interval is deemed active if it contains at least one (unidentified) point of the Pareto Sum. At each level of the recursion, we split all active intervals into two halves. The key observation is the following: Given S and an arbitrary list of intervals $I_1, \dots, I_s$, we can compute for each I_j whether it contains a point in the Pareto Sum (but not already in S) using only $\tilde{O}(1)$ All-ints $\mathsf{FOP}_{\mathbb{Z}}$ (n, n, s) queries, which in turn can be reduced to $\tilde{O}(1)$ All-ints 3-SUM(n, n, k) instances. After $O(\log n)$ levels of the recursion, each interval consists of at most a single integer, and can be resolved fully. The details can be found in Section 4.

Succinct independent set in unit interval graphs

Consider the maximum independent set problem (MIS) in unit interval graphs: Given a set of (open) unit intervals $\mathcal{I}$ in $\mathbb{Z}$, compute a maximum independent set S^*, i.e., a pairwise non-intersecting subset of intervals maximizing its size. A simple greedy algorithm is well known to solve the problem in time $O(n \log n)$, see, e.g. [26]. Taking the output size $k = |S^*|$ into account, this can be slightly improved to time $O(n \log k)$, see [38].

For studying its connection with All-ints 3-SUM(n, n, k), the technical work spent for the equivalence to Pareto Sum pays off: On the lower bound side, we establish a fine-grained reduction from All-ints 3-SUM(n, n, k) to succinct-MIS in unit interval graphs using a similar approach as for the Pareto Sum. In particular, we may reduce immediately from All-ints MixedConv3-SUM(n, n, k), and transfer conceptually similar arguments to the MIS setting.

On the upper bound side, we are not quite able to establish the reverse reduction. However, we show how to reduce computing a *maximal* independent set to All-ints 3-SUM(n, n, k), again exploiting careful parallel binary searches in a divide-and-conquer approach. It remains an interesting open problem to reduce the MIS problem in unit interval graphs to All-ints 3-SUM(n, n, k). The details can be found in the full version of the paper.

Summary

For the problems equivalent to All-ints 3-SUM(n, n, k), our results show that improving over decompress-and-solve running time $N^{1\pm o(1)}$ is possible whenever the output size k is small enough, specifically $k = O(N^{1/2-\epsilon}) = O(n^{1-\epsilon})$ for some $\epsilon > 0$. We refer to Figure 1 for an overview of the reductions shown. In the full version of the paper, we provide a table which summarizes runtimes for various problems, when defined succinctly and explicitly.

2 Preliminaries

We make use of basic tools in fine-grained complexity, for a comprehensive overview; see the survey [43]. For our purposes, the basic notion of *fine-grained reduction* is as follows. Consider problems P_1, P_2, with presumed time complexities T_1,T_2, we say there is a fine-grained reduction from P_1 to P_2 if for all $\epsilon > 0$ there exists $\delta > 0$ and an algorithm A solving P_1 with oracle access to P_2 such that if all calls to P_2 are replaced by an $O(T_2(n)^{1-\epsilon})$-time algorithm, then A runs in time $O(T_1(n)^{1-\delta})$. We say problems P_1, P_2 are fine-grained equivalent if there exist fine-grained reductions from P_1 to P_2 and P_2 to P_1. Throughout the paper, M will always denote a large number which we set as follows: With $A, B, C \subseteq \mathbb{Z}^d$, we let $M := 10 \cdot \max\{\|a\|_1 + \|b\|_1 + \|c\|_1 : a \in A, b \in B, c \in C\}$.

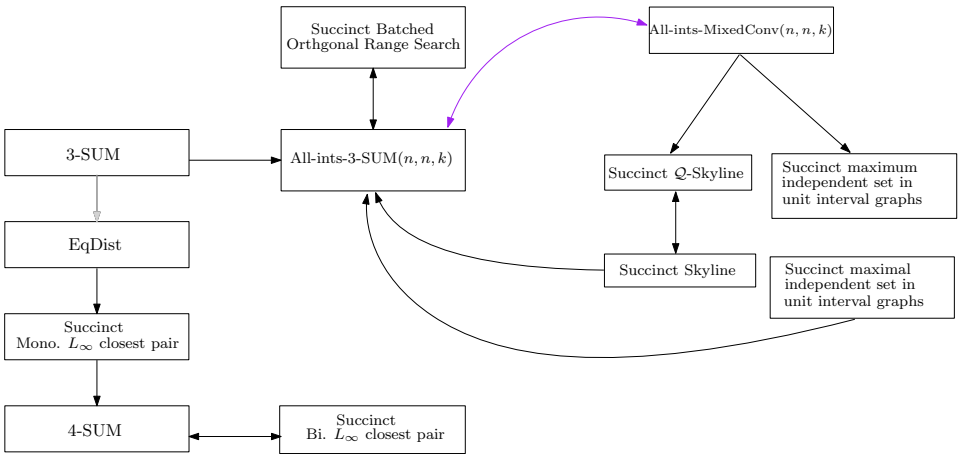

Figure 1 The above figure gives an overview of our most important results. The black arrows, denote the reductions, we have shown. The single gray arrow (reduction) from 3-SUM to the problem EqDist has been shown by Barequet and Har-Peled [6]. Furthermore, the violet arrow i.e the equivalence between All-ints 3-SUM(n,n,k) and All-intsMixedConv(n,n,k), holds only for the case, when $k \in \Theta(n^\alpha)$, where $\alpha \in (0,1]$.

3 Multivariate 3-SUM variants

Let us recall two central problems for our results.

▶ **Definition 4** (All-ints 3-SUM(n,n,k)). *Given sets A, B of size n, and a set C of size k. Determine for each c in C if there exists $(a,b) \in A \times B$ with $a+b=c$.*

▶ **Definition 5** (All-ints-MixedConv3-SUM(n,n,k)). *Given sequences of sets $A[0]\dots A[k-1]$ and $B[0]\dots B[k-1]$ of size $O(n/k)$, and a sequence of integers $C[0],\dots,C[k-1]$, determine for every $C[i]$ if there exists an element $a \in A[j]$ and an element $b \in B[i-j]$ such that $a+b=C[i]$.*

In order to show an equivalence between All-ints 3-SUM(n,n,k) and All-ints-MixedConv3-SUM, we make use of a family of hash functions $h : [U] \to [m]$ defined by $h(x) := x \bmod p$ over a uniform random prime p selected in $[m/2, m)$. These hash functions satisfy the following properties.

- almost-linearity, that is $h(x+y) = h(x) + h(y)$ or $h(x+y) = h(x) + h(y) - p$.
- $O(\log(U))$-universality, that is $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] = O(\frac{\log(U)}{m})$.

Throughout, we assume that $A, B, C \subseteq [U]$ with $U = O(n^3)$ which can be achieved using a standard universe reduction. Furthermore, the above hash functions exhibit the following additional property.

▶ **Lemma 6** (Overfull buckets [11]). *Let $\mathcal{H}$ be a family of d-universal hash functions $[U] \to [m]$. Furthermore, let $h \in \mathcal{H}$ be chosen randomly. Let S be a set of n integers in $[U]$. We call x bad iff the number of elements equal to $h(x)$ (excluding x) exceeds a parameter t. The expected number of bad elements x in S is at most $\frac{dn}{m} \cdot \frac{1}{t} \cdot n$.*

We adapt the approach of Chan and He [11] (originally used to show a deterministic reduction from Convolution 3-SUM to 3-SUM) to show the following.

▶ **Lemma 7.** *Let $k = \Theta(n^\alpha)$, where $\alpha \in (0, 1]$. If we can solve All-ints-MixedConv3-SUM in time $O((nk)^{1-\epsilon})$ for some $\epsilon > 0$, then we can solve All-ints 3-SUM(n, n, k) in time $O((nk)^{1-\epsilon'})$ for some $\epsilon' > 0$.*

Proof. We reduce from All-ints 3-SUM(n, n, k). Let p be a prime number chosen from $[m/2, m)$ where $m = \frac{dk}{R}$ with $d = O(\log(U)) = O(\log(n))$ and R to be chosen later. Let $h(x) := x \bmod p$ and $C'[i] := \{c \in C : h(c) = i\}$, $A'[i] := \{a \in A : h(a) = i\}$, $B'[i] := \{b \in B : h(b) = i\}$. We call elements in $c \in C$ bad if $|C'[i]| > t$ where $i = h(c)$. Furthermore, we call elements $a \in A$ bad if $|A'[i]| > t \cdot n/m$ where $i = h(a)$. Similarly, we call elements $b \in B$ bad if $|B'[i]| > t \cdot n/m$ where $i = h(b)$.

The expected number of bad elements in C is $O\left(\frac{dk}{\frac{dk}{R}} \cdot \frac{1}{t}\right) \cdot k = O\left(\frac{Rk}{t}\right)$. The expected number of bad elements in A and B is $O\left(\frac{dn}{m} \cdot \frac{1}{\frac{n}{m} \cdot t} \cdot n\right) = O\left(\frac{nd}{t}\right)$.

By making use of Markov's inequality for the sets A, B, C separately, the probability that the number of bad A, B and C elements exceed 4 times its expectation is at most $1/4$ each. Thus by applying the union bound, there exists a hash function h, represented by some suitable prime p, where the number of bad elements for each set A, B, C is bounded by at most 4 times the number of expected bad elements.

We find such a hash function by going naively through all $O(m/\log(m))$ primes $p \in [m/2, m)$. The primes can be generated in time $\tilde{O}(m)$ by the sieve of Eratosthenes, and computing the bucket sizes takes time $O(m/\log(m)) \cdot O(n) = \tilde{O}(\frac{dkn}{R})$.

We proceed now as follows: For the bad elements in C, we perform a bruteforce 2-SUM algorithm, which needs time $O(n)$. There are at most $O(\frac{Rk}{t})$ many bad elements, thus this takes time $O(\frac{Rkn}{t})$.

For the bad elements in A and B, we also perform a simple bruteforce algorithm, which needs time $O(k)$, resulting in a runtime $O(k \cdot \frac{nd}{t}) = \tilde{O}(\frac{nk}{t})$.

For the remaining good elements in C, which occur at most t times, we proceed as follows: For each possible candidate in $[t]$, we perform an All-ints-MixedConv3-SUM instance for both cases of the almost-additiveness. For this, we get a runtime of $\tilde{O}(t \cdot (n \cdot \frac{k}{R})^{1-\epsilon})$. Concluding, we get a total runtime of

$$\tilde{O}\left(\frac{kn}{R}\right) + O\left(\frac{Rkn}{t}\right) + \tilde{O}\left(\frac{nk}{t}\right) + \tilde{O}\left(t \cdot (n \cdot \frac{k}{R})^{1-\epsilon}\right).$$

By setting $t = R^2$, we equate the first two terms, reducing to a runtime of:

$$\tilde{O}\left(\frac{kn}{R}\right) + \tilde{O}\left(\frac{nk}{R^2}\right) + \tilde{O}\left(R^2 \cdot \left(n \cdot \frac{k}{R}\right)^{1-\epsilon}\right),$$

which is bounded by $\tilde{O}\left(\frac{kn}{R}\right) + \tilde{O}\left(R^{1+\epsilon} \cdot (n \cdot k)^{1-\epsilon}\right)$. We balance the expression by setting $R = (nk)^{\epsilon/(2+\epsilon)}$, resulting in the final runtime $\tilde{O}((nk)^{1-\epsilon'})$, for an $\epsilon' > 0$. We remove the polylog factors by choosing an ϵ^* such that $0 < \epsilon^* < \epsilon'$, and get the desired runtime $O((nk)^{1-\epsilon^*})$. ◀

The reverse reduction is simpler and follows the standard reduction from convolution 3-SUM to 3-SUM.

▶ **Lemma 8.** *Let $k = \Theta(n^\alpha)$, where $\alpha \in (0, 1]$. If All-ints 3-SUM(n, n, k) can be solved in time $T(n, k)$, then we can solve All-ints MixedConv3-SUM(n, n, k) in time $O(T(n, k))$.*

Proof. We first convert the All-intsMixed-Conv3-SUM (n, n, k) problem into an All-ints 3-SUM (n, n, k) problem. Consider the newly created sets A', B', C', and a large number M as follows:

$$A' := \{a + iM : i \in \{0, \dots, k-1\} \wedge a \in A[i]\}$$
$$B' := \{b + iM : i \in \{0, \dots, k-1\} \wedge b \in B[i]\}$$
$$C' := \{c[i] + iM : i \in \{0, \dots, k-1\}\}$$

Due to the large choice of M it holds that there exist $a' \in A', b' \in B', c' \in C'$ such that $a' + b' = c'$ iff there exists $i, j, k \in \{0, \dots, k-1\}$ such that $i + j = k$ and there exists $a \in A[i], b \in B[j]$ such that $a + b = C[k]$.

◀

▶ **Lemma 9** (All-ints $\mathsf{FOP}_{\mathbb{Z}}$ queries [24]). *Let $A_1 \subseteq \mathbb{Z}^{d_1}, A_2 \subseteq \mathbb{Z}^{d_2}, A_3 \subseteq \mathbb{Z}^{d_3}$, with $|A_1| = k, |A_2| = |A_3| = n$, and $\varphi(a_1, a_2, a_3)$ be a quantifier free linear arithmetic formula. If we can solve All-ints 3-SUM(n, n, k) in time $T(n, k)$, then we can determine for each $a_1 \in A_1$, whether there exist $a_2 \in A_2, a_3 \in A_3$ such that $\varphi(a_1, a_2, a_3)$ is satisfied in time $\tilde{O}(T(n, k))$.*

The details can be found in the full version of the paper. We conclude with a simple 3-SUM lower bound for All-ints 3-SUM(n, n, k).

▶ **Lemma 10.** *Let $k = \Theta(n^\alpha)$ for $\alpha \in (0, 1]$. If we can solve All-ints 3-SUM (n, n, k) in time $O((nk)^{1-\epsilon})$ for some $\epsilon > 0$, then the 3-SUM hypothesis fails.*

Proof. We reduce from 3-SUM and separate the set C into $O(n/k)$ sets of size $O(k)$. We perform then $O(n/k)$ instances of All-ints 3-SUM(n, n, k) that run in time $O(n/n^\alpha) \cdot O((n \cdot n^\alpha)^{1-\epsilon}) = O(n^{2-\epsilon'})$ for some $\epsilon' > 0$. ◀

4 Output sensitive Skyline

We recall the two variants of the Pareto Sum problem.

▶ **Definition 11** (Succinct Skyline/Pareto Sum). *Given a succinctly defined set $X = A + B \subseteq \mathbb{Z}^2$, the skyline (aka Pareto front) of X consists of all non-dominated vectors in X. More formally compute the following set: $\mathsf{PS}(A, B) := \{x \in X : \text{There is no } x' \in X \setminus \{x\} : x \leq x'\}$, where we denote the output size by $k := |\mathsf{PS}(A, B)|$.*

▶ **Definition 12** ($\mathcal{Q}$-Succinct Skyline/$\mathcal{Q}$-Pareto Sum). *Given a succinctly defined set $X = A + B \subseteq \mathbb{Z}^2$, and a quadrant $\mathcal{Q} = [\alpha_1, \infty) \times [\alpha_2, \infty)$ the skyline (aka Pareto front) of X consists of all non-dominated vectors in X inside $\mathcal{Q}$. More formally compute the following set: $\mathsf{PS}_{\mathcal{Q}}(A, B) := \{x \in X = A + B : x \in \mathcal{Q} \text{ and there is no } x' \in X \setminus \{x\} : x \leq x'\}$, where again we denote the output size by $k := |\mathsf{PS}_{\mathcal{Q}}(A, B)|$.*

In the following section, we show that the above problems are fine-grained equivalent, which will be necessary to show the fine-grained equivalence between the Pareto Sum problem and All-ints 3-SUM(n, n, k).

4.1 On the equivalence of succinct skyline and $\mathcal{Q}-$succinct skyline

▶ **Theorem 13.** *If we can compute the Pareto Sum in time $T(n, k)$, then the $\mathcal{Q}$-Pareto Sum of an instance $A, B, \mathcal{Q}$ with $|A| = |B| = n$ and output size k can be computed in time $\mathcal{O}(T(n, k))$.*

Proof. Assume the quadrant to be given in the form $\mathcal{Q} = [L.x, \infty) \times [L.y, \infty)$. The goal is to construct a Pareto Sum instance where only Pareto points $c \in A + B$ with $c[0] \geq L.x$ and $c[1] \geq L.y$ are part of the output. The reduction consists of the following steps:

- Step 1: Transform the point coordinates in A, B such that the minimum coordinate value is at least 1 and adapt $L.x$ and $L.y$ accordingly. Define M larger than the largest possible point coordinate in $A + B$ and set $A' := A \cup A^+$ and $B' = B \cup B^+$ where $A^+ = \{(-5M + L.x, 7M), (-3M + L.x, 5M), (9M, -7M), (11M, -9M)\}$ and $B^+ = \{(-5M, 7M), (-M, 3M), (3M, -M)\}$.
- Step 2: Transform the point coordinates in A' and B' such that the minimum coordinate values are again at least 1 and adjust $L.y$ accordingly. Let M' be larger than the new maximum coordinate of any point in $A' + B'$. Augment A' with points A^{++} and B' with points B^{++}, which are derived from A^+, B^+ by swapping their x- and y-coordinates, respectively, and replacing M by M' and $L.x$ by (the transformed) $L.y$.
- Step 3: Compute the Pareto Sum C' of A', B'.
- Step 4: Delete all points with coordinates smaller than 1 or larger than M' from C', transform the coordinates of the remaining points in C' by reversing the transformations applied in Step 4, delete again elements with coordinate values smaller than 1 or larger than M and finally transform the coordinates once more by reversing the transformation applied in Step 2.

Applying the steps 1 and 2 takes time $\mathcal{O}(n)$ each. Step 3 takes time $T(\max\{|A'|, |B'|\}, |C'|)$ with $\max\{|A'|, |B'|\} = n+8$. Step 4 takes time in $\mathcal{O}(|C'|)$. Below, we argue that the returned result is correct and that $|C'| \in \mathcal{O}(k)$.

After the coordinate transformation in Step 1, the elements in A and B have coordinates in $\{1, \ldots, M-1\}$. Figure 2 illustrates the impact of adding A^+ and B^+ to A and B, respectively. Firstly, we observe that all elements $c = a + b$ with $a \in A^+$ or $b \in B^+$ – except for $A_2^+ + B_3^+$ – have either $c[0] < M$ or $c[1] < M$. Accordingly, none of these elements can dominate any element in $A + B$ as their maximum coordinate is bounded by M. The point $A_2^+ + B_3^+ = (L, 4M)$, however, dominates all points in $A + B$ with an x-coordinate of at most $L.x$. These points are all outside of $\mathcal{Q}$ based on our definition of $L.x$. Next, we observe that only 12 points $c = a + b$ with $a \notin A$ or $b \notin B$ are non-dominated, see again Figure 2. For cells having a colored point in Figure 2 we argue in the following that they dominate the corresponding colored ranges by comparing the x- and y-coordinates of the dominating point to the x- and y-coordinates of the dominated range.

- $A_1^+ + B_3^+ = (-2M + L, 6M)$ dominates $A_2^+ + B$ (yellow) since $-2M + L > -3M + L + 1$ and $-2M + L > -2M + L - 1$ for the x-coordinates and for the y-coordinates $6M > 6M - 1$ and $6M > 5M + 1$.
- $A_2^+ + B_2^+ = (-4M + L, 8M)$ dominates two ranges, namely $A_1^+ + B$ and $A + B_1^+$ (orange). The former is dominated since $-4M + L > -5M + L - 1$ and $-4M + L > -4M + L - 1$ hold for the x-coordinates and $8M > 8M - 1$ and $8M > 7M + 1$ for the y-coordinates. For the latter range we conclude that $-4M + L > -5M + 1$ and $-4M + L > -4M - 1$ for the x-coordinates and $8M > 8M - 1$ and $8M > 7M + 1$ for the y-coordinates.
- $A_2^+ + B_3^+ = (L, 4M)$ dominates $A + B_2^+$ (purple) since $L > -M + 1$ and $L > -1$ for the x-coordinates and $4M > 4M - 1$ and $4M > 3M + 1$ for the y-coordinates.
- $A_3^+ + B_1^+ = (4M, 0)$ dominates $A + B_3^+$ (light green) since $4M > 3M + 1$ and $4M > 4M - 1$ for the x-coordinates and $0 > -1$ and $0 > -M + 1$ for the y-coordinates.
- $A_3^+ + B_3^+ = (12M, -8M)$ dominates $A_4^+ + B$ (turquoise) since $12M > 11M + 1$ and $12M > 12M - 1$ for the x-coordinates and $-8M > -8M - 1$ and $-8M > -9M + 1$ for the y-coordinates.

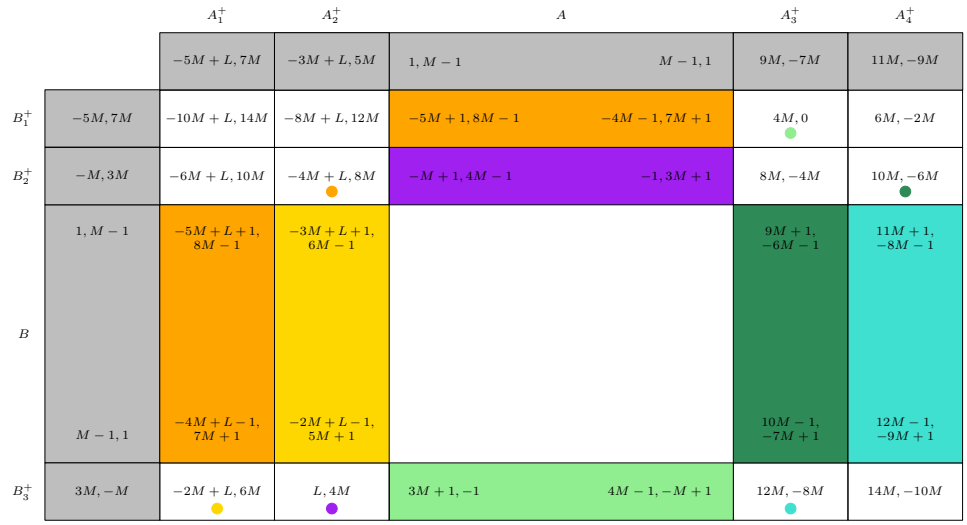

Figure 2 $\mathcal{Q}$-Pareto Sum instance augmented with A^+, B^+. The colored regions are dominated by the element in the cell with the dot of matching color. L refers to $L.x$. By definition, we have $L \leq M$.

- $A_4^+ + B_2^+ = (10M, -6M)$ dominates $A_3^+ + B$ (dark green) since $10M > 9M + 1$ and $10M > 10M - 1$ for the x-coordinates and $-6M > -6M - 1$ and $-6M > -7M + 1$ for the y-coordinates.

Step 2 repeats the very same process of coordinate transformation and augmentation, now with the goal to introduce a filter for the points with a y-coordinate smaller than $L.y$. The arguments from above apply again. Therefore, computing the Pareto Sum on A', B' in Step 3 results in C' of size at most $k + 24$. Thus, the filtering and back transformation in Step 4 takes time in $\mathcal{O}(k)$. Summing up over all steps, the running time is in $\mathcal{O}(n + k + T(n, k)) = \mathcal{O}(T(n, k))$.

After completing Step 4, C' is guaranteed to contain all points of the $\mathcal{Q}$-Pareto Sum C of A, B that are strictly inside $\mathcal{Q}$, as the points with an x-coordinate of at most $L.x$ and a y-coordinate of at most $L.y$ are dominated by the newly introduced elements in the augmented sumset, and Pareto points that result from the augmentation steps are filtered. ◀

We remark that the reduction in the other direction is trivial. Given an instance A, B of Pareto Sum, we can construct an instance of $\mathcal{Q}$-Pareto Sum with the same solution by enclosing all points in $A+B$ inside a large quadrant i.e $\mathcal{Q} = [-M, \infty) \times [-M, \infty)$. Accordingly, these two problems are fine-grained equivalent.

Moreover, our reduction technique can be leveraged to show that there are arbitrarily large Pareto Sum instances, where A and B consist of incomparable vectors, for which the output size k is constant. In prior work, it was only shown that for an instance with $|A| = 2$ and $|B| = 5$ an output size $k = 4 < \max(|A|, |B|)$ is possible [33]. But the question whether there exist sets A, B with $|A|, |B| = n$ and a Pareto Sum of size $k \in o(n)$ has been open. Given any Pareto Sum instance, we can define $L.x$ as the maximum x coordinate of any Pareto point in $A + B$. Using the augmentation described in Step 2 in the proof of Theorem 13, all Pareto points of $A + B$ will be dominated and only 12 new Pareto points will be introduced. Thus, we get the following corollary.

▶ **Corollary 14.** *For any $n \in \mathbb{N}$, there exist sets A, B of incomparable vectors with $|A| = |B| = n$ and output size of the Pareto Sum $k \in \mathcal{O}(1)$.*

4.2 A lower bound from All-ints 3-SUM(n, n, k)

▶ **Lemma 15.** *If we can compute the output-sensitive $\mathcal{Q}$-Pareto Sum of size $k = \Theta(n^\alpha)$ for $\alpha \in (0, 1]$ for sets A, B in time $O(T(n, k))$, then we can solve All-ints MixedConv3-SUM(n, n, k) in time $O(T(n, k))$.*

Proof. Consider an All-ints MixedConv3-SUM(n, n, k) instance given by array of sets $A[0, \ldots, k-1], B[0, \ldots, k-1]$ of size $O(n/k)$, and the array $C[0, \ldots, k-1]$. Let $A' := \{a + 100iM : i \in \{0, \ldots, k-1\} \wedge a \in A[i]\}$, $B' := \{b + 100iM : i \in \{0, \ldots, k-1\} \wedge b \in B[i]\}$ $C' := \{C[i] + 100iM : i \in \{0, \ldots, k-1\}\}$. It is clear that the newly constructed sets A', B' are of size at most n and the set C' of size at most k. Let $t := 2M$, and by the above construction, it holds that for an index $i \in \{0, \ldots, k-1\}$ there does not exist a $j \in \{0, \ldots, k-1\}$, where $j \leq i$ and $a \in A[j], b \in B[i-j]$ such that $a+b = C[i]$ if and only if for all $a' \in A', b' \in B'$ there exists $c' \in C'$, where one of the following two inequalities holds: $c' + 1 \leq a' + b' \leq c' + 1 + t$ or $c' - t - 1 \leq a' + b' \leq c' - 1$. Let us create the following set of the left side of the inequalities observed, that is $\tilde{C} := \bigcup_{c \in C'} \{c' + 1, c' - t - 1\}$.

We will construct the following sets in $\mathbb{Z}^2$, in order to integrate both above inequalities in a Pareto Sum problem.

$$A^* := \left\{ \begin{pmatrix} a \\ -a \end{pmatrix} : a \in A' \right\}, B^* := \left\{ \begin{pmatrix} b \\ -b \end{pmatrix} : b \in B' \right\}, C^* := \left\{ \begin{pmatrix} c + t \\ -c \end{pmatrix} : c \in \tilde{C} \right\}$$

We perform an output sensitive $\mathcal{Q}$-Pareto Sum computation on the following sets, with a quadrant $\mathcal{Q}$, which we determine in a bit.

$$H = A^* \cup \left(C^* + \left\{ \begin{pmatrix} 1000kM \\ -1000kM \end{pmatrix} \right\} \right), G = B^* \cup \left\{ \begin{pmatrix} -1000kM \\ 1000kM \end{pmatrix} \right\}$$

The sumset $H + G$ then consists of the following sumsets:

$$A^* + B^*,\ A^* + \left\{ \begin{pmatrix} -1000kM \\ 1000kM \end{pmatrix} \right\},\ C^* + B^* + \left\{ \begin{pmatrix} 1000kM \\ -1000kM \end{pmatrix} \right\},\ C^*.$$

We set the quadrant $\mathcal{Q} = [\min_{c \in C^*} c[0], \infty) \times [\min_{c \in C^*} c[1], \infty)$, thus forcing the output-sensitive $\mathcal{Q}$-Pareto Sum to ignore everything besides the points in $A^* + B^*$ and C^*.

Let us first argue that the points in C^* need to be part of the output-sensitive skyline. We argue based on the indices $i \in \{0, \ldots, k-1\}$, which separate the points in $A^* + B^*$ and C^* into clusters of points. For each $i, j \in \{0, \ldots, k-1\}$ the points in $A^* + B^*$ are of the form $\begin{pmatrix} a + 100iM + b + 100jM \\ -a - b - 100iM - 100jM \end{pmatrix} = \begin{pmatrix} (a+b) + (i+j)100M \\ -(a + b + (i+j)100M) \end{pmatrix}$. For each $i \in \{0, \ldots, k-1\}$ the two points originally induced by index i, now in C^* are of the form $\tilde{c}_i^1 := \begin{pmatrix} c[i] + 100iM + 1 + t \\ -(c[i] + 100iM + 1) \end{pmatrix}, \tilde{c}_i^2 := \begin{pmatrix} c[i] + 100iM - t - 1 + t \\ -(c[i] + 100iM - t - 1) \end{pmatrix} = \begin{pmatrix} c[i] + 100iM - 1 \\ -(c[i] + 100iM - t - 1) \end{pmatrix}$. For each $i \in \{0, \ldots, k-1\}$, we now observe the points which are included in the Pareto Sum. Clearly, the Pareto front, will contain $\tilde{c}_i^1$ as it is the point with the largest x-coordinate among the points landing at the cluster of points induced by index i. Furthermore, the point $\tilde{c}_i^2 = \begin{pmatrix} c[i] + 100iM - 1 \\ -(c[i] + 100iM - t - 1) \end{pmatrix}$ is contained in the Pareto Sum as it is the point with the highest y-coordinate among the points landing at the cluster of points induced by index i. Let us prove the following claim.

▷ Claim 16. The Pareto front of the cluster of points induced by index i will contain precisely the points $\{\tilde{c}_i^1, \tilde{c}_i^2\}$ if and only if there is no $j \in \{0, \ldots, k-1\}$ such that there exists an $a \in A[j]$ and $b \in B[i-j]$ with $a + b = C[i]$.

Proof. As argued before the points $\{\tilde{c}_i^1, \tilde{c}_i^2\}$, will always be in the Pareto Sum induced by index i. Assume there exists an index j such that there exists an $a \in A[j]$ and $b \in B[i-j]$ such that $a + b = C[i]$, then in the sumset $A^* + B^*$ there will be a point γ_i of the form:

$$\gamma_i = \begin{pmatrix} a + 100jM + b + 100(i-j)M \\ -(a + 100jM + b + 100(i-j)M) \end{pmatrix} = \begin{pmatrix} c[i] + 100iM \\ -(c[i] + 100iM) \end{pmatrix}.$$

The point $\tilde{c}_i^1 = \begin{pmatrix} c[i] + 100iM + 1 + t \\ -(c[i] + 100iM + 1) \end{pmatrix}$ does not dominate γ_i because of the y-coordinate. Furthermore, the point $\tilde{c}_i^2 = \begin{pmatrix} c[i] + 100iM - 1 \\ -(c[i] + 100iM - t - 1) \end{pmatrix}$ does not dominate γ_i because of the x-coordinate. Lastly, it is easy to see that the only points not dominated by $\tilde{c}_i^1$ or $\tilde{c}_i^2$ in the cluster of points induced by index i need to be of the form γ_i. Concluding, the Pareto front of the cluster of points induced by index i will contain precisely the points $\{\tilde{c}_i^1, \tilde{c}_i^2, \gamma_i\}$ if and only if there exists a $j \in \{0, \ldots, k-1\}$ and $a \in A[j]$ and $b \in B[i-j]$ with $a + b = C[i]$. ◀

After having proven the claim, it remains to scan through the Pareto Sum and identify whether the Pareto front induced by each index i consists of two points, concluding the reduction from All-ints MixedConv3-SUM(n, n, k). Lastly, notice that the size of the Pareto front evaluated is at most $O(k)$. ◀

By a simple padding argument we can now show the following lower bound.

▶ **Lemma 17.** *Let $\epsilon > 0$. There is no output-sensitive algorithm to compute the Pareto Sum of sets A, B of size n in time $O(n^{2-\epsilon})$ with guaranteed output size $\Theta(n^\alpha)$, where $1 \leq \alpha < 2$ and $k := n^\alpha$, unless All-ints 3-SUM(n, n, k) can be solved in time $O(n^{2-\epsilon'})$ for an $\epsilon' > 0$.*

Proof. Case $\alpha \leq 1$ is shown in Lemma 15, by making use of the fine-grained equivalence between All-ints 3-SUM(n, n, k) and All-ints-MixedConv-3-SUM(n, n, k).

We reduce from the case $\alpha = 1$, let $\beta = \alpha - 1$. We construct an artificial blowup of the Pareto Sum output set of size n^β. To this end, take a subset of B' of size n^β from the set B. Let the vectors in B' be numbered arbitrarily $b_1, \ldots, b_m$. Construct the following set:

$$\bar{B} := B \cup \underbrace{\bigcup_{i=1}^{m} \left\{ b_i + \begin{pmatrix} i \cdot M \\ -i \cdot M \end{pmatrix} \right\}}_{:=\tilde{B}}.$$

The Pareto Sum C will now consist of the original Pareto Sum of size n combined, with $A + \tilde{B}$ of size $n^{1+\beta}$. This holds due to the fact that all vectors in $A + \tilde{B}$ are incomparable to each other, and incomparable to the vectors $A + B$, due to the shift chosen for each vector. ◀

Let us now make use of the 3-SUM hardness of All-ints 3-SUM(n, n, k) to show the following lower bound, by a simple combination of Lemma 15 and Lemma 17.

▶ **Theorem 18.** *Let $k = \Theta(n^\alpha)$, where $\alpha \in (0, 2)$. There is no algorithm to solve output sensitive Pareto Sum of output size k, in time $O(\min\{n^2, nk\}^{1-\epsilon})$ for an $\epsilon > 0$, unless the 3-SUM hypothesis fails.*

Proof. We perform a simple case distinction depending on α, if $0 < \alpha \leq 1$ we apply Lemma 15 together with the fine-grained equivalence between All-ints 3-SUM(n, n, k) and All-ints MixedConv-3-SUM(n, n, k). In the case of $1 < \alpha < 2$, we similarly apply Lemma 17 together with the fact that All-ints 3-SUM(n, n, k) is 3-SUM hard. ◀

4.3 A reduction to All-ints 3-SUM(n, n, k)

▶ **Lemma 19.** *Given sets $A, B \subseteq \mathbb{Z}^2$ of size n and closed intervals $I_1 = [\alpha_1, \omega_1], \ldots, I_k = [\alpha_k, \omega_k]$. We can compute for all intervals I_j the highest point in $A + B$ with x-coordinate in I_j in time $\tilde{O}(T(n, k))$ if All-ints 3-SUM(n, n, k) can be solved in time $T(n, k)$.*

Proof. We consider a parallel binary search approach. Let $\gamma_1, \ldots, \gamma_k, \in [-M, M]$ and $C := \{(\alpha_i, \omega_i, \gamma_i) : i \in \{1, \ldots, k\}\}$. An application of All-ints$\mathsf{FOP}_{\mathbb{Z}}$ query to the set C on the formula

$$\exists a \in A \exists b \in B \exists c \in C : c[0] \leq a[0] + b[0] \wedge a[0] + b[0] \leq c[1] \wedge a[1] + b[1] \geq c[2],$$

gives us an answer for each interval I_j, where $j \in \{1, \ldots, k\}$, whether there exist points $a \in A, b \in B$ such that the x-coordinate of $a + b$ lies on the interval I_j and the y-coordinate of $a + b$ is higher than γ_j. By appropriately changing the value of γ_j, for each interval and constructing a new formula in a logarithmic number of steps, we can find the height y_j^* of the highest point in each interval I_j via a parallel binary search.

We now continue computing the x-coordinate of the highest point for each interval, which we can also compute by a binary search. Consider $\delta_1, \ldots, \delta_k$, where $\delta_i = \lfloor (\omega_i + \alpha_i)/2 \rfloor$ and construct the following set $C' = \{(\alpha_i, \omega_i, \delta_i, y_i^*) : i \in \{1, \ldots, k\}\}$. An application of All-ints $\mathsf{FOP}_{\mathbb{Z}}$ query to the set C' of the formula

$$\exists a \in A \exists b \in B \exists c' \in C' : c[0] \leq a[0]+b[0] \wedge a[0]+b[0] \leq c[1] \wedge a[0]+b[0] \geq c[2] \wedge a[1]+b[1] = c[3]$$

gives us an answer for each Interval I_j, whether there exists $a \in A, b \in B$ such that the x-coordinate of $a + b$ lies on or to the right of δ_j and the y-coordinate of $a + b$ is precisely the y-coordinate of the highest point in the interval I_j.

Thus, by appropriately changing the value of δ_i for each interval and constructing a new formula, in a logarithmic number of steps, we can find a corresponding x-coordinate to the highest y-coordinate in each interval via a parallel binary search. Notice that we can choose to binary search in such a way as to find for each interval the rightmost point, which equals the highest y-coordinate. Lastly, we can also deduce if there is no point $a + b$ inside an interval at all.

For the runtime notice that a parallel binary search on an All-ints $\mathsf{FOP}_{\mathbb{Z}}$ query runs in time $\tilde{O}(T(n, k)) \cdot O(\log U) = \tilde{O}(T(n, k))$ by Lemma 9, where $U \in O(poly(n))$ denotes the size of the universe. ◀

▶ **Theorem 20.** *If we can solve All-ints 3-SUM(n, n, k) in time $T(n, k)$, then we can compute an output sensitive Pareto Sum in time $\tilde{O}(T(n, k))$.*

Proof. We perform a divide and conquer approach, where we keep a collection of disjoint intervals $\mathcal{I}$, with the invariant that every interval $I \in \mathcal{I}$ contains at least one point the Pareto front of $A + B$ whose x-coordinate lies in I. Furthermore, keep a set S which stores the points of the skyline of $A + B$.

Start with the set of intervals given by one interval concluding the x-coordinate of all points, $\mathcal{I} := \{I_1 = [-M, M]\}$. We now state the general procedure.

Firstly, compute the highest points inside all intervals in $\mathcal{I}$ by an application of Lemma 19. In the beginning we directly, store this point in the set S as it must be part of the Pareto front of $A + B$.

Now, for each interval $I_j = [\alpha, \omega]$ in $\mathcal{I}$, we create two intervals $I_j^{(1)} := [\alpha, m-1]$ and $I_j^{(2)} := [m, \omega]$, where $m = \lfloor(\alpha+\omega)/2\rfloor$ is the middle x-coordinate of the interval I_j. Perform a call on Lemma 19 with A, B and the set of intervals $\mathcal{I}' := \{I_1^{(1)}, I_1^{(2)} \ldots, I_l^{(1)}, I_l^{(2)}\}$, to compute the highest points of the non-empty intervals. In the case of several highest points in an interval, Lemma 19 computes the rightmost highest point. We can now deduce for each of the intervals in $\mathcal{I}'$ if it contains at least one point in the Pareto front of $A+B$. Clearly if the interval is empty, we can disregard it and keep the non-empty half. Assume for I_j, we have that $I_j^{(1)}$ and $I_j^{(2)}$ are both non-empty, with the highest points being s_1 and s_2 respectively. If s_2 dominates the point s_1, we can disregard the interval $I_j^{(1)}$, as it can not be part of the Pareto front of $A+B$, any further subdivision of $I_j^{(1)}$ would create only more dominated points by monotonicity. Otherwise, we know $s_1[0] < s_2[0]$ and $s_1[1] > s_2[1]$. Let d_1 and d_2, be the rightmost highest points of the intervals $[m, M]$ and $[\omega+1, M]$ respectively, which can be computed by a call to Lemma 19 for all non-empty intervals $I_1^{(1)}, I_1^{(2)} \ldots, I_l^{(1)}, I_l^{(2)}$. We will keep the interval $I_j^{(1)}$ (store s_1 in S) iff d_1 does not dominate s_1, and will keep the interval $I_j^{(2)}$ (and store s_2 in S) iff the point d_2 does not dominate s_2. Continue recursively, with the set of non-disregarded intervals $\mathcal{I}'$, and the stored Pareto front points S, where we terminate the recursion and return the set S when all intervals in $\mathcal{I}$ are of length 1.

For the correctness, we remark that the invariant holds that every interval in $\mathcal{I}$ holds at least one point in the Pareto front. For the runtime, notice that the length of the intervals is halved in every recursion step. Finally, we have a recursion depth of at most $\log_2 U$, where $U \in O(poly(n))$ denotes the size of the universe, and in each recursion, we perform a call to Lemma 19 with at most $4k$ intervals (including the computation of d_1 and d_2). Thus, we get a final runtime of $\tilde{O}(T(n,k))$. ◀

5 Conclusion

We conclude with some open questions. It appears difficult to reduce All-ints 3-SUM(n,n,k) to 3-SUM when $k \ll n$, which would lead to a fine-grained equivalence between the problems. A first step might be to investigate if All-ints 3-SUM(n,n,k) admits an efficient self-reduction. Furthermore, it remains to classify further problems into being succinctly-easy or succinctly-equivalent to variants of k-SUM. Finally, it is interesting to study good representations of arbitrary sets X as sumsets $A+B$. In recent work [3], it was shown that determining perfect representability is NP-hard. However, perhaps we can efficiently compute approximate representations of sumsets, which may even lead to efficient (approximation) algorithms for explicit versions of various geometric primitives.

References

1 Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 192–203. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.26.

2 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, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/645e6bfdd05d1a69c5e47b20f0a91d46-Abstract.html.

3 Amir Abboud, Nick Fischer, Ron Safier, and Nathan Wallheimer. Recognizing sumsets is np-complete. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 4484–4506. SIAM, 2025. doi:10.1137/1.9781611978322.153.

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, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, 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 and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.53.

6 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.

7 Antonio Hernández Barrera. Finding an o(n2 log n) algorithm is sometimes hard. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, page 289–294. Carleton University Press, 1996.

8 Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973. doi:10.1016/S0022-0000(73)80033-9.

9 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Puatracscu, and Perouz Taslakian. Necklaces, convolutions, and X+Y. *Algorithmica*, 69(2):294–314, 2014. URL: https://doi.org/10.1007/s00453-012-9734-3, doi:10.1007/S00453-012-9734-3.

10 Timothy M. Chan. Orthogonal Range Searching in Moderate Dimensions: k-d Trees and Range Trees Strike Back. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SoCG.2017.27, doi:10.4230/LIPIcs.SoCG.2017.27.

11 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, Salt Lake City, UT, USA, January 6-7, 2020*, pages 1–7. SIAM, 2020. doi:10.1137/1.9781611976014.1.

12 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the ram, revisited. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13-15, 2011*, pages 1–10. ACM, 2011. doi:10.1145/1998196.1998198.

13 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 Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1501–1514. ACM, 2022. doi:10.1145/3519935.3520032.

14 Wei-Mei Chen, Hsien-Kuei Hwang, and Tsung-Hsi Tsai. Maxima-finding algorithms for multidimensional samples: A two-phase approach. *Comput. Geom.*, 45(1-2):33–53, 2012. URL: https://doi.org/10.1016/j.comgeo.2011.08.001, doi:10.1016/J.COMGEO.2011.08.001.

15 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

16 Mina Dalirrooyfard, Andrea Lincoln, and Virginia Vassilevska Williams. New techniques for proving fine-grained average-case hardness. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 774–785. IEEE, 2020. doi:10.1109/FOCS46700.2020.00077.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.

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

19 Greg N. Frederickson and Donald B. Johnson. The complexity of selection and ranking in X+Y and matrices with sorted columns. *J. Comput. Syst. Sci.*, 24(2):197–208, 1982. doi:10.1016/0022-0000(82)90048-4.

20 Michael L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976. doi:10.1016/0304-3975(76)90078-5.

21 Daniel Funke, Demian Hespe, Peter Sanders, Sabine Storandt, and Carina Truschel. Pareto sums of pareto sets: Lower bounds and algorithms. *Algorithmica*, pages 1–34, 2025.

22 Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. Scaling and related techniques for geometry problems. In Richard A. DeMillo, editor, *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 135–143. ACM, 1984. doi:10.1145/800057.808675.

23 Anka Gajentaan and Mark H. Overmars. On a class of o(n2) problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

24 Geri Gokaj and Marvin Künnemann. Completeness theorems for k-sum and geometric friends: Deciding fragments of linear integer arithmetic. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 55:1–55:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.ITCS.2025.55, doi:10.4230/LIPICS.ITCS.2025.55.

25 Shreya Gupta, Boyang Huang, Russell Impagliazzo, Stanley Woo, and Christopher Ye. The computational complexity of factored graphs. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 58:1–58:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. URL: https://doi.org/10.4230/LIPIcs.ITCS.2025.58, doi:10.4230/LIPICS.ITCS.2025.58.

26 Udaiprakash I. Gupta, D. T. Lee, and Joseph Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982. URL: https://doi.org/10.1002/net.3230120410, doi:10.1002/NET.3230120410.

27 Demian Hespe, Peter Sanders, Sabine Storandt, and Carina Truschel. Pareto sums of pareto sets. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ESA.2023.60, doi:10.4230/LIPICS.ESA.2023.60.

28 Donald B. Johnson and Samuel D. Kashdan. Lower bounds for selection in x + y and other multisets. *J. ACM*, 25(4):556–570, October 1978. doi:10.1145/322092.322097.

29 Donald B. Johnson and Tetsuo Mizoguchi. Selecting the kth element in X + Y and x_1 + x_2 + ... + x_m. *SIAM J. Comput.*, 7(2):147–153, 1978. doi:10.1137/0207013.

30 Haim Kaplan, László Kozma, Or Zamir, and Uri Zwick. Selection from heaps, row-sorted matrices, and X+Y using soft heaps. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San*

Diego, CA, USA, volume 69 of *OASIcs*, pages 5:1–5:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: https://doi.org/10.4230/OASIcs.SOSA.2019.5, doi:10.4230/OASICS.SOSA.2019.5.

31 Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. *Comb.*, 40(4):539–573, 2020. URL: https://doi.org/10.1007/s00493-019-4113-1, doi:10.1007/S00493-019-4113-1.

32 David G. Kirkpatrick and Raimund Seidel. Output-size sensitive algorithms for finding maximal vectors. In Joseph O'Rourke, editor, *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 89–96. ACM, 1985. doi:10.1145/323233.323246.

33 Kathrin Klamroth, Bruno Lang, and Michael Stiglmayr. Efficient dominance filtering for unions and minkowski sums of non-dominated sets. *Computers & Operations Research*, 163:106506, 2024.

34 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016. URL: https://doi.org/10.1137/1.9781611974331.ch89, doi:10.1137/1.9781611974331.CH89.

35 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010. doi:10.1145/1806689.1806772.

36 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Texts and Monographs in Computer Science. Springer, 1985. doi:10.1007/978-1-4612-1098-6.

37 MI Shamos. Computational geometry[ph. d. thesis]. 1978.

38 Jack Snoeyink. Maximum independent set for intervals by divide and conquer with pruning. *Networks*, 49(2):158–159, 2007. URL: https://doi.org/10.1002/net.20150, doi:10.1002/NET.20150.

39 Godfried T Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

40 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.67.

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

42 Ryan Williams. On the difference between closest, furthest, and orthogonal pairs: Nearly-linear vs barely-subquadratic complexity. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1207–1215. SIAM, 2018. doi:10.1137/1.9781611975031.78.

43 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.