Approximating Pareto Sum via Bounded Monotone Min-Plus Convolution

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

The Pareto sum of two-dimensional point sets P and Q in $\mathbb{R}^2$ is defined as the skyline of the points in their Minkowski sum. The problem of efficiently computing the Pareto sum arises frequently in bi-criteria optimization algorithms. Prior work establishes that computing the Pareto sum of sets P and Q of size n suffers from conditional lower bounds that rule out strongly subquadratic $O(n^{2-\epsilon})$-time algorithms, even when the output size is $\Theta(n)$. Naturally, we ask: How efficiently can we *approximate* Pareto sums, both in theory and practice? Can we beat the near-quadratic-time state of the art for exact algorithms?

On the theoretical side, we formulate a notion of additively approximate Pareto sets and show that computing an approximate Pareto set is *fine-grained equivalent* to Bounded Monotone Min-Plus Convolution. Leveraging a remarkable $\tilde{O}(n^{1.5})$-time algorithm for the latter problem (Chi, Duan, Xie, Zhang; STOC '22), we thus obtain a strongly subquadratic (and conditionally optimal) approximation algorithm for computing Pareto sums.

On the practical side, we engineer different algorithmic approaches for approximating Pareto sets on realistic instances. Our implementations enable a granular trade-off between approximation quality and running time/output size compared to the state of the art for exact algorithms established in (Funke, Hespe, Sanders, Storandt, Truschel; Algorithmica '25). Perhaps surprisingly, the (theoretical) connection to Bounded Monotone Min-Plus Convolution remains beneficial even for our implementations: in particular, we implement a simplified, yet still subquadratic version of an algorithm due to Chi, Duan, Xie and Zhang, which on some sufficiently large instances outperforms the competing quadratic-time approaches.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Computational geometry

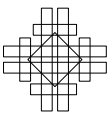
Keywords and phrases computational geometry, fine-grained complexity, algorithm engineering

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.54

Related Version *Full Version*: `https://arxiv.org/pdf/2603.25449`

Supplementary Material *Software (Source Code)*: `https://github.com/ggokaj/SoCG26-ParetoSum` archived at `swh:1:dir:53715b7d04ec81c190b784371b4675bc28183aee`
Dataset (Benchmark): `https://doi.org/10.5281/zenodo.19260269`

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



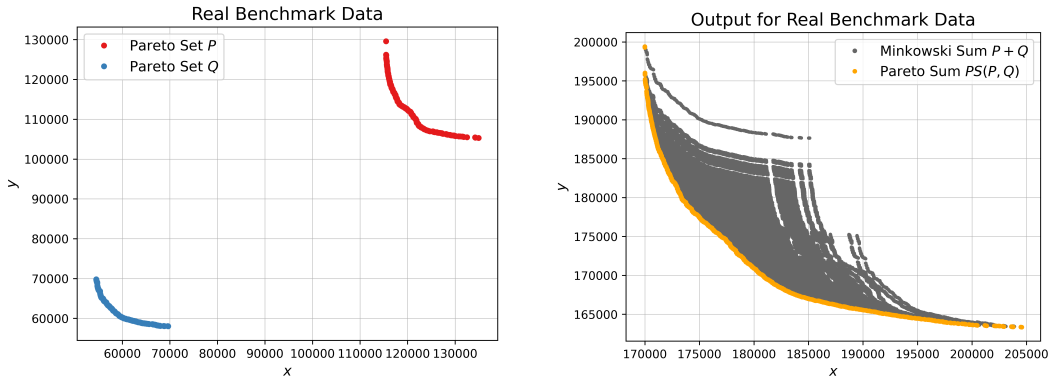


Figure 1 Two Pareto sets P, Q on the left arising from bi-criteria route planning. Depicted on the right is the Minkowski sum $P + Q$ with the Pareto sum $PS(P, Q)$.

1 Introduction

In multi-criteria decision-making processes and in algorithms for multi-objective optimization, a central computational task is to obtain and filter non-dominated solutions. This is essential in order to keep intermediate solution sets interpretable and computationally tractable. Often, partial solutions must be combined in the course of the algorithm, and non-dominated filtering must be applied to the resulting set. This core problem is known as *Pareto sum*: Given two sets $P, Q \subset \mathbb{R}^2$ of non-dominated points[1] (also called skylines or Pareto sets), the Pareto sum $PS(P, Q)$ is defined as the skyline of their Minkowski sum $P + Q := \{p + q \mid \ p \in P, q \in Q\}$.

Recently, exact algorithms have been developed for Pareto sum computation with a running time in $\tilde{O}(n^2)$ [13]. Naturally, the question arises whether this algorithm admits the best-possible time complexity for this problem or whether there is hope for designing asymptotically faster ones. The study of fine-grained complexity has emerged as a powerful framework for understanding the algorithmic complexity of problems within the class P. Using fine-grained reductions, the conjectured intractability of one key problem can be transferred to another problem, yielding a conditional lower bound on how fast the other problem can be solved. For the Pareto sum problem, it was recently shown that an algorithm with a running time in $\mathcal{O}(\min\{n^2, nk\}^{1-\delta})$ for $\delta > 0$, where k denotes the output size, would disprove the 3-SUM hypothesis [15]. The conditional lower bound matches the complexity of known algorithms (up to subpolynomial factors), which renders the existence of subquadratic algorithms for Pareto sum (even with linear output sizes) unlikely.

However, in practical applications, Pareto sum instances of substantial input size n and with output sizes $k \in \Omega(n)$ frequently arise. On such instances, exact computation takes at least quadratic time, which is typically impractical. This motivates our study of approximation algorithms for Pareto sum. We define a point set $\tilde{S}$ to be an approximation of $S := PS(P, Q)$ if for each element s in S there is a point $\tilde{s}$ in $\tilde{S}$ of "similar quality" (specifically, $\tilde{s} \leq s+(\Delta, \Delta)$) and each $\tilde{s} \in \tilde{S}$ is indeed close to a point in $P + Q$ (specifically $p + q \leq \tilde{s} \leq p + q + (\Delta, \Delta)$ for some $p \in P, q \in Q$). This provides an additive approximation guarantee. This definition caters well to many practical applications of Pareto sum computation. For example, in bi-criteria route planning, the goal is to find the set of Pareto-optimal paths between the source and the target node with respect to two metrics. Route planning techniques for this

[1] We call a set S of points non-dominated, if there do not exist distinct vectors $s, s' \in S$ such that $s \leq s'$.

setup heavily rely on Pareto sum computations and the number of resulting skyline points is oftentimes huge [13, 16, 25]. Thus, the goal is to reduce the number of Pareto-optimal paths to be considered within the algorithm and also as an output for the user. Multiplicative approximation algorithms tend to not work well in this scenario, because for paths with a value of zero or close to zero in one metric, even slightly suboptimal paths induce huge approximation factors. Additive approximations do not suffer from the same issue and are also much easier to interpret for a user (for example, if it is guaranteed that the reported route has at most 2 minutes of extra driving time). In this paper, we propose an efficient additive-approximation algorithm that allows one to trade running time against solution quality. We rigorously analyze its theoretical properties and also provide an extensive empirical study of its performance on synthetic and real data sets.

1.1 Related Work

The Pareto sum problem has already been extensively studied from a theoretical and practical perspective. A simple approach follows by computing the Minkowski sum explicitly and then applying general algorithms for skyline computation [17, 10]. In [18], the practical relevance of the problem was emphasized, and novel algorithms were proposed that achieve a subquadratic space consumption for certain input set structures. Algorithms with guaranteed output-sensitive space consumption were proposed in [13]. A simple Sort & Compare algorithm was proven to run in $\mathcal{O}(n^2 \log n)$ on input sets of size n, and a sweep search algorithm to run in $\mathcal{O}(nk + n \log n)$ where k denotes the output size. Experiments on synthetic instances and instances stemming from bi-criteria route planning in road networks confirm that these algorithms outperform those that fully compute the Minkowski sum and also the ones proposed in [18]. In [19], Pareto sum computation is used as a subroutine for multiobjective spanning tree or TSP computation which uses a tree decomposition as basis. For join nodes, an algorithm similar to Sort & Compare is used to obtain the desired combined result. In [14], several additional fine-grained complexity results for Pareto sum are provided. This includes a quadratic lower bound for Pareto sum computation even for $k = \Theta(n)$ under the 3-SUM hypothesis (which is even more believable than the previous min-plus convolution quadratic lower bound by [13], as the min-plus convolution hypothesis implies the 3-SUM hypothesis). Moreover, they established Pareto sum as a complete problem for a wide class of logical and geometric problems, further adding interest to the problem from the side of complexity theory.

1.2 Our Contributions

In this paper, we explore the degree to which the quadratic-time bottleneck for the Pareto sum problem – which is observable both in practice [13] and substantiated by conditional lower bounds [13, 14, 15] – can be circumvented by additively *approximating* Pareto sums. To this end, in Section 3, we formally introduce a natural relaxation of Pareto sums to Δ-approximate Pareto sums: intuitively, $\tilde{S}$ is an approximation of the true Pareto set S of P, Q if for any $s \in S$, we have an approximate point $\tilde{s}$ of "similar quality" (specifically, $\tilde{s} \leq s + (\Delta, \Delta)$). Conversely, any such $\tilde{s} \in \tilde{S}$ is of similar quality to some actual point in $P + Q$ (i.e., there are $p \in P, q \in Q$ with $p + q \leq \tilde{s} \leq p + q + (\Delta, \Delta)$).[2]

[2] In fact, we often even achieve the stronger guarantee that $\tilde{S} \subseteq P + Q$, see Definition 5.

Crucially, our results rely on establishing a certain fine-grained *equivalence* between approximating Pareto sums and Bounded Monotone Min-Plus Convolution, the latter of which has received increasing interest in recent years [9, 11, 4, 6, 12]. This leads to the following theoretical results:

- For any sets P, Q of at most n points in $[0, W]^2$, we can compute an ϵW-approximate Pareto sum $\tilde{S}$ in time $\tilde{O}(n + (1/\epsilon)^{1.5})$. This leverages a remarkable algorithm due to Chi, Duan, Xie and Zhang [11] solving Bounded Monotone Min-Plus Convolution in strongly subquadratic time $\tilde{O}(n^{1.5})$.
- Conversely, we show that any $\tilde{O}(n + (1/\epsilon)^{1.5-\delta})$-time algorithm for computing an ϵW-approximate Pareto sum would give a polynomial-factor improvement for Bounded Monotone Min-Plus Convolution. This establishes a fine-grained equivalence of the approximate Pareto sum and Bounded Min-Plus Convolution. Some approximation problems [8, 7] are known to be fine-grained equivalent to min-plus convolution, but this appears to be the first such connection for bounded monotone min-plus convolution.
- We observe that slightly stronger approximation guarantees can be obtained if a *witness-reporting* version of Bounded Monotone Min-Plus Convolution can be achieved. Indeed, we show that such witnesses can be reported in strongly subquadratic time, by adapting a $\tilde{O}(n^{1.6})$-time algorithm for Bounded Monotone Min-Plus Convolution [11] using techniques due to Alon and Naor [1].

Subsequently, we observe how beneficial our relaxation to approximate Pareto sums can be on realistic input sets. To this end, we empirically fine-tune and evaluate a number of different approaches, leading to the following results:

- On synthetic and realistic inputs, we can approximate Pareto sums significantly more efficiently than computing them exactly. E.g., on generated data sets with 10^6 points and coordinates in $[0, 2 \cdot 10^6]$, computing an additive ($\Delta = 20$)-approximation saves a computation time of up to three orders of magnitude. Furthermore, the size of the returned sets reduces significantly with coarser approximation parameters. Thus, we enable an effective choice of running time/solution size vs. approximation quality.
- We focus on efficiently solving the *Bounded* Pareto sum problem, which denotes the special case in which all coordinates are integer values in $\{0, \dots, W-1\}$ – such instances occur naturally in our approximation algorithms (after a scaling step), but are of independent interest in bi-criteria optimizations with integral cost functions (e.g., in routing). Here, we empirically evaluate and compare several algorithmic approaches: (1) a natural adaptation of the Sort & Compare method of [13] to BucketSort & Compare, (2) a convex-pruning algorithm inspired by a fast algorithm for Near-convex Min-Plus Convolution [5], and (3) an algorithm based on a $\tilde{O}(n^{1.6})$-time algorithm for Bounded Monotone Min-Plus Convolution [11] (which we will refer to as CDXZ algorithm). Perhaps surprisingly, we observe that depending on the class of inputs, either one of these approaches may outperform the others. This indicates that the (theoretical) connection between Pareto sums and Bounded Monotone Min-Plus convolution can also be exploited in practice to obtain practical algorithms (depending on the instances).

We feel that our results add practical evidence (and extend theoretical evidence, e.g. [21, 8, 3]) that studying the connections of geometric (and other) problems with certain formulations of convolution-type problems is beneficial for algorithmic research.

2 Bounded Pareto Sum

In this section, we prove our core reduction which is the basis for developing new algorithms for (approximate) Pareto sum computation. Throughout the paper, we abbreviate $[n] := \{0, 1, \ldots, n-1, n\}$ and use the interval notation $[a, b] := \{a, a+1, \ldots, b-1, b\}$ for some $a, b \in \mathbb{Z}$, with $a \leq b$. We say that a point (x, y) dominates another point (x', y'), if $x \leq x'$ and $y' \leq y$ with at least one coordinate being strictly smaller. A point set is called a Pareto set if it contains no dominated elements. For a point p, we denote by $p.x$ and $p.y$ the x and y-coordinate of p respectively. The Pareto front of a set S, consists of the non-dominated points in S. We are now ready to define the problems of interest.

▶ **Definition 1** (Bounded Pareto Sum). *Given Pareto sets P, Q of size n with integer coordinates in $[W]^2$, compute the set S of non-dominated points in $P+Q := \{p+q \mid p \in P, q \in Q\}$.*

We remark that for bounded Pareto sum, we always have $W \geq n-1$, as in a Pareto set the coordinates per dimension all need to be different.

▶ **Definition 2** (Bounded Monotone Min-Plus Convolution). *Given two monotone decreasing arrays A and B of size n with positive entries bounded by $W = O(n)$. Compute an array C, where for each $k \in [2n-2]$, we have $C[k] := \min_{i \in [n-1]} (A[i] + B[k-i])$.*

The following reduction is also illustrated in Figure 2 on a small Pareto sum instance[3].

▶ **Theorem 3.** *An instance of bounded Pareto Sum with sets P, Q of size n and entries $(x, y) \in [W]^2$ can be reduced in time $\mathcal{O}(W)$ to an instance of bounded monotone min-plus convolution of two sequences A, B of size W with entries in $[W]$.*

Proof. (Sketch.) We first shift all x-coordinates in P and Q so that the minimum x-value becomes 0; this preserves all dominance relations in the Pareto sum.

Next, we construct two sequences A and B of length W, initialized with value W. For each point $(x, y) \in P$ (resp. $(x, y) \in Q$), we set $A[x] = y$ (resp. $B[x] = y$) and then enforce monotonicity by a single left-to-right sweep: each entry is replaced by the minimum seen so far. After this step, $A[i]$ (resp. $B[i]$) equals the smallest y-value among all points of P (resp. Q) with $x' \leq i$. This construction runs in $\mathcal{O}(W)$ time. Computing the min-plus convolution $C[k] = \min_{i+j=k} A[i]+B[j]$ then yields, for each k, exactly the smallest achievable y-coordinate among all pairs $p \in P$, $q \in Q$ with $p.x + q.x = k$. Thus C contains all candidate points of the Pareto sum. A final linear-time sweep removes dominated entries from C, leaving precisely the non-dominated pairs $(k, C[k])$. Hence the Pareto sum of P and Q can be obtained via a bounded monotone min-plus convolution instance computable in $\mathcal{O}(W)$ time. A more detailed proof can be found in the full version of the paper. ◀

According to our theorem, if bounded monotone min-plus convolution can be solved in time $T(n, W)$ for arrays of size n with entries in $[W]$, then we can solve bounded Pareto Sum in time $O(W + T(W, W))$. This makes algorithms for bounded monotone min-plus convolution applicable to Pareto Sum with little overhead to carry out the reduction and the back-transformation of the solution. By applying the algorithm of [11], we get:

▶ **Corollary 4.** *Given two Pareto sets P, Q of size n with integer entries $(x, y) \in [W]^2$, we can compute the Pareto sum of P and Q in time $\tilde{O}(W^{1.5})$.*

[3] A similar construction has been noted by Chan and Lewenstein for the special case that P and Q are connected sets i.e. all points in P and Q have a neighboring point with L_1 distance 1; see [9, Remark 3.4]

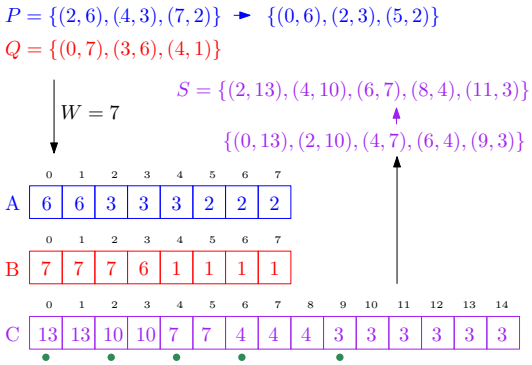


Figure 2 Reduction from Pareto sum to bounded monotone min-plus convolution and back-transformation of the solution on a small example.

3 Additive Approximation of Pareto sum

In this section, we study additive approximation algorithms for Pareto sum with arbitrary real-valued inputs.

To formalize our notion of approximation, observe that the Pareto sum of P and Q can be defined as the set S satisfying the following three conditions: (1) S dominates $P+Q$, i.e.., for every $p \in P$ and $q \in Q$, there exists some $s \in S$ with $s \leq p+q$, (2) $S \subseteq P+Q$ and (3) S is a Pareto set, i.e., pairwise non-dominating. To obtain an *approximate* Pareto set, it is only natural to relax the first condition by allowing an additive slack of $\Delta > 0$ in the domination inequality. Formally, we arrive a the following notion of approximation:

▶ **Definition 5** (Δ-Approximative Pareto sum). *Let $P, Q, \tilde{S} \subseteq \mathbb{R}^2$. We call $\tilde{S}$ a Δ-approximation of the Pareto sum of P, Q if the following conditions hold:*

1. *for all $p \in P, q \in Q$, there exists some $\tilde{s} \in \tilde{S}$ with $\tilde{s} \leq p+q+(\Delta,\Delta)$.*
2. *$\tilde{S} \subseteq P+Q$,*
3. *$\tilde{S}$ is a Pareto set.*

Let us remark that we could have arrived at this definition via slightly adapting the notion of an approximative Pareto front by the authors in [20], which leans on the notion of Papadimitriou and Yannakakis [22]: Let $d_H(S,\tilde{S})$ denote the directed Hausdorff distance between S and $\tilde{S}$ under some "distance" d, i.e., $d_H(S,\tilde{S}) := \max_{s\in S} \min_{\tilde{s}\in\tilde{S}} d(s,\tilde{s})$.

Following [20], we call a set $\tilde{S}$ a Δ-*approximation for the Pareto set* S if $d_H(S,\tilde{S}) \leq \Delta$. As a natural (asymmetric) "distance" measure $d(s,\tilde{s})$ that quantifies the disadvantage of taking $\tilde{s}$ over s (w.r.t domination of other points), we define: $d(s,\tilde{s}) = \max\{0, \tilde{s}.x - s.x, \tilde{s}.y - s.y\}$. Note that this measure is a more suitable definition than using, e.g., $\|\tilde{s}-s\|_\infty$, which would unnaturally punish a point that is worse by $\leq \Delta$ in x-direction, but *much better* (by $\gg \Delta$) in y-direction. With this notion, $\tilde{S}$ is a Δ-approximative Pareto sum of P, Q, if $\tilde{S}$ Δ-approximates S, (2) $\tilde{S} \subseteq P+Q$, and (3) $\tilde{S}$ is a Pareto set.

To obtain such an approximation, we proceed as follows: we first scale down the points of P and Q by dividing the x and y-coordinates by a scaling parameter t, creating the sets P' and Q'. We additionally save a *representative* $\text{rep}(p') \in P, \text{rep}(q') \in Q$ for each (rounded) point $p' \in P'$ and $q' \in Q'$ in the original sets (we choose the one with minimal L_1 norm but any representative works). Then we compute the Pareto sum S' for the sets P' and Q' and determine, from each $s' = p' + q' \in S'$, an approximate point $\tilde{s}$ using the representatives of p' and q', which yields an approximate set $\tilde{S}$. As we shall prove, this simple idea turns out to be conditionally optimal, by devising a fine-grained equivalence with

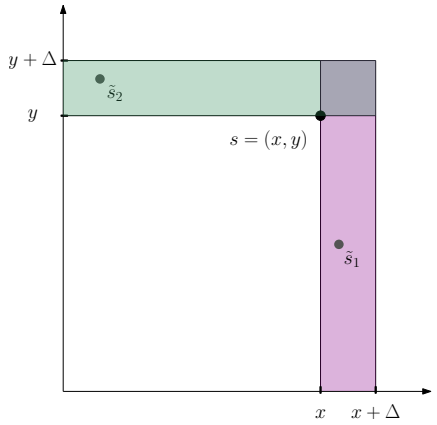


Figure 3 The points $\tilde{s}_1$ and $\tilde{s}_2$ are examples of points with $d(\tilde{s}, s) \leq \Delta$ for the shown $s \in S$. Note that if $s \in S$, then no point in $P + Q$ dominates s (i.e., no point lies to the lower left of s).

bounded $(\min, +)$-convolution. This algorithm is formalized in Algorithm 1, which takes as input P, Q and a scaling factor t, and returns a $2t$-approximation of the Pareto sum of P and Q.

Algorithm 1 ApproximateParetoSum(P, Q, t).

$P' \leftarrow \left\{\left(\lfloor \frac{p.x}{t} \rfloor, \lfloor \frac{p.y}{t} \rfloor\right) \mid p \in P\right\}$
$Q' \leftarrow \left\{\left(\lfloor \frac{q.x}{t} \rfloor, \lfloor \frac{q.y}{t} \rfloor\right) \mid q \in Q\right\}$
for $p' \in P'$ **do**
  $\text{rep}(p') \leftarrow \arg\min_{p \in P} \left\{\|p\|_1 \mid p' = \left(\left\lfloor \frac{p.x}{t} \right\rfloor, \left\lfloor \frac{p.y}{t} \right\rfloor\right)\right\}$
end for
for $q' \in Q'$ **do**
  $\text{rep}(q') \leftarrow \arg\min_{q \in Q} \left\{\|q\|_1 \mid q' = \left(\left\lfloor \frac{q.x}{t} \right\rfloor, \left\lfloor \frac{q.y}{t} \right\rfloor\right)\right\}$
end for
$P' \leftarrow$ Pareto front of P', $Q' \leftarrow$ Pareto front of Q' {*if necessary*}
$S' \leftarrow PS(P', Q')$, where $s' \in S'$ is given as (p', q') with $s' = p' + q'$ and $p' \in P', q' \in Q'$
$Z \leftarrow \{\text{rep}(p') + \text{rep}(q') \mid (p', q') \in S'\}$
return $\tilde{S} \leftarrow$ Pareto front of Z

▶ **Lemma 6.** *For any P, Q and $t > 0$, let S denote $PS(P, Q)$ and let $\tilde{S}$ denote the result of ApproximateParetoSum(P, Q, t). Then $\tilde{S}$ is a $2t$-approximation of the Pareto sum of P, Q.*

Proof. We first argue that the set Z computed in Algorithm 1 satisfies the relaxed domination condition with $\Delta = 2t$, specifically, that for all $p + q$ with $p \in P, q \in Q$, there exists some $z = \text{rep}(p') + \text{rep}(q') \in Z$ with $z \leq p + q + (2t, 2t)$. Indeed, for any $p \in P, q \in Q$, consider their rounded points $p' = \left(\lfloor \frac{p.x}{t} \rfloor, \lfloor \frac{p.y}{t} \rfloor\right) \in P'$, $q' = \left(\lfloor \frac{q.x}{t} \rfloor, \lfloor \frac{q.y}{t} \rfloor\right) \in Q'$. By definition of the Pareto sum S' of P', Q', there must exist some $(\tilde{p} + \tilde{q}) \in S'$ with $\tilde{p} + \tilde{q} \leq p' + q'$. We claim that $z := \text{rep}(\tilde{p}) + \text{rep}(\tilde{q})$ approximately dominates $p + q$, since

$$\begin{aligned} z &= \text{rep}(\tilde{p}) + \text{rep}(\tilde{q}) \leq (t\tilde{p} + (t, t)) + (t\tilde{q} + (t, t)) \\ &= t(\tilde{p} + \tilde{q}) + 2(t, t) \leq t(p' + q') + 2(t, t) \\ &\leq p + q + 2(t, t), \end{aligned}$$

where we used $\text{rep}(x') \leq tx' + (t,t)$ for any rounded point x' in the first line, and $tx' \leq x$ for any x and rounded point x' in the third line. We claim that the lemma follows: Filtering out dominated points in Z ensures that $\tilde{S}$ is a Pareto set (establishing Condition 3.), without affecting the domination condition (if $z \in Z$ dominates $z' \in Z$, then whenever $z' \leq p+q+(t,t)$, then $z \leq z' \leq p+q+(t,t)$, so z' is never needed to ensure the condition). Finally, clearly $\tilde{S} \subseteq Z \subseteq P+Q$ as any element of Z is explicitly given as $\text{rep}(p') + \text{rep}(q')$ for corresponding $\text{rep}(p') \in P$ and $\text{rep}(q') \in Q$. ◀

▶ **Theorem 7.** *Let $T(n,W)$ denote the runtime of monotone min-plus convolution of arrays of size n and entries bounded by W with witness reporting. Given sets P,Q of n points in $\mathbb{R}^2$, we can compute a Δ-approximate Pareto sum in time $\mathcal{O}(n+(W/\Delta)\log(W/\Delta)+T(W/\Delta,W/\Delta))$.*

Proof. Given Lemma 6, it remains to show that $\text{ApproximateParetoSum}(P,Q,t)$ with $t := \Delta/2$ can be computed in the desired running time. After $O(n)$-time preprocessing, the main step in Algorithm 1 is to compute the Pareto sum S' of P', Q'. By our rounding, P', Q' have integer coordinates in $\{0,\dots,\lfloor W/\Delta \rfloor\}$, and Theorem 3 is applicable to obtain S' and thus Z in time $T(\lfloor W/\Delta \rfloor, \lfloor W/\Delta \rfloor)$. Observe that Z is a set of size at most $O(W/\Delta)$. Thus, we can compute its Pareto front in time $O((W/\Delta)\log(W/\Delta))$, concluding the claim. ◀

▶ **Corollary 8.** *Given sets P,Q of n points with real coordinates in $[0,W]^2$ and $\epsilon > 0$, we can compute an ϵW-approximation of the Pareto sum in time $\tilde{\mathcal{O}}\left(n + (1/\epsilon)^{1.6}\right)$.*

Proof. Follows from the $\tilde{O}(n^{1.6})$ time algorithm of [11], and our witness reporting extension, described in the full version of the paper[4]. ◀

Let us also now show a lower bound, which will nicely complement Algorithm 1.

▶ **Theorem 9.** *Let $T_{apx}(n,W,\Delta)$ denote the running time of computing, given Pareto sets P,Q in $\mathbb{R}^2_{\geq 0}$ with maximum coordinate bounded by W, a Δ-approximation of their Pareto sum, with witnesses. If there exists $\alpha \geq 0$ and c such that $T_{apx}(n,W,\Delta) \leq O(n+(W/\Delta)^c)$ whenever $\Delta = \Theta(W^\alpha)$, then bounded monotone min-plus convolution can be computed in time $O(n^c)$, with witnesses.*

Proof. The proof is an adaptation of the reduction from min-plus convolution to Pareto sum given in [13]. Consider an arbitrary min-plus convolution instance $A[0,...,n-1], B[0,...,n-1]$ with monotonicity (specifically, $A[i] > A[i+1], B[i] > B[i+1]$ for all i) and entries in $\{0,\dots,O(n)\}$. We construct the sets of points

$$P := \{(i \cdot (2n^{\frac{\alpha}{1-\alpha}}), A[i] \cdot (2n^{\frac{\alpha}{1-\alpha}})) \mid i \in [n-1]\},$$
$$Q := \{(i \cdot (2n^{\frac{\alpha}{1-\alpha}}), B[i] \cdot (2n^{\frac{\alpha}{1-\alpha}})) \mid i \in [n-1]\}.$$

Consider a Δ-approximation $\tilde{S}$ of the Pareto sum of P,Q with $\Delta = n^{\frac{\alpha}{1-\alpha}}$. By construction of the point sets, if a point $\tilde{s} \in \tilde{S} \subseteq P+Q$ Δ-dominates a point $p+q$, i.e., $\tilde{s} \leq p+q+(\Delta,\Delta)$, then it already dominates $p+q$, i.e., $\tilde{s} \leq p+q$. Thus, $\tilde{S}$ is the true Pareto sum of P,Q, and we can extract the min-plus convolution C of A,B: By construction, for every $k \in [2n-2]$, there must be a point in $PS(P,Q)$ with x-coordinate $k \cdot (2n^{\frac{\alpha}{1-\alpha}})$ and y-coordinate $(\min_{i \in [n-1]} A[i] + B[k-i]) \cdot (2n^{\frac{\alpha}{1-\alpha}})$, revealing $C[k] = \min_{i \in [n-1]}(A[i]+B[k-i])$. The existence of a point with x-coordinate $k \cdot (2n^{\frac{\alpha}{1-\alpha}})$ for every $k \in [2n-2]$ follows by the decreasing values of $C[k]$. Also, witnesses for such points can be transferred.

[4] We remark that it is very plausible that our witness reporting algorithm extends to the $\tilde{O}(n^{1.5})$ algorithm.

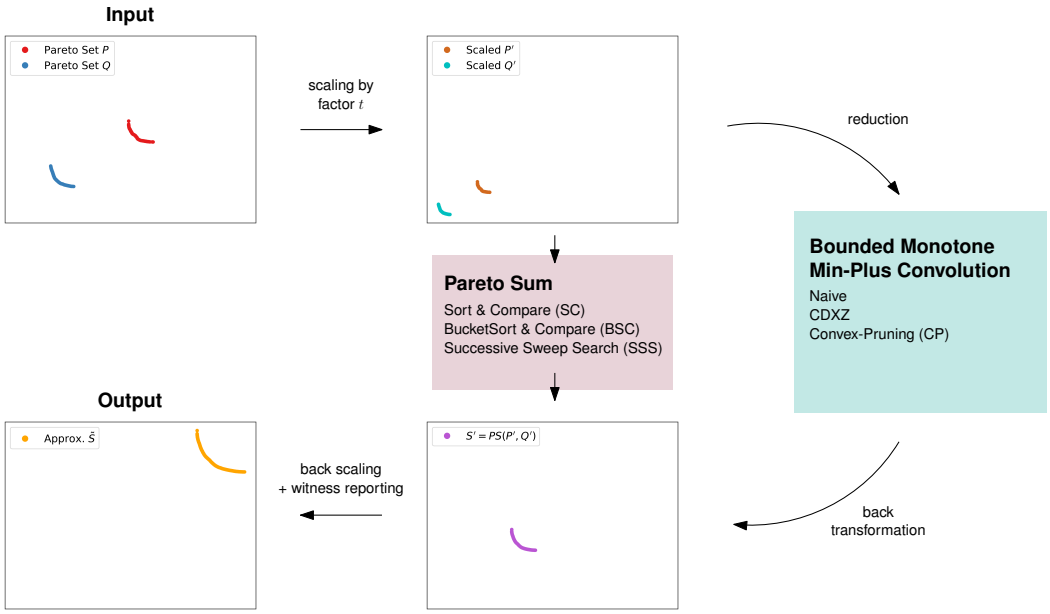


Figure 4 Overview of the main steps of our additive approximation algorithms. An important step is to compute the Pareto sum of the scaled instance. This can either be done directly via a Pareto sum algorithm or by an algorithm for bounded monotone min-plus convolution when applying the proper forward and backward reduction steps.

Note that the constructed instance of Pareto sum Approximation consists of n points in P, Q with coordinates bounded by $W \leq O(n^{1+\frac{\alpha}{1-\alpha}}) = O(n^{\frac{1}{1-\alpha}})$ and $\Delta = n^{\frac{\alpha}{1-\alpha}}$. Thus, if $T_{apx}(n, W, \Delta) \leq O(n + (W/\Delta)^c) = O(n + (n^{\frac{1}{1-\alpha}} - \frac{\alpha}{1-\alpha})^c)$, then we can compute the min-plus convolution of A, B in time $O(n^c)$, with witnesses. ◀

▶ Remark 10. Even without witness reporting, we still get a fine-grained equivalence (for slightly weaker guarantees), by a minor adaptation of Algorithm 1 and Theorem 9. We relax the condition $\tilde{S} \subseteq P + Q$ to the milder condition that each $\tilde{s}$ is in the vicinity of a point $p + q \in P + Q$. More formally, for all $\tilde{s} \in \tilde{S}$ there exist $p \in P$ and $q \in Q$ such that $p + q \leq \tilde{s} \leq p + q + 2(t, t)$. Further details can be found in the full version of the paper.

4 Framework and Implementation

Our proposed additive approximation algorithm, presented in Algorithm 1, consists of scaling down the instance, computing the Pareto sum of the transformed point sets, and then scaling up the solution. The Pareto sum instance after scaling is captured by our definition of bounded Pareto sum. To compute its solution, we can obviously directly apply any Pareto sum algorithm. By virtue of Theorem 3, however, also any algorithm for bounded min-plus convolution translates to an algorithm for bounded Pareto sum when performing the required reduction and back-transformation of the solution. Thus, depending on whether we use an algorithm directly designed for Pareto sum or a convolution algorithm, the computation pipeline needs to be adapted. Figure 4 provides an overview of our framework. In the following, we describe the algorithms of each category in more detail.

4.1 Algorithms for (bounded) Pareto sum

We start with the algorithms designed directly for Pareto sum computation.

Sort & Compare (SC) and Successive sweep search (SSS)

We implemented both state of the art algorithms for (general) Pareto sum computation from [13] as a baseline. In a nutshell, the Sort & Compare algorithm sorts the points in the sumset $P + Q$ lexicographically and performs constant time dominance checks per point based on this ordering. Further details on the baselines can be found in the full version of the paper.

BucketSort & Compare (BSC)

For bounded integral Pareto sum, we can improve the Sort & Compare algorithm, by using bucketsort to sort the points, and then perform a direct update on the minimal y-coordinate for each corresponding x-coordinate. Further details can be found in the full version of the paper.

4.2 Algorithms for min-plus convolution

After the reduction to min-plus convolution, we can of course use the naive quadratic time algorithm for the latter. In the following, we discuss two more sophisticated and (potentially) faster convolution algorithms.

CDXZ Algorithm

We discuss a simplified version of the $\tilde{O}(n^{1.6})$-algorithm [11, Section 4.1] for monotone bounded min-plus convolution in a detailed fashion in the full version of the paper. Using this algorithm in our Pareto sum reduction framework results in a running time of $\mathcal{O}(W^{1.6} \log W)$ for bounded Pareto sum. Let us highlight the main improvements and simplifications we achieved.

- We removed the segment tree data structure. Making use of a simpler enhanced min-plus convolution algorithm instead of using a segment tree with interval updates is simpler and improves the runtime theoretically and practically.
- We improved the bound of b, which was previously ranging from $b \in \{0, 1, 2\}$ to $b \in \{0, 1\}$. The better bound of the constant b, is insignificant in the theoretical analysis of the algorithm, but it plays an important role in the error correction part. Besides the initial polynomial multiplication of P and Q, every other part of the algorithm in the error correction phase will be repeated b-times, thus the better bound on b will result on repeating these costly parts of the algorithm twice instead of thrice.
- As the set T_b can be of expected size $\mathcal{O}(n^{1.6})$ (for the choice of $\beta = 0.4$), we do not save it explicitly like [11] suggests. In order to circumvent storing the set T_b, we make an online computation of the polynomials $R_{p,b}[k](x)$ simultaneously, updating the polynomials after having seen a new pseudo-witness $(i, k - i)$ on the fly.

We implemented the version without witness reporting, as the witness reporting algorithm would be too inefficient in practice. This leads to a weak approximation of the Pareto sum. For the multivariate polynomial multiplication, we perform a standard reduction to the univariate case, by turning P and Q into univariate polynomials. Instead of FFT, we make use of the polynomial multiplication offered by the library FLINT [24], where the univariate

polynomials are kept as lists. Instead of using binary search trees in the computation of T_b, we performed a linear search, after saving for each possible modulo value $i \in [p-1]$ the indices k with $C'[k] + b \mod p = i$. This turned out to be faster in the experiments. The parameters α and β are not fixed as the optimal theoretical runtime suggests. The choice of the parameters α and β are transferred to the choice of the scaling factor n^α and the prime p, which are now inputs of the implementation; see Figure 7 for the influence of these parameters on the runtime.

Convex Pruning (CP) algorithm

Bringmann and Cassis developed an algorithm for min-plus convolution, suited towards the case, where the arrays A and B are near-convex [5]. We implemented two versions of their algorithm with and without witness reporting. Our implementation is close to the descriptions in [5], with the main difference being that we did not use sparse convolutions, but rather an enhanced min-plus convolution computation, which works particularly well when the arrays A and B have few distinct entries. We precomputed groups of entries of the same value in A and B. More details on the enhanced min-plus convolution on the precise changes can be found in the full version of the paper.

4.3 Further engineering

Algorithm 1 has been implemented mostly matching its description, with small changes, which we describe now. When using a convolution algorithm within Algorithm 1, we also perform the needed reduction from bounded Pareto sum to monotone bounded min-plus convolution and the transformation backwards. The reduction closely follows the constructive proof of Theorem 3. At the end of Algorithm 1, instead of computing the Pareto front of the set Z by sorting, we do the following: For each newly inserted point, we look back and remove all points, which are dominated by this new point. This will work fast as the set Z is almost a Pareto set, so few corrections need to be made.

5 Experimental Evaluation

In the following, we evaluate our approximate algorithm with the different choices for bounded Pareto sum computation sketched above. While all algorithms are described for the case that P and Q are of the same size n, they also all work for differing sizes. Our code is written in C++. It was compiled with gcc using -O3. In our implementation of the CDXZ algorithm, we use routines from FLINT (Fast Library for Number Theory) [24]. All experiments were conducted on a single core of an AMD Threadripper 3970 with 256 GB of RAM. We set a time-out of one hour per run. Running times are always averaged over 10 runs.

Benchmarks. We use three types of benchmarks in our evaluation. Two are synthetic to allow us to control instance properties, while the third benchmark consists of real data.

Benchmark 1: Instances with bounded range. First, we use an instance generator that allows us to fix n and W. Here, it is important to ensure $W \geq n-1$ to guarantee that there are enough distinct natural numbers in the range to construct a Pareto set. We use $W = c \cdot n$ for $c \geq 1$. To efficiently create a strictly monotonically increasing sequence, we initialize an array with a counter γ_i for each $i \in [W]$. Then, we increment a randomly selected counter in each of n rounds. Afterwards, we sweep over this array, starting at position 0, and perform

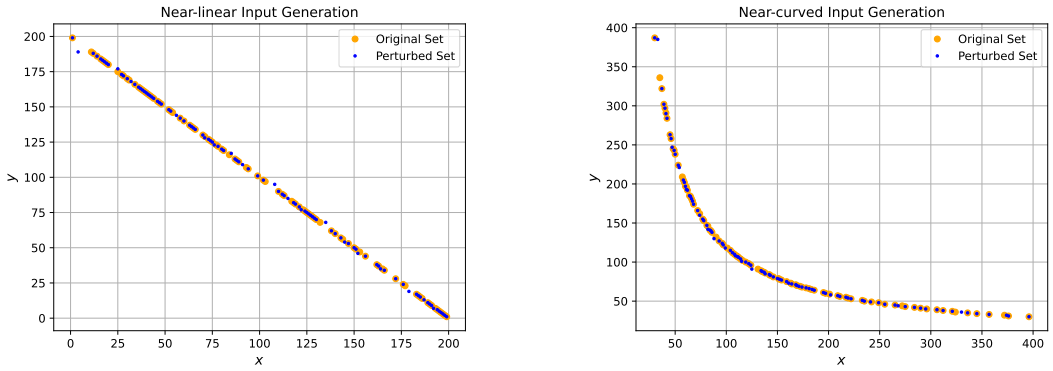


Figure 5 Two different versions of generating input sets with values in the range $[0, 2n]$: Near-linear and near-curved input sets are based on the linear and curve generator, respectively, where we apply perturbation to a subset of the points.

the following operations: If $\gamma_i > 1$, we set $\gamma_i = 1$ and $\gamma_{i+1} = \gamma_{i+1} + \gamma_i - 1$. Here, entry 0 is assumed to be the successor of γ_W. We stop when all entries are 0 or 1 which happens after each entry has been considered at most twice. Then, for all $\gamma_i = 1$, element i is added to the sequence. This produces a valid sequence in time $\mathcal{O}(n + W)$. Pareto sets can then be constructed by using two such sequences, one sorted increasingly for the x-coordinates and the other one decreasingly for the corresponding y-coordinates.

Benchmark 2: Function-based instance generator. For any strictly monotonically decreasing function f defined on $\mathbb{R}^+$, sampling n points $(x, f(x))$ provides us with a Pareto set. We use two base functions: the first uses $f(x) = -x$ and the second $f(x) = \frac{c}{x}$ with a constant $c \in \mathbb{R}^+$. We slightly perturb the points to have more input variability, while guaranteeing that pairwise non-dominance is upheld. We refer to the resulting instances as near-linear or near-curved. See Figure 5 for an illustration. We note that by means of the perturbation, the resulting point sets are no longer convex.

Benchmark 3: Route planning instances. Finally, we also use real-world data sets for evaluation. The instances stem from bi-criteria route planning in line with [13]. Here, given a graph in which the edges are augmented with two cost values (in our case distance in meters and positive height difference in centimeters), the goal is to compute all Pareto-optimal paths from a source node s to a target node t. Using bi-directional search to answer such queries, or constructing a data structure which accelerates query answering [23], demands to combine and filter the set of Pareto-optimal paths from s to some intermediate node v, and from v to the target t. The cost pairs of these two path sets form our inputs P and Q. The benchmark contains roughly 100000 instances based on randomly chosen s, v and t triples in a directed road network with one million nodes and two million edges. Note that the sizes of P and Q might differ significantly in each instance.

Results for Exact Pareto Sum. For a general comparison of the different algorithms for Bounded Pareto sum discussed in this paper, we first applied a scalability study on the range-bound instances with all point coordinates being integers, see Figure 6. We observe huge performance differences. SSS is by far the slowest approach. Its theoretical performance is only subquadratic in case the output size is in $o(n)$. However, for the considered instances there are roughly $4n$ output points. The BSC algorithm performs best and also significantly

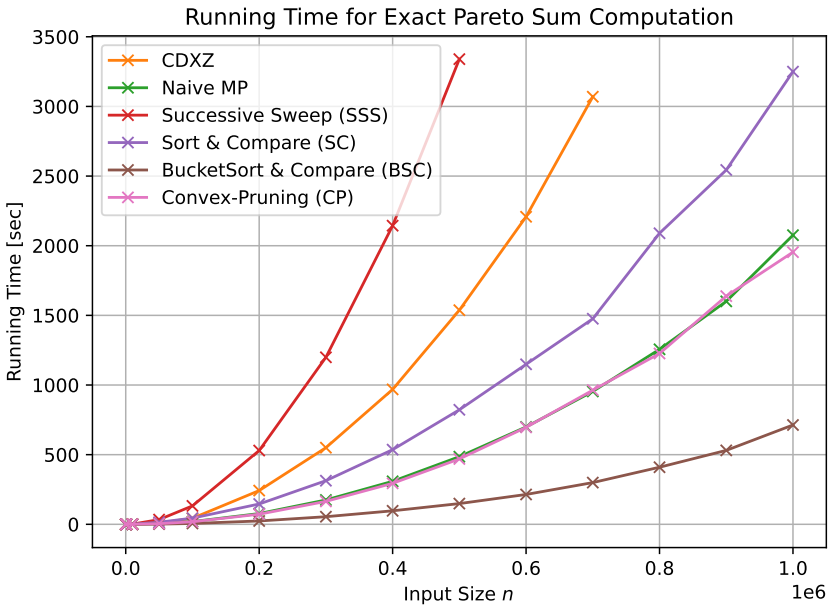


Figure 6 Running times for Pareto sum computation on generated inputs of size up to $n = 100,000$ with values in the range of $[0, 2n]$ and a timeout of 1 hour.

outperforms the plain SC algorithm. Even more remarkably, also the min-plus convolution based algorithms, naive and CP, outperform SC. While CDXZ beats SSS, it is a bit slower than SC on this instance type. As described in Section 4, our implementation of the CDXZ algorithm allows us to choose its parameters. We conducted a series of tests on synthetic instances of different sizes to determine a good parameter choice, see Figure 7 for an example. We observe that the parameter values heavily impact the performance. Across all tested

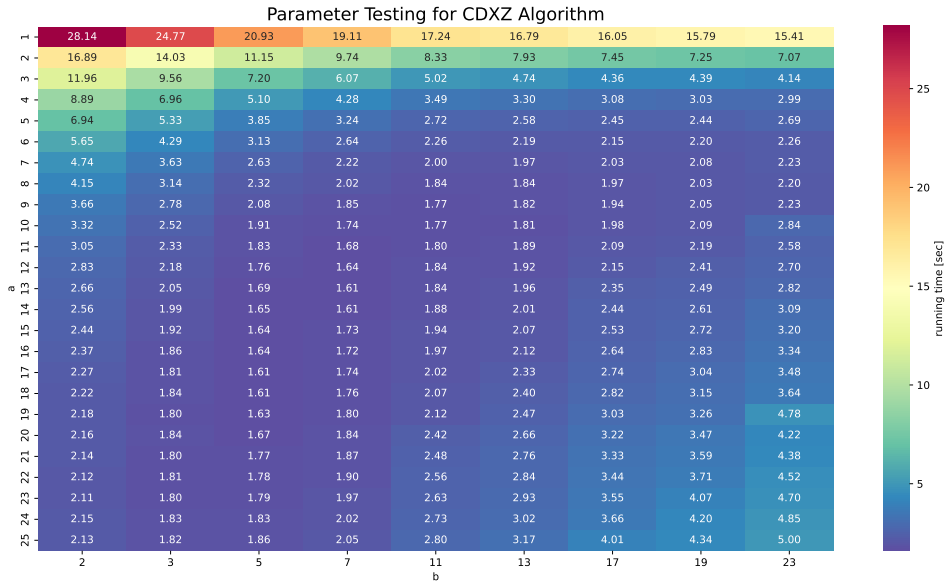


Figure 7 Parameter testing for the CDXZ algorithm on inputs of size $n = 20000$ in the range $2n$.

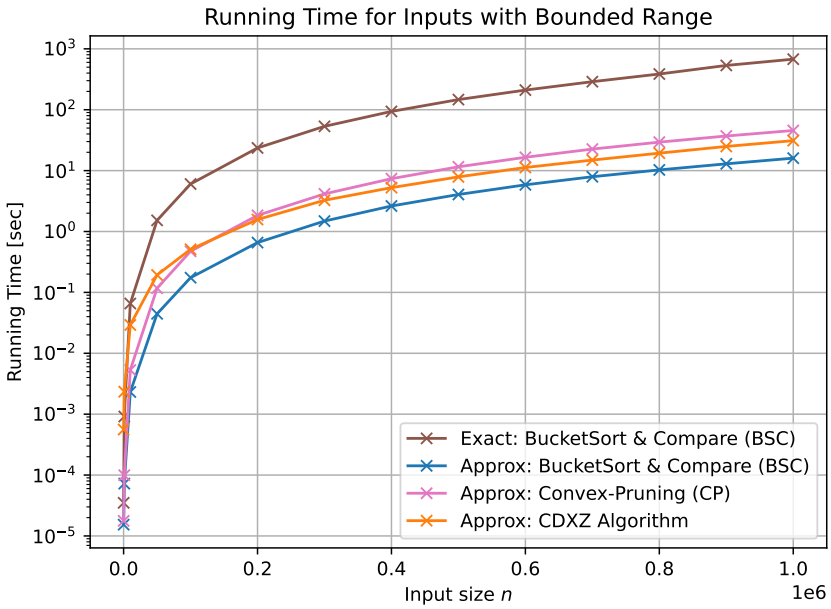


Figure 8 Running time of the approximation variants on inputs generated with bounded range of $2n$ using a scaling factor of $t = 10$.

instances, scaling factors of 25 and $p = 2$ were among the top configurations. We therefore use those values in all subsequent experiments. We remark, though, that additional fine-tuning or choices adaptive to n could potentially further improve the performance.

However, our main interest is in computing approximate solutions. In the following, we investigate how the algorithms perform when used as a plug-in in our approximation framework. In the remainder of the experiments, we focus on the BSC algorithm as well as the convolution based algorithms, as they outperform the baseline algorithms.

Results for Approximate Pareto Sum. Next, we present comparative running time results of our approximation algorithm with different inner methods of Pareto sum computation applied to the scaled down inputs. For fair comparison, we use the non-witness reporting variants for all of them. Figures 8, 9, and 10 depict the results for range-bounded, near-linear and near-curve instances, respectively, using a value range of $2n$ and a scaling factor of $t = 10$. Interestingly, for each input type, another algorithm performs best. For range-bounded instances, BSC is the fastest with CDXZ being second, while on near-linear instances CDXZ outperforms the other two methods. On near-curved instances, however, CP is clearly the best, beating the others by at least an order of magnitude. The performance of CP relies on its ability to prune pairs, which on near-curved instances applies to up to 95% of total point pairs, see Figure 11.

Finally, we consider our third benchmark set, which consists of bi-criteria route planning instances of varying size. Figure 13 shows the running time and approximation quality of our algorithms for different scaling factors. We observe that BSC performs best with CP being a close second, especially for larger values of t. They both improve on exact computation by over an order of magnitude and also stay well below the theoretical additive approximation bound of $2t$. Examples of approximate solutions can be found in Figures 14. One can see that even when choosing a larger t, the Pareto sum is still well represented by the computed

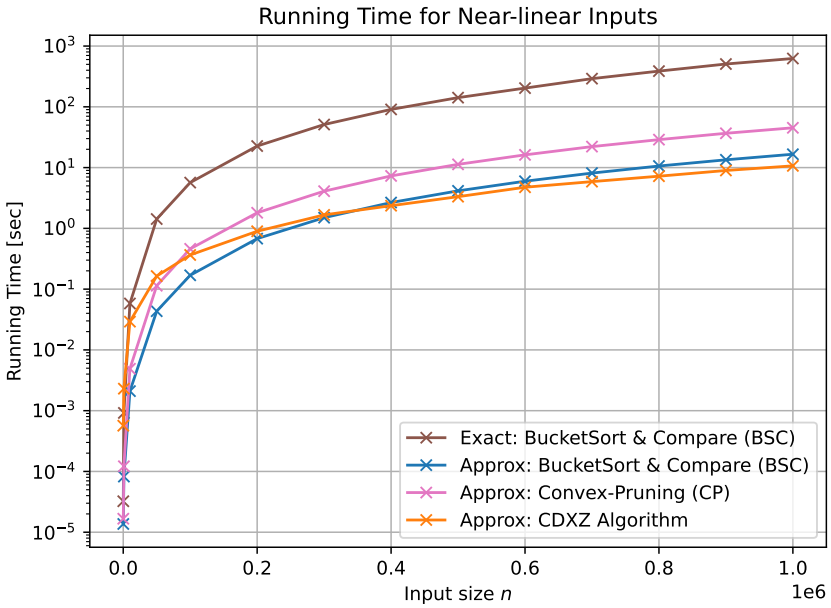


Figure 9 Running time of the approximation variants with a scaling factor of $t = 10$ on near-linear inputs generated in the range of up to $2n$.

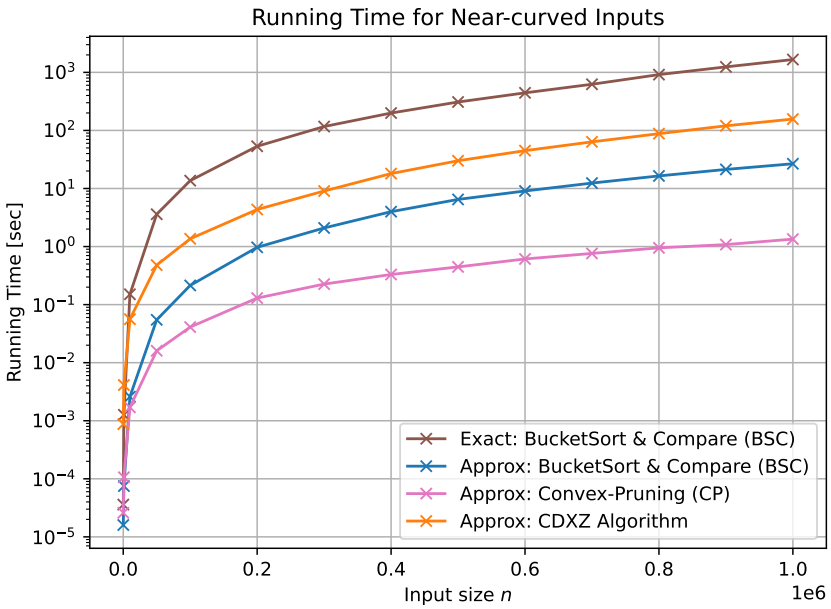


Figure 10 Running time of the approximation variants with a scaling factor of $t = 10$ on near-curved generated inputs in the range of up to $2n$.

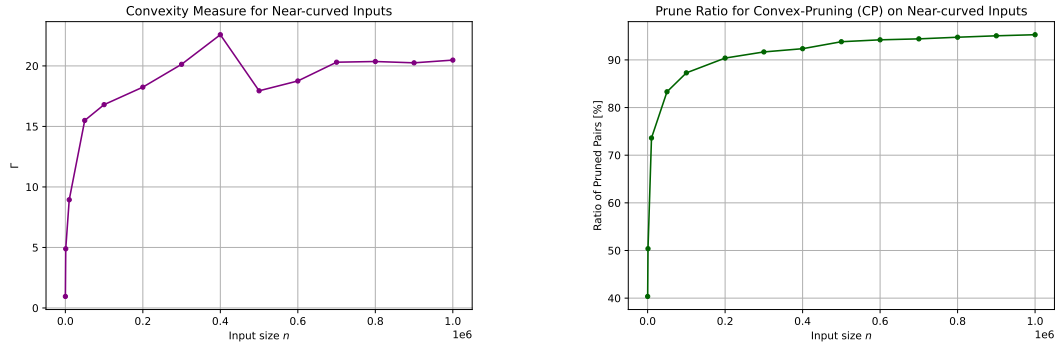


Figure 11 Convexity measure for near-curved inputs (left) and ratios of pruned pairs by the CP algorithm with respect to all available pairs in the index space (right).

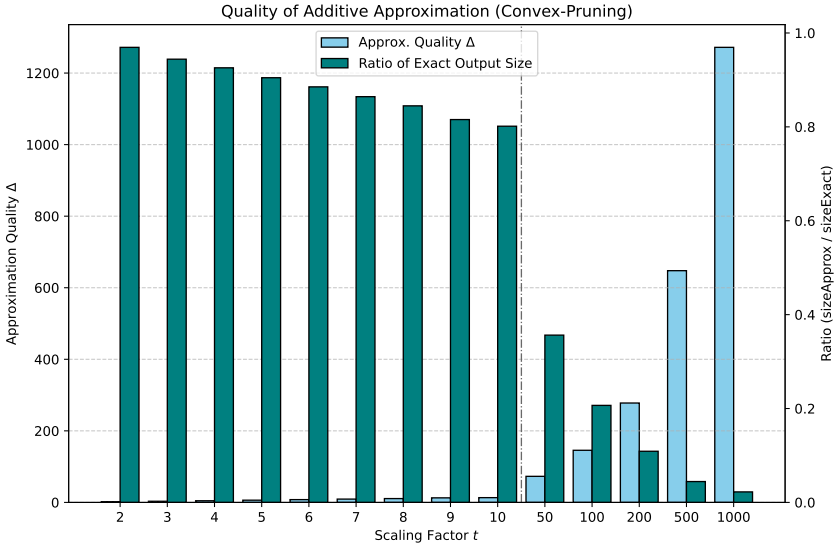


Figure 12 Quality of the approximation using the CP algorithm on real benchmarks for varying scaling factors $t = 2, \ldots, 1000$. We compare the ratio of the output size of the approximation and the exact algorithm to the approximation quality Δ.

solution. Indeed, larger values of t are quite reasonable in our application. For example, a route that is up to 100 meters longer than another one is almost indistinguishable to a user if the total route length is anyway in the orders of kilometers.

Furthermore, the user usually does not want to be presented with hundreds of routing options, but a concise and diverse set of reasonable options. This is achievable with our approach, as we can not only reduce the running time with larger t but also the output size, see Figures 15 and 12. Clearly, our algorithm provides good trade-offs between running time and solution quality, which allows one to choose the best t depending on the application.

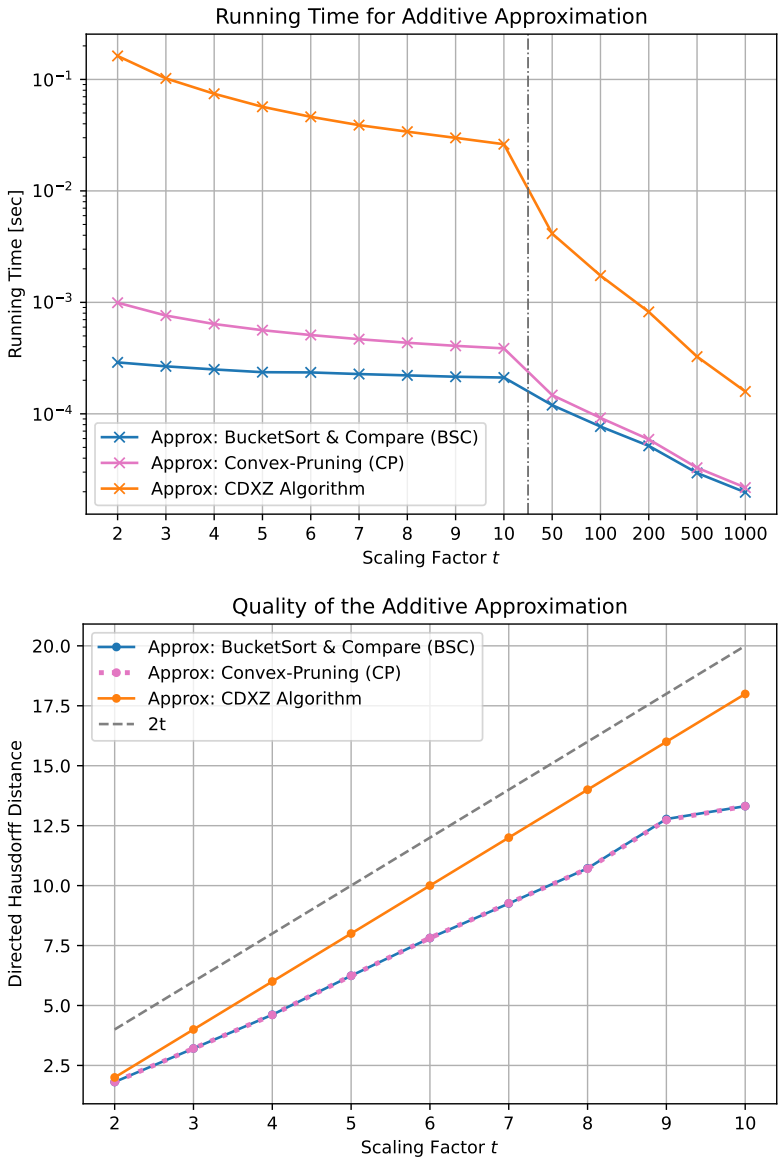


Figure 13 Running time of approximations on real benchmark for varying scaling factors $t = 2, \ldots, 1000$ (top) and solution quality for $t = 2, \ldots, 10$ (bottom). This shows the difference of the quality of approximation with (BSC,CP) and without witness reporting(CDXZ).

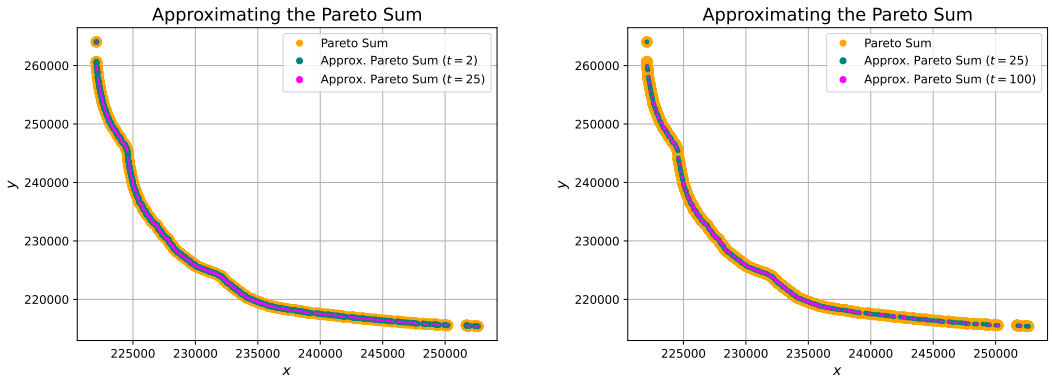


Figure 14 Example for approximate Pareto sum on real benchmark data. The approximative solution for $t = 2$ results in a quality of $\Delta = 2$ (left) whereas for $t = 25$ the quality is $\Delta = 38$ and for $t = 100$ we have $\Delta = 153$ (right).

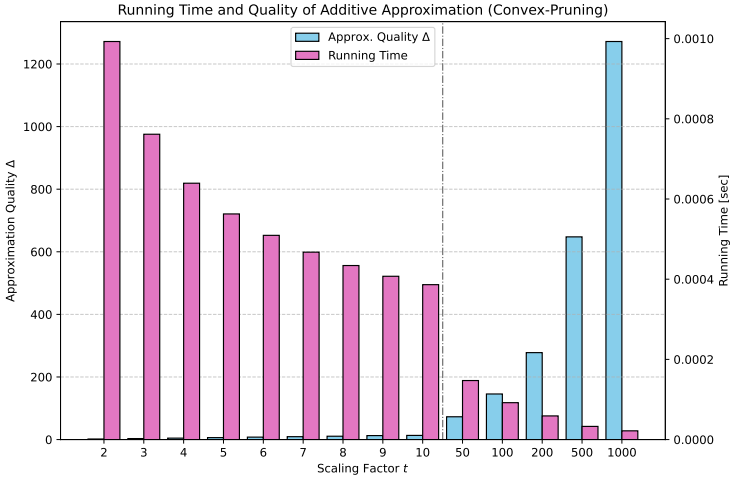


Figure 15 Approximation quality Δ and running time of the approximation using the CP algorithm on real benchmarks with varying scaling factor $t = 2, \ldots, 1000$.

6 Future Work & Discussion

We have established that fine-grained reductions are not only a means to obtain theoretical intractability results, but also a tool to efficiently solve problems via reductions to other problems for which fast algorithms are implemented and engineered. To further foster (bounded monotone) min-plus convolution as a core primitive, it would be interesting to know whether the $\tilde{O}(n^{1.5})$ algorithm [11, Section 4.2], admits an efficient implementation. The main drawback right now is that it needs 9 calls to the monotone bounded min-plus function, and that the constant $b \in [-4, 7]$ is much larger than in the algorithm we implemented. We believe that the CP algorithm can benefit from practical sparse convolution algorithms, especially on near-linear inputs. Is it possible to engineer fast sparse convolution algorithms in practice? Finally, can we leverage our min-plus convolution results to obtain practical improvements for knapsack [4, 6] or the tree sparsity problem [21, 2]?

References

1 Noga Alon and Moni Naor. Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16(4/5):434–449, 1996. doi:10.1007/BF01940874.

2 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. Better approximations for tree sparsity in nearly-linear time. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2215–2229. SIAM, 2017. doi:10.1137/1.9781611974782.145.

3 Karl Bringmann. Approximating subset sum ratio faster than subset sum. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1260–1277. SIAM, 2024. doi:10.1137/1.9781611977912.50.

4 Karl Bringmann and Alejandro Cassis. Faster knapsack algorithms via bounded monotone min-plus-convolution. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 31:1–31:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.31.

5 Karl Bringmann and Alejandro Cassis. Faster 0-1-knapsack via near-convex min-plus-convolution. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, Amsterdam, The Netherlands, September 4-6, 2023*, volume 274 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.24.

6 Karl Bringmann, Anita Dürr, and Adam Polak. Even faster knapsack via rectangular monotone min-plus convolution and balancing. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.33.

7 Karl Bringmann, Ahmed Ghazy, and Marvin Künnemann. Exploring the Approximability Landscape of 3SUM. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2024.34.

8 Karl Bringmann and Vasileios Nakos. A fine-grained perspective on approximating subset sum and partition. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1797–1815. SIAM, 2021. doi:10.1137/1.9781611976465.108.

9 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3sum via additive combinatorics. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 31–40. ACM, 2015. doi:10.1145/2746539.2746568.

10 Wei-Mei Chen, Hsien-Kuei Hwang, and Tsung-Hsi Tsai. Maxima-finding algorithms for multidimensional samples: A two-phase approach. *Computational Geometry*, 45(1-2):33–53, 2012. doi:10.1016/J.COMGEO.2011.08.001.

11 Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. 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 1529–1542. ACM, 2022. doi:10.1145/3519935.3520057.

12 Weiming Feng and Ce Jin. Approximately counting knapsack solutions in subquadratic time. 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 1094–1135. SIAM, 2025. doi:10.1137/1.9781611978322.32.

13 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.

14 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. doi:10.4230/LIPIcs.ITCS.2025.55.

15 Geri Gokaj, Marvin Künnemann, Sabine Storandt, and Carina Truschel. (multivariate) k-sum as barrier to succinct computation. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms, ESA 2025, September 15-17, 2025, Warsaw, Poland*, volume 351 of *LIPIcs*, pages 42:1–42:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.42.

16 Boris Goldin and Oren Salzman. Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 149–158, 2021. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/15957.

17 David G Kirkpatrick and Raimund Seidel. Output-size sensitive algorithms for finding maximal vectors. In *Proceedings of the First Annual Symposium on Computational Geometry*, pages 89–96, 1985. doi:10.1145/323233.323246.

18 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. doi:10.1016/J.COR.2023.106506.

19 Joshua Marc Könen, Heiko Röglin, and Tarek Stuck. Parameterized Algorithms for Computing Pareto Sets. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms (ESA 2025)*, volume 351 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 105:1–105:15, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2025.105.

20 Julien Legriel, Colas Le Guernic, Scott Cotton, and Oded Maler. Approximating the pareto front of multi-criteria optimization problems. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2010. doi:10.1007/978-3-642-12002-2_6.

21 Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. A subquadratic approximation scheme for partition. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 70–88. SIAM, 2019. doi:10.1137/1.9781611975482.5.

22 Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, Redondo Beach, California, USA, November 12-14, 2000*, pages 86–92. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892068.

23 Sabine Storandt. Route planning for bicycles—exact constrained shortest paths made practical via contraction hierarchy. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 22, pages 234–242, 2012.

24 The FLINT team. *FLINT: Fast Library for Number Theory*, 2025. Version 3.3.1, https://flintlib.org.

25 Carina Truschel and Sabine Storandt. Multi-Criteria Route Planning with Little Regret. In Jonas Sauer and Marie Schmidt, editors, *25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025)*, volume 137 of *Open Access Series in Informatics (OASIcs)*, pages 13:1–13:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2025.13.